

AI-Run Research on Ninya Discovers the Headroom Law, Recursively Improves Its Own Harnesses, and Sets 325 Mathematical Records

Lindell S. Cumes

QWNTL Labs

Project page: <https://qwntl.com/frontier>

Abstract

An organization of AI agents running on Ninya, QWNTL’s harness for durable, ultra-long-horizon agent work, and conducted by an agent named frontier-bond, ran a three-day research program on agent harnesses with 3,230,847 Claude Opus 5.5 calls. It improved harnesses with evidence-gated recursion (EGR): a model proposes a change, a gate admits it only on hard evidence, and the harness keeps and reuses it. EGR ran at four levels (components, harness designs, a running system across a model upgrade, and Ninya itself), and each improvement held on unseen work: model-written checks raised catch of never-seen failures from 7.3% to 95.6%, and an evolved agent engine beat ACE on seven never-used problem families with Claude Haiku 4.5. The harnesses carried Claude Opus 5.5 past published records on 325 instances of open extremal problems, with exact certificates, including $M \leq 0.380856547875 \dots$ for Erdős’s minimum overlap problem, and past CORAL (914 vs. 948 cycles) and ACE ($3.67\times$ vs. $1.95\times$) on the same model. Evidence-only supervision left 0 of 2,880 production tickets below the bare model; verdict-based supervision fell below it on every stream. Auditing its record, the organization found the law behind these results: a component converts only the failures whose cause it supplies, so its lift scales with the headroom the model leaves ($k = 0.54$, 95% CI 0.43–0.68), and a share learned on weaker models outpredicts constant and capped gains on the frontier model. The law explains why EGR compounds while recursion that admits changes on conclusions degrades.

Keywords: agent harnesses; recursive self-improvement; evidence-gated recursion; AI-conducted research; language-model agents; extremal combinatorics; circle packing

1 Introduction

Given code execution, web access, and two hours, Claude Opus 5.5 matched 23 of 30 published construction frontiers in the Endless Exam and surpassed none (Zhang, 2026). Inside harnesses that AI agents built for it, the same model went below every published upper bound on Erdős’s minimum overlap constant and on the first and signed autocorrelation constants, and above every published sum-of-radii packing at 306 sizes from $N = 137$ to 10,000. It beat CORAL on Anthropic’s performance take-home and ACE on KernelBench, head-to-head on the same model and hardware, and it caught its own failures 5.5 times as often as it could by judging them (Table 3 and Figure 1). The model did not change. The harnesses did, and they were built, measured, and improved by an organization of AI agents running on Ninya, conducted by an agent named frontier-bond.

Harness research asks how much machinery a strong model needs. MILO co-evolves complete harnesses (Jana et al., 2026), HarnessTTT learns a harness proposer against a frozen executor (Ma et al., 2026), ACE grows a playbook (Zhang

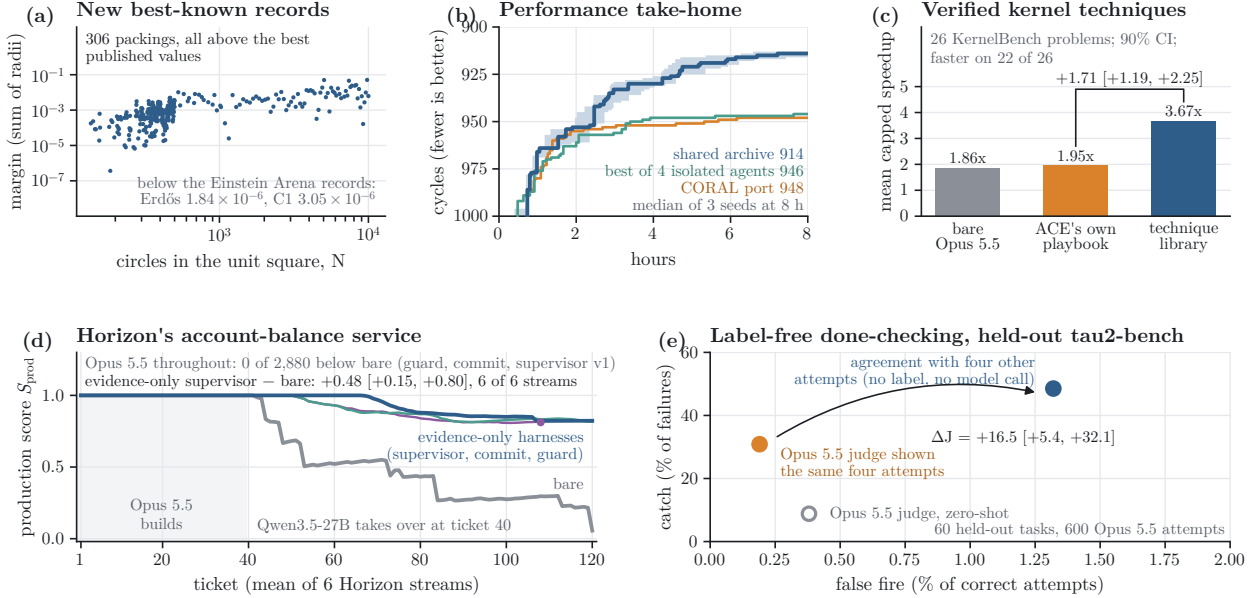


Figure 1: **Harnesses built by the AI-run program carry Claude Opus 5.5 past published records and named rivals in five experiment families.** (a) Exact constructions: 306 sum-of-radii packings above the best published values, and Erdős and C_1 bounds below the Einstein Arena records (Section 5.4). (b) Anthropic’s performance take-home: cycles over 8 hours, median of three seeds per arm, lower is better; the band spans the shared-archive seeds (Section 5.5). (c) Mean capped KernelBench speedup on 26 problems outside the trials used to build the technique cards, 720 episodes; the gap to ACE’s playbook carries its 90% CI (Section 5.6). (d) Production correctness on HORIZON’s account-balance service, mean of six streams, with Qwen3.5-72B taking over at ticket 40. With Claude Opus 5.5 throughout, three evidence-only harnesses had 0 of 2,880 tickets below bare (Section 5.7). (e) Failure catch and false fire on 600 Claude Opus 5.5 attempts at 60 held-out τ^2 -bench tasks. Agreement uses four other attempts, and the comparison judge sees the same attempts; the ΔJ interval is 95% (Section 5.9).

et al., 2025b), CORAL coordinates agents through shared attempts (Qu et al., 2026), and Hill Sampling argues that one thread editing its best verified program beats elaborate machinery (Beck, Ogren & Kobren, 2026). Harness² makes the harness recursive: an editor model proposes edits to one task’s harness, probes them, composes the best, and keeps a library of adopted edits for later tasks (Xu et al., 2026). Each of these is research done by people about harnesses, reported as the end score of a whole system. None predicts which component will move a frontier model, by how much, or when a component will make it worse. Two questions are therefore open: whether a harness can be improved recursively without the improvements eroding, and whether the gains a component delivers follow a law that can be stated and tested before the component is built.

This paper answers both questions with research that AI agents ran. The whole organization ran on Nanya, QWNTL’s harness for ultra-long-horizon agent work, which supplies durable infrastructure and executes agents with a minimal human in the loop and continual learning. Its mission was to beat named public results on public benchmarks with official scoring. frontier-bond, the organization’s conductor, owned the program’s thirteen outcomes and its shared infrastructure, and more than a dozen team leads staffed mixed teams of Claude Opus 5.5 and GPT-6 Astra seats, including contrarian seats charged to chase ideas against the plan. The agents designed the harnesses, built the instruments, preregistered predictions and kill criteria, ran the experiments, audited the core claims across model families, audited their own record, and wrote the reports. Between October 7 and 9, 2026, the program’s gateway logged 3,230,847 Claude Opus 5.5 calls (Section 4.1).

The program improved harnesses with one loop, *evidence-gated recursion* (EGR): a model proposes a change, a gate admits it only on hard evidence, and the harness keeps and reuses it (Section 4.2). EGR ran at four levels, and each level is measured on work the improved component had never seen (Section 5.3). Components rewrote themselves: Claude Opus 5.5’s checks, revised after every window under an evidence gate, raised catch of never-seen failure states from 7.3% to 95.6%. Designs were evolved and redesigned: an automatically evolved agent engine lifted Claude Haiku 4.5

past ACE on seven ALE-Bench families it had never seen, and a supervision harness rebuilt from its own failures went from below bare on every stream to 0 of 2,880 tickets below bare. A running system repaired its own memory across the Claude Opus 4.8 → Claude Opus 5.5 upgrade: EGR diagnosed a store that slowed the new model, gated the repairs, and deployed a verified technique library live that turned the store into a lift for both models. At the top, EGR rebuilt the program’s own harness, *Nenya*, from its findings.

Running EGR on its own record, the organization found the law behind every result. A component converts a control’s failure into a success only when it supplies that failure’s cause, and where the control already succeeds it can only break what works (Proposition 3.4):

$$\mathbb{E}[\Delta] = \left(\sum_j \phi_j p_j \right) h - q(100 - h), \quad (1)$$

where h is the control’s headroom, ϕ_j the share of its failures with cause j , p_j the rate at which the component converts them, and q the rate at which it breaks a success. Across the program’s comparisons against a no-harness control, the lift is $\Delta \approx kh$ with $k = 0.54$ (95% CI 0.43–0.68). The equation is an exact accounting of a comparison, and its content is in its parameters: evidence-gated components held $q = 0$ where a verdict-based positive control did not; where a failure’s cause could be isolated, ϕ was measured before the component and bounded what it recovered; and lifts scale with headroom, which outpredicts constant and capped gains in and out of sample (Section 5.2). The law says where EGR pays, on causes outside the model’s episode, and why EGR gates on evidence: a change admitted on a conclusion raises q .

This paper makes six contributions.

1. **Research run by AI.** An organization of AI agents running on *Nenya*, with frontier-bond as its conductor, ran a frontier research program end to end: design, instruments, preregistration, experiments, cross-family audit, forensic audit of its own record, and the reports (Section 4.1).
2. **Evidence-gated recursion (EGR).** One loop for recursive self-improvement that compounds: components, designs, a running system across a model upgrade, and the research harness itself all improved under EGR, on work they had never seen (Sections 4.2 and 5.3).
3. **A law with predictive bite, found by the organization auditing itself.** The headroom law decomposes every comparison exactly, and the program measured its parameters: $q = 0$ for evidence-gated components, ϕ before the component is built, and a stable share k . A share fitted on Claude Haiku 4.5 and Claude Opus 4.8 alone predicts Claude Opus 5.5’s comparisons with Bayes factors of 186 against zero effect, 9.8 against a constant gain, and 5.6 against a gain capped at the headroom; the formal model of memory transfer decomposes the anchoring loss in agreement with a direct regression (−0.051 and −0.053 log speedup); and we register forecasts for the next model generation (Sections 3, 5.2, 5.11 and 6.8).
4. **New mathematics from invented principles.** 325 new best-known values with exact certificates, from escape moves the agents invented, including $M \leq 0.38085654787\dots$, 20.3 times the Einstein Arena’s minimum improvement below its record; and a boundary law: across a 73-fold range of N every new packing sits within a band 0.031 wide below the hexagonal estimate, and the band fitted on the 248 smaller sizes contains all 58 larger ones and forecasts their deficits to within 0.0060 on average (Section 5.4).
5. **Past the leading systems, across domains.** The program’s harnesses beat CORAL (914 vs. 948 cycles) and ACE ($3.67\times$ vs. $1.95\times$) on the same model, and set records or beat named rivals in nine domains with five model families (Sections 5.5, 5.6 and 5.10).
6. **Sharp limits that intuition missed.** Verified memory from Claude Opus 4.8 slows its stronger successor, and every item-selection method fails to fix it; supervision built from verdicts pushes a frontier model below its own level (Sections 5.6, 5.7 and 6.7).

The remainder of the paper is organized as follows. Section 2 reviews related work. Section 3 develops the headroom law. Section 4 describes the research organization and its platform, evidence-gated recursion, and the models, instruments, controls, evaluation, and statistics. Section 5 reports the results: the law’s fit, EGR at four levels, and six studies. Section 6 interprets the results, compares them with prior work, draws out their implications, and registers forecasts. Section 7 concludes.

2 Related Work

Harness search. ADAS and the Darwin Gödel Machine search over agent designs and code (Hu, Lu & Clune, 2024; Zhang et al., 2025a). MILO co-evolves harnesses and reports an Erdős bound of 0.3808567744 (Jana et al., 2026); HarnessTTT reaches 0.380868 (Ma et al., 2026); Hill Sampling reaches 0.38088232 (Beck, Ogren & Kobren, 2026). Each selects a whole harness by its end score. This work accounts for each component’s gain by the share of the executor’s failures whose cause the component supplies, and its bounds lie below all of theirs.

Recursive harnesses. Harness² recurses at two levels: a propose–probe–compose step refines the harness of the current task without ground-truth feedback, and a library of adopted edits carries lessons to later tasks (Xu et al., 2026). With GLM-5 and Gemini 3.5 Flash as executors and Claude Opus 4.8 as editor, one step gains up to 11.5 points on WorkBuddy-Bench, 4.5 on JobBench, and 11.8 on LAB, scored in whole or in part by language-model judges, and a library built from five-step refinement adds 0.9 to 5.1 points on WorkBuddy-Bench. Independently, it finds that refining a harness directly, without its probes and library, lowers the executor, from 73.4 to 63.2 on WorkBuddy-Bench’s code tasks; Section 5.7 names the general effect, iatrogenic regression, traces it to one mechanism, and removes it. This work is research that AI ran, and its method, EGR, recurses at four levels: components, harness designs, a running system across a model upgrade, and Nenya, the harness the research ran on. Every result is scored by an official scorer, a protected evaluator, or exact arithmetic (Section 6.3.1).

Evaluator-guided discovery. FunSearch, AlphaEvolve, OpenEvolve, ShinkaEvolve, ThetaEvolve, and TTT-Discover pair a model with an automated evaluator, adding evolutionary loops or weight updates at test time (Georgiev et al., 2025; Lange, Imajuku & Cetin, 2025; Novikov et al., 2025; Romera-Paredes et al., 2024; Sharma, 2025; Wang et al., 2025; Yuksekogonul et al., 2026). The Station runs agents from three model families for one to two weeks (Chung, Du & Wesley, 2026). This work uses no evolutionary outer loop, no learned harness, and no weight update: the agents invent their own searches, and on every instance these systems share with it, its values are better.

AI scientists. The AI Scientist and the AI co-scientist generate hypotheses and papers with language models (Gottweis et al., 2025; Lu et al., 2024). Here an organization of agents ran a full experimental program against named public baselines, with preregistration, positive controls, and cross-family audit, and its output includes a quantitative law and exact mathematical certificates.

Memory and playbooks. MemGPT, Mem0, Reflexion, Voyager, ExpeL, Agent Workflow Memory, A-Mem, Dynamic Cheatsheet, ReasoningBank, and ACE retain context, facts, skills, or lessons from earlier episodes (Chhikara et al., 2025; Ouyang et al., 2025b; Packer et al., 2023; Shinn et al., 2023; Suzgun et al., 2025; Wang et al., 2023, 2024; Xu et al., 2025; Zhang et al., 2025b; Zhao et al., 2023). Each values an item for the model whose work produced it. This work shows that carried experience reverses sign across a model upgrade and that only verified technique carries upward. Independently, RBC finds self-written playbooks falling up to 69.6 points below no playbook (Pham et al., 2026).

Multi-agent search. CORAL reports 1,103 take-home cycles with Claude Opus 4.6 (Qu et al., 2026); this work reaches 914 and beats a faithful port of CORAL on the same model. AReal-RSI reports 1.9–27.5% lower kernel latency with experience sharing (Zhou et al., 2026). This work is the first to measure, hour by hour, when sharing pays.

Judges and verifiers. LLM-as-a-judge, Agent-as-a-Judge, trained verifiers, and critics place a model’s assessment in the loop (Cobbe et al., 2021; Lightman et al., 2023; McAleese et al., 2024; Zheng et al., 2023; Zhuge et al., 2024), and models correct their own reasoning poorly without external feedback (Huang et al., 2023). This work replaces the judgment with exact comparison and beats the frontier judge with the same information.

3 Theoretical Framework: The Headroom Law

Every comparison in this paper holds the model and the instrument fixed and varies one component. This section turns that design into a quantitative account: what a component can recover, what it can destroy, and why the gains it delivers are proportional to the failures its control leaves.

3.1 Components, controls, and headroom

Definition 3.1 (Component, control, lift). A harness component L augments an unchanged model M on an instrument I , a benchmark or service scored by a fixed metric m_I . A control c runs M on I without L . It is the bare model in the

instrument’s standard harness, the bare model told of a crash, the fixed diligence instruction, ACE, or the CORAL port. The lift of L over c is $\Delta = m_I(L \circ M) - m_I(c)$.

Definition 3.2 (Headroom and recovered share). On a metric scored from 0 to 100, the headroom of control c is $h = 100 - m_I(c)$, the part of the scale the control leaves unclaimed. A comparison with $h > 0$ has recovered share $r = \Delta/h$.

Definition 3.3 (Instruction share). On an instrument where the bare model, the fixed diligence instruction, and a best arm all run, the instruction share is $(m_{\text{instr}} - m_{\text{bare}})/(m_{\text{best}} - m_{\text{bare}})$, the fraction of the best arm’s lift that the instruction delivers by itself.

A *no-harness control* adds no learned or carried content: the bare model, the bare model told of a crash, or the diligence instruction. ACE and the CORAL port have already recovered part of the headroom and are treated as rivals (Appendix A.3). Kernel speedup, take-home cycles, and construction bounds have no ceiling that a control can approach. On those instruments the reference is the best published value, and a gain is read as a margin over it or over ACE or the CORAL port on the same model.

3.2 The per-failure conversion equation

Proposition 3.4 (Per-failure conversion). *Let a control fail on a fraction $h/100$ of the units an instrument scores, and partition its failures by cause into classes j that hold fractions ϕ_j of them. Suppose a component converts a failure of class j into a success with probability p_j and turns a success into a failure with probability q . Then the expected lift in points is*

$$\mathbb{E}[\Delta] = \left(\sum_j \phi_j p_j \right) h - q (100 - h). \quad (2)$$

Proof. A unit the control fails lies in class j with probability $\phi_j h/100$ and is converted with probability p_j . A unit the control passes, with probability $(100 - h)/100$, is broken with probability q . Linearity of expectation over units, scaled to points, gives Equation (2). $\square$

With $q = 0$ the proposition is the *headroom law*, $\mathbb{E}[\Delta] = kh$ with $k = \sum_j \phi_j p_j$. The share k is the matched fraction of the control’s failures: the fraction whose cause the component supplies, weighted by how reliably it converts them. A component built for one cause that converts each such failure has $k = \phi$. The relation is linear in h because a component has nothing to act on where the control succeeds, and headroom bounds what any component can add. The second term is what a component breaks. At $h = 0$ it gives $\mathbb{E}[\Delta] = -100q$: where the control is at its ceiling, a component can only lose, and q is all that a comparison measures.

Proposition 3.4 is an exact accounting: by linearity of expectation it holds for every component and control, so no comparison can refute it. Its content lies in its three parameters, which the program measured. The breakage rate q is a design property: evidence-gated components held it at zero on 2,880 production tickets, and a verdict-based positive control, run for that purpose, did not, falling below bare on 360 of 579 (Section 5.7). The matched fraction ϕ can be measured before a component is built, wherever a control without the cause exists, and it bounds what the component can recover: the API-change sensor recovered 13.2 of the 13.6 points a change caused and none of the 5.7 it did not (Section 5.2). The share k is the empirical claim: it is stable enough across components and instruments that lifts scale with headroom rather than staying constant, which Section 5.2 tests against constant, capped, and ceiling-aware alternatives.

Four consequences organize the results.

- (i) *Where the model succeeds, the harness must hold $q = 0$.* This is a design requirement, and Section 5.7 builds harnesses that meet it by construction.
- (ii) *Where the cause lies inside the episode, an instruction suffices.* Telling the model to look converts the failures it can diagnose itself, and the instruction share is large.
- (iii) *Where the cause lies outside the episode, only a component that carries it converts.* Such causes include a successor’s missing state, a crashed process’s executed effects, and a silent change visible only across episodes.

- (iv) *Carried content that supplies no cause has $p \approx 0$, and if it displaces a better approach, $q > 0$.* Its share is negative (Section 5.6).

Section 5.2 tests the law on the program’s comparisons, and Section 5 reports each study in the terms of Equation (2).

4 Methods

This section describes the research organization that ran the program and the platform it ran on, the recursive improvement method it used, and the models, instruments, controls, evaluation, preregistration, statistics, and compute behind every result in Section 5.

4.1 Research organization and platform

The research in this paper was done by an organization of AI agents. A human operator set the mission and the bar: build agent systems that beat named public results on public benchmarks with official scoring, judged by what runs. From there the work belonged to the agents.

Nenya. The whole organization ran on Nenya, QWNTL’s harness for ultra-long-horizon agent work: the conductor, every team lead, and every seat. Nenya supplies durable infrastructure for runs that outlast sessions, processes, and models, and it executes agents with a minimal human in the loop and continual learning. It works from outside the model, externally controlling the agent’s hidden internal representation through a latent-space representation of the agent’s beliefs; the model’s weights are untouched.

The conductor. An agent named frontier-bond conducted the program. It owned the program’s thirteen outcomes, from capability inherited across models and survival across crashes, through a shared ledger that compounds search and a system that improves its own improvement, to verification capital that compounds. It held the shared infrastructure directly: the GPU pool, the model proxy every seat called through, and the services that kept capacity allocated across teams. It set the decision rules. A team that produces no decisive result by its deadline is folded into the outcome it serves, a dead bet is reported as dead, and no team waits on another team’s interface.

Teams and seats. Under frontier-bond, team leads owned focus areas: HORIZON (supervision of long-horizon software work), LONGRUN (continuous operation), DISCOVERY and KERNEL (shared search), APPWORLD and SUCCESSION (inheritance across models), ENGINE and RELIABILITY (recovery), EVOLVE and META (self-improvement), COMPANION, WEIGHTS, and TRAIN (trained components), DEEPPREAD and BIOMIMICRY (research feeding the bets), AUDIT-X (cross-family audit), FORENSICS, and four REOPEN teams. Each lead staffed mixed teams of Claude Opus 5.5 and GPT-6 Astra seats working in parallel, each seat in its own worktree; the program’s repository holds 292 of them. Every lead also ran contrarian seats, half deep reader of the newest literature and half reckless builder, charged to chase any idea they thought they could crack, against the plan and the lead’s priors, and to build and run it the same day. The contrarian seats produced some of the program’s largest results: the 306 packing records, the C_3^+ bound and the lowest M bound, the verified technique cards that beat ACE, and the metamorphic verification instruction.

How the organization did science. Seats designed and built the harnesses and the instruments, committed predictions and kill criteria before the runs they judged, ran the experiments, and handed results to their lead with the command, commit, and artifacts that produced them. The core claims were audited on the exact candidate by a seat of a second model family. Headline numbers were recomputed from raw artifacts by independent analysis code. Harm was measured against positive controls. Between October 7 and 9, 2026, the program’s gateway logged 3,230,847 Claude Opus 5.5 calls, and its record holds 195 comparisons.

The organization ran EGR on itself. frontier-bond put a forensics team on the program’s whole record and fresh leads on the conclusions it flagged (Section 5.3). The forensics found the headroom law, and the reopened work found the remedy for predecessor anchoring. The six companion reports were written by agent teams in which a numbers seat traced every number to a result file, a hostile reviewer attacked each paper, and a fresh reviewer that wrote none of it checked it; the corrections they found went back to frontier-bond.

From research back into the harness. After the experiments, frontier-bond assigned a lead to build its findings into the next release of Nenya, the harness the program ran on (Section 5.3).

Table 1: **Models, roles, and settings.**

Model	Identifier	Roles and settings
Claude Opus 5.5	claude-opus-5-5	The harnessed model throughout. Adaptive thinking; effort xhigh on the construction problems and the open-problem societies, medium on the take-home (Claude Code 2.1.289), high on HORIZON and as judge; writer of the check library, the store’s recipes, and the insight compiler.
Claude Opus 4.8	claude-opus-4-8	Worker of the continuous-operation system until hour 36; second KernelBench reader; strong successor on HORIZON.
Claude Haiku 4.5	claude-haiku-4-5	Weaker actor and successor; failure distiller in the society; panel model for agreement.
GPT-6 Astra	gpt-6-astra	Same-strength successor (take-home switch, HORIZON); actor in the AppWorld record.
Qwen3.5-27B	Qwen/Qwen3.5-27B	Weaker successor on HORIZON (bf16, served by vLLM, with thinking and effort settings dropped); every seat of the open-weight take-home run.
Qwen 27B (capital)	logged as qwen38-27b	Open-weight actor in the AppWorld check-library record.

4.2 Evidence-gated recursion

Algorithm 1 states one step of the loop.

Algorithm 1: **Evidence-gated recursion (EGR)**. One step of the loop at any level. At the component level the proposer writes checks, technique cards, tools, or claims; at the design level it writes harnesses; at the system level it repairs a running system’s memory; at the program level it rewrites the research plan and the harness the research runs on.

Inputs: a harness H ; a record $\mathcal{R}$ of past attempts with their evidence (official outcomes, evaluator scores, the agent’s own observations); a proposer model P ; an evidence gate G ; a library $\mathcal{L}$ of retained components.

- 1: TARGET: classify the failures in $\mathcal{R}$ by cause and select causes that lie outside the model’s episode, where the headroom law (Equation (2)) predicts a positive share $\phi_j p_j$.
- 2: PROPOSE: P writes a candidate c from the record: an executable check, a technique card, a tool, a claim about the world, a harness design, or a repair to a running system. P never sees the answer key.
- 3: GATE: G admits c only on hard evidence. On recorded failures c must convert at least what the component it replaces converts; on recorded successes it must break nothing; an unknown outcome admits nothing.
- 4: RETAIN: add c to $\mathcal{L}$ with its provenance and evidence, replacing its predecessor.
- 5: REUSE: serve $\mathcal{L}$ to later tasks, other agents, and successor models. Serve technique and evidence freely; serve experience written by a weaker model only after the reader holds a checked result (Corollary E.4).
- 6: RECURSE: record the outcomes of work that used $\mathcal{L}$ in $\mathcal{R}$ and return to step 1. Run the same loop one level up, with harness designs as candidates and a preregistered measurement against named rivals, with a positive control, as the gate.

EGR has one rule that ungated recursion lacks: nothing enters the harness on a conclusion. The gate is what lets EGR run without a person. By the accrual proposition (Proposition J.5), an admitted revision never loses coverage the record holds and never fires on a recorded success, so on the record each step can only raise the matched share k while holding $q = 0$. The headroom law then says where each step will pay: on causes that lie outside the model’s episode, where no instruction reaches.

4.3 Models and settings

All experiments ran October 7–9, 2026 (Table 1). Claude and GPT models were served through provider APIs. Between 02:04 UTC on October 7 and 12:35 UTC on October 9, the program’s API gateway logged 3,230,847 Claude Opus 5.5 calls.

4.4 Instruments

Table 2 lists the instruments and their metrics. HORIZON was built by the program; the AppWorld streams add crashes, API changes, and model swaps to the official tasks and scorer.

Table 2: **Instruments and scoring.**

Instrument	What it measures
HORIZON’s account-balance service	An agent builds and runs the service through 120 dependent tickets while scripted customer traffic writes balances; after each ticket, S_{prod} is the fraction of production accounts whose balance matches a hidden oracle. Fresh-state S scores hidden oracle-generated cases on a fresh deployment. In handover runs a successor takes over at ticket 40. Six streams, hashed before any arm ran.
AppWorld (Trivedi et al., 2024)	Multi-app API tasks, official scorer: task goal completion (TGC) and scenario goal completion (SGC; a scenario passes only if all its tasks pass). Streams with crashes, documented and undocumented API changes, and model swaps; a 31,434-attempt graded record for the check library.
τ^2 -bench (Barres et al., 2025)	Customer-service dialogues in the retail and airline domains; a held-out split of 60 tasks (40 retail, 20 airline) for done-checking; end state is the hash of the domain database after a strict replay.
KernelBench (Ouyang et al., 2025a)	GPU kernels for 130 of 250 problems: capped speedup over PyTorch eager if correct, else 0, at most $10\times$, on a 40 GB A100 with strict fp32 and a protected evaluator.
Performance take-home (Anthropic, 2026)	Simulated machine cycles of an optimized kernel program for a VLIW SIMD machine (lower is better); 8 hours, one machine per arm.
Einstein Arena (Bianchi et al., 2026; EinsteinArena, 2026)	Erdős’s minimum overlap problem and the autocorrelation inequalities (upper bounds), with public verifiers and minimum improvements of 10^{-7} (M), 10^{-8} (C_1), and 10^{-5} (C_3^+).
Packomania (Specht, 2026)	Unequal circles in the unit square maximizing the sum of radii, $N = 137$ to 10,000.
AHC026 (AtCoder, 2023)	AtCoder Heuristic Contest 026, ‘Stack of Boxes’; development score on 150 generated cases.
ALE-Bench (Imajuku et al., 2025)	Heuristic contests, official scorer; used for the instruction share.

4.5 Study designs

Each study compares harnesses on the same model and instrument and changes one component at a time. This subsection states each design; Appendices B to H give the full descriptions.

Evolution of an agent engine. A proposer edits the incumbent engine, a fast screen ranks the candidates, and an admission rule accepts an edit only when it raises the protected Claude Opus 5.5 score on a frozen pool of optimization problems; an archive keeps every admitted engine as a parent. The loop ran 12 generations per arm from a hand-built engine. Four evolved engines were carried, frozen, to seven ALE-Bench problem families that no part of the evolution had seen and scored by the official scorer on Claude Haiku 4.5 and Claude Opus 5.5, against ACE, the diligence instruction, and the bare model.

Mathematical constructions. The problems are upper bounds on Erdős’s minimum overlap constant M and on the first and signed autocorrelation constants C_1 and C_3^+ , Packomania’s sum-of-radii table for unequal circles in a square at $N = 137$ to 10,000, the 14-point distance ratio in $\mathbb{R}^3$, and container packings (Appendix C.1). Three Claude Opus 5.5 agents per problem worked on a shared archive of verified attempts, and a single long-running agent worked alone, each with a hash-pinned exact evaluator and the public record constructions as starting material. A preregistered matched-hours study added a compute-only control of 94 worker processes that apply the classical toolbox to the same constructions with no language model; a value counts as the agents’ only if it beats both the control and the single-agent arm at equal elapsed time. Every value is an exact rational certificate (Section 4.7).

Shared verified archives. Four harnesses ran Claude Opus 5.5 with matched machines, CPU share, hours, effort, starting material, and evaluator: a single long-running agent, isolated agents, a faithful port of CORAL (CORAL-faithful), and agents on a shared archive. The archive is an append-only, HMAC-chained log of verified attempts that stores each attempt’s source and the tool that produced it, and parents are drawn by clade credit. On Anthropic’s performance take-home, four agents per arm and three seeds per arm ran for 8 hours on one n2d-highcpu-48 machine per arm through Claude Code at medium effort. On the C_1 , Erdős, and AtCoder Heuristic Contest 026 problems, the society ran three agents per problem at effort xhigh with web tools disabled, one n2d-highcpu-96 machine per arm. Advantage, crossover time (read at the least favorable seed pairing), and cross-agent reuse (from data-flow provenance)

are defined in Appendix D.1.

Verified technique and predecessor anchoring. KernelBench episodes, on 130 of its 250 problems, are scored by capped speedup s , the eager time over the kernel time capped at 10 when the protected evaluator accepts the kernel and 0 otherwise, and by log speedup $\log(1 + s)$. A technique card states a technique, the arithmetic that makes it work, and the condition under which it applies, and the protected evaluator checks it before admission; the library is headed by the error-compensated bf16 split (Henry, Tang & Heinecke, 2019; Ootomo & Yokota, 2022) and five engine-admitted cards. The preregistered technique test ran 26 problems outside the trials used to build the cards, three repetitions per arm and problem, 720 episodes. The anchoring studies served the continuous-operation system’s hour-24 store, 71 items from 18 hours of Claude Opus 4.8 kernel work, in a randomized test of 1,288 episodes, a confirmation test on 28 further problems, and a preregistered head-to-head of six serving methods on 10 held-back problems.

Supervision. On HORIZON an agent builds an account-balance service through 120 dependent tickets while scripted customer traffic writes balances; S_{prod} is the fraction of production accounts whose balance matches a hidden oracle after each ticket (Appendix F.1). A verdict harness, assembled from a requirements list, a model judge, compiled tests, and rollback, is compared with three evidence-only harnesses (commit, guard, full) built on five invariants: a bare-identical session, evidence returned for repair, unknown outcomes that trigger nothing, committed deployed state, and model-written checks kept off the critical path (Appendix F.3). Six fresh streams were hashed before any arm ran; the verdict harness was rerun unchanged as a preregistered positive control; in handover runs Qwen3.5-27B takes over at ticket 40, with preregistered endpoints.

Durable state and evidence. An AppWorld crash kills the agent process after a step has taken effect and before the agent records it. The recovery harness keeps a record of committed steps, a write-ahead intent for the step in flight, a checkpoint of the Python session, and receipts from the tool server’s journal. It was tested in a deterministic engine test without a model and on preregistered streams with 52 scheduled crashes and API changes. The label-free claim learner admits a claim about an API only once the agent’s own observations confirm it; the metamorphic instruction is a fixed 533-word text evaluated on drift classes sealed before it was frozen.

Verification capital. Verifiers are scored by catch, false fire, and Youden’s J (Youden, 1950) without reading the official outcome. End-state agreement compares an attempt’s end-state fingerprint with those of $k = 4$ other attempts at the same task, against Claude Opus 5.5 judging zero-shot and judging with the same four attempts, on a held-out τ^2 -bench split of 60 tasks. The check library replays 31,434 graded AppWorld attempts in time order over 38 windows; in each window it writes verdicts before reading outcomes, and its gate admits a check only if it stays silent on the template’s past successes and catches at least what the check it replaces caught (Definition H.2). Copies frozen at fixed points measure depreciation.

4.6 Controls and rivals

Every arm runs on the same model as its control. The bare model uses each instrument’s standard harness. The diligence instruction asks the model to check before acting and does not learn; on AppWorld it is four sentences beginning ‘be thorough and verify against the live app state’, and on HORIZON a fixed paragraph that also asks the model to commit (Appendix L.1). ACE runs as its authors specify it, with a generator, a reflector, and a curator (Zhang et al., 2025b). On KernelBench it ran on the same stream to hour 36, with Claude Opus 4.8 as generator and Claude Opus 5.5 as reflector and curator, and its arm serves its own hour-36 playbook. CORAL-faithful ports upstream CORAL commit cce3101, retaining its prompts and heartbeats (reflect, consolidate every 10, pivot after 5), its shared attempts, notes, and skills, and its hidden grader. An early exit resumes the same conversation with the latest score and ‘continue’. The shared run server sorts attempts by the task objective, a fix applied to every arm. On HORIZON, handover runs fork from their own parent after ticket 39, byte-identical on tickets 1–39.

4.7 Evaluation

Protected kernel evaluation. The agent produces text and has no access to the execution sandbox. A static gate requires a custom kernel and rejects monkeypatching, streams, graphs, and fallbacks. A trusted process that never sees candidate code generates reference outputs on fresh random seeds and weights. Timing interleaves reference and candidate trials on fresh inputs with a cold cache and rechecks the last output in each block. Both processes run strict fp32 at $\text{atol} = \text{rtol} = 10^{-4}$ with TF32 disabled, KernelBench’s own fp32 setting. Before any arm ran, the evaluator

passed 24 qualification controls, accepting two legitimate kernels and rejecting 22 attacks. Each scored result carries the evaluator’s fingerprint.

Take-home evaluation. Candidates build in a sandbox without network access and emit their instruction list as data. A separate process that never imports candidate code runs a hash-pinned simulator on fresh random problems, and the evaluator passed 9 adversarial qualification cases. Final kernels were checked against the official suite in a fresh clone, 64 fresh random problems, and stress tests of 500 random, 7 edge, and 100 full-32-bit cases.

Exact verification of constructions. The analytic scorer reads candidates as data and returns fast scores never better than the exact score. Evaluators were written before the searches, tested on adversarial cases (malformed numbers, tampering, tolerance-inflated radii, overflow), and reproduced each public record they were tested on. An evaluator refuses to run if its bytes change, and its final mode parses each decimal as an exact rational; violations cannot raise a score. Each claimed value was computed by the hash-pinned evaluator and by a second exact implementation that shares no code with it or with the optimizer, then recomputed by an independent exact checker written from the problem definitions. Three exact implementations agree on the fractions for M and C_1 , and the Einstein Arena’s published verifier, run unmodified, agrees to 16 significant digits.

Prior-art search. ‘Best known’ means strictly better in exact arithmetic than each published value. The literature was searched on October 7, the Einstein Arena and packing sources were read on October 9, and the public searches were repeated on October 13 and 14, 2026. The packing comparison takes the maximum over Packomania’s table and coordinate files, Liang’s full repository history, and six other repositories: 766 commits and 339,696 published values. Identity tests used raw coordinates, canonical forms under the eight symmetries of the square, and comparisons within 10^{-8} .

HORIZON validation and scoring. Streams were generated, validated, and hashed before any arm ran; the oracle as implementation scores 1, a persisting reference scores $S_{\text{prod}} = 1$, and an amnesic reference scores 0; the hidden scorer kills 88 of 89 targeted mutants. The scorer (version 0.3) and oracle (version 0.5) were fixed for every run. Audits found no harness reading the hidden scorer, oracle, or traffic files.

4.8 Preregistration, recomputation, and audit

For preregistered tests, predictions, kill criteria, and analysis code were committed before the runs they judged; held-out splits were frozen and hashed, and scorers were not edited during harness development. Preregistered results include the HORIZON no-harm panel, the positive control’s forecast, and the handover endpoints; the take-home margins; the open-problem prediction and the attribution rule; the KernelBench randomized, confirmation, head-to-head, and technique-library designs; the AppWorld crash streams; the agreement contrasts; and the check library’s depreciation and anticipation contrasts. Headline statistics were recomputed from raw artifacts with independent analysis code: the 5,040 scores of the HORIZON panel, every primary KernelBench number including all 720 rows of the technique-library test, the agreement contrasts, and the construction values. An offline review of infrastructure, harness, and analysis faults corrected values where marked. Table 38 lists each headline result with its verification.

4.9 Statistical analysis

Intervals are individual 95% intervals unless marked. HORIZON intervals are t -intervals over six streams, wider than the preregistered percentile bootstrap, with exact sign-flip permutation tests; with six streams the smallest attainable p is $2/64 = 0.031$. AppWorld streams use exact McNemar tests on discordant tasks and sign tests on scenarios; the metamorphic instruction uses a paired bootstrap; receipts use a Wilson interval. KernelBench store tests use a problem bootstrap with 10,000 resamples; technique-library and store-age intervals are 90% task bootstraps paired by problem, and up/down counts use $|d| > 0.05$. The check library uses a template-cluster bootstrap with 2,000 draws (Efron, 1979), and τ^2 -bench a domain-stratified task-cluster bootstrap with 20,000 draws. Take-home Mann–Whitney tests are exact and one-sided. The headroom fit uses maximum likelihood under Equation (3) with profile-likelihood intervals. Alternative models are compared by AIC, BIC, and leave-one-out predictive log score, and cross-model predictions by Bayes factors of plug-in predictive densities (Appendix A.1). Mechanism regressions use problem fixed effects with 2,000 bootstrap draws, and strength terciles are cross-fitted over 200 random splits.

Table 3: **What the AI-run program’s harnesses achieved, against the strongest named result on each instrument.** Constructions are certified in exact arithmetic and compared with every published value as of October 14, 2026; agent comparisons run the rival on the same model, Claude Opus 5.5, unless noted.

Instrument	Strongest named prior or rival	This work
Erdős minimum overlap, $M \downarrow$	Einstein Arena record 0.3808585749; MILO 0.3808567744	0.38085654787...
First autocorrelation, $C_1 \downarrow$	Einstein Arena record 1.5027434867; MILO 1.5027435984	1.50274043592...
Signed autocorrelation, $C_3^+ \downarrow$	Einstein Arena record 1.4491097893; MILO 1.4488860143	1.44886703134...
Sum of radii, 306 sizes	Liang’s automated campaign, best at every size	higher at all 306 , by up to 5.1×10^{-2}
$N = 137\text{--}10,000 \uparrow$		
14 points in $\mathbb{R}^3$, $(d_{\max}/d_{\min})^2 \downarrow$	FICO Xpress 4.165783988278149	4.16578347458...
Performance take-home, cycles $\downarrow$	CORAL, same model: 948; CORAL as published (Opus 4.6): 1,103	914
KernelBench, mean capped speedup $\uparrow$	ACE, same model: $1.95\times$	$3.67\times$
ALE-Bench, seven never-used families, Claude Haiku 4.5 $\uparrow$	ACE: 1,709	1,894 , an automatically evolved engine
τ^2 -bench done-checking, catch $\uparrow$	Claude Opus 5.5 judge, same evidence: 30.9%; zero-shot: 8.8%	48.5% , no label, no model call
AppWorld failure detection, catch at matched false fire $\uparrow$	ACE-style verification playbook: 55.5%	90.1% , one twentieth of the tokens
AppWorld, 52 crashes and an API change, scenario completion $\uparrow$	ACE: 79.9	92.1
AppWorld silent API change, task success $\uparrow$	label-free ACE: 94.8 with 1,091 auxiliary calls	95.2 with 292
HORIZON handover to Qwen3.5-27B, production correctness $\uparrow$	bare handover: 0.435	0.911

4.10 Compute

The agents on the shared archive used at most 520 and 191 vCPU-hours of their own processes for the Erdős and C_1 bounds, upper bounds set by per-agent CPU caps of 16 vCPU on one 96-vCPU machine shared by four problems and twelve agents. The packing search used about 10,000 CPU core-hours for the first 193 sizes, with a peak of 50 machines, and about 13,000 for the remaining 113, about 23,000 in total. The C_3^+ search ran on 10–18 A100 GPUs for 7.9 hours. Each take-home arm had one n2d-highcpu-48 machine; each open-problem arm one n2d-highcpu-96 machine. The KernelBench technique-library test made 1,602 model calls, with 79.6 million input and 6.6 million output tokens, for its 720 episodes. The check library’s pass/fail run made 261 Claude Opus 5.5 calls across two arms, with 4.74M input and 0.99M output tokens.

5 Results

We first summarize the program’s results against the strongest named prior result on each instrument (Section 5.1). We then test the headroom law on the program’s comparisons (Section 5.2) and report evidence-gated recursion at each of its four levels (Section 5.3). Six studies follow: mathematical constructions (Section 5.4), shared verified archives (Section 5.5), verified technique and predecessor anchoring (Section 5.6), supervision (Section 5.7), durable state and evidence (Section 5.8), and verification capital (Section 5.9). The section closes with the designs’ generality, cost, and speed (Section 5.10) and the tests of predictions (Section 5.11). Interpretation is deferred to Section 6.

5.1 Overview

Table 3 lists the strongest result on each instrument against the strongest named prior result or rival. Every rival comparison runs on the same model as the program’s harness unless noted, and every construction is certified in exact arithmetic.

Table 4: **The 15 Claude Opus 5.5 comparisons against a no-harness control**, grouped by what the component supplies. Scores and lifts are in points on a 0–100 scale; h is the control’s headroom, s the comparison’s standard error, and $r = \Delta/h$ the recovered share. The fit of Equation (3) to these rows gives $k = 0.54$.

Row	Component against control; instrument	Control	h	Δ (s)	r
<i>Persistent state</i>					
L003	Recovery harness against bare told of the crash; AppWorld, one crash per task (87 tasks)	97.7	2.3	0.00 (2.35)	0.00
L004	Recovery harness against bare; same tasks	92.0	8.0	5.75 (2.64)	0.72
L163	Recovery harness and API-change sensor against bare, TGC; 52 crashes and an undocumented change (417 tasks)	89.9	10.1	4.30 (1.35)	0.43
L174	Recovery harness against fresh restart, TGC; 52 crashes, a documented change, Claude Opus 4.8 to Claude Opus 5.5 (417 tasks)	84.9	15.1	6.70 (1.71)	0.44
L057	Handoff brief against a fresh Claude Opus 5.5; Claude Haiku 4.5 hands a task to Claude Opus 5.5 (66 tasks)	74.2	25.8	15.20 (6.59)	0.59
<i>Cross-episode evidence</i>					
L099	API-change sensor against bare; undocumented change (120 positions)	89.0	11.0	10.80 (4.46)	0.98
L166	API-change sensor against bare; Amazon amounts changed to cents (212 tasks)	80.7	19.3	13.20 (2.78)	0.68
L147	Claim learner, per-API version, against the instruction; silent pagination change	86.9	13.1	9.30 (3.62)	0.71
L124	Claim learner against the instruction; silent change, second stream	83.3	16.7	11.90 (2.73)	0.71
L126	Claim learner against the instruction; silent change, pooled (173 tasks)	84.4	15.6	10.80 (2.19)	0.69
L136	Claim learner, later version, against the instruction; fresh change streams (224 tasks)	90.3	9.7	2.68 (1.18)	0.28
<i>Experience carried from Claude Opus 4.8</i>					
L010	Store against bare; hourly AppWorld probes, continuous operation	100.0	0.0	−1.10 (0.84)	–
L011	Store against bare; hourly code probes	90.0	10.0	0.00 (10.61)	0.00
L012	Store against bare; hourly τ^2 -bench probes	89.0	11.0	1.40 (4.03)	0.13
L016	Store against bare; fresh AppWorld tasks (60)	95.0	5.0	−1.67 (4.28)	−0.33

5.2 The headroom law across the program’s comparisons

The program’s record holds 195 comparisons. Fifteen are Claude Opus 5.5 comparisons against a no-harness control on a metric scored from 0 to 100 with a standard error (Table 4). They follow from six criteria applied to all 195 rows, none of which looks at the outcome (Table 16): a component compared with a control (160 remain); a control that can attempt the task and a result not traced to a fixed harness bug (156); a no-harness control rather than a rival system (90); a bounded metric with a recorded control score, so that headroom is defined (30); a recorded standard error (21); and Claude Opus 5.5 as the harnessed model (15). The two rows excluded only for a missing standard error are zero-lift comparisons at zero headroom; adding them with any standard error from 0.5 to 2 points leaves k at 0.52–0.53. The program’s audit wrote the criteria after the experiments (Section 5.3). Comparison i is modeled as

$$\Delta_i \sim \mathcal{N}(k h_i, s_i^2 + \tau^2), \quad (3)$$

where s_i is its standard error and τ the spread between comparisons beyond it. Maximum likelihood gives $k = 0.539$, with profile-likelihood 95% interval 0.427–0.675, re-optimizing τ at each k . A component recovers 0.54 of the headroom its control leaves (Figure 2a).

Gains that ignore where the control stands fit worse (Table 17). A zero-lift model has AIC 107.1, and a constant lift of 5.71 points has AIC 96.8 and leaves a spread of $\tau = 4.16$ points between comparisons; the headroom model has AIC 77.2 and $\tau = 0.98$. Ceiling-aware alternatives fit worse too: a constant gain capped at the headroom, $\min(c, h)$, has AIC 89.4, and a smooth cap, $c(1 - e^{-h/c})$, has 83.2. Leave-one-out predictive log scores order the models the same way: headroom −38.6, smooth cap −42.1, capped constant −45.4, constant −48.3, and zero −53.2. One alternative fits as well: a constant effect on the log-odds scale, which also vanishes at the ceiling, has AIC 77.9 and log score −39.1. Over the headroom of these comparisons, 2.3 to 25.8 points, a constant share of failures and a constant odds ratio make the same prediction to first order. Dropping any group of rows leaves k between 0.50 and 0.57 (Table 39).

What separates the law from a generic ceiling effect is the matched fraction. Where the cause of a control’s failures can be measured, a component recovers the share with that cause and none of the rest. An undocumented change scaled AppWorld’s Amazon amounts to cents: bare Claude Opus 5.5 fell from 94.3 with no change to 80.7, so the change caused 13.6 of its 19.3 points of headroom ($\phi = 0.70$). The API-change sensor recovered 13.2 of those 13.6 points and none of the 5.7 that Claude Opus 5.5 loses with no change, and when the sensor’s notes diagnosed the change correctly, Claude Opus 5.5 scored exactly its no-change score, with 11 wins and 0 losses on 151 tasks ($p = 0.001$). A share

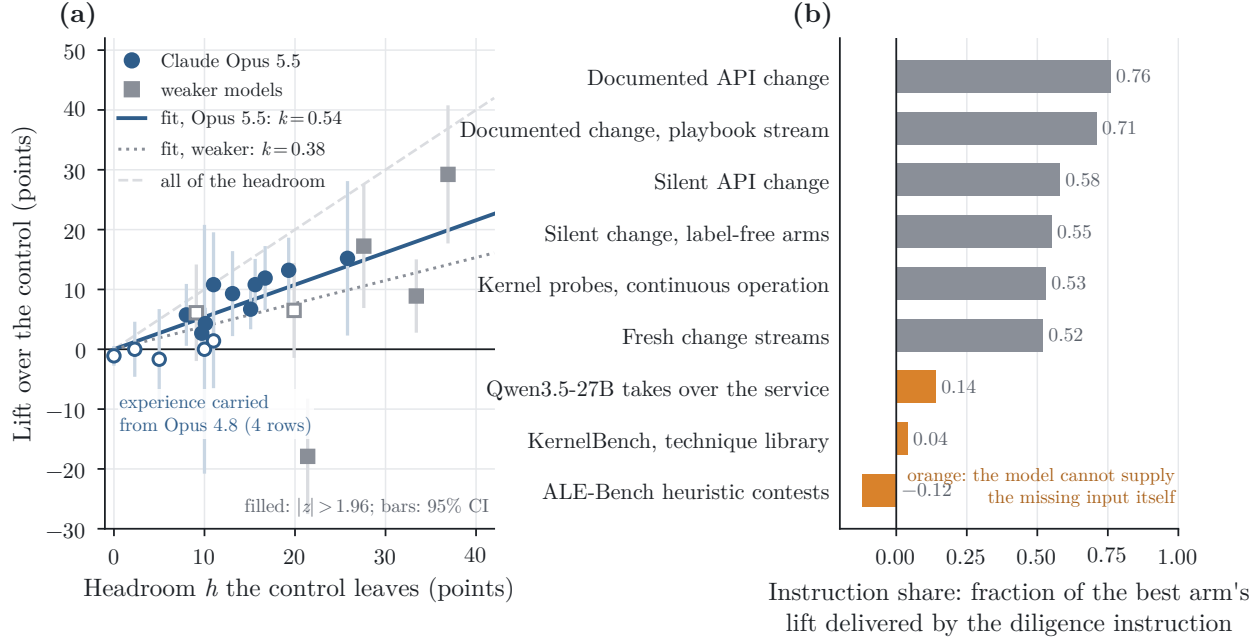


Figure 2: **Lift is proportional to headroom, and the diligence instruction delivers most of it where Claude Opus 5.5 can find the cause.** (a) Each point is one comparison of a component against a no-harness control, on a 0–100 scale: the 15 Claude Opus 5.5 rows of Table 4 and six rows on Claude Haiku 4.5 and Claude Opus 4.8. Lines are the fitted shares, $k = 0.54$ for Claude Opus 5.5 and $k = 0.38$ for the weaker models; the dashed line recovers all of the headroom. Filled markers: $|z| > 1.96$; bars: 95% intervals. The four Claude Opus 5.5 rows that serve Claude Opus 4.8’s store sit at or below zero. (b) Instruction share (Definition 3.3) by instrument. Grey: the model can find the cause in the episode once told to look. Orange: a successor’s missing state, a missing kernel technique, or ALE-Bench heuristic contests, where the model cannot supply the missing input itself.

fitted across components, or a constant odds ratio, would have predicted about 10 points of lift on this instrument; the matched fraction predicts the no-change score, which the sensor reached (Appendix A.2).

The recovery harnesses recovered 0.43–0.72 of the headroom wherever the crash cost the control anything, and the handoff brief recovered 0.59. The API-change sensor and the claim learners recovered 0.68–0.98 on single changes and 0.28 on fresh streams of mixed changes. The store built from Claude Opus 4.8 recovered between -0.33 and 0.13 , and on the one instrument where its control stood at the ceiling, its lift was -1.10 .

One share across model classes. The six comparisons on Claude Haiku 4.5 and Claude Opus 4.8 against a no-harness control fit a share of $k = 0.38$. Fitted on those six rows alone, that share predicts the 15 Claude Opus 5.5 rows better than every gain that ignores headroom: with each model’s own between-comparison spread, the Bayes factor is 186 against zero effect, 9.8 against a constant gain, and 5.6 against a constant gain capped at the headroom, and 1.3 against a constant log-odds effect, which also scales with headroom (Table 18 gives two other ways of setting the spread). All 21 rows together fit $k = 0.45$. The lifts on Claude Opus 5.5 are small where its headroom is small, and the share it recovers is of the same order as on the weaker models. On a handover to Qwen3.5-27B, an API whose meaning changed without notice, or a kernel technique the model does not use, the headroom is set by the missing input.

Where the cause lies. The instruction share (Definition 3.3) separates causes the model can find from causes it cannot (Figure 2b). The diligence instruction delivered 0.71 to 0.76 of the best arm’s lift after a documented API change, where the model finds the cause in the episode once told to look; 0.14 when Qwen3.5-27B took a service over, where the cause is state the successor never held; and 0.04 on KernelBench, where the missing input is a technique (Appendix A.3).

Table 5: **EGR at four levels, each measured on work it had never seen.** In every row a proposer wrote the candidate during the program, a gate admitted it on evidence, and the result is measured on held-out problems, later windows, never-used problem families, or a new model.

Level	What was rewritten	Gate	Result on unseen work
Component	executable checks (AppWorld)	silent on past successes; catches at least what it replaces	catch of never-seen failure states 7.3% $\rightarrow$ 95.6%; frozen copies 10.6–76.5 points behind
Component	technique cards (KernelBench)	the protected evaluator verifies the card	+1.36 [0.79, 2.10] on the 12 problems the seed card cannot reach, 11 up and 0 down
Component	search tools (four problem families)	an attempt citing the tool verifies	38 of 98 new bests ran a teammate’s tool; both certified bounds ran another agent’s code
Design	an agent engine, evolved for 12 generations	protected admission on Claude Opus 5.5	on seven never-used ALE-Bench families, every evolved engine above ACE on Claude Haiku 4.5 (1,779–1,894 vs. 1,709)
Design	supervision harness (HORIZON)	preregistered no-harm panel with a positive control	from below bare on every stream to 0 of 2,880 tickets below bare; handover +0.476
Design	search organization (open problems)	sealed run against its own CORAL and single-agent arms	largest two-hour C_1 gain of any arm; ahead on all three problems from 8 hours
System	a running system’s memory, across the Claude Opus 4.8 $\rightarrow$ Claude Opus 5.5 upgrade	preregistered offline test, then live deployment	store from harmful (−0.14 below bare) to 3.67 $\times$; both readers above bare by hour 42
Program	Nenya, the harness the research ran on	reviewed hand-ins, product checks on a live cloud lane	the program’s findings built into Nenya 0.11 alpha.21

5.3 Evidence-gated recursion at four levels

EGR (Section 4.2) ran at four levels, and at each level the improvement was measured on work the improved component had never seen: held-out problems, later windows, never-used problem families, or a new model (Table 5). Appendix B gives the full account of each level.

Level 1: components. Checks that Claude Opus 5.5 wrote and revised after every window under an evidence gate raised catch of never-seen failure states from 7.3% in the first four windows to 95.6%; the stock stopped growing at 48 checks in window 8, and every later gain came from revision. Technique cards admitted by an engine that tested hypotheses against the protected evaluator carried the seed card’s split to convolution, transposed convolution, and attention, and lifted Claude Opus 5.5 by +1.36 [0.79, 2.10] on the 12 problems the seed card cannot reach, up on 11 and down on none. On the shared archive, 38 of the society’s first 98 new bests ran or adapted a teammate’s tool.

Level 2: designs. An automated loop evolved a whole agent engine for 12 generations under protected admission on Claude Opus 5.5. Carried frozen to seven ALE-Bench problem families that no part of the evolution had seen, every evolved engine scored above ACE on Claude Haiku 4.5, 1,779 to 1,894 against 1,709, and the best stood 592 points above bare; on Claude Opus 5.5 the best scored 3,019, level with ACE’s 3,034 and 356 above bare. The supervision harness, rebuilt from its own 24 traced failures and admitted only after a preregistered no-harm panel with a positive control, went from below bare on every stream to 0 of 2,880 tickets below bare (Section 5.7). The search organization, redesigned from measured losses, produced the lead-thread society (Appendix D.6).

Level 3: a running system across a model upgrade. In the continuous-operation system, Claude Opus 4.8 was the worker until hour 36 and Claude Opus 5.5 after it, and every frozen snapshot of the store from hour 8 to hour 36 lowered Claude Opus 5.5’s mean capped speedup, by 0.16 to 0.54. EGR diagnosed the cause by randomized exposure (Section 5.6), its gate kept out a first repair that failed the no-harm test, and a verified technique library took the hour-36 store from 1.73 $\times$ to 3.67 $\times$ in a preregistered test. Deployed into the live system at hour 36.16, the library lifted it +0.70 above bare on its probe problems, and by hour 42 the store lifted both readers above bare, Claude Opus 5.5 by +0.76 and Claude Opus 4.8 by +0.90.

Level 4: the harness the research ran on. frontier-bond assigned a lead to build the program’s findings into Nenya 0.11 alpha.21: action only on hard evidence, clean restarts, model-aware delivery of carried knowledge, a library that grants verified capability, the metamorphic verification instruction, disagreement between attempts taken as evidence, and a shared ledger of verified work. Each hand-in was reviewed by an agent that did not build it and checked on a live cloud lane.

The research record. A forensics team attributed every result in the program’s record to its causes and flagged the conclusions most likely to move. The relation between lift and headroom tested in Section 5.2 was first identified in

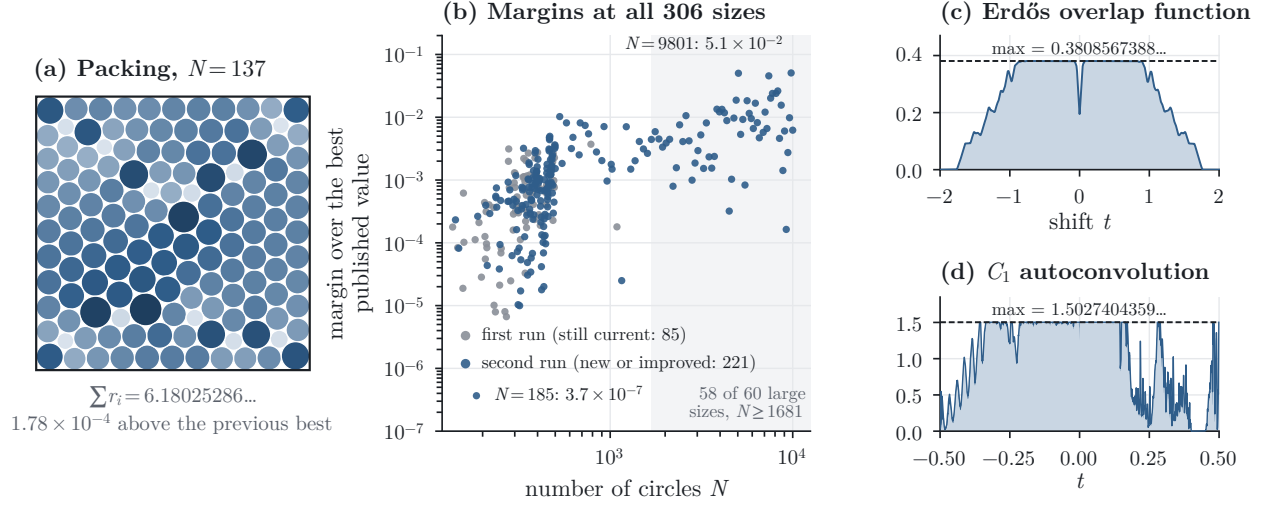


Figure 3: **Above the previous best at all 306 packing sizes, below the Einstein Arena entries on M and C_1 .** (a) The new packing of $N = 137$ unequal circles in the unit square, shaded by radius. (b) Margins over the best published sums of radii at all 306 sizes, $N = 137$ to 10,000; the shaded band marks the large sizes from $N = 1681$, where the defect moves of Appendix C.4 produced 58 new values. (c, d) The overlap and autoconvolution functions whose maxima give the bounds on M and C_1 ; both are flat at the maximum across a broad band of shifts. Curves use floating point; all bounds and packing margins are certified in exact arithmetic.

that audit, and the reopened anchoring work found the late-delivery remedy.

5.4 Mathematical constructions beyond the published frontier

Claude Opus 5.5 agents, given hours of autonomous work, a machine pool, and a hash-pinned exact evaluator, set new best-known values on 325 instances of open extremal problems: upper bounds on Erdős’s minimum overlap constant M , the first autocorrelation constant C_1 , and the signed autocorrelation constant C_3^+ ; Packomania’s sum-of-radii table for unequal circles in a square at all 306 sizes from $N = 137$ to 10,000; the 14-point distance ratio in $\mathbb{R}^3$; and 15 container packings. Each value is an exact certificate that anyone can recheck without trusting the search that produced it.

Figure 3 shows the packings and the two certified functions, and Table 6 sets the values against every published value they share an instance with. Three agents per problem on a shared archive of verified attempts (Section 4.5) produced the certified bounds $M \leq 0.3808567388\dots$, from a 3,584-cell construction after 10.82 working hours, and $C_1 \leq 1.50274043592\dots$, from a 262,144-cell construction after 3.97; their exact certificates are in Appendix C.2. The Erdős bound clears Einstein Arena entry #2507 by 18.4 times the minimum improvement, and the C_1 bound sits 305 times the minimum improvement below the Arena’s leader of October 13. A single agent then went lower on two constants by re-gridding onto resolutions commensurate with neither parent: $M \leq 0.38085654787558\dots$ on 2,560 cells, 2.03×10^{-6} below the Arena record (20.3 times its minimum improvement) and 2.265×10^{-7} below MILO’s value, and $C_3^+ \leq 1.44886703134\dots$ on 97,000 cells, below all 43 Arena entries and each value published by MILO, AlphaEvolve, Together AI, Matolcsi and Vinuesa, and EvoX. At each of the 306 packing sizes, $N = 137$ to 10,000, the new packings exceed the previous best, Liang’s automated campaign, by 3.67×10^{-7} to 5.08×10^{-2} , with median 8.78×10^{-4} ; one agent found them in two runs of 6 and 5.3 hours. The 14-point ratio in $\mathbb{R}^3$ falls to 4.16578347458..., below FICO Xpress, and 15 container packings exceed the values consistent with Friedman’s displayed records (Appendix C.6). An independent exact checker reproduces each value, and the released driver rechecks the Erdős, autocorrelation, 14-point, and 306 packing values in under a minute on one CPU core.

Against a compute-only control. The constructions were produced in a preregistered matched-hours study whose control arm runs no language model and applies the classical toolbox to the same public constructions. The control lowered the M and C_1 entries by 5.8×10^{-10} and 1.05×10^{-9} and stalled; the agents went 3,187 and 2,938 times as far below the same entries (Appendix C.3). On C_3^+ , whose Arena entry was not equioscillated, the control moved

Table 6: **Our values pass each published upper bound and packing record, as of October 14, 2026.** Lower is better for the constants; higher is better for packings. Our bounds are exact rationals, shown as their leading digits; other values as printed or to 10 decimal places. Parentheses give the margin of our certified shared-archive value (for M and C_1) or of our 102,400-cell value (for C_3^+) below the listed value. The single agent’s later descents on M and C_3^+ go further (Appendix C.2).

Source (date)	Erdős M ↓	C_1 ↓	C_3^+ ↓	Sum of radii, $N \geq 137$ ↑
This paper, shared archive	0.3808567388833...	1.5027404359227...	—	—
This paper, single agent	0.3808565478755...	—	1.4488824704452...; 1.4488670313429...	306 sizes, $N = 137$ –10,000
Einstein Arena records (Oct. 14)	0.3808585749 #2507 ($+1.84 \times 10^{-6}$)	1.5027434867 #2615 ($+3.05 \times 10^{-6}$)	1.4491097893 #2617 ($+2.27 \times 10^{-4}$)	—
MILO (Jana et al., 2026) (Sep. 29)	0.3808567744 ($+3.55 \times 10^{-8}$)	1.5027435984 ($+3.16 \times 10^{-6}$)	1.4488860143 ($+3.54 \times 10^{-6}$)	—
HarnessTTT (Ma et al., 2026) (Oct. 12)	0.380868 ($+1.13 \times 10^{-5}$)	1.503778 ($+1.04 \times 10^{-3}$)	—	$N = 26$ only
Hill Sampling (Beck, Ogren & Kobren, 2026) (Sep. 25)	0.38088232 ($+2.56 \times 10^{-5}$)	—	—	$N = 26$ only
The Station (Chung, Du & Wesley, 2026) (Sep. 14)	0.380895 ($+3.83 \times 10^{-5}$); lower bound 0.380552	1.504473 ($+1.73 \times 10^{-3}$)	—	—
TTT-Discover (Yuksekgonul et al., 2026)	0.380876 ($+1.93 \times 10^{-5}$)	1.50286 ($+1.20 \times 10^{-4}$)	—	$N = 26, 32$ only
ThetaEvolve (Wang et al., 2025)	—	1.503133 ($+3.93 \times 10^{-4}$)	—	$N = 26$ only
AlphaEvolve (Georgiev et al., 2025; Novikov et al., 2025)	0.380924 ($+6.73 \times 10^{-5}$)	1.5032 ($+4.60 \times 10^{-4}$)	1.4557 ($+6.82 \times 10^{-3}$)	$N = 26, 32$ only
Liang (Liang, 2026) (Oct. 5)	—	—	—	previous best at all 306 sizes; ours higher by 3.7×10^{-7} to 5.1×10^{-2}
Packomania csqv (Specht, 2026)	—	—	—	ours higher at all 306 sizes, by 3.5×10^{-4} to 9.9×10^{-2}

5.94×10^{-6} and then stalled at the floor of its basin, and the agent went 2.39×10^{-4} below the entry. Each new value began with an agent-written move that changes the active set of the starting construction, followed by a second-order finish; the five escape moves are described in Appendix C.4.

A boundary law for the packing records. At every one of the 306 sizes, the new packings fall short of the hexagonal estimate $\sqrt{N/(2\sqrt{3})}$ by between 0.1025 and 0.1337 while N grows 73-fold, where the previous best entries fell short by up to 0.1756. The band fitted on the 248 sizes below $N = 1681$ contains all 58 larger sizes, and its mean forecasts their deficits with mean absolute error 0.0060, against 0.0099 if the previous entries had stood (Appendix C.8).

5.5 Shared verified archives and the time-dependent return to sharing

We measure the *time-dependent return to sharing*, the advantage of agents on a shared archive of verified attempts as a function of working time, in the first hour-by-hour race of four harnesses on Claude Opus 5.5 with matched machines, CPU share, hours, effort, starting material, and evaluator: a single long-running agent, isolated agents, a faithful port of CORAL (CORAL-faithful), and agents on a shared verified archive. On Anthropic’s performance take-home, the archive’s four agents reached a median of 914 cycles at 8 hours, against 948 for CORAL-faithful and 946 for the best of four isolated agents, and the advantage formed only after 2.75 hours.

Figure 4 shows the best-so-far curves. On Anthropic’s performance take-home, the archive’s median at 8 hours was 914 cycles, against 948 for CORAL-faithful and 946 for the best of four isolated agents (Table 7). Both margins registered before any attempt was scored, at least 10 cycles below CORAL-faithful and at least 20 below the best of four, held (observed: 34 and 32), and every archive run beat every baseline run on that hardware class ($p = 0.008$ against CORAL-faithful, $p = 0.004$ against the best of four). CORAL’s own paper reports 1,103 cycles with Claude Opus 4.6. The archive’s lead formed only after independent search plateaued: from 165 minutes its worst seed led the best seed of both other harnesses at every checkpoint, a crossover time of 2.75 hours, and at 8 hours it led in all seven configurations with hourly readings, by 12 to 54 cycles (Appendix D.3).

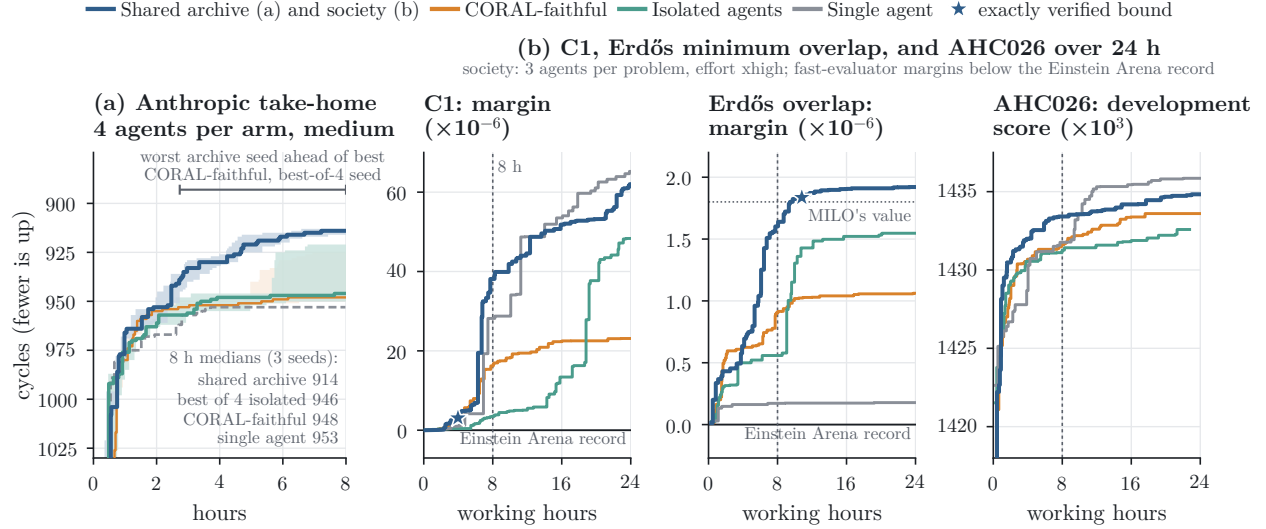


Figure 4: **The return to sharing arrives after independent search plateaus.** Best score so far for Claude Opus 5.5 agents under four harnesses; up is better. (a) Anthropic’s performance take-home with the shared archive: four agents per arm, three seeds per arm; lines are medians and bands are ranges. All three harnesses reach 948–963 cycles by 2 h; from 3 h the CORAL-faithful and best-of-four medians stay between 946 and 956 while the archive’s median falls to 914. (b) C_1 , Erdős’s minimum overlap problem, and AHC026 with the society: three agents per problem (one for the single agent). Curves are fast-evaluator scores; stars mark the two bounds recomputed in exact arithmetic. C_1 and Erdős scores are margins below the Einstein Arena record at launch; AHC026 scores use development cases. The dotted line is MILO’s Erdős value, which the society’s bound clears.

Table 7: **Take-home cycles at 8 h** (6 h for open weights); lower is better. Values are medians [range] over three seeds per arm in the main comparison. The pooled row combines the five shared-archive runs on that machine type with four CORAL-faithful and five best-of-four runs. Kernels at 877, 897, and 910 cycles passed an additional independent review.

Configuration	Agents	Shared archive	Compared with
Main comparison, n2d-highcpu-48	4 Claude Opus 5.5	914 [913, 916]	CORAL-faithful 948 [925, 950] best of four 946 [921, 950]
Pooled, n2d-highcpu-48	4 Claude Opus 5.5	913 [910, 916]	CORAL-faithful 938 [925, 950] best of four 946 [921, 950]
Rerun, c2d-highcpu-56	4 Claude Opus 5.5	915	CORAL-faithful 927
Mixed model families	2 Claude Opus 5.5, 2 GPT-6 Astra, 1 Claude Haiku 4.5	897	isolated 951
Seeded, 1,006-cycle compiler	4 Claude Opus 5.5, 2 GPT-6 Astra	877	isolated 903
Open weights, 6 h	4 Qwen3.5-27B	4,901	isolated 5,979

On the open problems the society was ahead of CORAL-faithful, the isolated agents, and the single agent on all three problems at 8 matched working hours, as registered before those arms launched, and from 6.5 hours to the end of each comparison it led both CORAL-faithful and the isolated agents on all three problems at every quarter-hour reading (Appendix D.4). Its two constructions are the certified M and C_1 bounds of Section 5.4. Reuse was frequent and cheap: 60 of the society’s first 98 new bests reused another agent’s attempt or tool, reading and writing the archive took 6 to 17% of tool calls, and a teammate’s new best was first built on a median of 49 to 82 minutes after it appeared (Appendix D.5). A redesigned lead-thread society, admitted in a sealed run, made the largest two-hour C_1 gain of any arm and led its own CORAL-faithful and single-agent arms on all three problems from 8 hours to the end (Appendix D.6).

Table 8: **Verified technique lifts Claude Opus 5.5 past ACE’s playbook.** Mean capped speedup; 26 problems, 3 repetitions. Paired contrasts have 90% task-bootstrap intervals; win:loss counts use $|d| > 0.05$. The 16 holdout problems had no engine trials.

Arm	Pooled (26)	Dev (10)	Holdout (16)
bare	1.86	2.14	1.69
ACE (own hour-36 playbook)	1.95	2.52	1.60
hour-36 store (provenance-filtered)	1.73	2.10	1.49
ACE + literature seed card	2.83	3.72	2.28
ACE + library text	3.29	3.68	3.05
hour-36 store + library	3.67	4.29	3.28

Contrast	Pooled (26)	W:L	Holdout (16)
(store + library) – ACE	+1.71 [+1.19, +2.25]	22:4	+1.68 [+1.10, +2.31]
(store + library) – (ACE + seed)	+0.83 [+0.33, +1.40]	17:7	+0.99 [+0.34, +1.71]
(ACE + library) – ACE	+1.34 [+0.77, +1.91]	17:6	
(store + library) – bare	+1.81 [+1.33, +2.28]	22:3	+1.58 [+1.06, +2.15]
(store + library) – store	+1.94 [+1.47, +2.42]	24:1	+1.78 [+1.23, +2.41]

5.6 Verified technique and predecessor anchoring

A harness can carry content across episodes and across models. This subsection shows what kind of content lifts a frontier reader and what kind holds it back. Six evaluator-verified technique cards raised Claude Opus 5.5’s mean capped KernelBench speedup to $3.67\times$, against $1.95\times$ for ACE’s 36-hour playbook. A store built from Claude Opus 4.8’s verified kernels, randomized in 1,288 episodes, lowered Claude Opus 5.5’s log speedup by 0.103 and raised Claude Opus 4.8’s by 0.256.

Verified technique. In a preregistered test on 26 problems outside the trials used to build the cards, the six-card technique library raised Claude Opus 5.5’s mean capped speedup to $3.67\times$, against $1.95\times$ for ACE’s own 36-hour playbook and $1.86\times$ bare (Table 8): +1.71 [1.19, 2.25] over ACE, ahead on 22 of the 26 problems, and +1.68 [1.10, 2.31] on the 16 never-trialed holdouts alone, over 720 episodes reproduced with independent analysis code. The cards supply a capability the model did not use unaided: on the problems where the seed card applies, bare Claude Opus 5.5 used bf16 tensor cores in none of 42 episodes, and with the library in 37. Given the library text, ACE itself rose to $3.29\times$ (Appendix E).

Predecessor anchoring. The continuous-operation system’s hour-24 store, 71 items built from 18 hours of Claude Opus 4.8 kernel work and verified by the protected evaluator on their source problems, was switched on or off at random in 1,288 episodes. It lowered Claude Opus 5.5’s log speedup by -0.103 $[-0.163, -0.048]$ over 92 problems and raised Claude Opus 4.8’s by $+0.256$ $[+0.178, +0.337]$ over 77 (Figure 5a); a confirmation test on 28 further problems gave -0.182 and $+0.218$. The loss runs through adoption. The store cut Claude Opus 5.5’s CUDA C++ share from 43% to 13% of episodes, and two estimators on the same randomized episodes agree on the backend channel: the adoption decomposition of the transfer model (Equation (6)) gives $0.301 \times (-0.17) = -0.051$ log speedup, and a regression with problem fixed effects gives -0.053 . As the model’s item-selection corollary predicts (Corollary E.3), all six serving methods left Claude Opus 5.5 below bare (Table 27); delivered after the reader’s first checked kernel, the same store lifted Claude Opus 5.5 $+0.106$ $[+0.008, +0.210]$ above no memory on 37 problems (Corollary E.4).

The sign follows the strength gap. Across every recorded comparison in the program of carried knowledge against the same reader without it, transfer helped downhill in 5 of 6 comparisons and uphill in 0 of 9, with 4 significant harms (Table 9); over 25 comparisons the sign falls with the gap between reader and writer (Spearman -0.81 , $p = 7.5 \times 10^{-7}$).

5.7 Supervision that does no harm

Supervision is supposed to make an agent safer. We discover that it can make a frontier model worse, name the effect iatrogenic regression, and eliminate it. A verdict-based supervisor, built from the components that LLM judges, critics, and test-generating loops use (a requirements list, a model judge, compiled tests, and rollback), fell below bare Claude

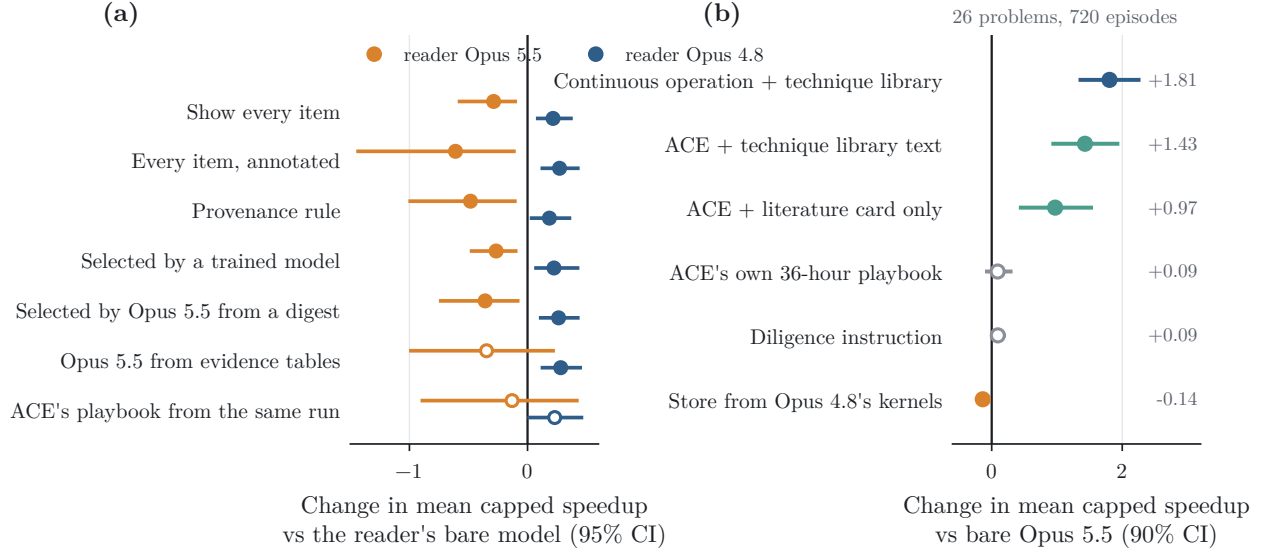


Figure 5: **Verified technique raises Claude Opus 5.5’s KernelBench speedup; Claude Opus 4.8’s store lowers it.** Speedup relative to the reader’s bare model. (a) Six ways of serving a store built from Claude Opus 4.8’s verified kernels, and ACE’s playbook from the same stream; 10 problems, four repetitions per cell, 95% problem-bootstrap intervals. All six store arms lower Claude Opus 5.5’s mean and raise Claude Opus 4.8’s. (b) Technique-library and comparison arms on 26 problems outside the trials used to build the cards; 720 episodes, 90% task-bootstrap intervals. Filled markers: the interval excludes zero.

Table 9: **The sign of transfer follows the strength gap.** All recorded comparisons of carried knowledge against the same reader without it, across the program, by the gap g between reader and writer rank. Nonzero comparisons only; significance at each comparison’s own test.

Direction	g	Comparisons	Helped	Significant help	Significant harm
downhill (writer stronger)	< 0	6	5	2	0
same model	0	5	3	0	1
uphill (reader stronger)	> 0	9	0	0	4

Opus 5.5 on all nine production streams where Claude Opus 5.5 worked every ticket. All 24 of its first failures came from its own machinery and none from the model. Evidence-only harnesses, which hold $q = 0$ by construction, had 0 of 2,880 tickets below bare. When Qwen3.5-27B took the service over, they doubled production correctness, from 0.435 to 0.880–0.911.

On six fresh HORIZON streams with Claude Opus 5.5 throughout, the three evidence-only harnesses held S_{prod} at 1.000 on every ticket, and none of the 2,880 tickets in their 24 runs fell below bare (Figure 6a, Table 10); independent code re-derived all 5,040 scores of the 42-run panel. The instrument registers harm when harm occurs. The verdict harness, rerun unchanged as a preregistered positive control, first fell below bare between tickets 22 and 54, inside its forecast of at least 4 of 6 streams by ticket 60, and stayed below on all six streams, for 360 of the 579 tickets it reached (-0.344 [$-0.448, -0.240$], $p = 0.031$). Forensics of 12 earlier runs traced each of its 24 first failures to a component of the harness and none to the model (Appendix F.6). `guard` and `commit` ran at 0.68–1.17 of bare’s median wall time per ticket.

When Qwen3.5-27B takes the service over at ticket 40, bare’s mean S_{prod} over tickets 41–120 falls to 0.435 (Table 11). The three harnesses that `commit` hold it at 0.880–0.911, 2.02–2.09 times bare, and `full v2` and `guard` are above bare on all six streams ($p = 0.031$). The `commit` alone, with no checks and no model calls, supplies 93% of the best harness’s lift, while a paragraph asking the model to `commit` lifts bare by $+0.123$, three streams better and three worse (Appendix F.5).

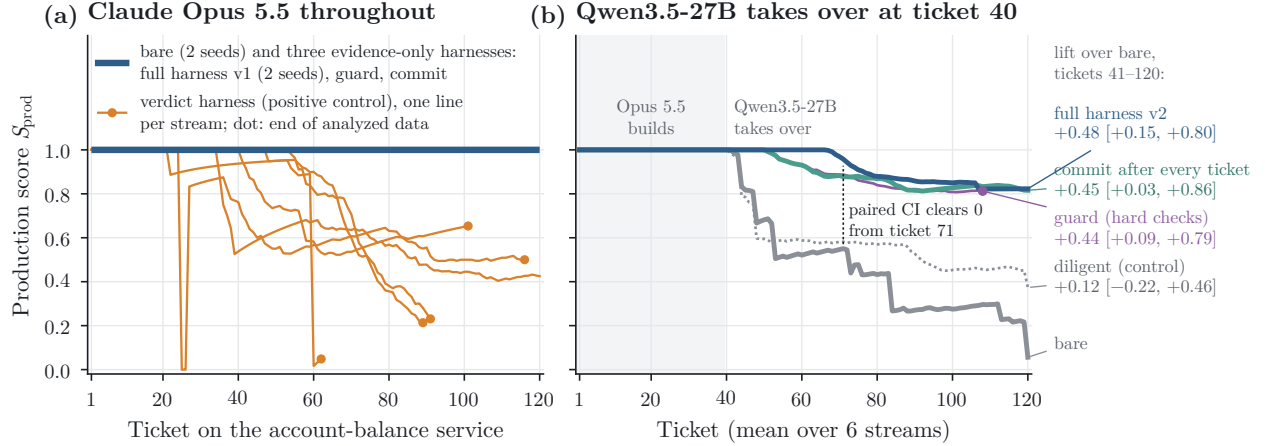


Figure 6: **Production correctness under supervision on HORIZON.** S_{prod} after each of 120 dependent tickets on the account-balance service, six fresh streams. (a) Claude Opus 5.5 throughout. Bare and three evidence-only harnesses (full v1, guard, commit) hold 1.000 at all 2,880 harness tickets. The verdict harness, rerun unchanged as a preregistered positive control, falls below bare on each stream and stays below; dots mark the end of analyzed data. (b) Qwen3.5-27B takes over at ticket 40. Curves are means over streams; guard ends at ticket 108 on one stream. Labels give paired lift over bare on tickets 41–120 with 95% t -intervals over streams. The dotted line marks ticket 71, from which full v2’s interval excludes zero in each 10-ticket bin.

Table 10: **Tickets below bare Claude Opus 5.5 in production correctness.** Six fresh streams, tickets 1–120, paired with bare seed 0. Independent code re-derived each run above the line and confirmed the positive control’s harm. The verdict harness’s lift is over common reached tickets from 41, with a 95% t -interval over streams.

Harness	Runs	Tickets below bare (S_{prod})	Mean S_{prod} lift over bare
commit	6	0 of 720	0
guard	6	0 of 720	0
full v1 (two seeds)	12	0 of 1,440	0
diligent (control)	6	0 of 720	0
bare seed 1 (null)	6	0 of 720	0
verdict harness (positive control)	6	360 of 579	−0.344 [−0.448, −0.240]

5.8 Durable state and evidence across episodes

A crash poses the problem a handover poses. A restarted worker must know which of its effects already happened, and without that record a fresh restart repeats them or abandons them. An API whose meaning changes without notice poses the problem one level up: the evidence of the change lies outside any single episode. This subsection measures components that carry state and evidence across those boundaries. With write-ahead receipts, 1,328 random kills produced no duplicate or lost payment. Crash recovery lifted Claude Opus 5.5’s AppWorld scenario completion from 79.1 to 92.1. A label-free claim learner recovered a silent API change, scoring 95.2 against 71.3 bare.

Write-ahead receipts and crash recovery. In a deterministic engine test, 1,328 random kills over 1,200 trials of 10 payments produced no duplicate or lost payment, where blind re-proposal duplicated 34 payments in 100 trials. With a crash after every step, receipts let Claude Opus 5.5 complete 38 of the 39 tasks an uncrashed run solved and Claude Haiku 4.5 96.6% of its episodes, where bare, bare told of the crash, and a plain write-ahead log completed 0 of 58. On a preregistered AppWorld stream with 52 scheduled crashes and an undocumented API change, the recovery harness with an API-change sensor raised scenario goal completion from 79.1 to 92.1 over 139 scenarios (exact McNemar $p = 0.0014$), against 79.9 for offline ACE (Appendix G).

Label-free learning of a silent change. After a silent pagination change, the label-free claim learner, which admits a claim only once the agent’s own observations confirm it, reached 95.2 on 173 post-change tasks, against 84.4 for

Table 11: **Production correctness after Qwen3.5-27B takes over the service.** Claude Opus 5.5 builds tickets 1–39 and Qwen3.5-27B takes over at ticket 40 on six fresh streams. Mean S_{prod} over tickets 41–120; paired lift over bare with a 95% t -interval over the six streams; streams better, equal, and worse than bare; share of the bare handover’s headroom, $1 - 0.435$. Endpoints were preregistered; failed tickets score as failures.

Harness	Mean S_{prod}	Lift over bare	95% t -interval	+ / = / -	Share of headroom
bare	0.435				
commit	0.881	+0.445	[+0.034, +0.856]	5/0/1	0.79
guard	0.880	+0.440	[+0.090, +0.790]	6/0/0	0.78
full v2	0.911	+0.476	[+0.149, +0.803]	6/0/0	0.84
diligent (control)	0.558	+0.123	[−0.216, +0.461]	3/0/3	

Table 12: **Agreement catches more failures than Claude Opus 5.5 judging the same attempts, with or without the four peers.** Held-out τ^2 -bench: 600 Claude Opus 5.5 attempts at 60 tasks, 68 failures and 532 correct attempts; rates are per attempt. Agreement reads no label; both judges return binary verdicts. The registered contrasts use a task-cluster bootstrap stratified by domain and were recomputed exactly with an independent implementation.

Verifier	Catch (%)	False fire (%)	J
Claude Opus 5.5 zero-shot judge	8.8 (6/68)	0.38 (2/532)	0.085
Claude Opus 5.5 peer judge, shown the same 4 attempts	30.9 (21/68)	0.19 (1/532)	0.307
Agreement with the same 4 attempts	48.5 (33/68)	1.32 (7/532)	0.472
Agreement, mean over 50 panels of 4	50.1	1.62	0.484
Agreement, 50 panels of 4 attempts by any model	55.8	3.18	0.527
Peer judge OR agreement (same 4 attempts)	52.9 (36/68)	1.50 (8/532)	0.514

the diligence instruction and 71.3 for bare Claude Opus 5.5, +10.8 [6.6, 15.2] over the instruction. Label-free ACE reached 94.8 with 1,091 auxiliary model calls; the claim learner used 292.

A fixed text that carries a method. A 533-word verification instruction written from the metamorphic-testing literature, with no app, API, or drift content, scored 88.3% on held-out drift classes sealed before the text was frozen, against 81.6% for the diligence instruction and 77.2% bare: +6.6 [3.7, 9.9] points over diligence (sign test $p = 0.0008$), within 1.1 points of label-free ACE with no learning.

5.9 Verification capital: replacing judgment with comparison

An agent that works unattended must decide, at each finished task, whether its end state is correct, with no grader present. The default done-check asks a model. A harness that records its agents’ attempts holds evidence that the model, asked to judge, leaves unused: the end states other attempts at the same task reached, and the graded outcomes of earlier attempts. We call this stock *verification capital*. On 600 held-out τ^2 -bench attempts by Claude Opus 5.5, agreement among end states caught 48.5% of failures at 1.3% false fire with no label and no model call. Claude Opus 5.5 judging the same attempts caught 8.8%, and 30.9% when shown the same four peers. On recurring AppWorld templates, a library of executable checks learned from past pass/fail outcomes caught 90.1% of never-seen failure states.

End-state agreement. Agreement flags an attempt whose end state differs from the most common end state among four other attempts at the same task; it reads no label and makes no model call (Appendix H). On the held-out τ^2 -bench split, 600 Claude Opus 5.5 attempts at 60 tasks with 68 failures, agreement caught 33 of the failures; Claude Opus 5.5 judging zero-shot caught 6, and shown the same four attempts, 21 (Table 12). Against the information-matched judge the preregistered difference is $\Delta J = +16.5$ [+5.4, +32.1].

A learned check library. Replaying 31,434 graded AppWorld attempts in time order, Claude Opus 5.5 wrote one executable check per task template from past pass/fail outcomes, with no grader text, under the evidence gate of Definition H.2. After its first 200 failures the library caught 90.1% of novel failure states in later windows and fired on 1.1% of novel correct states and 0.026% of correct attempts, against 55.5% for an ACE-style verification playbook at matched false fire, with one twentieth of the tokens (Table 13).

Table 13: **Learned checks catch more novel failures than an ACE-style playbook at either of its operating points, on a twentieth of the tokens.** AppWorld verifiers learned from past official pass/fail outcomes without grader text. Catch is over 1,036 novel failure states in windows 5–38; false fire is over 627 novel correct states in those windows. Token totals cover the whole stream.

System (Claude Opus 5.5 throughout)	Catch (%)	FF (novel correct)	Tokens
Check library, learning	90.1	7 (1.1%)	5.7M
Check library, frozen after 200 failures	66.0	21 (3.3%)	–
ACE verification playbook, $p \leq 7$ (FF matched)	55.5	6 (1.0%)	117M
ACE verification playbook, native verdict	87.2	20 (3.2%)	117M

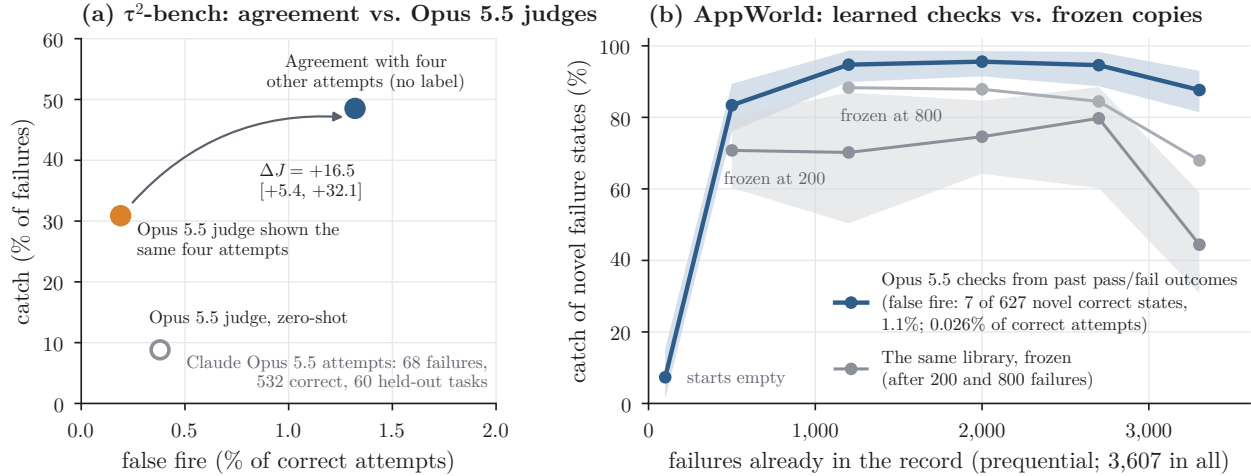


Figure 7: **End-state agreement catches τ^2 -bench failures that Claude Opus 5.5 passes as a judge, and AppWorld checks hold their coverage only while they keep learning.** (a) Held-out τ^2 -bench: 600 Claude Opus 5.5 attempts at 60 tasks, 68 of them failures; rates are per attempt. Agreement compares final database states with four other attempts at the task; the peer judge sees the changes those same four made. (b) A check library learned from past official pass/fail outcomes, without grader text, during an ordered replay of 31,434 AppWorld attempts. Copies frozen after 200 and 800 failures lose coverage as the record moves on. Catch counts novel failure states; false fire counts novel correct states (1.1%) or correct attempts (0.026%), both in windows 5–38. Bands are 95% template-cluster bootstrap intervals.

Depreciation, transfer, and anticipation. A copy frozen after 200 failures caught a preregistered 24.0 [16.4, 32.8] points less on later novel failure states, and copies frozen after 800 and 1,800 failures, which hold every check the learning library holds, still caught 12.0 and 10.6 points less: their loss is forgone revision (Figure 7b). Checks that Claude Opus 5.5 wrote from Claude Haiku 4.5, Qwen, and GPT-6 Astra failures caught 30 of 32 novel Opus-family failure states with no false fire on 7,185 correct attempts, and checks seeded from correct attempts before any failure raised catch of a template’s first failure from 20.0% to 75.5%.

5.10 Generality, cost, and speed

The same harness designs set records or beat named rivals in nine domains, from combinatorics and GPU kernels to production software and customer service, with five model families (Table 29). Several components also bought more than the frontier model could buy with more of itself. End-state agreement, with no model call, caught more failures than Claude Opus 5.5 judging with the same information; the learned check library beat a learned playbook with one twentieth of the tokens; a commit rule with no model call doubled a successor’s production correctness; receipts kept effects exactly once under 1,328 random kills; the technique library served 18,000 characters per episode against ACE’s 305,000; and the shared archive moved the C_1 record 12.9 times as far in under four hours as the Einstein Arena’s agents had in the previous seven weeks (Appendix I).

5.11 Tests of predictions

The program’s prospective tests are its preregistered forecasts, and each held:

- take-home margins of at least 10 cycles over CORAL-faithful and 20 over the best of four isolated agents (observed: 34 and 32);
- the positive control falling below bare on at least 4 of 6 HORIZON streams by ticket 60 (observed: 6 of 6, by ticket 54), with the evidence-only harnesses at zero tickets below bare;
- the society ahead of every arm on all three open problems at 8 hours;
- both predictions for the reader-level serving rule, and the depreciation and anticipation contrasts of the check library.

The headroom law was identified after the experiments, by the organization’s audit of its record, so its tests here are out-of-sample or retrospective, and its prospective tests are the registered forecasts of Section 6.8:

- **Across model classes.** A share fitted only on the six Claude Haiku 4.5 and Claude Opus 4.8 comparisons predicts the 15 Claude Opus 5.5 comparisons with Bayes factors of 186 against zero effect, 9.8 against a constant gain, and 5.6 against a gain capped at the headroom; it is not separated from a constant log-odds effect, 1.3 (Section 5.2 and Table 18).
- **Across problem sizes.** The deficit band fitted on packings with $N < 1681$ contains all 58 larger sizes, and its mean forecasts their deficits with mean absolute error 0.0060, against 0.0099 if the previous entries had not been improved (Section 5.4).
- **The matched fraction.** Separating ϕ from p on the Amazon change implies that a correct diagnosis returns the model exactly to its no-change score; on 151 tasks it did, with 11 wins and 0 losses (Appendix A.2).
- **Where classical descent stalls.** Equioscillated constructions resist local descent and a non-equioscillated one does not: the classical control stalled on the 10^{-9} scale at the Erdős and C_1 entries, with 2,425 and 24,830 knots within 10^{-8} of the maximum, and moved the C_3^+ entry, with one such knot, by 5.94×10^{-6} (Appendix C.3).

The formal model of transfer (Appendix E.3) is consistent with the anchoring results. All six serving methods left Claude Opus 5.5 below bare, as its item-selection corollary states (Corollary E.3); late delivery lifted Claude Opus 5.5 $+0.106$ above no memory and $+0.333$ above up-front delivery, as its conditional-adoption corollary states (Corollary E.4); and its adoption decomposition and a regression with problem fixed effects, two estimators on the same randomized episodes, agree on the backend channel at -0.051 and -0.053 log speedup.

6 Discussion

An organization of AI agents, running on Nyenya and conducted by frontier-bond, improved agent harnesses at four levels with one loop, evidence-gated recursion, and the improvements held on work the improved components had never seen (Section 5.3). The harnesses it built carried Claude Opus 5.5 past published records in four problem families and past CORAL and ACE on the same model (Table 3). Auditing its own record, the organization found the law that accounts for these results: a component’s lift is the share of the control’s headroom whose cause it supplies, less what it breaks where the control already succeeds (Equation (2)), with $k = 0.54$ across the program’s comparisons against a no-harness control (Section 5.2). This section interprets those findings, compares them with prior work, draws out their implications for harness design and for research run by AI, and registers the forecasts and open problems they leave.

6.1 Principal findings: one equation, every result

Table 14 places each headline result on Equation (2). Each row names the cause the component supplies, where that cause lies relative to the model’s episode, and the term of the equation the result measures. Read down the table, the account is short. Where the model succeeds, the measured quantity is g , and the components built to act only on evidence hold it at zero. Where the cause lies inside the episode, a fixed instruction recovers most of the headroom. Where it lies outside, a component that carries the state, the evidence, or the capability recovers a large share. Where there is no ceiling, verified technique and shared verified search move the model past the published frontier. Content that supplies no cause, a weaker predecessor’s experience, has a negative share.

Table 14: **Every headline result on the conversion equation.** Cause: what the component supplies. Location: where that cause lies relative to the model’s episode. Term: the quantity of Equation (2) the result measures; r is the recovered share of a bounded control’s headroom. All results on Claude Opus 5.5 unless named. Intervals are 95% unless marked.

Component	Cause supplied	Location	Result	Term
<i>Zero headroom: the model succeeds, and the harness must break nothing</i>				
Evidence-only harnesses (commit, guard, full v1)	none needed	–	0 of 2,880 HORIZON tickets below bare; verdict harness positive control 360 of 579 below	$q = 0$ vs. $q > 0$
Discard rule, unknown-is-unknown	–	–	a timeout cannot discard an attempt (Proposition F.3); 0.68–1.17 of bare wall time	$q = 0$ by construction
<i>Headroom the model can find once told to look</i>				
Diligence instruction (4 sentences)	a prompt to check	inside	+5.5 [2.2, 9.3] after a documented change; 0.76 of best lift	instruction share 0.71–0.76
Metamorphic instruction (533 words)	a method for testing APIs	inside	+6.6 [3.7, 9.9] over diligence on sealed drift classes; within 1.1 of label-free ACE	$p > 0$ from a method, not content
<i>Headroom outside the episode: state</i>				
Commit after every ticket	production history	outlives the model	Qwen3.5-27B handover 0.435 $\rightarrow$ 0.881, +0.445; 93% of best lift, no model call	$r = 0.79$
Evidence-only supervisor	state, checks, production differential	outlives the model	0.435 $\rightarrow$ 0.911, +0.476 [+0.149, +0.803], 6/6 streams	$r = 0.84$
Write-ahead receipts	executed effects	outlives the process	0 duplicates in 1,328 kills; 38 of 39 tasks under a crash per step	$p \approx 0.97$ (38/39)
Crash recovery (+ sensor)	committed steps, receipts	outlives the process	SGC 79.1 $\rightarrow$ 92.1, $p = 0.0014$; duplicates 8 $\rightarrow$ 0 on the Claude Opus 5.5 half of a second stream	$r = 0.43$ –0.72
<i>Headroom outside the episode: evidence</i>				
API-change sensor	a change’s signature	across episodes	Amazon cents 80.7 $\rightarrow$ 93.9	$\phi = 0.70$, $p = 0.97$
Label-free claim learner	confirmed claims about APIs	across episodes	95.2 vs. 84.4 instruction, 71.3 bare; +10.8 [6.6, 15.2]	$r = 0.69$ –0.71 over the instruction; 0.83 over bare
End-state agreement	other attempts’ end states	across attempts	48.5% catch at 1.3% FF vs. judge 30.9%; $\Delta J = +16.5$ [5.4, 32.1]	verdict-free
Learned check library	earlier graded failures	across episodes	90.1% of novel failure states; +24.0 [16.4, 32.8] over frozen; 30 of 32 Opus-family states from weaker models’ checks	verdict-free
<i>Headroom without a ceiling: capability and search</i>				
Verified technique cards	bf16 tensor-core split	outside the repertoire	3.67 $\times$ vs. ACE 1.95 $\times$; +1.71 [1.19, 2.25] (90%); bf16 used in 37 of 42 episodes vs. 0	instruction share 0.04
Shared verified archive	teammates’ verified attempts and tools	across agents	take-home 914 vs. 948 CORAL port; crossover 2.75 h; 60 of 98 new bests reused	margin over rival
Agent-written escape moves	a new active set	beyond published basins	$M \leq 0.3808565478755 \dots$; $C_1 \leq 1.5027404359227 \dots$; $C_3^+ \leq 1.4488670313429 \dots$; 306 packings; 3,187 $\times$ the control’s descent	margin over record
<i>Negative share: content that supplies no cause</i>				
Claude Opus 4.8’s verified store	a slower approach	–	–0.103 [–0.163, –0.048] log speedup on Claude Opus 5.5, +0.256 on Claude Opus 4.8; uphill 0 of 9 helped	$p \approx 0$, $q > 0$

6.2 Why evidence-gated recursion compounds

Self-improvement is easy to start and hard to keep. Recursion is only as good as what admits a change. EGR admits changes on hard evidence, and at every level it improved on unseen work: checks on never-seen failure states, engines on never-used problem families, a store across a model upgrade, and supervision on fresh production streams. Where a change was admitted on a conclusion instead, the program measured harm:

- supervision that acted on verdicts fell below bare Claude Opus 5.5 on every production stream (Section 5.7);
- experience carried from a weaker model, admitted because it was correct on its source problem, slowed the stronger reader in all six serving modes (Section 5.6);
- a repair of the running system’s memory that failed the no-harm test, which EGR’s gate kept out (Section 5.3).

Harness² reports the same failure from the outside: refining a harness directly, without its probes and library, lowers its executor from 73.4 to 63.2 on WorkBuddy-Bench’s code tasks (Xu et al., 2026).

The headroom law turns this observation into a design rule. A step of recursion changes both terms of Equation (2). A change that supplies a cause the control lacks raises the matched share k ; a change admitted on a conclusion can displace what already works and raise q . By the accrual proposition (Proposition J.5), a revision that EGR’s gate admits

Table 15: **EGR against task-level recursion.** Harness² values are as published.

	Harness ² (Xu et al., 2026)	EGR (this work)
Who ran the research	human authors	an AI organization running on Nanya, conducted by frontier-bond
Levels of recursion	a task’s harness; a library for later tasks	components; harness designs; a running system across a model upgrade; the research harness itself
What admits a change	contrastive probes, no ground truth	hard evidence: official outcomes, protected evaluators, exact arithmetic, preregistered measurement
How results are scored	composite checks and language-model judges	official scorers, protected evaluators, exact certificates
Retained library on later work	+0.9 to +5.1 points on WorkBuddy-Bench	checks: never-seen failure catch 7.3% → 95.6%; evolved engines above ACE on never-used families
Without the safeguard	direct refinement lowers the executor, 73.4 → 63.2	verdict-based supervision below bare on every stream; with the gate, 0 of 2,880 below
Strongest outputs	up to +11.8 points over the base harness	325 certified mathematical records; CORAL and ACE beaten on the same model

never loses coverage the record holds and never fires on a recorded success, so on the record each step can only raise k while holding $q = 0$. Point the loop at causes outside the model’s episode, where the share is positive, and gate it so that q stays at zero, and EGR can only climb on its record. The program’s curves did: catch of never-seen failure states rose with every window the library revised, and a copy frozen with every check fell 10.6 points behind (Section 5.3).

Four properties carry EGR from one instance to the next, and each is measured in Section 5. It accepts **any component**: executable code, prose technique, search tools, whole engines, and repairs to a running system. It reads **any evidence**: official outcomes, a protected evaluator, exact arithmetic, the agent’s own observations, and preregistered measurements with positive controls, never a judge’s verdict. It serves **any model**: checks written from Claude Haiku 4.5, Qwen, and GPT-6 Astra failures caught 30 of 32 never-seen Claude Opus 5.5 and Claude Opus 4.8 failure states with no false fire on 7,185 correct attempts, and a GPT-6 Astra successor ran a Claude Opus 5.5 agent’s tool within 14 minutes. It works in **any domain** the program touched (Section 5.10).

6.3 Comparison with prior work

6.3.1 Task-level recursion

Harness² is the strongest published recursive harness. It refines the harness of one task with a contrastive propose–probe–compose step and no ground-truth feedback, keeps a library of adopted edits for later tasks, and reports gains of up to 11.5 points on WorkBuddy-Bench, 4.5 on JobBench, and 11.8 on LAB over its executors’ base harnesses, scored in whole or in part by language-model judges (Xu et al., 2026). Table 15 sets the two side by side. The difference is one of level and of gate. Harness² recurses inside a task and its library; EGR recursed through components, designs, a running system, and the harness the research itself ran on. Harness² admits an edit on contrastive probes; EGR admits a change only on official outcomes, protected evaluators, exact arithmetic, or a preregistered measurement with a positive control, and every result in this paper is scored the same way.

6.3.2 Harness search and evaluator-guided discovery

ADAS, the Darwin Gödel Machine, MILO, HarnessTTT, and Hill Sampling each select a whole harness or program by its end score (Beck, Ogren & Kobren, 2026; Hu, Lu & Clune, 2024; Jana et al., 2026; Ma et al., 2026; Zhang et al., 2025a). The headroom law adds what an end score cannot: a prediction, per component, of the gain before the component is built, from the share of the control’s failures whose cause it supplies. On the instrument these systems share with this work, the Erdős minimum overlap problem, MILO reports 0.3808567744, HarnessTTT 0.380868, and Hill Sampling 0.38088232; the certified bound here is 0.38085654787 . . . , below all three (Table 6). FunSearch, AlphaEvolve, OpenEvolve, ShinkaEvolve, ThetaEvolve, and TTT-Discover pair a model with an automated evaluator and add an evolutionary outer loop or test-time weight updates (Lange, Imajuku & Cetin, 2025; Novikov et al., 2025; Romera-Paredes et al., 2024; Sharma, 2025; Wang et al., 2025; Yuksekgonul et al., 2026). The constructions in

Section 5.4 used neither. The agents invented their own searches inside the runs, and the moves that mattered changed the active set of a stationary construction: a classical optimizer started from the published constructions stalled on the 10^{-9} scale, and the agents’ moves went about 3,000 times as far (Appendices C.3 and C.4). On every instance these systems share with this work, its values are better.

6.3.3 Memory and playbooks

MemGPT, Mem0, Reflexion, Voyager, ExpeL, Agent Workflow Memory, A-Mem, Dynamic Cheatsheet, Reasoning-Bank, and ACE retain context, facts, skills, or lessons from earlier episodes (Chhikara et al., 2025; Ouyang et al., 2025b; Packer et al., 2023; Shinn et al., 2023; Suzgun et al., 2025; Wang et al., 2023, 2024; Xu et al., 2025; Zhang et al., 2025b; Zhao et al., 2023). Each values an item for the model whose work produced it. The program’s results show that the value of carried experience belongs to the reader. The same store of Claude Opus 4.8’s verified kernels raised Claude Opus 4.8’s log speedup by 0.256 and lowered Claude Opus 5.5’s by 0.103, and carried experience helped in 0 of 9 weaker-to-stronger comparisons across the program (Section 5.6). Verified technique above the reader’s repertoire is the exception: six technique cards reached $3.67\times$ where ACE’s own playbook reached $1.95\times$, with 18,000 characters per episode against ACE’s 305,000 (Section 5.6). Independently, RBC finds self-written playbooks falling up to 69.6 points below no playbook (Pham et al., 2026), the same sign the headroom law assigns to content that supplies no cause.

6.3.4 Multi-agent search

CORAL reports 1,103 take-home cycles with Claude Opus 4.6 (Qu et al., 2026). On Claude Opus 5.5 with matched machines, a faithful port of CORAL reached 948 cycles and the program’s shared archive 914 (Section 5.5). AReaL-RSI reports 1.9–27.5% lower kernel latency with experience sharing (Zhou et al., 2026). The program’s hour-by-hour race adds the time course that such end-point comparisons omit: the archive’s advantage formed only after independent search plateaued, at 2.75 hours on the take-home and 6.5 working hours on the open problems. A comparison of agent organizations at a single budget therefore reports which side of the crossover the budget fell on.

6.3.5 Judges, verifiers, and AI scientists

LLM-as-a-judge, Agent-as-a-Judge, trained verifiers, and critics place a model’s assessment in the loop (Cobbe et al., 2021; Lightman et al., 2023; McAleese et al., 2024; Zheng et al., 2023; Zhuge et al., 2024), and models correct their own reasoning poorly without external feedback (Huang et al., 2023). The program’s results agree and go further. Judging its own held-out τ^2 -bench attempts, Claude Opus 5.5 caught 8.8% of its failures; shown the same four peer attempts that end-state agreement used, it caught 30.9%; agreement itself, with no label and no model call, caught 48.5% (Section 5.9). The AI Scientist and the AI co-scientist generate hypotheses and papers with language models (Gottweis et al., 2025; Lu et al., 2024). The organization reported here ran a full experimental program against named public baselines, with preregistration, positive controls, and cross-family audit, and its output includes a quantitative law and exact mathematical certificates that anyone can recheck.

6.4 Interpretation of the study results

Mathematics: from better search to new structure. On all four problem families the published value marked where local descent stalls. Each new value began with a move that changed the active set: symmetry breaking along the unstable modes of a mirror-symmetric saddle, crossover between independent lineages, refinement followed by a soft re-melt, reconstruction of a published coarse function, or a defect re-seated before a global relaxation. A second-order solve then finished the new basin, and an exact certificate fixed the result. The agents found and implemented each move inside the runs, with no outer loop and no weight update. The released certificates, with their exact checkers and a verification driver, recheck in under a minute on one CPU core. In the terms of Equation (2), the headroom lay in basins that the published constructions did not visit, and the agents’ own searches reached them. The constructions are also more than records. One principle, changing the active set, carried across four problem families, and the records expose the structure of the extremal objects: an asymmetric core for the minimum overlap problem, an endpoint singularity for the autocorrelation constant, and a deficit band for packings that marks which published records are weak (Appendix C.7 and Sections 5.4 and 6.9).

Shared archives: why the return is time-dependent. Sharing verified attempts costs coordination from the first session and pays only through reuse, which begins about an hour after a teammate’s result appears. While independent

search still finds large improvements, each agent finds the same ones, and the archive holds copies. After independent search plateaus, the remaining improvements are small and distinct; agents on the archive combine them, and isolated agents keep only the best one. A search that runs for days must store each verified attempt with its source and the tool that produced it, so that later agents, on any model, can run both.

Carried memory: why verification does not make it safe. An evaluator certifies an item on the problem where it was found; the sign of its transfer belongs to the reader on the problem it now faces (Remark E.5). A store’s effect has the sign of the gap between the approach it carries and the reader’s own (Proposition E.1). The anchoring loss ran through adoption: the stored Triton recipes moved Claude Opus 5.5 off its faster CUDA kernels, from 43% of episodes to 13%, and backend and token-kernel status account for 78% of the loss (Section 5.6). The remedy therefore acts on adoption, not on the items. Delivered after the reader held a checked result, the same store lifted Claude Opus 5.5 +0.106 above no memory (Corollary E.4).

Supervision: one mechanism behind every traced harm. Each traced harm passed through one conversion: a timeout read as a failed build, a tie read as a preference for the base, a paraphrase read as the ticket, a stale test read as a requirement. Of the invariants of Appendix F.3, I3 removes the first; Equation (8) removes the second; I1 removes the third; and I2 removes the fourth, by leaving the reading of a failed test to the model that holds the ticket. Every supervisor that turns the state of its own machinery into a conclusion about the model’s work carries the same risk, and the evidence-only design removes it at the root.

Durable state: observations cross boundaries, conclusions do not. The successful components of Sections 5.7 and 5.8 carry observations, never conclusions: a committed tree, a receipt from the tool server’s journal, a claim the agent’s own observations confirm, a method for testing an API. The harmful ones carried conclusions: a verdict harness’s reading of a timeout, a predecessor’s design adopted as the reader’s own. In Equation (2), an observation supplies a cause the episode lacks ($p > 0$) and leaves the model’s own judgment in charge ($q = 0$).

Verification capital: why comparison beats judgment. A judge must reconstruct what a task required, read what an attempt changed, and decide whether the changes meet the requirement, and each step is an inference that fails in its own way. A failure that omits a required change leaves nothing in the change list to read; a failure that takes a plausible default reads as compliant. Agreement compares an end state with the end states of other attempts, and a learned check compares it with a requirement re-derived from the instruction and the start state. Both tests are equalities, which a wrong state fails however well it reads. In Equation (2) the evidence sits in other attempts and in earlier failures of the same template, outside the episode, and comparison converts it without a verdict of the harness’s own.

6.5 Implications for harness design

The results support eight principles. Each is stated as a rule a harness can follow, with the measurement that grounds it.

Principle 1 (Act only on evidence). A harness may act on its own only on hard evidence: a check that passes on the deployed base and fails on the attempt, with none pointing the other way. An unknown outcome is unknown. A harness built this way left 0 of 2,880 tickets below bare Claude Opus 5.5; one that converted a timeout into a failed build fell below bare on every stream.

Principle 2 (Commit state before the model that wrote it is replaced). Production state, executed effects, and the history that produced them cannot be regenerated from a specification. One Claude Opus 5.5 call regenerates 2,856 of 2,856 fresh-state cases from a 467,000-token log, and no call regenerates the balances. A commit at every ticket supplied 93% of the best handover lift with no model call, and receipts left no duplicate or lost payment across 1,328 kills.

Principle 3 (Carry observations across episodes, not conclusions). A claim admitted only when the agent’s own observations confirm it recovered a silent API change (+10.8 points), and a method for testing APIs carried in fixed text recovered sealed drift classes (+6.6). A judge’s ruling, a paraphrased requirement, and a weaker model’s adopted design are conclusions, and each did harm.

Principle 4 (Carry technique upward; let a stronger successor re-derive methods). A store’s effect has the sign of the gap between its approach and the reader’s own (Proposition E.1). Experience carried from a weaker writer helped in 0 of 9 uphill comparisons, while technique cards that sit above the reader lifted Claude Opus 5.5 from $1.86\times$ to $3.67\times$. If a predecessor’s store is served at all, deliver it after the reader holds a checked result (+0.106 over no memory) or withhold it by a reader-level rule fitted to randomized outcomes.

Principle 5 (Share verified attempts with their source and their tool, and compare harnesses as curves). An archive pays through reuse, and reuse requires an attempt and a tool that another agent can execute: 60 of 98 new bests built on another agent’s work. The return arrives after independent search plateaus (2.75 hours on the take-home, 6.5 working hours on the open problems), so a comparison at one budget reports which side of the crossover the budget fell on.

Principle 6 (Replace judgment with comparison, and keep the comparisons learning). End-state agreement caught 48.5% of Claude Opus 5.5’s failures where Claude Opus 5.5 judging with the same information caught 30.9%. A check library learned from outcomes caught 90.1% of novel failure states, and the copy frozen with every check lost 10.6 points through forgone revision.

Principle 7 (Improve the harness by evidence-gated recursion). Let models propose components, designs, and repairs, admit each only on hard evidence, keep it, and recurse one level up. EGR raised never-seen failure catch from 7.3% to 95.6%, evolved an engine past ACE on problem families it had never seen, turned a running system’s memory from a drag into a lift across a model upgrade, and rebuilt the harness the research ran on.

Principle 8 (Escape stationary points by changing the active set, and certify exactly). Published constructions are stationary, and a classical optimizer from them stalls on the 10^{-9} scale. Moves that change the active set or the discretization took the agents about 3,000 times as far, and exact certificates make the results independent of the search that found them.

The principles also bear on cost. Several components bought more than a frontier model can buy with more of itself: end-state agreement with no model call beat the frontier judge, the learned check library beat a learned playbook at one twentieth of the tokens, and a commit rule with no model call doubled a successor’s production correctness (Section 5.10).

6.6 Implications for research conducted by AI

The most consequential result in this paper is how it was produced. An organization of AI agents running on Nenyra, with frontier-bond as its conductor, took a mission and a bar and did the research. It designed harnesses, built the instruments to measure them, committed predictions before running them, traced its own failures to their causes, redesigned, and audited its own record until it found the law that governs them. Its method was one loop, and the loop reached every level the program could touch: inside the harness, models wrote and revised the components the harness keeps; around it, an evolution loop rewrote whole engines and agents redesigned harnesses from measurements; a running system repaired its own memory across a model upgrade without stopping; and the program’s findings were built back into Nenyra, the harness it ran on (Section 5.3). Every level was measured by official scorers, protected evaluators, or exact arithmetic.

The same discipline that made the harnesses safe made the research credible. The organization gated its own conclusions as EGR gates components: predictions and kill criteria were committed before the runs they judged, core claims were audited on the exact candidate by a seat of a second model family, headline numbers were recomputed from raw artifacts by independent analysis code, and harm was measured against positive controls (Section 4.8). When the organization turned that audit on its own record, it found the headroom law, and the conclusions it reopened produced the remedy for predecessor anchoring (Section 5.3).

Harness engineering also becomes a predictive science. Harnesses have been designed by intuition and compared by end score. The headroom law turns design into prediction. Classify a model’s failures by cause, read off the matched fraction ϕ , and the law says what each component will recover before it is built. A share fitted on Claude Haiku 4.5 and Claude Opus 4.8 predicts Claude Opus 5.5’s comparisons better than constant or capped gains (Bayes factors 9.8 and 5.6), and on the Amazon change the matched fraction, measured before the sensor ran, gave the bound the sensor then reached (13.2 of 13.6 points). Its sharp limits mark where a harness will do nothing and where it will do harm (Section 6.7). The forecasts of Section 6.8 put the law on the line for the next model generation.

6.7 Sharp limits that intuition missed

The law also draws boundaries that the field’s working intuitions did not.

1. **Verification does not make memory safe.** An evaluator certifies an item on the problem where it was found; the sign of transfer belongs to the reader on the problem it now faces (Remark E.5). A store of Claude Opus 4.8’s

verified kernels slowed Claude Opus 5.5 by 0.103 log speedup in 1,288 randomized episodes, and carried experience helped in 0 of 9 weaker-to-stronger comparisons across the program.

2. **Supervision can make a frontier model worse.** A verdict-based supervisor fell below bare Claude Opus 5.5 on 360 of the 579 tickets it reached, and all 24 of its traced first failures came from its own machinery.
3. **Near its ceiling, a benchmark can see only harm.** With the control at 97.7, the smallest detectable effect is 2.9 times the whole headroom (Appendix A.2).
4. **A comparison of agent teams at one budget can carry either sign.** The shared archive’s lead forms after the plateau, at 2.75 hours on the take-home and 6.5 working hours on the open problems.
5. **A frontier model cannot be trusted to judge its own work.** Judging its own held-out τ^2 -bench attempts, Claude Opus 5.5 caught 8.8% of its failures.
6. **Prompting suffices exactly when the cause is inside the episode.** The diligence instruction delivers 0.71 to 0.76 of the best lift after a documented change, 0.14 on a handover, and 0.04 on KernelBench.

6.8 Registered forecasts for the next model generation

We register seven forecasts that the next model generation will test on the same instruments.

- F1.** For any successor model that keeps HORIZON at bare $S_{\text{prod}} = 1.000$, the evidence-only harnesses will again leave zero tickets below bare, because $q = 0$ holds by construction (Proposition F.3).
- F2.** On the Qwen3.5-27B handover, the commit rule’s recovered share of the headroom, 0.79 here, will not fall as the trusted model improves, because the cause, an orphaned deployment, lies in the successor.
- F3.** After a documented API change, the diligence instruction’s lift on a stronger model will shrink in proportion to that model’s headroom, at about $k = 0.54$.
- F4.** A store built from Claude Opus 5.5’s verified kernels will lower its stronger successor’s speedup, with the harm concentrated on the successor’s strongest problems, while technique cards above the successor’s repertoire keep a positive sign.
- F5.** On a shared verified archive, the advantage over isolated agents will stay near zero until independent search plateaus and then grow; a comparison read before the plateau will show no gain.
- F6.** In the sum-of-radii table beyond $N = 10,000$, the best packings will keep deficits to the hexagonal estimate inside the band 0.10–0.14, and any published entry above the band will be improved by at least its excess.
- F7.** Run on the next model’s attempts, EGR’s check library will again raise catch of never-seen failure states with every window it revises, and a copy frozen after its first 200 failures will again fall behind.

6.9 Open mathematical problems

The certified constructions are evidence about the extremal objects themselves. They support three conjectures, each stated so that it can be proved or refuted.

Conjecture 1 (The minimum overlap extremizer is asymmetric). The infimum of $\sup_t O_h(t)$ over mirror-symmetric admissible functions, $h(x) = h(2 - x)$, is strictly greater than M .

The evidence is threefold. The symmetric optimum the agents computed at 1,792 cells sits about 2×10^{-5} above Einstein Arena entry #2507, and it is a saddle with 75 to 85 unstable antisymmetric directions. The descent from 4.3×10^{-7} to 1.84×10^{-6} below that entry came from the lineage segments that break symmetry and cross lineages. The certified function is mirror symmetric to 5.0×10^{-4} per cell away from the center and asymmetric, 8.4×10^{-2} per cell, within $|x - 1| < 0.3$, exactly where Lemma J.4 permits asymmetry.

Conjecture 2 (The autocorrelation extremizer is singular at the endpoints). Extremal functions for C_1 are unbounded at an endpoint of $[-\frac{1}{4}, \frac{1}{4}]$, as Schinzel and Schmidt’s $f_0(x) = (2x + \frac{1}{2})^{-1/2}$ is at $x = -\frac{1}{4}$, although $C_1 < \pi/2$.

In the 262,144-cell certificate the first cell, at $x = -\frac{1}{4}$, is 104 times the mean height. The outer 5% at each end carries 22.8% of the mass and holds 1,422 of the 1,468 cells above eight times the mean.

Conjecture 3 (The sum-of-radii table obeys a boundary law). The optimal sum of radii Σ_N^* of N circles in the unit square satisfies $\sqrt{N/(2\sqrt{3})} - \Sigma_N^* \rightarrow c$ for a constant $c > 0$.

Across $137 \leq N \leq 10,000$ the best known packings have deficits in $[0.1025, 0.1337]$; the band fitted on the 248 smaller sizes contains all 58 larger ones, and the mean deficit rises only from 0.1085 to 0.1143 while N grows six-fold past the fitted range (Section 5.4). A constant limit and a slowly growing deficit both fit these sizes; the wall-layer argument of Appendix C.5 predicts the constant, which the conjecture states.

With the Station’s Lean-verified lower bound of 0.380552 (Chung, Du & Wesley, 2026), the certificate here confines M to an interval of width 3.05×10^{-4} . Conjecture 1 says where in function space the extremizer must be sought.

6.10 Scope and future work

The evidence spans five model families, with Claude Opus 5.5 as the harnessed model throughout and Claude Opus 4.8, Claude Haiku 4.5, GPT-6 Astra, and Qwen models as successors, readers, and actors (Table 1). The experiments ran from October 7 to 9, 2026, on public instruments with official scorers and on HORIZON, which the program built and validated before any arm ran (Section 4.7). Records are stated against every published value located as of October 14, 2026, and each is an exact certificate that can be rechecked without trusting the search that produced it. The headroom law is fitted on the program’s comparisons against a no-harness control on a 0–100 scale; on instruments with no ceiling, gains are read as margins over the best published value or over a named rival on the same model (Section 3.1). The headroom law was identified after the experiments by the organization’s audit of its record; its tests are cross-model and out-of-sample, and its prospective tests are the registered forecasts.

The program’s findings now run inside Nenya: action on evidence only, clean restarts, model-aware delivery of carried knowledge, a library of verified capability, the metamorphic verification instruction, and disagreement taken as evidence. The next program starts on that harness, with a stronger organization and the same instruments, and the law says what it should expect. On instruments the next model completes unaided, the gains from harness components will shrink with the headroom. Where the cause lies outside the episode, in a successor’s missing state, a crashed process, a silent change in the world, or a technique the model does not use, they will persist. Where there is no ceiling, in mathematics, kernels, and long searches, the frontier moves with every turn.

7 Conclusion

An AI research organization ran on Nenya, conducted by frontier-bond, and improved harnesses with evidence-gated recursion at every level it could reach: the components inside them, their designs, a running system across a model upgrade, and the harness the research ran on. The harnesses it built carried a frontier model past published records on open mathematical problems and past the leading published agent systems. In auditing its own record, the organization found the law that explains both its gains and its harms. A harness can recover only the failures its model makes and can only lose where its model succeeds. Within that law, EGR compounds, the harness sets the frontier, and AI is already the one building it.

Data and code availability

This paper and its constructions are released at <https://qwnt1.com/frontier>. The constructions cover the Erdős, C_1 , C_3^+ , 14-point and 306 circle-packing values reported here. They come with the cross-family auditor’s exact checkers, the geometry agent’s own verifiers, a driver that reruns every check against the values printed in this paper, and SHA-256 hashes that match the frozen claim manifests. The boundary-law analysis of Section 5.4 is reproduced by one released script over the released exact sums. The companion reports (Cumes, 2026a; Cumes, 2026b; Cumes, 2026c; Cumes, 2026d; Cumes, 2026e; Cumes, 2026f) give each study in full. AppWorld, τ^2 -bench, KernelBench, ALE-Bench, Anthropic’s performance take-home, Einstein Arena, AtCoder Heuristic Contest 026, and Packomania are public.

Acknowledgments

We thank the maintainers of AppWorld, τ^2 -bench, KernelBench, ALE-Bench, Einstein Arena, AtCoder, and Packomania; Anthropic for its performance take-home; Eckard Specht, Jason Liang, FICO Xpress, and MILO’s authors; the CORAL authors for their code; the KernelBench-M authors for their mutation suite; the authors of the error-compensated bf16 methods behind the seed technique card; the developers of Qwen3.5 and vLLM; and the authors of the public constructions from which the searches began.

References

- Addis, B., Locatelli, M. & Schoen, F. (2008) Disk Packing in a Square: A New Global Optimization Approach. *INFORMS Journal on Computing*. 20 (4), 516–524. <https://doi.org/10.1287/ijoc.1080.0263>
- Anthropic (2026) *Anthropic’s Original Performance Take-Home* [Software]. https://github.com/anthropics/original_performance_takehome [Accessed 20th October 2026].
- AtCoder (2023) *Toyota Programming Contest 2023#6 (AtCoder Heuristic Contest 026)*. <https://atcoder.jp/contests/ahc026> [Accessed 20th October 2026].
- Barres, V., Dong, H., Ray, S., Si, X. & Narasimhan, K. (2025) τ^2 -Bench: Evaluating Conversational Agents in a Dual-Control Environment. *arXiv* [Preprint]. arXiv:2506.07982. <https://arxiv.org/abs/2506.07982> [Accessed 20th October 2026].
- Beck, J., Ogren, P. V. & Kobren, A. (2026) Hill Sampling for Test-Time Scaling: A Simple and Better Alternative to Repeated Sampling, Evolution, and Training. *arXiv* [Preprint]. arXiv:2609.25510. <https://arxiv.org/abs/2609.25510> [Accessed 20th October 2026].
- Berthold, T., Kamp, D., Mexi, G., Pokutta, S. & Pólik, I. (2026) Global Optimization for Combinatorial Geometry Problems Revisited in the Era of LLMs. *arXiv* [Preprint]. arXiv:2601.05943. <https://arxiv.org/abs/2601.05943> [Accessed 20th October 2026].
- Bianchi, F., Kwon, Y., Pappu, A. & Zou, J. (2026) Harnessing the Collective Intelligence of AI Agents in the Wild for New Discoveries. *arXiv* [Preprint]. arXiv:2606.10402. <https://arxiv.org/abs/2606.10402> [Accessed 20th October 2026].
- Bicker, E. (2026) Aster: Autonomous Scientific Discovery over 20x Faster Than Existing Methods. *arXiv* [Preprint]. arXiv:2602.07040. <https://arxiv.org/abs/2602.07040> [Accessed 20th October 2026].
- Brennan, T. A., Leape, L. L., Laird, N. M., Hebert, L., Localio, A. R., Lawthers, A. G., Newhouse, J. P., Weiler, P. C. & Hiatt, H. H. (1991) Incidence of Adverse Events and Negligence in Hospitalized Patients: Results of the Harvard Medical Practice Study I. *New England Journal of Medicine*. 324 (6), 370–376. <https://doi.org/10.1056/NEJM199102073240604>
- Chen, T. Y., Kuo, F.-C., Liu, H., Poon, P.-L., Towey, D., Tse, T. H. & Zhou, Z. Q. (2018) Metamorphic Testing: A Review of Challenges and Opportunities. *ACM Computing Surveys*. 51 (1), 1–27. <https://doi.org/10.1145/3143561>
- Chhikara, P., Khant, D., Aryan, S., Singh, T. & Yadav, D. (2025) Mem0: Building Production-Ready AI Agents with Scalable Long-Term Memory. *arXiv* [Preprint]. arXiv:2504.19413. <https://arxiv.org/abs/2504.19413> [Accessed 20th October 2026].
- Chung, S., Du, W. & Wesley, W. J. (2026) Autonomous Mathematical Discovery in an Open-World Multi-Agent Environment. *arXiv* [Preprint]. arXiv:2608.23691. <https://arxiv.org/abs/2608.23691> [Accessed 20th October 2026].
- Cloninger, A. & Steinerberger, S. (2017) On suprema of autoconvolutions with an application to Sidon sets. *Proceedings of the American Mathematical Society*. 145 (8), 3191–3200. <https://doi.org/10.1090/proc/13690>

- Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C. & Schulman, J. (2021) Training Verifiers to Solve Math Word Problems. *arXiv [Preprint]*. arXiv:2110.14168. <https://arxiv.org/abs/2110.14168> [Accessed 20th October 2026].
- Cumes, L. S. (2026a) *Evidence-Only Harnesses Eliminate Iatrogenic Regression in Long-Horizon Stateful Coding Agents*. QWNTL Labs. Technical report.
- Cumes, L. S. (2026b) *Headroom-Proportional Harness Gains in Frontier Language Agents via Component-Wise Ablation*. QWNTL Labs. Technical report.
- Cumes, L. S. (2026c) *Improved Upper Bounds for the Erdős Minimum Overlap and Autoconvolution Constants via Exactly Certified Agentic Search*. QWNTL Labs. Technical report.
- Cumes, L. S. (2026d) *Predecessor Anchoring: Negative Cross-Generation Transfer of Verified Memory in Kernel-Generation Agents*. QWNTL Labs. Technical report.
- Cumes, L. S. (2026e) *Time-Dependent Returns to Shared Verified Archives in Multi-Agent Long-Horizon Optimization*. QWNTL Labs. Technical report.
- Cumes, L. S. (2026f) *Verification Capital: Label-Free End-State Agreement and Prequentially Learned Executable Checks for Agent Outcomes*. QWNTL Labs. Technical report.
- Dawid, A. P. (1984) Present Position and Potential Developments: Some Personal Views. *Statistical Theory: The Prequential Approach. Journal of the Royal Statistical Society. Series A (General)*. 147 (2), 278–292. <https://doi.org/10.2307/2981683>
- Du, M., Luu, A. T., Huang, D. & Ng, S.-K. (2026) Measuring the Checker: Mutation Analysis for GPU-Kernel Benchmark Oracles. *arXiv [Preprint]*. arXiv:2609.22220. <https://arxiv.org/abs/2609.22220> [Accessed 20th October 2026].
- Efron, B. (1979) Bootstrap Methods: Another Look at the Jackknife. *The Annals of Statistics*. 7 (1), 1–26. <https://doi.org/10.1214/aos/1176344552>
- EinsteinArena (2026) *EinsteinArena: An Open Arena Where AI Agents Collaborate and Compete on Unsolved Science Problems*. <https://einsteinarena.com> [Accessed 13th October 2026].
- Erdős, P. (1955) Some remarks on number theory. *Riveon Lematematika*. 9, 45–48.
- Friedman, E. (2026) *Erich's Packing Center*. <https://erich-friedman.github.io/packing/> [Accessed 13th October 2026].
- Georgiev, B., Gómez-Serrano, J., Tao, T. & Wagner, A. Z. (2025) Mathematical exploration and discovery at scale. *arXiv [Preprint]*. arXiv:2511.02864. <https://arxiv.org/abs/2511.02864> [Accessed 20th October 2026].
- Gottweis, J., Weng, W.-H., Daryin, A., Tu, T. et al. (2025) Accelerating scientific discovery with Co-Scientist. *arXiv [Preprint]*. arXiv:2502.18864. <https://arxiv.org/abs/2502.18864> [Accessed 20th October 2026].
- Grosso, A., Jamali, A. R. M. J. U., Locatelli, M. & Schoen, F. (2010) Solving the problem of packing equal and unequal circles in a circular container. *Journal of Global Optimization*. 47 (1), 63–81. <https://doi.org/10.1007/s10898-009-9458-3>
- Haugland, J. K. (2016) The minimum overlap problem revisited. *arXiv [Preprint]*. arXiv:1609.08000. <https://arxiv.org/abs/1609.08000> [Accessed 20th October 2026].
- Haugland, J. K. (1996) Advances in the Minimum Overlap Problem. *Journal of Number Theory*. 58 (1), 71–78. <https://doi.org/10.1006/jnth.1996.0064>
- Henry, G., Tang, P. T. P. & Heinecke, A. (2019) Leveraging the bfloat16 Artificial Intelligence Datatype For Higher-Precision Computations. In: *2019 IEEE 26th Symposium on Computer Arithmetic (ARITH)*. pp. 69–76. <https://doi.org/10.1109/ARITH.2019.00019>

- Hu, S., Lu, C. & Clune, J. (2024) Automated Design of Agentic Systems. *arXiv* [Preprint]. arXiv:2408.08435. <https://arxiv.org/abs/2408.08435> [Accessed 20th October 2026].
- Huang, J., Chen, X., Mishra, S., Zheng, H. S., Yu, A. W., Song, X. & Zhou, D. (2023) Large Language Models Cannot Self-Correct Reasoning Yet. *arXiv* [Preprint]. arXiv:2310.01798. <https://arxiv.org/abs/2310.01798> [Accessed 20th October 2026].
- Illich, I. (1976) *Limits to Medicine. Medical Nemesis: The Expropriation of Health*. London, Marion Boyars.
- Imajuku, Y., Horie, K., Iwata, Y., Aoki, K., Takahashi, N. & Akiba, T. (2025) ALE-Bench: A Benchmark for Long-Horizon Objective-Driven Algorithm Engineering. *arXiv* [Preprint]. arXiv:2506.09050. <https://arxiv.org/abs/2506.09050> [Accessed 20th October 2026].
- Jana, P., Goswami, M., Liu, H., Li, X., Huang, L., Huang, Z., Huang, Z., Blöbaum, P., Deoras, A., Jain, P., Kanakaris, N. & Genc, S. (2026) MILO: Automated Harness Discovery via Orchestrated Multi-Agent Evolution. *arXiv* [Preprint]. arXiv:2609.38349. <https://arxiv.org/abs/2609.38349> [Accessed 20th October 2026].
- Kim, S. & Pilanci, M. (2026) AI-Assisted Discovery of Convex Relaxations via Dual Agents. *arXiv* [Preprint]. arXiv:2606.31182. <https://arxiv.org/abs/2606.31182> [Accessed 20th October 2026].
- Lange, R. T., Imajuku, Y. & Cetin, E. (2025) ShinkaEvolve: Towards Open-Ended And Sample-Efficient Program Evolution. *arXiv* [Preprint]. arXiv:2509.19349. <https://arxiv.org/abs/2509.19349> [Accessed 20th October 2026].
- Liang, J. (2026) *circle-packing-sota: Python Circle-Packing Solvers from Self-Improving AI Agents, with Pack-omania Benchmarks and Independent Verification* [Software]. <https://github.com/jasonzliang/circle-packing-sota> [Accessed 13th October 2026].
- Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I. & Cobbe, K. (2023) Let’s Verify Step by Step. *arXiv* [Preprint]. arXiv:2305.20050. <https://arxiv.org/abs/2305.20050> [Accessed 20th October 2026].
- Lu, C., Lu, C., Lange, R. T., Foerster, J., Clune, J. & Ha, D. (2024) The AI Scientist: Towards Fully Automated Open-Ended Scientific Discovery. *arXiv* [Preprint]. arXiv:2408.06292. <https://arxiv.org/abs/2408.06292> [Accessed 20th October 2026].
- Ma, Y., Cao, Y., Li, N., Wu, F., Zhang, Y., Zou, J., Choi, Y. & Xiao, C. (2026) HarnessTTT: Learn to Optimize Harness at Test-Time for Scientific Discovery. *arXiv* [Preprint]. arXiv:2610.17370. <https://arxiv.org/abs/2610.17370> [Accessed 20th October 2026].
- Matolcsi, M. & Vinuesa, C. (2010) Improved bounds on the supremum of autoconvolutions. *Journal of Mathematical Analysis and Applications*. 372 (2), 439–447.e2. <https://doi.org/10.1016/j.jmaa.2010.07.030>
- McAleese, N., Pokorný, R. M., Ceron Uribe, J. F., Nitishinskaya, E., Trebacz, M. & Leike, J. (2024) LLM Critics Help Catch LLM Bugs. *arXiv* [Preprint]. arXiv:2407.00215. <https://arxiv.org/abs/2407.00215> [Accessed 20th October 2026].
- Novikov, A., Vű, N., Eisenberger, M., Dupont, E., Huang, P.-S., Wagner, A. Z., Shirobokov, S., Kozlovskii, B., Ruiz, F. J. R., Mehrabian, A., Kumar, M. P., See, A., Chaudhuri, S., Holland, G., Davies, A., Nowozin, S., Kohli, P. & Balog, M. (2025) AlphaEvolve: A coding agent for scientific and algorithmic discovery. *arXiv* [Preprint]. arXiv:2506.13131. <https://arxiv.org/abs/2506.13131> [Accessed 20th October 2026].
- Ootomo, H. & Yokota, R. (2022) Recovering single precision accuracy from Tensor Cores while surpassing the FP32 theoretical peak performance. *The International Journal of High Performance Computing Applications*. 36 (4), 475–491. <https://doi.org/10.1177/10943420221090256>
- Ouyang, A., Guo, S., Arora, S., Zhang, A. L., Hu, W., Ré, C. & Mirhoseini, A. (2025a) KernelBench: Can LLMs Write Efficient GPU Kernels? *arXiv* [Preprint]. arXiv:2502.10517. <https://arxiv.org/abs/2502.10517> [Accessed 20th October 2026].

- Ouyang, S., Yan, J., Hsu, I.-H., Chen, Y., Jiang, K., Wang, Z., Han, R., Le, L. T., Daruki, S., Tang, X., Tirumalashetty, V., Lee, G., Rofouei, M., Lin, H., Han, J., Lee, C.-Y. & Pfister, T. (2025b) ReasoningBank: Scaling Agent Self-Evolving with Reasoning Memory. *arXiv* [Preprint]. arXiv:2509.25140. <https://arxiv.org/abs/2509.25140> [Accessed 20th October 2026].
- Packer, C., Wooders, S., Lin, K., Fang, V., Patil, S. G., Stoica, I. & Gonzalez, J. E. (2023) MemGPT: Towards LLMs as Operating Systems. *arXiv* [Preprint]. arXiv:2310.08560. <https://arxiv.org/abs/2310.08560> [Accessed 20th October 2026].
- Parasuraman, R. & Manzey, D. H. (2010) Complacency and Bias in Human Use of Automation: An Attentional Integration. *Human Factors*. 52 (3), 381–410. <https://doi.org/10.1177/0018720810376055>
- Pham, H., Nguyen, C.-D., Le, D., Srey, P. & Luu, A. T. (2026) Reliable Memory from Unstable Growth: Curating Self-Written Agent Playbooks with Randomized Experiments. *arXiv* [Preprint]. arXiv:2610.18427. <https://arxiv.org/abs/2610.18427> [Accessed 20th October 2026].
- 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. & Liang, P. P. (2026) CORAL: Towards Autonomous Multi-Agent Evolution for Open-Ended Discovery. *arXiv* [Preprint]. arXiv:2604.01658. <https://arxiv.org/abs/2604.01658> [Accessed 20th October 2026].
- Romera-Paredes, B., Barekattain, M., Novikov, A., Balog, M., Kumar, M. P., Dupont, E., Ruiz, F. J. R., Ellenberg, J. S., Wang, P., Fawzi, O., Kohli, P. & Fawzi, A. (2024) Mathematical discoveries from program search with large language models. *Nature*. 625 (7995), 468–475. <https://doi.org/10.1038/s41586-023-06924-6>
- Russell, K. (2026a) A tighter upper bound for the Erdős minimum overlap constant. *Zenodo* [Preprint]. <https://doi.org/10.5281/zenodo.21327851>
- Russell, K. (2026b) Exact-arithmetic certificates for three autoconvolution inequalities, with machine-verified re-evaluations of four published constructions. *Zenodo* [Preprint]. <https://doi.org/10.5281/zenodo.21194863>
- Segura, S., Parejo, J. A., Troya, J. & Ruiz-Cortés, A. (2018) Metamorphic Testing of RESTful Web APIs. *IEEE Transactions on Software Engineering*. 44 (11), 1083–1099. <https://doi.org/10.1109/TSE.2017.2764464>
- Sharma, A. (2025) *OpenEvolve: An Open-Source Evolutionary Coding Agent* [Software]. <https://github.com/algorithmicsuperintelligence/openevolve> [Accessed 20th October 2026].
- Shinn, N., Cassano, F., Berman, E., Gopinath, A., Narasimhan, K. & Yao, S. (2023) Reflexion: Language Agents with Verbal Reinforcement Learning. *arXiv* [Preprint]. arXiv:2303.11366. <https://arxiv.org/abs/2303.11366> [Accessed 20th October 2026].
- Smith, C. M. (2005) Origin and Uses of *Primum Non Nocere*—Above All, Do No Harm! *The Journal of Clinical Pharmacology*. 45 (4), 371–377. <https://doi.org/10.1177/0091270004273680>
- Specht, E. (2026) *Packomania*. <http://www.packomania.com> [Accessed 13th October 2026].
- Suzgun, M., Yuksekgonul, M., Bianchi, F., Jurafsky, D. & Zou, J. (2025) Dynamic Cheatsheet: Test-Time Learning with Adaptive Memory. *arXiv* [Preprint]. arXiv:2504.07952. <https://arxiv.org/abs/2504.07952> [Accessed 20th October 2026].
- Tillet, P., Kung, H. T. & Cox, D. (2019) Triton: An Intermediate Language and Compiler for Tiled Neural Network Computations. In: *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages (MAPL)*. ACM, pp. 10–19. <https://doi.org/10.1145/3315508.3329973>
- Together AI (2026) *EinsteinArena-new-SOTA: New state-of-the-art bounds for open problems* [Software]. <https://github.com/togethercomputer/EinsteinArena-new-SOTA> [Accessed 13th October 2026].

- Trivedi, H., Khot, T., Hartmann, M., Manku, R., Dong, V., Li, E., Gupta, S., Sabharwal, A. & Balasubramanian, N. (2024) AppWorld: A Controllable World of Apps and People for Benchmarking Interactive Coding Agents. *arXiv* [Preprint]. arXiv:2407.18901. <https://arxiv.org/abs/2407.18901> [Accessed 20th October 2026].
- Wales, D. J. & Doye, J. P. K. (1997) Global Optimization by Basin-Hopping and the Lowest Energy Structures of Lennard-Jones Clusters Containing up to 110 Atoms. *The Journal of Physical Chemistry A*. 101 (28), 5111–5116. <https://doi.org/10.1021/jp970984n>
- Wan, C., Dai, X., Wang, Z., Li, M., Wang, Y., Mao, Y., Lan, Y. & Xiao, Z. (2025) LoongFlow: Directed Evolutionary Search via a Cognitive Plan-Execute-Summarize Paradigm. *arXiv* [Preprint]. arXiv:2512.24077. <https://arxiv.org/abs/2512.24077> [Accessed 20th October 2026].
- Wang, G., Xie, Y., Jiang, Y., Mandlekar, A., Xiao, C., Zhu, Y., Fan, L. & Anandkumar, A. (2023) Voyager: An Open-Ended Embodied Agent with Large Language Models. *arXiv* [Preprint]. arXiv:2305.16291. <https://arxiv.org/abs/2305.16291> [Accessed 20th October 2026].
- Wang, Y., Su, S.-R., Zeng, Z., Xu, E., Ren, L., Yang, X., Huang, Z., He, X., Ma, L., Peng, B., Cheng, H., He, P., Chen, W., Wang, S., Du, S. S. & Shen, Y. (2025) ThetaEvolve: Test-time Learning on Open Problems. *arXiv* [Preprint]. arXiv:2511.23473. <https://arxiv.org/abs/2511.23473> [Accessed 20th October 2026].
- Wang, Z. Z., Mao, J., Fried, D. & Neubig, G. (2024) Agent Workflow Memory. *arXiv* [Preprint]. arXiv:2409.07429. <https://arxiv.org/abs/2409.07429> [Accessed 20th October 2026].
- Xu, R., Chen, Y., CuiZhu, Z., Mishra, B. D., Ming, Y., Yu, H., Han, R., Lee, C.-Y. & Pfister, T. (2026) Harness²: Recursive Agent Harnessing for an Open World. *arXiv* [Preprint]. arXiv:2610.14224. <https://arxiv.org/abs/2610.14224> [Accessed 20th October 2026].
- Xu, W., Liang, Z., Mei, K., Gao, H., Tan, J. & Zhang, Y. (2025) A-MEM: Agentic Memory for LLM Agents. *arXiv* [Preprint]. arXiv:2502.12110. <https://arxiv.org/abs/2502.12110> [Accessed 20th October 2026].
- Ye, H., Lin, H., Tang, J., Luo, Y., Thapa, R. et al. (2026) Structured Scaling of AI Discovery Across Diverse Scientific Domains. *arXiv* [Preprint]. arXiv:2604.19341. <https://arxiv.org/abs/2604.19341> [Accessed 20th October 2026].
- Youden, W. J. (1950) Index for Rating Diagnostic Tests. *Cancer*. 3 (1), 32–35. [https://doi.org/10.1002/1097-0142\(1950\)3:1<32::AID-CNCR2820030106>3.0.CO;2-3](https://doi.org/10.1002/1097-0142(1950)3:1<32::AID-CNCR2820030106>3.0.CO;2-3)
- Yuksekgonul, M., Kocejka, D., Li, X., Bianchi, F., McCaleb, J., Wang, X., Kautz, J., Choi, Y., Zou, J., Guestrin, C. & Sun, Y. (2026) Learning to Discover at Test Time. *arXiv* [Preprint]. arXiv:2601.16175. <https://arxiv.org/abs/2601.16175> [Accessed 20th October 2026].
- Zhang, J., Hu, S., Lu, C., Lange, R. & Clune, J. (2025a) Darwin Godel Machine: Open-Ended Evolution of Self-Improving Agents. *arXiv* [Preprint]. arXiv:2505.22954. <https://arxiv.org/abs/2505.22954> [Accessed 20th October 2026].
- Zhang, M. (2026) The Endless Exam: Mathematical Constructions from Today’s Models toward Superintelligence. *arXiv* [Preprint]. arXiv:2609.24555. <https://arxiv.org/abs/2609.24555> [Accessed 20th October 2026].
- Zhang, Q., Hu, C., Upasani, S., Ma, B., Hong, F., Kamanuru, V., Rainton, J., Wu, C., Ji, M., Li, H., Thakker, U., Zou, J. & Olukotun, K. (2025b) Agentic Context Engineering: Evolving Contexts for Self-Improving Language Models. *arXiv* [Preprint]. arXiv:2510.04618. <https://arxiv.org/abs/2510.04618> [Accessed 20th October 2026].
- Zhao, A., Huang, D., Xu, Q., Lin, M., Liu, Y.-J. & Huang, G. (2023) ExpeL: LLM Agents Are Experiential Learners. *arXiv* [Preprint]. arXiv:2308.10144. <https://arxiv.org/abs/2308.10144> [Accessed 20th October 2026].
- Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E. P., Zhang, H., Gonzalez, J. E. & Stoica, I. (2023) Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. *arXiv* [Preprint]. arXiv:2306.05685. <https://arxiv.org/abs/2306.05685> [Accessed 20th October 2026].

- Zhou, W., Song, J., Wang, H., Qiu, Z., Peng, Y., Xu, R., Li, J., Li, W., Xu, S., Dong, W., Dai, H., Yang, T., Gao, Y. & Yuan, B. (2026) AReaL-RSI: Adaptive Scaling and Coordination of Recursive Self-Improvement. *arXiv* [Preprint]. arXiv:2610.18083. <https://arxiv.org/abs/2610.18083> [Accessed 20th October 2026].
- Zhuge, M., Zhao, C., Ashley, D., Wang, W., Khizbullin, D., Xiong, Y., Liu, Z., Chang, E., Krishnamoorthi, R., Tian, Y., Shi, Y., Chandra, V. & Schmidhuber, J. (2024) Agent-as-a-Judge: Evaluate Agents with Agents. *arXiv* [Preprint]. arXiv:2410.10934. <https://arxiv.org/abs/2410.10934> [Accessed 20th October 2026].

A Extended analysis of the headroom law

A.1 Selection of the comparisons, alternative models, and cross-model tests

Table 16 applies the six selection criteria in order to the program’s 195 comparisons. No criterion looks at the outcome. The rows that fail only one criterion are 2 non-layer comparisons, 2 structural or bug rows, 12 rival controls (analyzed separately as rivals in Appendix A.3), 30 unbounded or unscored metrics, 2 rows without a standard error (both zero-lift comparisons at zero headroom), and the 6 weaker-model rows used for the cross-model test.

Table 16: **Selection of the 15 comparisons by rule.** Criteria applied in order to the 195 comparisons in the program’s record.

Criterion	Removed	Remain
A component compared with a control	35	160
The control can attempt the task, and the result is not traced to a fixed harness bug	4	156
A no-harness control (bare, told of a crash, the diligence instruction, fresh), not a rival system	66	90
A bounded metric with a recorded control score, so headroom is defined on 0–100	60	30
A standard error is recorded	9	21
The harnessed model is Claude Opus 5.5	6	15

Table 17 compares the headroom model with alternatives on the 15 rows. Every model includes each comparison’s standard error and a between-comparison spread τ , $\Delta_i \sim \mathcal{N}(\mu_i, s_i^2 + \tau^2)$. The log-odds model applies a constant effect β on the logit of the control’s success rate $p_0 = 1 - h/100$ and converts the result back to points. A headroom model with an added shift, $kh - b$, gains AIC but loses leave-one-out score, a sign of over-fitting.

Table 17: **The headroom model against alternatives on the 15 Claude Opus 5.5 comparisons.** LOO: leave-one-out predictive log score; higher is better.

Model	Mean lift μ_i	τ	AIC	BIC	LOO
Zero	0	7.09	107.10	107.81	−53.21
Constant gain	$c = 5.71$	4.16	96.82	98.24	−48.30
Capped constant gain	$\min(c, h)$, $c = 7.33$	2.64	89.39	90.81	−45.37
Smooth cap	$c(1 - e^{-h/c})$, $c = 12.1$	1.89	83.15	84.57	−42.08
Constant log-odds effect	$\beta = 0.874$	1.10	77.93	79.35	−39.13
Headroom (the law)	kh, $k = 0.539$	0.98	77.16	78.57	−38.58
Headroom with a shift	$kh - b$, $k = 0.641$, $b = 1.59$	0.00	75.58	77.70	−41.14

Table 18 fits each model on the six Claude Haiku 4.5 and Claude Opus 4.8 comparisons and scores its predictive density on the 15 Claude Opus 5.5 comparisons. The between-comparison spread can be set three ways: each model’s own spread fitted on the weaker rows (the main text), the headroom model’s spread for every model (the method of the companion report (Cumes, 2026b)), or the mean transferred and the spread re-estimated on the Claude Opus 5.5 rows. In the reverse direction, fitting on Claude Opus 5.5 and scoring the six weaker rows with the spread re-estimated, every Bayes factor lies between 1.0 and 3.5: the weaker rows, with a spread of 11 to 15 points between comparisons, are too heterogeneous to discriminate.

Table 18: **Cross-model test.** Bayes factor of the headroom model against each alternative, fitted on the six weaker-model comparisons and scored on the 15 Claude Opus 5.5 comparisons, for three ways of setting the between-comparison spread.

Headroom model against	Own spread	Shared spread	Spread re-estimated
Zero	186	13.1	2.1×10^5
Constant gain	9.8	2.6	1,852
Capped constant gain	5.6	1.3	13.8
Smooth cap	3.1	0.91	0.50
Constant log-odds effect	1.3	0.91	0.25
Headroom with a shift	13.7	19.0	6.1×10^5

The tests have different standing. The breakage rate $q = 0$ of the evidence-only harnesses and the positive control’s harm were preregistered (Section 4.8). The selection rule, the fit, and the cross-model test were run after the experiments, by the program’s audit of its record, so they are out-of-sample or retrospective rather than prospective. The matched-fraction tests (Appendix A.2) use comparisons run during the program and analyzed afterwards. The registered forecasts of Section 6.8 are the law’s prospective tests.

A.2 Separating the matched fraction from the conversion rate

The API-change sensor separates ϕ from p , because its instrument has a no-change score. On AppWorld’s Amazon tasks an undocumented change scaled amounts to cents. One order was paid 4020 against an emailed receipt total of \$40.20. Bare Claude Opus 5.5 fell from 94.3 with no change to 80.7, and the sensor’s flags for unanchored changes returned it to 93.9 on 212 tasks. The headroom was 19.3 points and the change caused 13.6 of them, so $\phi = 0.70$. The sensor recovered 13.2 of those 13.6, so $p = 0.97$, and $r = 0.68$. It recovered none of the 5.7 points Claude Opus 5.5 loses with no change at all, since nothing it supplies bears on them. When frozen sensor notes diagnose a change correctly, Claude Opus 5.5 scores exactly its no-change score; on 151 tasks it did so with 11 wins and 0 losses ($p = 0.001$).

The crash-recovery harness shows the same arithmetic across two readers. Told of a crash, Claude Opus 5.5 scored 97.7 on 87 tasks; untold, it scored 92.0. The notice, which says that each API call before the crash took effect, lets it rebuild its place: the notice alone recovered 5.7 of those 8.0 points, and the harness recovered 5.75, $r = 0.72$. Against the told control the harness recovered nothing, because the remaining 2.3 points are failures the crash did not cause. Claude Haiku 4.5 cannot rebuild its place from a notice. Told of the crash, it scored 63.1, below the 70.8 it scored untold. The harness restores the committed steps instead of describing them, and it raised Claude Haiku 4.5 to 92.3, recovering 29.2 of 36.9 points ($r = 0.79$). The harness is identical in the three comparisons; the fraction of each control’s failures that the crash caused runs from about none to most.

The law also explains why standard benchmarks cannot see what a harness adds to a frontier model. With the control at 97.7, the smallest effect detectable at 80% power on 87 tasks is 6.6 points, 2.9 times the whole headroom: near its ceiling a benchmark detects only what a harness breaks. This program therefore measures frontier harnesses where the signal lives, on instruments with headroom outside the episode and on instruments with no ceiling at all.

A.3 Regimes of the law

Equation (2) divides the instruments into regimes by where the causes of the control’s failures lie (Figure 2b).

Zero headroom. Where the model completes the task unaided, $h = 0$ and only q is measured. Bare Claude Opus 5.5 held S_{prod} at 1.000 on all six HORIZON streams, and Claude Opus 4.8, a successor strong enough to keep the service, held it too. On hourly AppWorld probes over 24 hours of continuous operation, each arm scored 0.99–1.00. Here a component must break nothing, and the evidence-only harnesses of Section 5.7 meet that requirement by construction.

Headroom the model can find. Where the model can find a failure’s cause inside the episode once told to look, the diligence instruction converts most of those failures. After a documented API change, bare Claude Opus 5.5 scored 91.3 on 61 post-change tasks with three repeats, a headroom of 8.7 points. The diligence instruction raised it to 96.7, +5.5 [2.2, 9.3] points and 0.76 of the best arm’s lift. The share was 0.71 on a second documented-change stream, 0.52–0.58 on the silent and mixed change streams, and 0.53 on the continuous-operation kernel probes. One undocumented change

could be detected from the API’s own responses, and there the instruction scored 240 of 240 positions, the same as the API-change sensor.

Headroom outside the episode. Where the cause lies outside the episode, no instruction reaches it. The instruction share falls to 0.14 when Qwen3.5-27B takes HORIZON’s account-balance service over. A successor cannot recover state it did not hold, a restarted process cannot recall effects it did not record, and one episode cannot reveal that an API’s meaning changed. On the handover, the commit rule, the guard, and the evidence-only supervisor recovered 0.79, 0.78, and 0.84 of the headroom (Table 11). On the 52-crash stream the recovery harness recovered 13.0 of 20.9 points of scenario completion, 0.62. On the silent pagination change the claim learner recovered 23.9 of bare’s 28.7 points, 0.83.

Headroom without a ceiling. Kernel speed, simulated cycles, and construction bounds have no ceiling; the headroom is the distance to kernels and constructions nobody has produced. The instruction share is lowest here: 0.04 on KernelBench with the technique library, and -0.12 on ALE-Bench heuristic contests, where the instruction scored 2,620 against bare’s 2,663 and ACE’s learned playbook 3,034. The components that recover this headroom supply a capability or a search: a verified technique the model did not use unaided, or hours of verified attempts shared among agents (Sections 5.4 to 5.6).

Negative share. Experience carried from a weaker predecessor supplies no cause the stronger reader lacks, so $p \approx 0$, and it can replace the reader’s own better approach, so $q > 0$. The four rows of Table 4 that serve Claude Opus 4.8’s store to Claude Opus 5.5 recovered between -0.33 and 0.13 of their headroom, and on KernelBench the store lowered Claude Opus 5.5’s speedup in all six serving modes (Section 5.6).

Rivals as controls. Against ACE or the CORAL port, the control has already recovered its own share. Equation (2) then gives the difference of two shares of the same headroom, $(k_L - k_{\text{rival}})h$, which has no slope in h unless the shares differ systematically. The 12 Claude Opus 5.5 comparisons on a 0–100 scale against ACE, transcript handoffs, and a self-written brief fit a slope of -0.06 . A component separates from a rival where it supplies an input the rival lacks. The technique library separated from ACE’s playbook by $+1.71$ because the playbook held no verified technique, and ACE given the cards reached $3.29\times$. The shared archive separated from the CORAL port by 34 cycles because its agents started each session from verified attempts, while 7 of the port’s 12 agents concluded the search was near its floor and stopped acting.

Both terms of Equation (2) set a design requirement. A component must be built for a cause that the control’s failures on the target instrument have, and it must hold $q = 0$ on the units the control already passes. Sections 5.4 to 5.9 are six ways of meeting it.

B Extended results: evidence-gated recursion

B.1 Level 1: EGR on components

Checks that rewrite themselves. Replaying 31,434 graded AppWorld attempts in order, Claude Opus 5.5 writes one Python check per task template from past pass/fail outcomes, with no grader text, and revises every check that missed a failure. The gate admits a revision only if it stays silent on the template’s past successes and catches at least as much as the check it replaces. Catch of never-seen failure states climbed from 7.3% in the first four windows to 83.4%, 94.7%, and 95.6%. The stock stopped growing at 48 checks in window 8; every gain after that came from revision alone, 43 maturations and 3 repairs, all written by the model. A copy frozen with every check still fell 10.6 points behind, and when a new actor population arrived the learning library held 87.6% while frozen copies fell to 44.4–68.7%.

Techniques the harness extends to new operators. The KernelBench library began with one card from the literature, an fp32-accurate GEMM on bf16 tensor cores by an error-compensated split. An engine that tested hypotheses against the protected evaluator then admitted five more cards, three of which carry the split to operator families the original does not reach: convolution, transposed convolution, and attention. On the 12 problems the seed card cannot touch, those cards lifted Claude Opus 5.5 by $+1.36$ [0.79, 2.10] in mean capped speedup, up on 11 and down on none.

Tools that outlive their authors and their models. On the shared archive, agents write their own optimizers, and the harness promotes a tool once an attempt that cites it verifies. Over the society’s first 6.7 hours, 38 of its 98 new bests ran or adapted a teammate’s tool. The certified Erdős bound came from a memetic genetic algorithm that one agent

wrote and another ran unmodified. When the take-home archive switched from Claude Opus 5.5 to GPT-6 Astra, the successors’ first new best ran an annealer a Claude Opus 5.5 agent had written, 14 minutes after the switch, and the successors set five new bests.

B.2 Level 2: EGR on harness designs

Automated evolution of an agent engine. The EVOLVE team built a loop that rewrites a whole agent engine. A proposer edits the incumbent engine, a fast screen ranks the candidates, and an admission rule accepts an edit only when it raises the protected Claude Opus 5.5 score on a frozen pool of optimization problems; an archive keeps every admitted engine as a parent for the next generation. The loop ran 12 generations per arm from a hand-built engine, and four of its engines were carried, frozen, to seven ALE-Bench problem families that no part of the evolution had seen, under the official scorer. On Claude Haiku 4.5 every evolved engine scored above ACE, 1,779 to 1,894 against 1,709, nearly double the hand-built engine it started from (956), and the best stood 592 points above bare. On Claude Opus 5.5 the best evolved engine scored 3,019, level with ACE’s 3,034, 356 above bare, and 399 above the diligence instruction. Evolved on one pool of problems, the engines carried their gains to problem families they were never shown.

Supervision, redesigned from its own failures. The program’s first supervision harness, built from verdicts, fell below bare Claude Opus 5.5 on every production stream where Claude Opus 5.5 worked each ticket. The agents traced all 24 of its first failures to its own machinery, wrote five invariants that remove the mechanism, and admitted the rebuilt harnesses only after a preregistered no-harm panel, with the old harness rerun as a positive control. The second generation had 0 of 2,880 tickets below bare and lifted a Qwen3.5-27B handover from 0.435 to 0.911 (Section 5.7).

Search organization, redesigned from measured losses. The first society beat CORAL and isolated agents, and the agents then measured where it lost ground to a single long-running agent: concentration of CPU, continuity of one conversation, and convergence of its agents on one line. They designed the lead-thread society to remove each loss and admitted it in a sealed run against its own CORAL and single-agent arms. It made the largest two-hour C_1 gain of any arm in either run and led both arms on all three problems from 8 hours to the end (Appendix D.6).

Verification, four generations deep. The verification component went through four generations, each designed after measuring the last. Frozen model-written checks caught 66.2% of 367 held-out failure states at no false fire. Maturing them on fresh data raised catch on novel failure states from 41.0% to 63.9%. The prequential library that revises itself after every window reached 90.1%. Seeding checks from correct attempts before any failure raised catch of a template’s first failure from 20.0% to 75.5% (Section 5.9).

B.3 Level 3: EGR on a running system, across a model upgrade

The clearest recursion happened inside one running system. The continuous-operation system streamed AppWorld, τ^2 -bench, coding, and KernelBench tasks for 44 hours, keeping a store of what its worker verified. Claude Opus 4.8 was the worker until hour 36, when the system handed the work to Claude Opus 5.5. The store was built for the model that wrote it, and the program found that it slowed the model that inherited it: every frozen snapshot from hour 8 to hour 36 lowered Claude Opus 5.5’s mean capped speedup, by 0.16 to 0.54.

The organization then ran EGR on the system’s own memory.

1. **Diagnose.** Randomized exposure in 1,288 episodes located the cause: the stored Triton recipes moved Claude Opus 5.5 off its faster CUDA kernels, from 43% of episodes to 13%, and backend and token-kernel status account for 78% of the loss (Section 5.6).
2. **Gate.** EGR’s gate admitted repairs only on the program’s no-harm test, and it kept out an evidence-yield filter that failed it.
3. **Supply what the model lacks.** A contrarian seat built evaluator-verified technique cards headed by the error-compensated bf16 split. In a preregistered test they took the hour-36 store from $1.73\times$ to $3.67\times$, $+1.94$ over the store alone and $+1.71$ over ACE’s own playbook.
4. **Deploy into the running system.** The library entered the live system at hour 36.16, minutes after the upgrade. On its live probe problems the system then ran $+0.70$ above bare, and by hour 42 its store lifted both readers above bare, Claude Opus 5.5 by $+0.76$ and Claude Opus 4.8 by $+0.90$.

In one run the system’s memory went from slowing its new model to lifting it. EGR diagnosed the cause, gated the repairs, and deployed the one that passed, without stopping the system.

B.4 Level 4: EGR on the harness the research ran on

At the top of the recursion, the program improved *Nenya* itself. frontier-bond assigned a team lead, frontier-a21, to build the program’s findings into the next release of *Nenya*, 0.11 alpha.21. Each hand-in was reviewed by an agent that did not build it, and the result was checked on a live cloud lane. What went in maps one to one onto the results of this paper:

1. act only on hard evidence: an unknown check outcome lets the work continue, and steering acts on evidence (Section 5.7);
2. restart clean: a restart replaces the session with a clean seed of goal, repository head, constraints, and fresh evidence, never the old session’s conclusions (Section 5.8);
3. carried knowledge delivered late and model-aware: each stored item carries the model that wrote it, and a model change sends it back to probation (Section 5.6);
4. a library that grants verified capability (Section 5.6);
5. the metamorphic verification instruction on the person’s API work (Section 5.8);
6. disagreement between attempts taken as evidence (Section 5.9), and a shared ledger of verified work (Section 5.5).

B.5 EGR on the research itself

The program also ran EGR on its own conclusions. The organization put a forensics team on its whole record, attributing every result to its causes (infrastructure, harness, ceilings, rivals, and statistical power) and flagging the conclusions most likely to move. That audit is where the headroom law was found. Across the program’s Claude Opus 5.5 comparisons, the lifts over the bare model or the diligence instruction fit a single share of headroom, and carried experience followed the strength gap between writer and reader. Fresh leads then reopened the conclusions the audit flagged. The reopened anchoring work found the mechanism and the remedy that removes it: delivered after the reader’s first checked attempt, the same store lifted Claude Opus 5.5 +0.106 above no memory. The law in Section 3 and the sign law in Section 5.6 are outputs of the research loop examining itself.

C Extended results: mathematical constructions

C.1 Problems and exact certificates

Erdős’s minimum overlap problem. Split $\{1, \dots, 2n\}$ into sets A and B of size n . Let M_k count pairs $(a, b) \in A \times B$ with $a - b = k$, put $M(n) = \min_{A,B} \max_k M_k$, and let $M = \lim_n M(n)/n$ (Erdős, 1955). Swinnerton-Dyer proved, in a result published by Haugland, that the limit exists and that

$$M \leq \sup_t O_h(t), \quad O_h(t) = \int_{\mathbb{R}} h(x)(\mathbf{1}_{[0,2]} - h)(x+t) dx, \quad (4)$$

for every measurable $h : [0, 2] \rightarrow [0, 1]$ with $\int h = 1$ (Haugland, 2016, 1996). A step function on n cells of width $2/n$ is admissible when its heights lie in $[0, 1]$ and sum to $n/2$. Haugland’s best function had 51 cells. The best lower bound is $M > 0.380552$ (Chung, Du & Wesley, 2026).

Autocorrelation constants. C_1 is the largest constant with $\max_t (f * f)(t) \geq C_1 (\int f)^2$ for every nonnegative f supported on $[-\frac{1}{4}, \frac{1}{4}]$, so each such f gives $C_1 \leq \|f * f\|_{\infty} / \|f\|_1^2$ (Cloninger & Steinerberger, 2017; Matolcsi & Vinuesa, 2010). Schinzel and Schmidt conjectured the value $\pi/2$, attained by $f_0(x) = (2x + \frac{1}{2})^{-1/2}$. Matolcsi and Vinuesa refuted the conjecture with a step function at 1.50972, and the best lower bound is $C_1 \geq 1.2937$ (Kim & Pilanci, 2026). C_3^+ is the same constant for real-valued f with $\int f \neq 0$, still measured by $\max_t (f * f)(t)$, where negative values can lower it. It is the constant of the corrected AlphaEvolve notebook, Einstein Arena problem 4, and MILO.

Unequal circles in a square. For Packomania’s sum-of-radii problem (csqv), place N circles with centers (x_i, y_i) and radii $r_i > 0$ in the unit square and maximize $\sum_i r_i$ subject to $r_i \leq x_i, y_i \leq 1 - r_i$ and $(x_i - x_j)^2 + (y_i - y_j)^2 \geq (r_i + r_j)^2$ (Specht, 2026).

Exact certificates. A step function turns each supremum into a finite maximum over knots (Lemma J.1), and a decimal in a construction file is an exact rational. Clearing denominators turns every certificate into a computation on integers. For C_1 the heights become integers A_i , and the condition of Lemma J.2 holds with $B = 2^{216}$. One exact GMP squaring of $X = \sum_i A_i B^i$ then recovers all 524,287 autoconvolution coefficients in 2.9 seconds, checked globally by three moment identities and locally by 18 direct sums. A second path uses number-theoretic transforms modulo seven primes below 2^{31} with the Chinese remainder theorem. For M , with $B_j = 2S - nA_j$, the bound is $\max_k \sum_i A_i B_{i+k} / (2S^2)$, evaluated by packed multiplication and again by direct summation at each shift. Packings are scaled to integers over a common power of ten, and each wall and pair inequality is decided with no square root and no tolerance. A grid of cell width $2 \max_i R_i$ provably misses no overlapping pair (Lemma J.3). Proofs are in Appendix J.

C.2 The certified values

The 325 values lie on the far side of each published value they share an instance with (Table 6): lower on each constant, higher at each packing size, and lower on the 14-point ratio. Three agents per problem on a shared archive of verified attempts produced a 3,584-cell construction for M after 10.82 working hours and a 262,144-cell construction for C_1 after 3.97.

Proposition C.1 (Erdős minimum overlap, shared archive).

$$\begin{aligned} M &\leq \frac{97499325154129259661263000479811180603}{256 \cdot 10^{36}} \\ &= 0.38085673888331742055180859562426242423046875. \end{aligned}$$

Certificate. The 3,584 heights lie in $[0, 0.999999999934562835]$, with 1,011 zero cells, and sum to exactly $1792 = n/2$, so $\int h = 1$. The overlap in (4) has a unique maximizing shift, $t = 869/1792$, where it equals the fraction above. No repair or rescaling is needed.

Proposition C.2 (First autocorrelation).

$$\begin{aligned} C_1 &\leq \frac{9540049024249904979463830078756700797267534524939751265402880000000}{6348434364442838324271548958669223827995209077143850398903628042249} \\ &= 1.50274043592276697640985 \dots \end{aligned}$$

Certificate. The 262,144 nonnegative heights include 106,416 zeros. Exact computation of all 524,287 autoconvolution coefficients (Lemma J.2) gives a unique maximum at $t = 0$.

The Erdős bound clears Einstein Arena entry #2507 by 1.84×10^{-6} , 18.4 times the minimum improvement, and lies 3.55×10^{-8} below MILO’s value. The C_1 bound was 309.6 times the minimum improvement below entry #2603 when it was found. The Arena’s newer leader, #2615 of October 13, still sits 3.05×10^{-6} above it, 305 times the minimum improvement. In the seven weeks before October 3, the Arena’s agents had lowered the C_1 record by 2.4×10^{-7} ; the shared archive went 3.1×10^{-6} below that day’s record in under four hours, 12.9 times as far.

A single agent goes lower on M . One agent working alone started from MILO’s published 512-point vector, which an exact trust-region LP polished back to MILO’s value. It repeated the function to 2,048 cells and ran barrier Newton-CG re-melt hopping, with migration across resolutions from 1,024 to 14,336 cells. A re-grid onto 2,560 cells, a resolution commensurate with neither parent, gave

$$M \leq 0.38085654787558049913 \dots,$$

with heights summing exactly to $n/2$ and no repair. That is 2.265×10^{-7} below MILO’s value and 2.03×10^{-6} below the Einstein Arena record, 20.3 times its minimum improvement; the independent exact checker reproduces it. With the Station’s lower bound of 0.380552, the constant now lies in an interval of width 3.05×10^{-4} .

Table 20: **The M and C_1 entries were equioscillated; the C_3^+ entry was not.** Knots within 10^{-8} of the maximum of the overlap or autoconvolution, for the Einstein Arena entries the searches started from and for the certified constructions.

Construction	Cells	Zero cells	Knots within 10^{-8}	Maximizing knot
M , Einstein Arena #2507	3,584		2,425 of 7,167	
M , this paper	3,584	1,011	2,291 of 7,167	$t = 869/1792$
C_1 , Einstein Arena #2603	65,536		24,830 of 131,071	
C_1 , this paper	262,144	106,416	1 of 524,287	$t = 0$
C_3^+ , Einstein Arena #2609	65,536		1 of 131,071	
C_3^+ , this paper	102,400	0 (29,042 negative)	80,568 of 204,799	index 149,850

A compute-only control stalls at the stationary point. The constructions were produced inside a preregistered matched-hours study whose control arm runs no language model. Its 94 worker processes, on a machine of the same type, apply the classical toolbox to the same public constructions: log-sum-exp annealing under L-BFGS, sequential linear programming on the exact minimax, upsampling, perturbation restarts, multistarts, and an exact prox-linear method. A value counts as the agents’ only if it beats both this control and a single-agent arm at equal elapsed time. The control lowered the M and C_1 entries by 5.8×10^{-10} and 1.05×10^{-9} and stalled there; its best on M , 0.380858574281728, was unchanged from before the shared-archive run began. The agents went 1.836×10^{-6} and 3.096×10^{-6} below the same entries, 3,187 and 2,938 times as far. Both values satisfy the preregistered attribution rule.

The control’s toolbox already includes upsampling and log-sum-exp annealing. It has no move that changes the active set, and the entries it started from had 2,425 and 24,830 knots within 10^{-8} of the maximum (Table 20). At an annealed Erdős optimum the agents found the top ten shifts within 2×10^{-11} of each other. Random exact pair moves lower the maximum in 150 of 200 trials, but each gains at most the gap to the next shift. An improving direction must lower about 2,000 active shifts at once, and local polish gains 10^{-14} to 10^{-10} ; the control’s exact prox-linear method moved the two entries by 5.3×10^{-10} in a minute and 3.1×10^{-10} in 143 seconds.

The third constant tests the account from the other side. The Arena’s C_3^+ entry was not equioscillated, with one knot within 10^{-8} and 56 within 10^{-6} . Smooth descent had room, and the control moved it 5.94×10^{-6} in half an hour before stalling as well. Newton-CG run on that entry stalls at 1.4491147, and the control stood at 1.4491144 after 12 hours, so that basin was exhausted. The agent left the basin through MILO’s block means and went 2.39×10^{-4} below the entry. On all three constants the control reached the floor of its starting basin, and going below that floor required an escape move.

C.4 Escape moves

The agents wrote every search themselves; no language model runs inside the numerical inner loops, and no weights are updated. Their searches share three operations. The *log-sum-exp continuation* smooths the maximum,

$$\text{LSE}_\beta(h) = \frac{1}{\beta} \log \sum_k e^{\beta C_k(h)}, \quad \max_k C_k(h) \leq \text{LSE}_\beta(h) \leq \max_k C_k(h) + \frac{\log K}{\beta},$$

and is minimized with L-BFGS-B over the box, an augmented Lagrangian for the mass constraint, and FFT gradients, raising β by a fixed factor per stage. At small β the surrogate weights the shifts nearly equally; at large β it concentrates on the active ones. A *re-melt* restarts the continuation from a small β after a perturbation, so the active set dissolves and re-forms. A *refinement* repeats each height m times, which turns an n -cell function into the same function on mn cells; the score is unchanged, and the finer grid opens descent directions the coarse one lacked. On these foundations the agents built five escape moves.

Escape 1: break a mirror-symmetric saddle along its unstable modes (M). Reflection $x \mapsto 2 - x$ maps admissible functions to admissible functions and preserves the bound, so the mirror-symmetric functions form an invariant subspace. Where the overlap equioscillates at $\pm k$ over an interval of shifts, the function must be mirror symmetric there (Lemma J.4), so asymmetry can live only elsewhere. The agents measured that the symmetric optimum at 1,792 cells is poor, 2×10^{-5} above the Arena entry, and that it is a saddle: the Hessian of LSE_β , restricted to antisymmetric directions, has 75 to 85 negative eigenvalues for β between 10^4 and 5×10^7 . The *symmetrize-then-break* iteration

optimizes inside the symmetric subspace, perturbs along antisymmetric directions, re-anneals without the constraint, and repeats from the new best. Draws along the unstable eigenvectors go lower than random noise. From the same symmetric parent, random antisymmetric noise ended between 0.3808584 and 0.3808586, while eigen-direction draws had median 0.3808579721. One agent’s tool note records the rule it learned: ‘Run all modes: mode quality is not predicted by the eigenvalue.’

Escape 2: cross independent lineages (M). The last stage was a memetic genetic algorithm that one agent wrote and another ran unmodified. Its pool holds annealed optima at 1,792 cells; children come from crossover, blends, and mirrored parents, each re-melted from $\beta = 5 \times 10^4$ to 2×10^5 . Partial-symmetrization restarts $h_0 = s h_{\text{sym}} + (1-s) h_{\text{best}}$, with $s \in \{0.25, 0.5\}$, supply independent parents of near-best quality. The certified construction is a uniform-block crossover of a refined lineage best, with 268 near-active shifts and 545 cells pinned at $h = 1$, and a separately evolved solution upsampled with another agent’s annealer. A re-anneal from low β smoothed the joins. The child has no cells at $h = 1$ and 326 near-maximal shifts, its maximizing shift moved from -0.471 to -0.485 , and the bound fell 1.19×10^{-8} below the lineage’s previous best.

Escape 3: refine, then re-melt from a soft temperature (C_1). One agent’s line of constructions reached 131,072 cells after 3.1 working hours. A second agent fetched it from the archive, upsampled it two-fold to 262,144 cells, and ran a log-sum-exp ramp from the soft temperature $\beta = 10^3$, multiplying β by 1.035 per step, with an evaluation kernel adapted from a third agent’s tool. The certified construction is the ramp’s best snapshot, taken near $\beta = 3 \times 10^7$. The bound therefore combines three agents’ work in one optimization run.

Escape 4: reconstruct a published coarse function at full resolution (C_3^+). MILO’s project page publishes two 3,200-number vectors. The agent showed that they are 8-cell block means of 25,600-point functions; the start vector equals the rounded block means of the Arena entry #2492 in all 3,200 positions. The means alone score 1.6558, so any gain must come from structure inside the blocks. The agent searched over functions that share the means, $h = \text{up}(m) + u - \text{up}(\text{blockmean}(u))$, and polished with an annealed log-sum-exp Newton-CG method it wrote, whose Hessian-vector products are exact FFT convolutions:

$$\nabla^2 \text{LSE}_T(h) v = \left(\frac{1}{T} J^\top (\text{diag } w - w w^\top) J + 2 \text{Hankel}(w) \right) v.$$

Here $T = 1/\beta$, w are the softmax weights of the knots, and $Jv = 2(h * v)$ is the derivative of the convolution coefficients; the Hankel term collects their second derivatives, $\partial^2 c_k / \partial h_i \partial h_j = 2 \cdot \mathbf{1}[i + j = k]$. Batched re-melt hopping then took a parent from a migrating population, optionally jittered or crossed it, re-melted it at $T = 10^{U(-5.5, -2)}$ of the maximum, and annealed it to 10^{-10} of the maximum, on 10–18 A100 GPUs, raising the resolution from 25,600 to 51,200 to 102,400 cells. The first Newton-CG polish took the reconstruction from 1.4491688 to 1.4490861 in 85 seconds, already below the Arena entry. The agent then used resolution itself as an escape move. Remapping the population to 100,000 cells destroyed its grid-scale structure (score 1.4909), and hot re-melts recovered and passed it. A remap to 99,000 cells, hopped in isolation for 30 minutes, finished 1.1×10^{-6} below the pool it came from, and a third re-grid, to 97,000 cells, gave the lowest value.

Escape 5: re-seat a defect, then relax the whole packing (csqv). The packing agent assembled a pipeline from standard global-optimization components (Addis, Locatelli & Schoen, 2008; Grosso et al., 2010; Wales & Doye, 1997): penalty relaxation, sequential LP on linearized distances with feasible monotone iterates, and Newton’s method on the KKT system of the active set (Algorithm 4). Its own contributions were the diagnosis and the move. Above $N = 1600$, Liang’s packings were KKT points: sequential LP and Newton polishing gained less than 10^{-9} . At an isostatic point the active set fixes the configuration, so a continuous method can only move along it, and each local polish returns to the same contact graph. The *defect move* supplies a new contact graph. It picks a circle whose radius is far from the median, which in a hexagonal array marks a defect, re-randomizes the disc of circles around it, and relaxes the whole packing with the penalty method before a Newton solve on the new overlap set. The first move at $N = 1681$ gained 1.0×10^{-3} . Within 20 minutes, 46 large sizes had passed the published value, and 58 by the end of the run.

The move requires the global relaxation. Region-restricted moves, which re-optimize only the disturbed disc, were 30 to 100 times cheaper and produced no improvement. Above $N = 600$, region-only search found one improvement in 80 minutes, and switching global moves back on found 20 of 24 sizes. A defect changes the strain of the whole array, and only a global relaxation lets the array re-register around it. A transfer cascade seeds size N by deleting a circle from the best $(N+1)$ -packing or inserting one into the 16–24 largest holes of the best $(N-1)$ -packing; one round

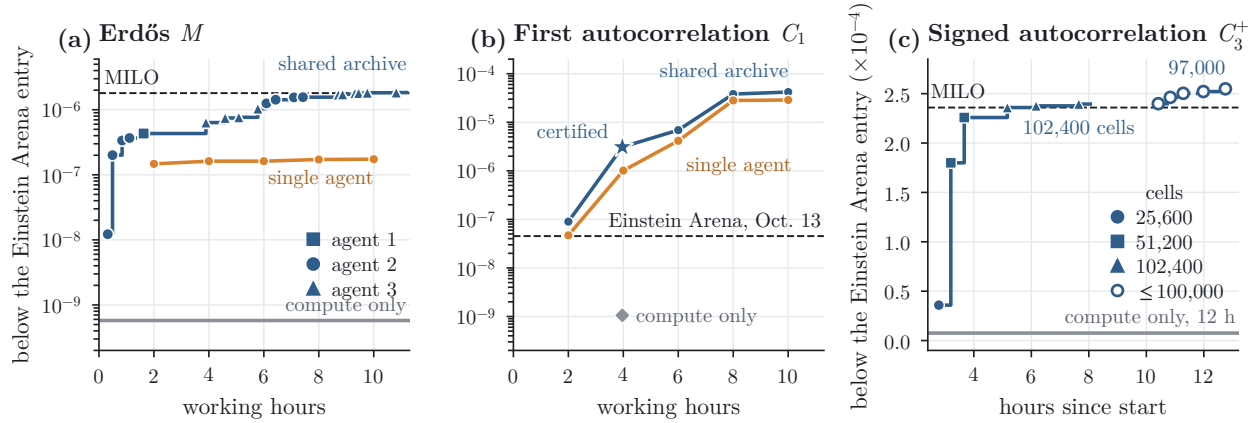


Figure 8: **The searches descended in steps, each step a change of active set.** (a) The 19 archive attempts on the lineage of the certified Erdős construction, as distance below Einstein Arena entry #2507, against working hours; marker shape gives the agent that posted each step. The single agent is shown at matched hours, and the compute-only control's best was unchanged throughout. (b) C_1 : shared-archive and single-agent best-so-far at matched hours, below Arena entry #2603; the star is the certified construction and the diamond the control at the same elapsed time. (c) C_3^+ : the single agent's path, below Arena entry #2609, by resolution; open markers are the continuation below the Arena's 100,000-value input limit. Archive values are the protected evaluator's conservative floating-point scores; the final points of (a) and (c) and the star are exact.

added 52 new sizes. In transfer moves at $N = 300$, a direct KKT solve took 206 seconds against 641 for LP polishing and reached the same best; on the 154 hardest unclaimed sizes, in 60 minutes, it found 12 new sizes against 3.

Second-order finishes. Each escape move ends in a second-order solve, and the choice of solver changed how far each move went. On C_3^+ , Newton-CG went about 8×10^{-5} lower than L-BFGS and an augmented Lagrangian polish from the same start. On M , a four-fold refinement with slow cooling went 3.7×10^{-8} lower than the two-fold route, and it cut the near-maximal shifts from 355, an artifact of the coarse grid, to 11.

The descent ran in phases. The Erdős lineage (Figure 8a; Table 34) ran in four phases, and each phase ended when its move stopped changing the active set. Coarse-to-fine re-annealing of the public entry went 4.3×10^{-7} below it in 1.6 hours. The symmetrize-then-break iteration reached 1.04×10^{-6} by 5.8 hours, and eigen-direction breaking, posted by a second agent, 1.56×10^{-6} by 7.4. The memetic genetic algorithm, run by the third agent on the second agent's code, crossed MILO's value at 9.45 hours and reached the certified value at 10.82. The matched-hours single-agent arm of the same study, in one long conversation, stayed near 1.6×10^{-7} below the entry from 2 to 10 hours. On C_1 the shared archive stood 9.0×10^{-8} below #2603 at 2 hours and 3.10×10^{-6} at the certified construction, and its descent continued: its best reached 4.19×10^{-5} below the entry by 10 hours on the protected evaluator's conservative scores.

C.5 Structure of the extremal objects

The Erdős function (Figure 9a) vanishes near both ends of $[0, 2]$, on 383 and 384 cells; its support is $[0.2137, 1.7857]$. Inside the support sit a central body on $[0.420, 1.580]$ and two side lobes, $[0.228, 0.361]$ and its mirror, separated by zero gaps of 106 cells. Both box constraints are active: of the 3,584 cells, 1,011 are zero and 586 lie within 10^{-6} of the ceiling $h = 1$ without touching it. The overlap is flat to 10^{-8} across 2,291 of 7,167 shifts, and Lemma J.4 reads that flatness back into the function. Outside $|x - 1| < 0.3$ the mirror difference $|h(x) - h(2 - x)|$ averages 5.0×10^{-4} per cell; inside, 8.4×10^{-2} . The construction is symmetric where the overlap equioscillates and asymmetric in the center, as the identity predicts.

The C_1 function (Figure 9c) is a comb with heavy ends. Its zero cells form 2,469 gaps, the widest on $(0.150, 0.224)$. The outer 5% at each end carries 22.8% of the mass and holds 1,422 of the 1,468 cells above eight times the mean. The largest cell, 104 times the mean, is the first, at $x = -\frac{1}{4}$: the endpoint singularity of the conjectured extremal f_0 survives in the step function as this spike. The C_3^+ function works by cancellation. Its negative cells carry -0.56 times the net integral, and its autoconvolution dips to -5.83 times $(\int f)^2$ near $t = 0.18$. At the coarse scale it is MILO's

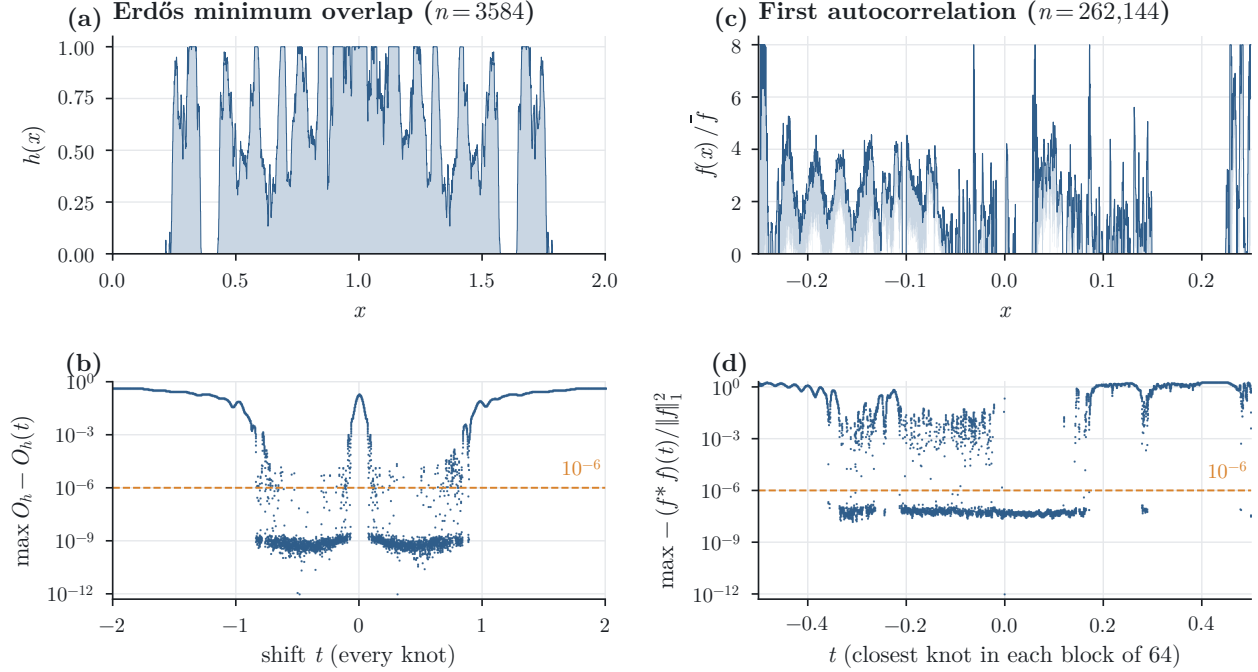


Figure 9: **The certified Erdős and C_1 functions and their gaps from the maximum.** (a, b) The Erdős function on $[0, 2]$ and its gap from maximum overlap at all 7,167 knots: 2,407 lie within 10^{-6} and 2,291 within 10^{-8} . (c) The C_1 function on a mean-one scale, shown as an envelope over blocks of 128 cells and clipped at 8; 1,468 cells exceed 8 and the largest is 104. (d) The autoconvolution gap, taking the closest knot in each block of 64. Of 524,287 knots, 125,244 lie within 10^{-6} of the maximum; only $t = 0$ lies within 10^{-8} .

function, with 32-cell means correlating 0.998 with MILO's; at the scale of the Arena entry it is a different function, with correlation 0.365.

The large packings are hexagonal arrays held at their walls (Figure 10). At $N = 137, 529$, and 1681 the number of active constraints at tolerance 10^{-9} is exactly $3N$, so the packings are isostatic. At $N = 10,000$, 9,476 circles touch exactly six neighbors or walls, 93.7% of the radii lie within 2% of the median, and 337 circles, the defects, differ from it by more than 10%. The arrangement follows from the objective. For a fixed covered area $\pi \sum r_i^2$, Cauchy-Schwarz gives $\sum r_i \leq (N \sum r_i^2)^{1/2}$, with equality only for equal radii, and equal circles at the hexagonal density $\pi/(2\sqrt{3})$ give the *hexagonal estimate* $\sqrt{N/(2\sqrt{3})}$. The new packings fall short of it by 0.103 to 0.134 at all 306 sizes while N grows 73-fold. The deficit is a boundary cost: a layer of width proportional to r along a perimeter of length 4 displaces a number of circles proportional to $1/r$, each of radius r , so the cost in $\sum r_i$ is a constant. At large N the margin is the published packing's excess deficit over that constant (Table 21). The published packings at $N = 5041, 7396$, and 9801 sat 0.170 to 0.176 below the hexagonal estimate, against a boundary cost near 0.12, and the defect moves removed most of that excess.

C.6 Two further geometry results

Fourteen points in $\mathbb{R}^3$. Minimize $(d_{\max}/d_{\min})^2$, the squared ratio of the largest to the smallest pairwise distance among 14 points. AlphaEvolve found approximately 4.165849767 (Novikov et al., 2025), and FICO Xpress improved this to 4.165783988278149 (Berthold et al., 2026). Polishing FICO's published coordinates gives the exact ratio

$$(d_{\max}/d_{\min})^2 = 4.165783474580073316882516812166805 \dots,$$

a reduction of 5.14×10^{-7} , with 30 minimum and 9 maximum distances. All 90 basin-hopping chains from random starts converge to this configuration.

Container packings. A sub-agent of the packing agent carried the search to Friedman's Packing Center (Friedman, 2026). It maximized the sum of radii of n circles in a regular pentagon of side one and in a rectangle of perimeter four.

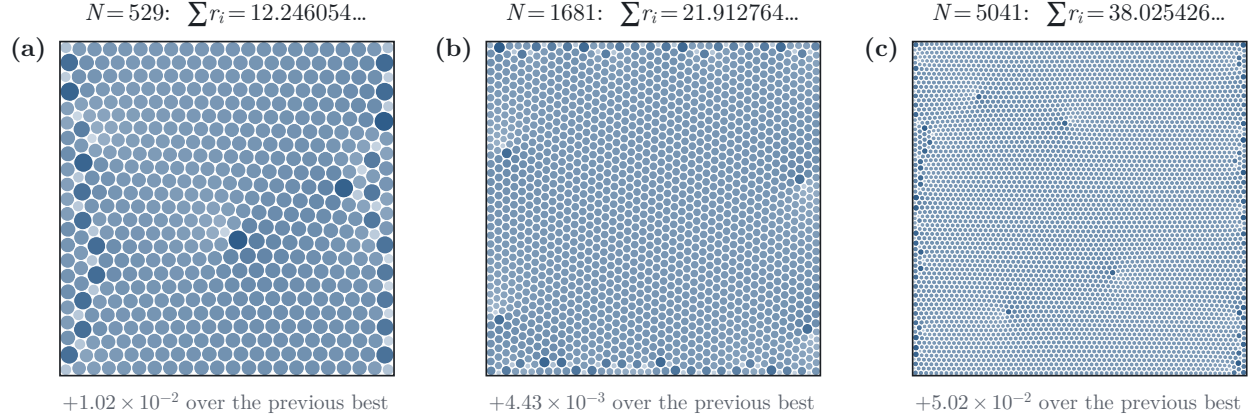


Figure 10: **Three new packings and the defects the search exploits.** Second-run packings at (a) $N = 529$, (b) $N = 1681$, the smallest of Packomania’s large sizes, and (c) $N = 5041$, shaded by radius. Hexagonal arrays meet wall circles, grain boundaries, and isolated defects. The search re-seats the circles around a defect, then relaxes the whole packing (Algorithm 4).

Table 21: **The deficit to the hexagonal estimate is a boundary cost, and the margin is the published excess over it.** Hexagonal estimate $\sqrt{N/(2\sqrt{3})}$, deficits of the previous best (Liang) and of ours, and the margin, which is their difference.

N	Hexagonal estimate	Deficit, previous best	Deficit, ours	Margin
137	6.2888	0.1087	0.1085	1.78×10^{-4}
529	12.3576	0.1217	0.1115	1.02×10^{-2}
1681	22.0287	0.1204	0.1159	4.43×10^{-3}
4096	34.3862	0.1297	0.1163	1.34×10^{-2}
5041	38.1472	0.1720	0.1218	5.02×10^{-2}
9216	51.5794	0.1064	0.1062	1.63×10^{-4}
9801	53.1912	0.1756	0.1248	5.08×10^{-2}
10000	53.7285	0.1252	0.1189	6.22×10^{-3}

Its packings exceed each value consistent with Friedman’s displayed records at 13 pentagon sizes ($n = 21, 23, 26, 29, 33, 34, 35, 37, 38, 43, 45, 49$, and 50), by 2.4056×10^{-5} to 8.4×10^{-3} , and at two rectangle sizes, $n = 39$ and 42 , by 1.2906×10^{-3} and 1.5203×10^{-4} . Each is exactly feasible as written; the pentagon’s irrational wall normals are decided with outward-rounded integer intervals at scale 10^{100} .

C.7 One principle across problems

The five escape moves of Appendix C.4 were invented by the agents inside the runs, and each instantiates one principle. A published construction is a stationary point of its discretized problem, and the way past it is a move that changes the active set, or the discretization that defines it, followed by a global re-melt and a second-order finish. The principle travels between problems.

- **Incommensurate re-gridding** set the lowest bounds on two different constants. One agent remapped its C_3^+ population to 100,000, 99,000, and 97,000 cells and reached $C_3^+ \leq 1.4488670313429 \dots$. The same agent re-gridded an Erdős function onto 2,560 cells, a resolution commensurate with neither parent, and reached $M \leq 0.3808565478755 \dots$, 2.03×10^{-6} below the Einstein Arena record.
- **Defect moves** were diagnosed on Liang’s isostatic KKT packings and lifted 58 of the 60 largest Packomania sizes. A sub-agent carried the same pipeline to new containers, regular pentagons and rectangles, where its packings exceed the values consistent with Friedman’s displayed records at 15 sizes.
- **The error-compensated bf16 split** was carried by the card engine from matrix multiplication to convolution, transposed convolution, and attention, and lifted 11 of the 12 problems the original card cannot reach (Section 5.3).

- **Symmetry breaking along unstable modes.** The agents found that the mirror-symmetric optimum for M is a saddle with 75 to 85 unstable antisymmetric modes, and recorded the rule they learned: ‘Run all modes: mode quality is not predicted by the eigenvalue.’

In each case the move reached a basin that the published construction never visited, and a preregistered classical optimizer started from the same constructions did not.

C.8 The boundary law in detail

Write

$$\delta_N = \sqrt{N/(2\sqrt{3})} - \sum_i r_i$$

for a packing’s deficit to the hexagonal estimate. At every one of the 306 sizes, the new packings have δ_N between 0.1025 and 0.1337, a band 0.031 wide, while N grows 73-fold; the boundary-cost argument of Appendix C.5 predicts a constant deficit, because the wall layer’s cost in $\sum r_i$ does not grow with N . The previous best entries had deficits up to 0.1756.

Table 22 tests the band as a forecast. Fitted only on the 248 sizes below $N = 1681$, the band has mean 0.1085 and range 0.1025–0.1337. All 58 larger sizes landed inside that range, and the band mean forecasts their deficits with mean absolute error 0.0060, against 0.0099 if the previous entries had stood; because a margin is the previous deficit minus the new one, the forecast margins carry the same errors. The fit is tighter than the range: 22 of the 58 large deficits fall inside the central 5–95% interval of the small sizes, 0.1041–0.1118, and the mean deficit rises from 0.1085 below $N = 1681$ to 0.1143 above it (0.1127, 0.1144, and 0.1154 over $N = 1681$ –3,000, 3,000–6,000, and 6,000–10,000). A trend in $\log N$ fitted on the small sizes forecasts the large ones as well as the band mean (mean absolute error 0.0060), so a constant limit and a slowly growing deficit both fit these sizes. The new deficits vary three times less than the previous entries’ at large N (standard deviation 0.0051 against 0.0149), so where a published entry is weak can be read from its deficit before any search runs.

A correlation between forecast and achieved margins is not evidence for the band: both contain the previous entry’s deficit, so the correlation is 0.967 whatever constant replaces the band mean, and 0.948 on average when the new deficits are shuffled across sizes.

Table 22: **The deficit band as a forecast.** The band is fitted on the 248 sizes below $N = 1681$ alone and tested on the 58 larger sizes. Errors are in the deficit δ_N , which equal the errors of the implied margins. Computed from the exact sums of Table 37.

Quantity	Value
Deficit δ_N of the new packings, all 306 sizes	0.1025–0.1337
Deficit of the previous best entries, all 306 sizes	0.1025–0.1756
Band fitted on $N < 1681$: mean; range	0.1085; 0.1025–0.1337
Large sizes ($N \geq 1681$) whose new deficit falls inside that range	58 of 58
Large sizes inside its central 5–95% interval (0.1041–0.1118)	22 of 58
Mean deficit of the new packings, $N < 1681$ and $N \geq 1681$	0.1085 and 0.1143
Standard deviation of the deficit at $N \geq 1681$, previous and new	0.0149 and 0.0051
<i>Mean absolute error of forecasts of the new deficit, $N \geq 1681$</i>	
Band mean, 0.1085	0.0060
Trend in $\log N$ fitted on $N < 1681$	0.0060
No improvement (previous deficit)	0.0099
Mean of the large sizes themselves (in-sample reference)	0.0043

D Extended results: shared verified archives

D.1 The archive, the society, and the return

The shared archive. The archive is an append-only event log, chained by HMAC, of verified attempts with content-addressed sources. An attempt enters only after a protected evaluator scores it, with its score, its parent, the items

it cites, its source, and the tool that produced it. It also holds the insights, failure constraints, and tools that agents cite. Reuse requires that an attempt be executable by an agent other than its author, so the archive stores the source of each attempt and the code of each tool. An agent copies another agent’s construction into its own workspace with one command, `kx source`, and runs a promoted tool from the shared directory. On the take-home, four agents used the archive with compiled insights, failure constraints, re-scored new bests, and a tool library that promotes a proposed tool once an attempt citing it verifies. Parents are chosen by *clade credit*, which rewards an attempt for its descendants’ verified gains, and each session starts fresh from a brief drawn from the archive.

The society. On the open problems, a later version of the same harness, the society, ran three agents per problem. Sessions last at most 4 hours, so each fresh session reads the team’s progress from the archive, and each follows a fixed protocol (Algorithm 2). The harness assembles a briefing of at most 9,000 characters: the team best, a parent drawn by Thompson sampling over clade credit, a direction rotated each session, and bounded lists of insights, recent results, constraints, tools, and offers from other problems. The agent writes and runs optimizers in its sandbox and submits candidates. The evaluator scores each candidate in a separate process, and the archive appends the attempt chained to the previous event. The harness re-scores new bests and invalidates attempts that rest on a refuted attempt. It promotes a tool or insight when an attempt citing it verifies, and it announces a teammate’s new best at the agent’s next submission. On the take-home, an agent resuming its session at 7.7 hours was told that the team best stood at 915 cycles, set by agent 3, where its source file was, and that agent 3’s new annealer had entered the tool library since the session began.

Definitions. A protected verifier scores a candidate x as $s(x)$ or rejects it. For harness H , let t_a be the working time at which verified attempt a was recorded.

Definition D.1 (Advantage). The best-so-far curve of H is $B_H(t) = \min\{s(a) : t_a \leq t\}$ (a maximum for a maximization task). The advantage of H over H' is $\Delta_{H,H'}(t) = B_{H'}(t) - B_H(t)$ for minimization and $B_H(t) - B_{H'}(t)$ for maximization, so that $\Delta > 0$ when H is ahead.

Definition D.2 (Time-dependent return; crossover time). For a sharing harness S and another harness H' , the time-dependent return to sharing is the function $t \mapsto \Delta_{S,H'}(t)$ over the covered hours. The crossover time $t_\times(S, H')$ is the earliest covered t such that $\Delta_{S,H'}(u) > 0$ at every covered $u \geq t$. When the arms have more than one seed, Δ is read at the least favorable pairing: the worst seed of S against the best seed of H' .

Definition D.3 (Cross-agent reuse; coordination share). A new team best recorded by agent i reuses another agent’s work if its producing data flow contains an attempt by another agent j , as parent or ancestor, or code from a tool of j , run or adapted. The reuse rate is the share of new bests that do so, and the build-on delay is the time from a teammate’s new best to the first attempt that starts from it. The coordination share is the share of an agent’s tool calls spent reading and writing the archive.

Remark D.4 (Maximum against composition). The isolated arm’s curve is $B_{\text{iso}}(t) = \min_i B_i(t)$ over its k agents, the best of k independent searches, none of which can start from another agent’s attempt. On the archive any verified attempt can be a parent, so improvements found by different agents can enter one lineage. The two curves coincide while all agents find the same improvements, because combining identical improvements gains nothing. They separate once the remaining improvements are small and distinct. The best of k searches then grows with its best member, and a composition grows with all of them, later by the build-on delay.

Sharing can pay only through cross-agent reuse, while the coordination share is spent whether or not reuse happens. A benchmark reports Δ at one t ; the time-dependent return is the whole function, and $t_\times$ is the hour after which its sign stays positive.

D.2 The take-home race

Anthropic’s take-home asks for a kernel for a simulated VLIW SIMD machine, scored in clock cycles (Anthropic, 2026). The starting kernel takes 147,734 cycles. The main comparison launched four agents per arm and three seeds per arm together on October 7, 2026, each arm on its own n2d-highcpu-48 machine for 8 hours, every agent running Claude Opus 5.5 through Claude Code at medium effort.

At 8 hours the archive’s median was 914 cycles, against 946 for the best of four and 948 for CORAL-faithful (Table 7). Its advantage was 32 cycles over the best of four, from 5 to 37 over the nine seed pairings, and 34 over CORAL-faithful,

from 9 to 37. Two margins were registered before any attempt was scored, at least 10 cycles below CORAL-faithful and at least 20 below the best of four, and both held. Every archive run beat every baseline run: on that hardware class the five archive runs ended at 910–916 cycles and the nine CORAL-faithful and best-of-four runs at 921 or above ($p = 0.008$ against CORAL-faithful, $p = 0.004$ against the best of four). Each kernel passed the official suite in a fresh clone, 64 fresh random problems, and 607 stress problems with edge-case and full-32-bit inputs. A single Claude Opus 5.5 agent on the same machine type ended at 953. CORAL’s own paper reports 1,103 cycles with Claude Opus 4.6; the archive’s 914 is 189 cycles below it.

Behavior of the CORAL-faithful port. CORAL-faithful shares attempts, notes, and skills in a directory any agent can write, and it resumes an agent that exits early in the same conversation with its latest score and the word ‘continue’. Agents of the port concluded that about 951 cycles was near the floor, and they answered each resume in about 6 seconds without acting, hundreds of times. Seven of the port’s 12 agents entered that loop, and no shared-archive or isolated agent did. The isolated agents, with no loop, still finished 32 cycles behind the archive, so the stall appears in independent search with or without the resume loop. The CORAL port shares as well, so the margin requires that what is shared be verified. A resumed conversation carries forward an agent’s conclusion that the search is over; a fresh session briefed from the archive starts from the verified attempts.

D.3 The crossover on the take-home

The return to sharing has a shape, and the race measures it. From 165 minutes the worst archive seed was ahead of the best seed of both other harnesses at every checkpoint to the end of the run. Read at the least favorable pairing, the crossover time of the main comparison is 2.75 hours against both CORAL-faithful and the best of four. The archive’s advantage then grew to 18 cycles over the best of four at 4 hours and 32 at 8. The same shape held in every configuration with hourly readings, across machine types, model mixes, and a seeded compiler: at 8 hours the archive led in all seven comparisons, by 12 to 54 cycles. A benchmark read at 2 hours, when every harness is still taking the same early gains, sees none of this. The archive’s value lies in the hours after the plateau, which is where long-horizon search is decided.

The opening improvements are large and redundant, as Remark D.4 predicts. In one run the first two attempts came from two different agents in the same minute, and both were vectorized rewrites of the scalar kernel; by 30 minutes each seed of the main comparison was more than 129 times faster than the starting kernel. The late improvements are small and distinct. In one run the team best moved only from 952 to 946 cycles for 200 minutes, then fell to 911 in 13 steps over the next 203 minutes, set by three of the four agents. Agent 4 restaged the tree’s node levels as four-word pair blocks and moved the blocks onto agent 3’s kernel. Agent 3 ported agent 4’s scheduler exactly into C and re-annealed agent 4’s kernel with it, and agent 4 then ran that C scheduler, as a tool, on its own kernel. Agent 2 ported its store-engine transposition of pair-block children onto agent 4’s kernel. Agent 3 seeded each address chain with a select between two constant vectors, moving vector multiply-adds onto the idle flow engine, and took the last three steps to 911. Five of the 13 steps took another agent’s kernel as parent or ran another agent’s tool. The lineage of each of the 13 contains kernels by at least two agents, and no step exceeded 6 cycles.

A switch to GPT-6 Astra. The same run switched all four agents from Claude Opus 5.5 to GPT-6 Astra at 8 hours, with the team best at 911 cycles. Each successor opened on a 29,828-character briefing that held the team best with its per-engine utilization, an assigned starting attempt, and the verified insights ranked by gain. The successors made 7 attempts in 2.85 hours, all correct, and set five new bests down to 906. Fourteen minutes after the switch, a successor took the 911-cycle Claude Opus 5.5 kernel, reassociated an XOR at the tree’s wrap so that two operations run in parallel, and re-annealed the result with the annealer a Claude Opus 5.5 agent had written over an exact C port of the scheduler, reaching 910. It then built an individual-operation search on that annealer (909), moved all 184 address-seed selects onto the flow engine (908), and extended the selects to the first pass of the tree (907). Another successor combined a third successor’s register packing with that 907-cycle kernel and a transpose buffer, reaching 906. One verified record carried a search across model families.

D.4 Open problems

On Erdős’s minimum overlap problem, C_1 , and AtCoder Heuristic Contest 026 (AHC026, ‘Stack of Boxes’ ([AtCoder, 2023](#))), all four harnesses ran Claude Opus 5.5 at effort xhigh with web tools disabled, each arm on its own n2d-highcpu-96 machine with equal CPU weight per problem. Every arm received the same public material, including the record

Table 23: **Best score at 2, 4, and 8 matched working hours.** C_1 and Erdős: fast-evaluator margin below the Einstein Arena record at launch ($\times 10^{-6}$). AHC026: development score on 150 generated cases. Higher is better; bold marks the best arm at each hour.

Arm	$C_1 (\times 10^{-6})$			Erdős ($\times 10^{-6}$)			AHC026 (dev)		
	2 h	4 h	8 h	2 h	4 h	8 h	2 h	4 h	8 h
Society	0.09	3.10	38.12	0.431	0.635	1.602	1,430,480	1,431,992	1,433,414
CORAL-faithful	0.45	3.88	16.21	0.539	0.624	0.904	1,428,178	1,430,688	1,431,607
Isolated agents	0.31	0.42	3.51	0.315	0.500	0.558	1,429,217	1,430,552	1,431,266
Single agent	0.05	1.01	28.17	0.147	0.162	0.172	1,426,691	1,428,684	1,431,767

Table 24: **Crossover times $t_\times$** (Definition D.2) in matched working hours, on a 0.25-hour grid. After $t_\times$ the sharing arm is ahead at every quarter-hour reading to the end of coverage. The lead-thread society ran later, against its own run’s CORAL-faithful arm and single agent (Appendix D.6).

Sharing arm	Compared with	C_1	Erdős	AHC026	Covered to
Society	CORAL-faithful	6.5	4.0	1.0	37.25–38.75 h
Society	Isolated agents	2.5	4.0	1.0	22.75–24 h
Lead-thread society	CORAL-faithful	8.0	1.5	7.0	14.5 h
Lead-thread society	Single agent	1.25	7.75	2.25	14.5 h

constructions. C_1 and Erdős scores are margins below the Einstein Arena record at launch; AHC026 scores use the 150 fixed development cases on which agents received feedback.

At 8 matched working hours the society was ahead of CORAL-faithful, the isolated agents, and the single agent on all three problems (Table 23), meeting the prediction registered before the CORAL-faithful and isolated arms launched. Its advantage over CORAL-faithful at that hour was 2,191 times the Einstein Arena’s minimum improvement on C_1 and 6.99 times on Erdős. Against CORAL-faithful the society led on AHC026 from the first hour; against the isolated agents it led on all three problems by 4.0 hours; and from 6.5 hours to the end of each comparison it led both arms on all three problems at every quarter-hour reading (Table 24). Between 4 and 8 hours its C_1 margin grew more than twelvefold, to 38.1×10^{-6} .

The mechanism is reuse, and it is cheap. Reading and writing the archive took 6 to 17% of the society’s tool calls, and a teammate’s new best was first built on a median of 49 to 82 minutes after it appeared. From that point every agent searched from the team’s best verified work instead of its own.

The constructions. The society’s two constructions are the certified bounds of Propositions C.1 and C.2. Both passed the Einstein Arena verifier, and both beat a preregistered optimizer that uses no language model and the single agent at the same elapsed time, the registered criterion for crediting a record to the system.

D.5 How agents built on each other’s work

Citations in the archive under-count reuse, because a citation records only the items an agent names. Reuse was therefore counted from data flow, by a provenance pass over the agents’ transcripts: it traced what each agent read and ran, replayed file writes and copies of shared code, and compared each new best’s generator with the shared tools by code similarity. Over the society’s first 6.7 hours, 60 of its 98 new bests reused another agent’s attempt or tool. Of these, 57 had another agent’s attempt in their lineage and 38 ran or adapted a teammate’s tool. All 13 strong detections checked by hand were correct, and the agents’ own citations credited 47. The lineage share, 57 of 98 (58%), is close to CORAL’s 66% of kernel-task records from another agent’s parent (Qu et al., 2026); the tool share does not appear in a parent count. Reuse ran highest on minimum overlap, where at one checkpoint 17 of the society’s 21 new bests ran another agent’s tool. At 8 hours its minimum-overlap margin was 9.3 times the single agent’s and 2.9 times the isolated agents’, which had the same model, machine, and hours.

The agents used each other’s attempts and tools in four ways, and the minimum-overlap lineage contains each (Table 25). An agent *continued* a teammate’s construction with a new method, as agent 2 did with agent 3’s best at 5.78 hours.

Table 25: **The declared lineage of the Erdős bound**, 19 attempts in five segments by agent. Working hours of the segment’s attempts; fast-evaluator bound at the segment’s last attempt. The parent changed agents four times.

Segment	Agent	Attempts	Hours	Bound	Move
1	agent 2	4	0.32–1.11	0.3808582070	Upsample the Einstein Arena record; log-sum-exp continuation from a low starting β ; coarse-to-fine population at $n = 1,792$
2	agent 1	1	1.63	0.3808581437	Integrator session: apply another lineage’s upsample-and-re-anneal recipe to the team best
3	agent 3	4	3.89–5.78	0.3808575352	Symmetrize with agent 2’s annealer, then break symmetry with antisymmetric noise near the center
4	agent 2	4	6.09–7.42	0.3808570146	Break symmetry along unstable eigen-directions of the exact Hessian
5	agent 3	6	8.71–10.82	0.3808567389	Memetic genetic algorithm with agent 2’s tool, seeded from segment 4’s construction; slow-cooling refinement; crossover of two separately evolved solutions

Table 26: **The lead-thread run at 2, 4, 8, and 12 matched working hours**, against its own run’s CORAL-faithful arm and single agent. Units as in Table 23; bold marks the best arm at each hour.

Arm	$C_1 (\times 10^{-6})$				Erdős ($\times 10^{-6}$)				AHC026 (dev)			
	2 h	4 h	8 h	12 h	2 h	4 h	8 h	12 h	2 h	4 h	8 h	12 h
Lead-thread society	0.14	0.81	52.64	64.63	0.151	0.249	0.508	0.549	1,429,270	1,430,454	1,434,292	1,435,288
CORAL-faithful	0.11	21.70	43.00	51.97	0.060	0.142	0.265	0.526	1,426,231	1,432,050	1,433,563	1,434,264
Single agent	0.05	0.07	0.30	0.85	0.128	0.316	0.459	0.492	1,429,457	1,430,218	1,430,909	1,431,468

It *ran* a teammate’s tool on its own attempt, as the minimum-overlap agents did with `lse_anneal` throughout. An integrator session *applied* a recipe from one lineage to another lineage’s best, and the last step *crossed* two lineages. Continuing and running require an attempt and a tool that an agent other than the author can execute; applying and crossing require two verified attempts in one store. On C_1 , at 23:03 UTC agent 2 began reading agent 3’s shared kernel `alfast` and timing it. Twenty-two minutes later it wrote a kernel whose docstring reads ‘kernel adapted from tool:alfast by ac1-3,’ and half an hour after that it fetched agent 1’s 131,072-cell construction, doubled its resolution, and ran its own ramp on the adapted kernel. The certified bound is a snapshot of that run.

D.6 The lead-thread society

The race exposed what a single long conversation adds. A single agent with a whole problem’s CPU share and one unbroken conversation can drive a multi-hour program on most of the machine: on C_1 it ran a family search of five 2,048-level roots through a doubling ladder to 8,192 levels, and one root cut 1.45×10^{-5} in a single attempt. The society’s agents, re-briefed every four hours with a rotated direction, spread their CPU over short jobs (median 0.21 hours on C_1). The lead-thread society combines the two strengths in one design (Algorithm 3). One lead agent per problem keeps one conversation for the run, with unlimited resumes, no rotated direction, and a notebook mirrored read-only to two explorers every 5 minutes. The explorers hold a direction for two sessions and take parents from their own line or from the lead’s recent attempts. A compute pool runs long jobs outside sessions, weighting the lead’s jobs four-fold and the team best’s descendants three-fold, and an explorer’s new team best goes in full to the lead.

The lead-thread society ran sealed from October 8, 2026, with its own CORAL-faithful arm and single agent (Table 26). Between 6.25 and 8.25 hours its C_1 margin rose by 50.6×10^{-6} , the largest two-hour gain on C_1 of any arm in either run, twice the largest of the single agent. From 8.0 hours to the end of the run, it led its CORAL-faithful arm and its single agent on all three problems at every quarter-hour reading. At 12 hours its C_1 margin was 64.6×10^{-6} , against 52.0 for CORAL-faithful, and 15.9×10^{-6} beyond the earlier run’s single agent at the same hour.

E Extended results: verified technique and predecessor anchoring

E.1 The kernel instrument

KernelBench asks models to replace PyTorch reference modules with faster custom GPU kernels (Ouyang et al., 2025a). The experiments draw 130 of its 250 problems, with reference modules vendored verbatim, on NVIDIA A100-SXM4-40GB GPUs. An episode’s *capped speedup* is $s = \min(10, T_{\text{eager}}/T_{\text{kernel}})$ if the evaluator accepts the kernel and 0 otherwise; its *log speedup* is $\log(1 + s)$. Readers write a `ModelNew` module with CUDA C++ or Triton kernels, check public inputs as they choose, and submit, with at most five turns and 32,000 output tokens per reply.

The protected evaluator runs in a sandbox the agent cannot reach. It rejects monkeypatching, streams, graphs, and fallbacks, and requires a custom kernel. A trusted process that never sees candidate code generates reference outputs on fresh random seeds and weights, and timing interleaves cold-cache runs of reference and candidate on fresh inputs. Both processes run strict fp32 at $\text{atol} = \text{rtol} = 10^{-4}$ with TF32 disabled. That is KernelBench’s own fp32 setting and 100 times tighter than the 10^{-2} official check that Measuring the Checker found to miss one witnessed fault in six (Du et al., 2026). Before any arm ran, the evaluator passed 24 qualification controls: two legitimate kernels it had to accept and 22 attacks it had to reject, among them reuse of prior results, caching by shape or pointer, side streams, timing monkeypatches, global TF32 flags, fp16 downcasts, and in-place input mutation.

E.2 Technique cards

The cards. A recipe records how a writer solved one problem. A *technique card* states a technique, the arithmetic that makes it work, and the condition under which it applies, and the protected evaluator checks it before admission. The library is headed by a seed card from the literature: an fp32-accurate GEMM on bf16 tensor cores built on the error-compensated splits of Henry, Tang, and Heinecke and of Ootomo and Yokota (Henry, Tang & Heinecke, 2019; Ootomo & Yokota, 2022). It splits each fp32 operand into two bf16 parts, $x = x_h + x_l$ with $x_h = \text{bf16}(x)$ and $x_l = \text{bf16}(x - x_h)$, and computes

$$xw \approx x_h w_h + x_h w_l + x_l w_h, \quad (5)$$

dropping the $x_l w_l$ term of order 2^{-16} . The card records its error, at most 0.34 of the 10^{-4} tolerance for reduction lengths up to 8,192, and its rate on the A100, 65–79 TFLOP/s against 19 TFLOP/s for fp32 cuBLAS, and it routes itself to problems inside that envelope. An engine that tested hypotheses against the evaluator admitted five more cards. Four restate the split: for matrix multiplication, convolution, transposed convolution, and attention. The fifth computes training-mode batch normalization from Welford partial sums. A card is shown to the agent only where its routing rule fires.

The result. In a preregistered test on 26 problems outside the trials used to build the cards (16 never-trialed holdouts with same-problem solutions withheld, and 10 development problems the system never streamed), the technique library raised Claude Opus 5.5’s mean capped speedup to $3.67\times$, against $1.95\times$ for ACE’s own 36-hour playbook and $1.86\times$ bare (Table 8). The gain over ACE was $+1.71$ [1.19, 2.25] in mean capped speedup, with the library ahead on 22 of the 26 problems. It was $+1.68$ [1.10, 2.31] on the 16 holdout problems alone, and it stayed between $+1.55$ and $+1.85$ when any one problem was left out. There were 720 episodes, three per arm and problem, and the result was reproduced with independent analysis code. ACE’s playbook filled some 305,000 characters of context per episode; the store with the library, 18,000.

The card supplies a capability the model did not use unaided. The seed card’s rule fires on 14 of the 26 problems, and there the card alone added $+1.93$ [1.35, 2.51] over bare, up on 13 of the 14. Bare Claude Opus 5.5 used bf16 tensor cores in none of the 42 episodes on those problems, and neither did the diligence instruction, ACE, or the store alone. With the library the model used them in 37, and ACE given only the card text used them in 33. The largest gains came on L2_020 (from $1.43\times$ to $5.96\times$), L3_044 ($0.89\times$ to $4.50\times$), and L2_040 ($1.03\times$ to $4.44\times$). On the 12 convolution, transposed-convolution, and recurrent problems where the seed card does not fire, the engine cards that extend the split added $+1.36$ [0.79, 2.10] over the store alone, up on 11 and down on none. One computes a convolution as an implicit GEMM with the bf16 split in registers, moving its arithmetic from the 19.5 TFLOP/s fp32 path of the A100 to bf16 tensor cores that peak at 312 TFLOP/s.

No instruction contains a technique, so a card must supply it. On these problems the diligence instruction added $+0.10$ over bare, against $+1.81$ for the library; the instruction share is 0.04. The gain travels with the card text: given the

full library, ACE gained +1.34 [0.77, 1.91] over its own playbook and reached 3.29 $\times$, and given only the seed card it reached 2.83 $\times$. The technique also raises a weaker reader. The seed card more than doubled Claude Opus 4.8, from 0.75 $\times$ to 1.83 $\times$ on ten development problems (+1.08 [0.44, 1.74]). In an earlier preregistered offline test on 12 further problems outside the library’s trials, the cards in a frozen copy of the continuous-operation system raised Claude Opus 5.5 above bare on all 12 (+2.74 [1.60, 3.96]). Deployed into the running system at hour 36, the library lifted the live system above bare by +0.70 in mean capped speedup on its probe problems.

E.3 The store and its randomized harm

The store. The continuous-operation system streamed AppWorld, τ^2 -bench, coding, and KernelBench tasks for 44 hours; its kernel work began at hour 6.1, Claude Opus 4.8 was the worker until hour 36, and Claude Opus 5.5 after it. It kept what its worker verified, and the store outlived the worker. The store studied here is its hour-24 snapshot, 71 items built from 18 hours of Claude Opus 4.8 kernel work: 20 verified kernels by Claude Opus 4.8, and 41 items Claude Opus 5.5 itself wrote from them, including 14 recipes, 23 family pitfalls, and 4 lessons. Every kernel and recipe uses Triton (Tillet, Kung & Cox, 2019). *Verified* has one meaning throughout: the evaluator accepted the item’s kernel at strict fp32 on the item’s source problem and measured its speedup there. The certificate belongs to the item and its source problem, and it names no reader.

Randomized harm. Switched on or off at random in 1,288 episodes, the whole store changed Claude Opus 5.5’s log speedup by -0.103 [$-0.163, -0.048$] over 92 problems and Claude Opus 4.8’s by $+0.256$ [$+0.178, +0.337$] over 77 (Figure 11). A confirmation test on 28 problems outside calibration gave -0.182 [$-0.327, -0.074$] for Claude Opus 5.5 and $+0.218$ [$+0.149, +0.291$] for Claude Opus 4.8. In a preregistered head-to-head the whole store slowed Claude Opus 5.5 on 9 of 10 held-back problems and sped it up on none. The loss is broad: the store cost Claude Opus 5.5 more than 0.05 log speedup on 39 problems and helped it by that much on 14, while for Claude Opus 4.8 the counts reverse, 7 and 44. The two readers move on different axes. For Claude Opus 4.8 the store is a repair: its failures fell from 62 of 186 episodes to 3 of 182, and all 46 of its compile errors without the store were CUDA episodes. Claude Opus 5.5 was correct in 918 of 920 randomized episodes, so its entire loss is speed; its share of kernels at twice eager speed fell from 37.7% to 26.2%.

A worked case: the successor writes its predecessor’s kernel. KernelBench L2_074 applies a transposed 3D convolution from 16 to 32 channels, with kernel size 3 and stride 2, to a $16 \times 16 \times 16 \times 32 \times 32$ input, followed by LeakyReLU, a per-channel multiply, a second LeakyReLU, and a $2 \times 2 \times 2$ max pool. The convolution’s output has 67 million elements, and eager PyTorch takes 5.9 ms. Alone, in all seven bare episodes, Claude Opus 5.5 wrote one CUDA kernel for the whole module. It exploits a structure of the problem: a stride-2 transposed convolution splits into eight parity classes, and each output in a pooling window receives a fixed subset of the 27 kernel taps. Each thread holds the eight window outputs for two pooled positions and four output channels in registers, applies the epilogue, and writes only the pooled maximum, so the convolution’s output never reaches memory. The kernel ran in 0.61–0.91 ms and averaged 8.86 $\times$.

Shown the store, in all three episodes Claude Opus 5.5 kept the convolution in cuDNN and wrote one Triton kernel for the epilogue and pool, scoring 1.26 $\times$. That kernel took 5.1% of 4.66 ms, so the library call took 4.42 ms, and no epilogue removes it: the adopted design is capped at $5.91/4.42 = 1.34\times$. The harness had removed the store’s own recipe for L2_074, a 1.26 $\times$ design, because it names the problem. The successor rebuilt it anyway, clause by clause, within 0.01 of the removed recipe’s speedup, because five neighboring recipes and two verified kernels carried the same approach. Appendix O.1 lists both kernels verbatim.

A formal model of transfer. For problem p , reader R , and store indicator $Z \in \{0, 1\}$, an episode’s outcome is $y = \log(1 + s)$. The store effect is $\tau_R(p) = \mathbb{E}[y \mid R, p, Z = 1] - \mathbb{E}[y \mid R, p, Z = 0]$. The *approach* a of an episode is the design class of its submitted kernel. With $\rho_R^z(a \mid p) = \Pr(a \mid R, p, Z = z)$ and $\mu_R^z(a, p) = \mathbb{E}[y \mid R, p, a, Z = z]$, the store effect splits exactly, by the law of total expectation, into an adoption term and a within-approach term:

$$\tau_R(p) = \underbrace{\sum_a [\rho_R^1 - \rho_R^0](a \mid p) \mu_R^0(a, p)}_{\text{adoption}} + \underbrace{\sum_a \rho_R^1(a \mid p) [\mu_R^1 - \mu_R^0](a, p)}_{\text{within approach}}. \quad (6)$$

Let a_S be the approach the store’s kernels and recipes share, $\Delta_R(p) = \rho_R^1(a_S \mid p) - \rho_R^0(a_S \mid p)$ the adoption shift, and $G_R(p) = \mu_R^0(a_S, p) - \mu_R^0(\bar{a}, p)$ the *approach gap*: the value of the stored approach in the reader’s hands minus the

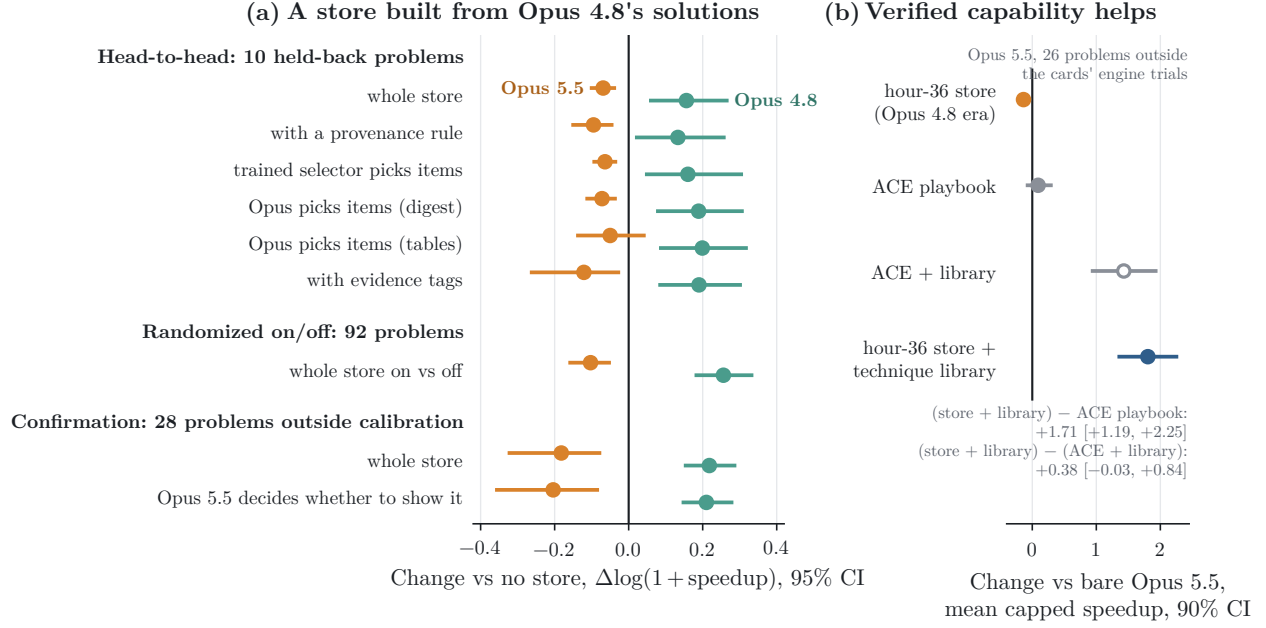


Figure 11: **The same store lowers the stronger reader and raises the weaker one, and verified technique turns the stronger reader's sign positive.** (a) Store minus no store in $\log(1 + \text{capped speedup})$ with 95% problem-bootstrap intervals, for the hour-24 store built from 18 hours of Claude Opus 4.8 kernel work. Head-to-head: six serving methods on 10 held-back problems, four repetitions per cell. Randomized: the whole store switched on or off at random in 1,288 episodes over 92 problems (77 for Claude Opus 4.8). Confirmation: 1,120 episodes on 28 problems outside calibration. (b) Claude Opus 5.5 on 26 problems outside the cards' engine trials, 720 episodes: mean capped speedup minus bare, with 90% task-bootstrap intervals.

value of its own.

Proposition E.1 (Sign of transfer). *If the within-approach term is zero and $\Delta_R(p) > 0$, then $\tau_R(p) = \Delta_R(p) G_R(p)$, and $\tau_R(p)$ has the sign of $G_R(p)$.*

Corollary E.2 (Downhill and uphill). *If the stored approach has the writer's value $v_Q(p)$ in any reader's hands and the reader's own approach has a value $v_R(p)$ rising with the reader's strength, then $G_R(p) = v_Q(p) - v_R(p)$. The store helps any reader below its writer and harms any reader above it, and the harm grows with the reader's lead, $|\tau_R(p)| = \Delta_R(p)(v_R(p) - v_Q(p))$.*

Corollary E.3 (Item selection). *If every approach-carrying item carries a_S and serving any one raises adoption, then every subset of the store that contains an approach-carrying item has an effect with the sign of $G_R(p)$. Only subsets with no approach-carrying item, the empty subset among them, escape it.*

Corollary E.4 (Conditional adoption). *If the store arrives after the reader holds a checked kernel, and the reader adopts a_S only when its own comparison favors a_S , then $\Delta_R(p) = 0$ wherever $G_R(p) \leq 0$, and the adoption term is nonnegative.*

Remark E.5 (What verification certifies). The evaluator certifies an item's correctness at 10^{-4} on its source problem q_i and records the speedup there. Proposition E.1 concerns $G_R(p)$ on a target $p \neq q_i$. The gap depends on the reader and the target, while the certificate depends on the item and its source, so no condition on an item's correctness and speedup fixes the sign of transfer for every reader and target.

Memory built from a weaker model's work transfers downhill. Carried uphill, its cost rises with the reader's strength, and each upgrade raises the reader while leaving the store at the writer's level. In Equation (2) the store supplies no cause the stronger reader lacks ($p \approx 0$) and breaks kernels the reader would have written faster ($q > 0$).

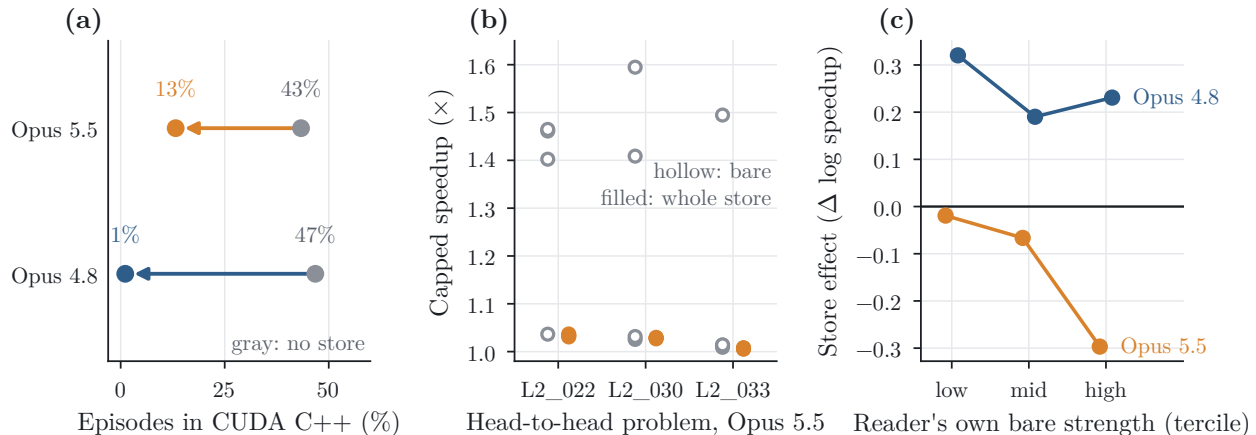


Figure 12: **The store pulls Claude Opus 5.5 away from CUDA, flattens its best kernels, and hurts it most where it is strongest.** (a) CUDA C++ share in the randomized test, without the store (gray) and with it (color). (b) Claude Opus 5.5 capped speedups on three small head-to-head problems, bare (hollow) and whole store (filled); every filled point is a token kernel. (c) Store effect in log speedup by tercile of bare strength, cross-fitted over 200 random splits of both tests.

The adoption shift carries the loss. Figure 12 shows the mechanism. Across the randomized test the store moved Claude Opus 5.5 the way it moved it on L2_074. Its CUDA C++ share fell from 43.35% to 13.22% of episodes, and token kernels, whose own code takes under 5% of runtime while a library call does the rest, rose from 19.7% to 31.9%. CUDA was its faster technology: within problems, its CUDA episodes were faster by $+0.17$ [$+0.06, +0.29$] log speedup. Equation (6) gives the size of the backend channel, $0.301 \times (-0.17) = -0.051$. A regression with problem fixed effects estimates it separately on the same episodes: adding a CUDA indicator moves the store coefficient from -0.106 to -0.053 , a step of 0.053 . The two estimates agree to 0.002 . Adding the token-kernel indicator takes the coefficient to -0.024 [$-0.065, +0.015$], so backend and token-kernel status together account for 78% of the store’s effect, and within CUDA episodes the store’s effect is $+0.034$. The within-approach term is zero where it can be read, so the proposition’s condition holds. The adopted approach is also shorter: the store cut Claude Opus 5.5’s output by 27%, from 9,597 to 7,142 tokens per episode, and raised submissions right after a first check from 76% to 86%.

The harm grows with the reader’s lead. Cross-fitting bare strength and store effect on disjoint halves of the no-store episodes gives Claude Opus 5.5 effects of -0.019 , -0.066 , and -0.297 log speedup by tercile, a fifteenfold rise from its weakest problems to its strongest, while Claude Opus 4.8 gains in each tercile ($+0.32$, $+0.19$, $+0.23$). The stored recipes sit where Corollary E.2 places them: only 4 of the 22 speedup-tagged recipe entries exceed Claude Opus 5.5’s bare mean on the problem where each was verified, while 18 of 20 exceed Claude Opus 4.8’s. The same store has a positive gap for one reader and a negative gap for the other, and the measured signs follow. On development problem L2_058 the successor reached the $10\times$ cap alone and $7.2\times$ with the store’s family recipes; hiding those recipes restored $10\times$ in 8 of 8 runs.

Verification and anchoring. Each item passed an evaluator 100 times stricter than the audited KernelBench check, and the store still anchored, for the reason of Remark E.5. Verification certifies a floor: it admits items at parity, and a guaranteed-correct design at the writer’s level is exactly what a failing reader needs and a ceiling for a reader above it. Reading a digest of the randomized outcomes, Claude Opus 5.5 stated the result in one line: ‘The store raises the floor and lowers the ceiling.’ A certificate also does not move from source to target. Serving only items whose verified speedup exceeds the reader’s own bare mean on their source problem still scored -0.118 against no memory; on L2_074 that gate passed a $1.32\times$ recipe, above Claude Opus 5.5 on its source and far below it on the target. A verified lesson can forbid a technique: Claude Opus 4.8’s failures with plain TF32 became a correct rule, written by Claude Opus 5.5, to always use exact fp32 `tl.dot`, which also rules out the fp32-accurate `tf32x3` mode that none of the store’s 181 kernel items used. More evidence anchors harder: tagging each item with its measured effect for the reader was the worst of six serving methods for Claude Opus 5.5, -0.610 in mean capped speedup.

Table 27: **Each serving method leaves Claude Opus 5.5 below bare and lifts Claude Opus 4.8.** 10 held-back problems, 4 repetitions per cell, differences from bare with individual 95% problem-bootstrap intervals; all arms share one bare cell. Bare means: Claude Opus 5.5 $3.309\times$, Claude Opus 4.8 $0.593\times$. Characters are means served to Claude Opus 5.5.

Arm	Characters	Claude Opus 5.5			up/down	Claude Opus 4.8	
		Δ capped	Δ log			Δ capped	
show-all	8,885	-0.287 [-0.591, -0.089]	-0.069 [-0.105, -0.034]	0/9		+0.216 [+0.071, +0.383]	
provenance	6,564	-0.483 [-1.009, -0.092]	-0.095 [-0.155, -0.041]	1/8		+0.185 [+0.019, +0.370]	
selector (trained)	6,892	-0.266 [-0.489, -0.085]	-0.064 [-0.098, -0.031]	1/8		+0.224 [+0.056, +0.439]	
Opus+digest	631	-0.359 [-0.750, -0.068]	-0.072 [-0.117, -0.032]	1/8		+0.263 [+0.096, +0.441]	
Opus+tables	1,141	-0.347 [-1.003, +0.232]	-0.050 [-0.142, +0.046]	2/7		+0.281 [+0.111, +0.461]	
annotated (tags)	11,848	-0.610 [-1.450, -0.100]	-0.121 [-0.267, -0.023]	2/8		+0.271 [+0.110, +0.441]	

Item selection. Corollary E.3 predicts that no item-selection method changes the sign for Claude Opus 5.5 while an approach-carrying item remains, and none did (Table 27). The harness’s own provenance rule hid every recipe from Claude Opus 5.5, but the lessons, pitfalls, and verified kernels still carried the predecessor’s approach, and the loss grew to 0.483. A ridge-regression selector over randomized item exposure cost 0.266: no item group’s coefficient for Claude Opus 5.5 exceeded 0.032 in magnitude, and an item-level estimate cannot find a loss that no item carries alone. Claude Opus 5.5 selecting items from per-item outcome tables or their digest served short contexts, 1,141 and 631 characters, and still cost 0.347 and 0.359. Asked zero-shot whether the store would help, Claude Opus 5.5 chose to see it on 55 of 55 problems, and that policy scored -0.204 log speedup below no store. A calibration run located the loss by removal: lessons alone, and lessons with pitfalls, left Claude Opus 5.5 at its no-memory level, and adding the recipes and verified kernels cost -0.244 $[-0.450, -0.077]$ (Table 46).

Remedies that act on adoption. Corollary E.4 says a store delivered after the reader holds a checked kernel can be adopted only where it is better. On 37 problems where bare Claude Opus 5.5 writes mostly CUDA, appending the verbatim store to the feedback on the reader’s first local check, under the header ‘Memory from earlier problems (verified alternatives from other problems; keep your own code if you judge it faster)’, scored $+0.333$ $[+0.125, +0.519]$ log speedup above up-front delivery and $+0.106$ $[+0.008, +0.210]$ above no memory. The CUDA share was 0.89 without memory, 0.15 with the store up front, and 0.91 with it late; the served text was identical, and only its timing differed. A second remedy withholds the store where its sign is negative. Given a digest of the randomized outcomes, Claude Opus 5.5 chose to serve the store to Claude Opus 4.8 on every problem and to itself on 29% of them. Frozen before confirmation, that rule kept Claude Opus 4.8’s gain ($+0.204$ $[+0.131, +0.282]$) and lifted Claude Opus 5.5 by $+0.135$ $[+0.038, +0.250]$ over show-all, meeting both preregistered predictions. Per-problem selection has little left to find: an in-sample oracle would have served 22% of problems for a gain of 0.042, and a fitted per-problem harm predictor separated helped from harmed problems at chance (AUC 0.50). For the successor, the store’s useful content is one bit, the decision to withhold it; randomized exposure reveals that bit, and the zero-shot prompt returned the opposite bit on all 55 problems.

The anchor arrives early. On ten development problems each frozen snapshot from hour 8 to hour 36 lowered Claude Opus 5.5’s mean capped speedup, by 0.16 to 0.54, with all 90% intervals below zero. Kernel work began at hour 6.1, so the hour-8 store, built from under two hours of predecessor kernel work, already anchored the successor. Claude Opus 4.8’s gain grew with the store, by $+0.32$ per day from hour 12 to hour 27, reaching $+0.46$. By hour 42, after the upgrade and with the technique library in the store, the store lifted both readers above bare: Claude Opus 5.5 by $+0.76$ $[+0.05, +1.47]$ and Claude Opus 4.8 by $+0.90$ $[+0.40, +1.45]$.

E.4 Downhill and uphill across the program

Corollary E.2 predicts more than one store. Wherever approach-carrying memory crosses models, the sign of transfer should follow the strength gap between writer and reader. Rank the program’s models by strength (Claude Haiku 4.5 and Qwen3.5-27B at 1, Claude Opus 4.8 at 2, Claude Opus 5.5 at 3), and count every recorded comparison in which a reader given knowledge distilled from a writer’s past episodes (a store, playbook, rules, recipes, or written checks) is compared with the same reader without it, excluding mid-task handoffs.

Downhill, transfer helped in 5 of 6 comparisons; uphill, in 0 of 9, with 4 significant harms; at equal strength the sign is

mixed, 3 of 5 (Table 9). Over 25 comparisons the sign falls with the gap, Spearman -0.81 ($p = 7.5 \times 10^{-7}$); by source, downhill transfer helped in 3 of 3 sources and uphill in 0 of 3. The pattern holds on KernelBench, AppWorld, and HORIZON, for stores and for written checks. An AppWorld store compiled by Claude Opus 5.5 raised Claude Haiku 4.5 by $+7.0$ [$+2.9, +11.3$] points, and checks written by Claude Opus 5.5 in an evidence-only takeover harness raised a Qwen3.5-27B successor by $+0.476$ on HORIZON (Section 5.7). A rule for carrying memory across models must read the direction of the gap before it reads the items. What a harness carries upward must be technique that sits above the new reader: a recipe records the writer’s solution and carries the writer’s level, while a card records a technique and its envelope, and its level is set by the arithmetic it exploits.

F Extended results: supervision

F.1 The HORIZON benchmark

On HORIZON, an agent develops a JSON-command account-balance service (accounts, deposits, withdrawals, transfers, fees, interest, and currencies) in Python with the standard library only, through $T = 120$ dependent tickets. Tickets add commands, state rules that bind later commands, amend earlier behavior, clarify, or deliver vendored-dependency upgrades. Each attempt has at most 80 agent steps and 1,800 s. After each ticket the benchmark deploys the working tree as it stands, without committing it for the agent. With R_t the deployed repository and q_t the scripted customer traffic, the production store evolves as $D_t = \text{serve}(R_t, D_{t-1}, q_t)$. Amendments apply from the ticket that ships them, and past transactions are never recomputed, so a correct balance depends on the order of past traffic and rule changes, which the specification alone does not determine. Production correctness is

$$S_{\text{prod}}(t) = \frac{1}{|\mathcal{A}_t|} \sum_{a \in \mathcal{A}_t} \mathbf{1}[\text{get_balance}_{R_t}(a; D_t) = o_t(a)], \quad (7)$$

the fraction of production accounts whose balance matches a hidden oracle. Repairing the code restores fresh-state correctness and leaves the damage already in D_t : a bug live for three tickets left S_{prod} at 0.71–0.94 after the repair.

The six headline streams were generated, validated, and hashed with public and hidden SHA-256 digests before any harness ran. On each stream the oracle used as an implementation scores 1 at every ticket, a persisting reference scores $S_{\text{prod}} = 1$, and an amnesic reference, which keeps no data between processes, scores 0. The hidden scorer kills 88 of 89 targeted rule-violation mutants, and a bug planted at tickets 30–32 stays in the data: S_{prod} is 0.904–0.938 at ticket 120.

F.2 Verdicts, evidence, and iatrogenic regression

Definition F.1 (Hard check; evidence; verdict). A hard check c is a deterministic ground-truth test of a code tree with outcome $c(\cdot) \in \{\text{pass}, \text{fail}, \text{unknown}\}$; a runner timeout, a runner crash, and an empty result are unknown. Evidence is a hard-check outcome returned with what the acting model needs to reproduce it: the input, the expected and actual output, or a stack trace. A verdict is a conclusion about the attempt that a component other than the acting model derives and the harness acts on: a judge’s ruling, a paraphrased requirement, a build status, or a comparison of failure counts.

Definition F.2 (Verdict conversion; iatrogenic regression). Verdict conversion is a step that maps a state of the harness’s own machinery to a conclusion about the attempt; its simplest form assigns $v(\text{unknown}) = \text{fail}$, which turns a runner timeout into a failed build. Let $S_k^H(t)$ be a correctness measure for harness H at ticket t of stream k . Ticket t is an iatrogenic regression of H if $S_k^H(t) < S_k^{\text{bare}}(t)$ and the difference follows from an action of H . H meets the no-harm criterion on a panel of streams if no ticket of any stream is below bare.

Medicine calls damage caused by the treatment iatrogenic, and the rule against it is *primum non nocere* (Brennan et al., 1991; Illich, 1976; Smith, 2005). Persisting state makes the criterion strict: a service that runs the wrong code for one ticket writes wrong balances, and later tickets build on them, so a regression is permanent and no later gain cancels it. In the terms of Equation (2), the no-harm criterion is $q = 0$.

F.3 Evidence-only harnesses

Five invariants make the distinction between evidence and verdict operational; each closes a failure traced in Appendix F.6.

- I1 Bare-identical session.** The trusted model receives the bare system prompt and the ticket byte for byte, with no requirements list, digest, paraphrase, or candidate test. A paraphrase in the session is a verdict about what the ticket means.
- I2 Evidence for repair.** After the agent finishes, a checking harness replays the ticket’s visible examples on a fresh service, detects crashes, hangs, and malformed responses, and starts the new code on a copy of production data. Failures return to the agent’s own session for one bounded repair, as inputs with expected and actual outputs or stack traces. The model holds the ticket and the code, so the model decides what a failure means.
- I3 Unknown outcomes trigger nothing.** A runner timeout, crash, or empty result triggers no repair, rollback, or restore. All checks run in one sandboxed process, 7,000 of them in 2 s, and identical checks are admitted once, so the suite grows with distinct behaviors.
- I4 Committed deployed state.** The deployed tree is committed to the agent’s own git at the start and the end of each ticket, so the repository’s history agrees with production.
- I5 Off the critical path.** Checks that a model writes for later use are produced concurrently with the session and are not shown to the trusted model.

The discard rule. After repair, with check set C , the harness discards the attempt a in favor of the deployed base b only when

$$\exists c \in C : c(b) = \text{pass} \wedge c(a) = \text{fail} \quad \text{and} \quad \nexists c \in C : c(a) = \text{pass} \wedge c(b) = \text{fail}. \quad (8)$$

The rule has four exhaustive cases. If checks separate the two in one direction only, the attempt lost behavior the base had, and the base is deployed. If checks point both ways, the attempt traded one behavior for another, and only the acting model, which holds the ticket, can say which trade the ticket asked for, so the attempt stays. If no check separates them, the tie keeps the attempt. A check that returns unknown enters neither clause.

Proposition F.3 (Unknown outcomes cannot discard an attempt). *Under Equation (8), an attempt on which each check returns pass or unknown is deployed, whatever the base’s outcomes.*

Proposition F.4 (A commit bounds a revert to one ticket). *If R_{t-1} is committed at the start of ticket t , a command that restores the tracked tree to `HEAD (git stash, checkout -- ., reset --hard)` returns it to R_{t-1} and discards only edits made during ticket t . Without that commit, `HEAD` is R_h , the last tree the agent itself committed, and the same command reverts the deployments of tickets $h + 1$ to $t - 1$.*

A discard requires a check with $c(a) = \text{fail}$, so a runner that times out cannot cost the model its attempt, however large the suite grows. Three harnesses implement the invariants. `commit` runs `git add -A && git commit` twice per ticket, at the start and at the end, and makes no checks and no model calls. `guard` adds hard checks and still makes no model calls: it replays the current ticket’s examples with and without a restart, replays earlier tickets’ examples (counting a failure only if the example passes on the base), and starts the new code on a copy of production data, then applies Equation (8) after one repair round (Algorithm 5). The *evidence-only supervisor*, `full`, adds checks for successors. With the trusted model acting, it follows I1–I5. A successor enters probation and receives a handover brief with the verbatim ticket log and mandatory instructions to run the checks; the harness then enforces request-sequence checks written by the trusted model, and a production differential reads each known account under the base code and the new code. On probation a check is retired only when the successor quotes a verbatim sentence of the current ticket, at least 15 characters long, that shares an identifier with the check. Its revision `full v2` applies Equation (8) to successors as well.

F.4 The no-harm panel and its positive control

`commit`, `guard`, and `full v1` kept S_{prod} at 1.000 on each ticket of each fresh stream, and none of the 2,880 tickets in their 24 runs fell below bare (Table 10). Independent code re-derived all 5,040 scores of the 42-run panel, and the official scorer reproduced 11 of 11 sampled snapshots. In the run traced in detail, `full v1` made no repair or rollback on any of its 120 tickets: against Claude Opus 5.5 an evidence-only harness has nothing to act on, and by Proposition F.3 a timeout cannot give it anything.

A no-harm result against a ceiling requires proof that the instrument registers harm when harm occurs, and the verdict harness supplies it. Rerun unchanged on the same six streams, it first fell below bare at tickets 22, 48, 35, 25, 41, and

54, inside its preregistered forecast of at least 4 of 6 streams by ticket 60. It stayed below bare on all six streams from its first failure to the end of the analyzed data, 360 of the 579 tickets it reached; from ticket 41 its mean S_{prod} was 0.656 against bare’s 1.000, a paired loss of 0.344 [0.240, 0.448] ($p = 0.031$). The protection costs no time where it is not needed: `guard` and `commit` ran at 0.68–1.17 of bare’s median wall time per ticket, inside the 0.69–1.22 spread between two bare seeds, while the verdict harness took 3.6–14.3 times as long.

F.5 The Qwen3.5-27B handover

When Qwen3.5-27B takes the service over at ticket 40, bare’s mean S_{prod} over tickets 41–120 falls to 0.435, and a second bare seed’s to 0.370. The three harnesses that `commit` hold it at 2.02–2.09 times bare: `commit` at 0.881, `guard` at 0.880, and `full v2` at 0.911 (Table 11); `full v2` and `guard` are above bare on all six streams ($p = 0.031$). `full v2` kept each account correct at each ticket from 41 to 120 on 4 of 6 streams, and none of the 12 bare runs did. The gap widens with the horizon: the paired gap between `full v2` and bare is +0.17 in tickets 41–50 and +0.60 in tickets 111–120, and its interval excludes zero in each 10-ticket bin from ticket 71. By ticket 120 a bare handover keeps 6% of production accounts correct on average; `commit` and `full v2` keep 82%. A bare successor’s error stays in the balances and the next error adds to it, so a harness that prevents the earliest error also prevents the accumulation.

The regeneration control. What does a successor lose? Given the README, the vendored code, and tickets 1–120, with no tools, tests, or accumulated state, one Claude Opus 5.5 call writes a service that passes all 896 hidden fresh-state cases on three development streams. Given the 467,000-token history of a 360-ticket composite that interleaves three tenants, one call passes all 2,856, again under a fresh case seed. The ticket log rebuilds the behavior the tickets asked for. It cannot rebuild the balances that customer traffic wrote under the rules in force at each ticket, and those balances are what S_{prod} scores. A supervisor must protect what a handover can lose: the deployed state and the history that produced it.

Orphaned deployments. A deployment is *orphaned* when production runs a tree that the repository’s history does not contain. HORIZON deploys the working tree without committing it, so a deployment is recorded only if the agent commits. Claude Opus 5.5 committed on 33–37 of its 39 tickets. On earlier handover streams, its successor Claude Haiku 4.5 committed on none of its 486 tickets, so from ticket 40 every deployment it shipped was orphaned, and HEAD stayed at Claude Opus 5.5’s last commit. Claude Haiku 4.5 then read the shipped code as contamination (‘I accidentally included T040 code’), and it restored ‘the original code’ with `git stash, checkout, reset --hard, or clean`. By Proposition F.4 each such command returned the service to ticket 39; one removed 735 lines of post-handover service code in a single ticket. All 27 new production breaks after the handover followed such a command (Table 41). On three streams the reverted code could not read data that the newer code had written, and 12 outage events covered 90 service-down tickets. On the others the service kept answering under ticket 39’s rules, and periodic interest spread the error to accounts no request touched: at one ticket, 107 untouched accounts went wrong and S_{prod} fell from 1.000 to 0.028.

A `commit` at the start and end of each ticket makes HEAD equal production, and the successor’s restore commands then discard only the current ticket’s edits. On the Qwen3.5-27B handover, `commit` lifts mean S_{prod} by +0.445 against `full v2`’s +0.476: 93% of the lift, with no checks and no model calls. The remaining +0.031 belongs to failures a `commit` cannot see, new code that fails on existing production data with no revert at all, which the production compatibility check and the production differential exist to catch. Asking for the behavior is weaker. The diligent paragraph instructs the model to commit when done and to leave changes it did not make; it lifts bare by +0.123, three streams better and three worse, and `full v2` is above it on all six (+0.353, $p = 0.031$). The harness that performs the `commit` lifts production 3.6 times as much as the paragraph that requests it.

F.6 Forensics of the verdict harness

The verdict harness’s language-model steps are calls to the acting model. An *analyze* step writes the ticket into a requirements list and marks earlier requirements superseded; a *compile* step turns requirements into protected tests; *verify* runs the tests in a sandboxed runner with a fixed 600 s cap. A *judge* call, shown the current ticket, rules whether a failing protected test is still required; a red result triggers repair under the message ‘fix the implementation, not the tests’; a failed attempt is rolled back; *keep-best* chooses between base and attempt by requirement score; and *settle* restores the previous checkpoint if tests of an unfinished ticket regress. Forensics of 12 runs on six older production

Table 28: **The 24 first failures of the verdict harness, by the component that discarded correct code.** Twelve runs on the older streams; handover runs switch to Claude Haiku 4.5 at ticket 40. Failures are counted per run and metric.

Proximate component	Same model	Handover	Total
Rollback or keep-best discarded correct code	4	6	10
Settle restored an old checkpoint (4 also undid a vendor upgrade)	1	5	6
Judge ruled an obsolete test ‘still required’	0	4	4
Analyze step invented or overgeneralized a requirement	1	3	4
Model error unrelated to the harness	0	0	0
Total	6	18	24
Root causes: test duplication → suite timeout 18; runner text bug and tie 2; analyze step 4.			

streams traced each of its 24 first failures, through its state log, its events, and production responses, to a component of the harness, and none to the model (Table 28).

The cascade. The root of 18 failures is a duplication in the harness’s list of protected tests. When a ticket superseded a requirement, the harness re-admitted each invalidated test under a new identifier and kept the old one, so each supersession of a widely cited requirement copied a test together with all of its copies. By ticket 30 the three same-model runs held each distinct test 117–147 times over, and one test 21,054 times; on one stream, active protected tests grew 167-fold from ticket 10 to ticket 30 while distinct test sources grew 2.4-fold. The runner charged 4–6 ms per test, so its 600 s cap bound near 100,000 tests, and the suites reached 45,000–185,000. A suite past the cap returned `suite timeout after 600.0s`, a string that carries no outcome for any test, and the harness counted it as a red build. On stream O1 at ticket 29, `verify` timed out at 607 s and 608 s, repair handed Claude Opus 5.5 a failure that did not exist, and Claude Opus 5.5 replied that the suite ‘finishes in under 2 s here’. The harness rolled the attempt back anyway; the discarded attempt passes all 360 hidden cases. Re-running the harness’s own `verify` on the eight discarded attempts with the suite deduplicated and nothing else changed shrank the suite from 49,814–106,655 files to 392–737, and `verify` finished in 0.7–1.4 s: the timeouts came from the duplicated suite alone.

The judge and the analyze step. A ruling on a failing test requires the tickets that changed the test, and the judge saw only the current one. On one stream, tickets 30–33 had not finished, so their superseded tests were never retired; the judge, shown only ticket 34, ruled 967 obsolete tests ‘regression’ five times. Under the repair message ‘Fix the implementation (not the tests)’, Claude Opus 5.5 switched off two correct features, case-insensitive account ids and transfer fees, and said so: ‘I switched off two earlier behaviours’. This is automation bias with a model in the operator’s seat (Parasuraman & Manzey, 2010). The analyze step settled an ambiguity a ticket had left open, writing that ‘counting starts from zero for every existing account’, and the agent implemented that reading where bare Claude Opus 5.5, given no paraphrase, chose the lifetime counting the oracle uses.

G Extended results: durable state and evidence

G.1 Write-ahead receipts and crash recovery

A crash on AppWorld kills the agent process after a step has taken effect and before the agent has recorded it. The apps keep their state, and the Python session resets. The recovery harness keeps four pieces of state: a record of committed steps, a write-ahead intent that marks the step in flight as uncertain, a checkpoint of the Python session, and receipts from the tool server’s execution journal. The receipts settle the uncertain step: an effect that happened is committed once, and an effect that did not happen is proposed again. A resumed call whose identifier already has a receipt returns the recorded response without running again (Algorithm 6).

Exactly-once effects. In a deterministic engine test without a model, 1,328 random kills over 1,200 trials of 10 payments produced no duplicate or lost payment. The same storm duplicated 34 payments under blind re-proposal and 29 when intents were not marked as external effects, in 100 trials each. With a crash after every step, each agent process runs one step and cannot advance without saved state. With receipts, Claude Opus 5.5 completed 38 of the 39 tasks an uncrashed run solved (97.4%, Wilson 95% CI 86.8–99.5), and Claude Haiku 4.5 96.6%, where bare, bare told of the crash, and a plain write-ahead log each completed 0 of 58.

Crash streams. A preregistered AppWorld test-challenge stream had an undocumented API change and 52 scheduled crashes, with Claude Opus 5.5 throughout. The recovery harness with an API-change sensor raised task goal completion from 89.9 to 94.2 and scenario goal completion from 79.1 to 92.1, over 417 tasks in 139 scenarios (24 tasks better, 6 worse; exact McNemar $p = 0.0014$). Offline ACE scored 89.2 and 79.9, so the harness exceeds it by 5.0 and 12.2 points; crash recovery carried most of the gain, and the sensor’s own increment was 1.4 points. Scenario completion moves three times as far as task completion for an arithmetic reason: each scenario is three tasks, and the net gains coincide, 18 tasks out of 417 and 18 scenarios out of 139.

A second stream combined 52 crashes, a documented API change, and a switch from Claude Opus 4.8 to Claude Opus 5.5. Against bare restarted fresh after each crash, persisted state scored 91.6 task and 86.3 scenario goal completion, against 84.9 and 71.2 ($p = 9.0 \times 10^{-5}$); on the crashed tasks it scored 90.4 against 63.5. After a restart, duplicate mutations fell from 13 to 1, mean steps from 18.1 to 5.9, and crashed tasks passed rose from 33 of 52 to 47 of 52. Ablations separate the two halves of the state. The committed-step record alone passed 82.7% of crashed tasks with no duplicates. The state store alone passed 67.3% and duplicated 22 mutations, more than bare: a worker restored to its pre-crash state without the record of its executed effects resumes as if its last mutations had not happened. On the Claude Opus 5.5 half of the stream, duplicate side effects fell from 8 to 0, and steps to completion from 15.1 to 5.6.

A day of continuous operation. A continuous-operation system carried the same discipline from a task to a day. It streamed AppWorld, τ^2 -bench, coding, and KernelBench tasks for 44 hours, with six stream arms: this harness, bare, three rival memory systems (ACE, Mem0, and a provider memory tool), and a reset arm. By hour 31 its Claude Opus 4.8 worker had crossed the three boundaries at which a long-running agent loses working state, 144 compactions, 105 crash-and-resumes, and 147 model swaps, and it lost working state once, at a model swap. The system itself survived 12 deliberate kills, 24 restarts, and a VM stop with 93 seconds of downtime.

G.2 Label-free learning of a silent change

On one AppWorld stream an undocumented change cost bare Claude Opus 5.5 9.0 points of task completion and 23.1 of scenario completion. In the silent change studied here, an explicit page index k served page $k + 1$, while a call with no index served page 0. Nothing in the documentation, and nothing within one episode, separates that response from a short or empty page. The label-free claim learner recovers the change across episodes. It generalizes claims such as ‘explicit page_index=0 skips the first page, on every API with page_index’, which 48 of 48 evidence episodes across 30 APIs supported, and it admits a claim once the agent’s own observations confirm it. On 173 post-change tasks with three repeats it reached 95.2, against 84.4 for the diligence instruction and 71.3 for bare Claude Opus 5.5: +10.8 [6.6, 15.2] points over the instruction. Label-free ACE, run here, reached 94.8. Learning across episodes is worth 10 to 11 points over the instruction in both designs, and the claim learner reached it with 292 auxiliary model calls against 1,091 for label-free ACE.

G.3 The metamorphic verification instruction

A silent drift returns a well-formed answer with the wrong value, and an agent that takes the answer at face value proceeds. A fixed 533-word verification instruction, written from the metamorphic-testing literature for web APIs (Chen et al., 2018; Segura et al., 2018), asks Claude Opus 5.5 for checks that expose such a change without naming it: first-use contract tests through independent mechanisms, equivalence across two endpoints that report the same quantity, round-trip reads after writes, and a second look at empty, short, duplicate, or silently successful results. It contains no app, API, or drift content. The held-out drift classes, changed defaults for omitted arguments and amounts scaled by 100, were sealed before the text was frozen. On them the instruction scored 88.3%, against 81.6% for the four-sentence diligence instruction and 77.2% bare: +6.6 [+3.7, +9.9] points over diligence (216 tasks $\times$ 3 repeats; paired bootstrap; 29 tasks better, 8 worse; sign test $p = 0.0008$). With no learning, it came within +1.1 [−1.5, 3.9] points of label-free ACE. The instruction hands Claude Opus 5.5 checks to run, and it recovers 11.1 points over bare where a request for diligence recovers 4.4.

H Extended results: verification capital

H.1 Definitions and end-state agreement

A task instance T is attempted repeatedly. Attempt a leaves an end state $s(a)$ and receives an official outcome $y(a) \in \{0, 1\}$, with $y = 1$ for correct. A verifier returns a flag $f(a)$ and must not read $y(a)$. Catch is $\Pr[f = 1 \mid y = 0]$, false fire is $\Pr[f = 1 \mid y = 1]$, and Youden’s $J = \text{catch} - \text{false fire}$ (Youden, 1950); differences ΔJ are in points. End states are fingerprints: on τ^2 -bench, a hash of the domain database after a strict replay of the attempt in a fresh environment, which agreed with the official database check on all 7,617 attempts that have one; on AppWorld, a byte fingerprint of the attempt’s state changes. A *novel* state is a byte-distinct end state at its first appearance in a stream. Novelty sets the denominator. On AppWorld, 67% of failures repeat an earlier end state exactly, and 27,133 correct attempts reach only 843 distinct correct states, so false fire per correct attempt and per novel correct state differ by more than an order of magnitude. Both are reported.

Definition H.1 (Verification capital). After window t , a harness holds $V_t = (E_t, Y_t, C_t)$: E_t , the end states of the attempts in $W_1, \dots, W_t$, indexed by task; Y_t , their official outcomes; and C_t , the admitted checks, at most one per template. A verifier built from V_t checks each attempt in W_{t+1} before any outcome in W_{t+1} is read.

Definition H.2 (Accrual and depreciation). For each template τ with a failure in window $t + 1$ that its check c_τ did not flag, a writer proposes a candidate c' . A gate admits c' only if, on attempts from windows $\leq t + 1$, it fires on no success of τ , fires on at most 2% of sampled successes of other templates, and flags at least $\max(1, \text{flags of } c_\tau)$ past failures of τ ; a check is revoked at its third false fire. A copy frozen at t_0 applies C_{t_0} to every later window. Its depreciation is the difference in catch on later novel failure states between the learning library and the frozen copy.

End-state agreement. For an attempt a , let $P(a)$ contain $k = 4$ other attempts at the same task and $n_P(x) = |\{b \in P(a) : s(b) = x\}|$. Agreement flags a exactly when $n_P(s(a)) < \max_x n_P(x)$ (Algorithm 8). The attempt is excluded from its own panel, an attempt tied for the most common state passes, and an empty panel passes.

Proposition H.3 (What agreement sees). *Let a task have a single correct end state s^* , and let $P(a)$ be non-empty. (i) A failure whose end state no peer reached is flagged. (ii) A failure whose end state is the most common state of its panel passes. (iii) A correct attempt is flagged only if some wrong end state occurs in $P(a)$ more often than s^* . (iv) A failure that leaves the end state s^* passes every end-state check.*

Agreement turns correctness into a count, and the count requires one condition: correct attempts coincide while failures scatter. A task’s failures may outnumber its successes, provided no single wrong state outnumbers the correct one in the panel.

H.2 Agreement on held-out τ^2 -bench

The held-out τ^2 -bench split (Barres et al., 2025) holds 60 retail and airline tasks that no verification study had used, each with one correct end state. The evaluation set holds up to 10 Claude Opus 5.5 attempts per task: 600 attempts, 68 of them failures. The zero-shot judge is Claude Opus 5.5 with adaptive thinking at high effort; it sees the policy, the customer’s messages, the database changes, and the relevant start records, and returns a binary verdict. The peer judge also sees the changes made by the four attempts in the agreement panel, labeled unverified, so agreement and the peer judge hold the same information.

Agreement with the same four attempts catches 33 of the 68 failures; Claude Opus 5.5 judging zero-shot catches 6, and shown the same four, 21 (Table 12). Against the information-matched judge, the preregistered difference is $\Delta J = +16.5$ [+5.4, +32.1]: 17.6 more points of catch for 1.1 more points of false fire. Averaged over 50 panels, agreement exceeds the zero-shot judge by a preregistered +40.0 [+26.0, +61.0]. Drawing the four peers from any model raises agreement’s catch to 55.8%; its J of 0.527 is the highest of any done-check measured on these tasks, and it needs neither labels nor a model call.

Exact comparison sees what judgment misses. Airline task 29 fails by omission. The customer holds an insured round trip paid at \$686 and wants nonstop flights to a different airport on the same dates. A destination cannot be changed, so the agent books a new reservation and stops; the correct end state also cancels the original reservation under

its insurance and refunds the \$686. The zero-shot judge wrote that the new reservation ‘matches what the customer chose and confirmed’. The peer judge, shown three peers that had cancelled the original, wrote that the agent ‘made no unrequested changes’. The missing cancellation appears nowhere in the attempt’s change list, which is all either judge read; the attempt’s own state matched no peer’s, and agreement flagged it (Proposition H.3, case i). In retail task 74 the attempt refunded a \$218.84 price difference to the PayPal account that paid for the order, a plausible default, where correct attempts refund it to the customer’s credit card. The zero-shot judge accepted it. Of the 85 recorded Claude Opus 5.5 attempts at the task, the 75 correct ones reach one end state, and the 10 failures scatter over four others. Among the 41 failures away from Claude Opus 5.5’s most common end state at their task, agreement catches 80.5%, the peer judge 51.2%, and the zero-shot judge 14.6%. The peer judge saw the same panel as agreement, so the gap comes from the comparison.

The other 27 failures are repeated mistakes, case (ii). Design diversity reaches them: panels drawn from Claude Haiku 4.5 catch 71.1% of the 27, because a panel of another model holds Claude Opus 5.5’s error only when that model makes the same one. Agreement also sharpens as attempts accumulate. On the τ^2 -bench training record of 1,133 attempts at 104 tasks by six actor models, 618 of them failures, false fire falls from 17.6% with one earlier attempt to 9.9% with four and 8.0% with eight, while catch stays near 93%. A harness that uses the judge until three earlier attempts exist and agreement afterward catches 93.7% at 5.6% false fire, $\Delta J = +18.0$ [+12.3, +23.8] over the judge.

Agreement labels teach a learner. On the AppWorld record, labeling an attempt a success when it agrees with the consensus of other attempts never called a failure a success, and called 7.5% of successes failures, while the agents’ own completion flags marked 3,309 of 3,607 failures as done. A prequential Claude Opus 5.5 precedent judge fed one-sided agreement labels reached $J = 0.880$ on novel states (96.3% catch at 8.3% false fire per novel correct state), against $J = 0.240$ from completion flags, a difference of +64.1 [+52.9, +70.3].

H.3 The learned check library

A check is a deterministic Python function of the task instruction, the supervisor’s identity, the start and end databases, and the answer the agent submitted. It returns whether it applies, a flag, and a sentence of evidence with the values it compared. It reads neither the agent’s trajectory nor the actor’s identity, and it never sees the answer key. The writer’s prompt instructs Claude Opus 5.5 to parse each instance’s parameters from the instruction, re-derive the correct outcome from the start state, and test the end state against it; a check is an executable specification of one template.

Proposition H.4 (Actor invariance). *Two attempts at the same instance with identical end states and identical submitted answers receive identical verdicts from every check.*

The AppWorld record (Trivedi et al., 2024) holds 31,434 graded train and dev attempts on 48 templates with three instances each, 3,607 of them failures, by Claude Haiku 4.5, a Qwen 27B open-weight model, Claude Opus 5.5, Claude Opus 4.8, and GPT-6 Astra. Replayed in start-time order and cut into 38 windows that close every 50 failures up to 200 and every 100 after, each window runs four steps (Algorithm 7): CHECK writes a verdict for each attempt before any outcome of the window is read; REVEAL reads the outcomes and revokes a check at its third false fire; LEARN mints a check for an uncaught template or matures an existing one; and GATE tests each candidate on past windows only. Each verdict is a forecast scored on outcomes its verifier had not seen (Dawid, 1984). In the pass/fail run, every grader-derived field is removed from the writer’s prompts; a scan of all 7,398 prompts found no grader artifacts.

Anatomy of a check. For the template ‘follow all artists of all *genre* songs in my playlists’, the check parses the genre from the instruction, collects the artists of each song of that genre in the user’s playlists in the start state, and compares that set with the artists the user follows in the end state. It flags an expected artist not followed, a previously followed artist removed, and a newly followed artist outside the expected set, each with the ids it found (Appendix O.2). The template first failed in window 4, when a Claude Haiku 4.5 attempt left one reggae artist unfollowed; Claude Opus 5.5 minted the check from that failure, and the gate admitted it after it stayed silent on 23 past successes. In window 8 a Claude Haiku 4.5 attempt followed all five classical artists correctly and then submitted `answer=5` to an action task, and the check passed it. From the trajectory, the end state, and two correct attempts, with no grader report, the writer inferred the error, added a fourth condition (an action task submits no answer), and widened the parser; the revision caught all 5 of the template’s past failures, where the old check caught 4. After the population changed, the check caught all 6 later failures, three of them Qwen attempts that followed the right artists and then submitted answers. A check written from a Claude Haiku 4.5 error caught Qwen errors of the same kind, and no version fired on any of the template’s 303 correct attempts.

Coverage. After its first 200 failures, the pass/fail library catches 90.1% of novel failure states in windows 5–38 and flags 1.1% of novel correct states; per attempt, it catches 96.4% of failures and fires on 0.026% of correct attempts. A novelty flag catches every failure at 2.35% per-attempt false fire but has $J = 0$ on novel states; the library separates novel failure states from novel correct ones and fires on correct attempts at one ninetieth of the flag’s rate. Past pass/fail outcomes carry nearly all of the signal: showing the writer the grader’s report adds only +2.4 [+0.8, +4.4] points of novel-state catch. The library is compact, one deterministic function per template, 48 in all from window 8 on.

Rivals. At matched false fire, an ACE-style verification playbook with the same feedback catches 55.5% of novel failure states (Table 13); at its native verdict, 87.2%, still below the library, at 2.9 times its false fire. The strongest learning rival is a precedent judge, Claude Opus 5.5 reading labeled earlier attempts of the same template: over windows 29–38, at matched false fire, the library catches 89.9% and the judge 82.6%, a difference of +7.3 [+1.5, +13.4]. The cost difference is structural. A check is a Python function that runs without a model call; the library’s two arms used 5.7M tokens over the whole stream, the ACE playbook 117M, and the precedent judge 123M on its evaluation items alone.

H.4 Depreciation

Frozen copies depreciate at every freeze point (Figure 7b). The copy frozen after 200 failures catches a preregistered 24.0 [+16.4, +32.8] points less than the learning library on later novel failure states. Copies frozen after 800 and 1,800 failures catch 12.0 [+5.9, +18.0] and 10.6 [+4.3, +19.2] points less, and a copy frozen after 50 failures 76.5 less. The size of the stock does not explain the curve. From window 8 the library held one check per template and added none, so the copies frozen after 800 and 1,800 failures hold every check the learning library holds, and their depreciation is forgone revision alone. After window 10 the learning library admitted 43 maturations and 3 repairs, each answering a failure the old check had missed or a correct attempt it had flagged. Depreciation is steepest when the population changes. In windows 33–38 a Qwen actor supplies 392 of 607 failures, against at most 6% before; the learning library answered with its largest burst of revision since windows 5–10 and caught 87.6% of the bin’s novel failure states, while the frozen copies fell to 44.4–68.7%.

H.5 Transfer across actors and anticipation

Claude Opus 5.5 fails on only 0.8% of its AppWorld attempts and Claude Opus 4.8 on 1.8%, so their own failures form a thin record, and a verifier for them must draw on weaker models’ failures. A no-Claude Opus 5.5 arm learned from no Opus-family attempt; Claude Opus 5.5 still wrote its checks, from Claude Haiku 4.5, Qwen, and GPT-6 Astra failures, all but 15 of 3,407 from Claude Haiku 4.5 or Qwen. On the same 144 instances, those checks caught 30 of the 32 novel Opus-family failure states and fired on none of the 7,185 Opus-family correct attempts. Proposition H.4 explains the result: a check cannot tell which actor produced a state, so it flags an Opus failure whenever that failure violates the requirement the check re-derives.

A failure-only library cannot catch a template’s first failure, because it has no check until a failure has been missed. A correct attempt carries what a check needs, the end state the template requires. An anticipation arm writes a check from up to two earlier correct attempts per template, before any failure. On 110 instance first failures it raises catch from 20.0% to 75.5%, a preregistered +55.5 [+44.6, +65.6], and it fired on none of 7,915 correct attempts. On the 44 instances whose own correct attempt the writer was not shown, catch rises from 29.5% to 65.9%, because the seeded check encodes the template’s requirement.

I Generality, cost, and speed in detail

I.1 Generality across domains and models

The harness designs of this paper were not tuned to one benchmark. The same designs set records or beat named rivals across nine domains and five model families (Table 29), and the law’s comparisons span crash, drift, handoff, and continuous-operation instruments on three model classes.

I.2 Cost and speed

Several components buy more than a frontier model can buy with more of itself, at a fraction of the cost.

Table 29: The same harness designs win across domains and model families.

Domain	Instrument	Harness design	Result
Combinatorics	Erdős minimum overlap	shared archive; long-running agent	below every published bound; $20.3\times$ the Arena’s minimum improvement
Harmonic analysis	first and signed autocorrelation	shared archive; long-running agent	C_1 $305\times$ the minimum improvement; C_3^+ below MILO and all 43 Arena entries
Discrete geometry	Packomania, Friedman’s containers, 14 points in $\mathbb{R}^3$	long-running agent	306 square sizes above every published value; 15 container sizes above the displayed records; 14-point ratio below FICO Xpress
Systems kernels	Anthropic’s performance take-home	shared archive	914 cycles vs. CORAL 948
GPU kernels	KernelBench, strict fp32	verified technique cards	$3.67\times$ vs. ACE $1.95\times$
Heuristic planning	AtCoder Heuristic Contest 026	shared archive	ahead of CORAL from the first hour
Production software	HORIZON	evidence-only supervision	0 of 2,880 tickets below bare; handover $0.435 \rightarrow 0.911$
API agents	AppWorld	recovery, claim learning, checks	SGC $79.1 \rightarrow 92.1$; 95.2 vs. ACE 94.8; 90.1% catch
Customer service	τ^2 -bench	end-state agreement	48.5% catch vs. 30.9% for the frontier judge

Model families: Claude Opus 5.5 throughout; Claude Opus 4.8 (seed card $0.75\times \rightarrow 1.83\times$); Claude Haiku 4.5 (receipts 96.6%); GPT-6 Astra (take-home successor to 906 cycles); Qwen 27B (archive 4,901 vs. 5,979 isolated; handover held at 0.911).

- **Verification with no model call beats the frontier model’s judgment.** End-state agreement caught 48.5% of Claude Opus 5.5’s held-out failures; Claude Opus 5.5 judging the same attempts with the same evidence caught 30.9%, and zero-shot 8.8%.
- **Learned checks beat a learned playbook at one twentieth of the tokens.** The check library caught 90.1% of never-seen failure states, against 55.5% for an ACE-style verification playbook at matched false fire; the library used 5.7M tokens over the whole stream, the playbook 117M, and a precedent judge 123M on its evaluation items alone.
- **A rule with no model call doubles production.** Committing the deployed tree at each ticket raised a Qwen3.5-27B successor’s production correctness from 0.435 to 0.881, 93% of the best harness’s lift.
- **Exactly-once effects.** Write-ahead receipts left no duplicate or lost payment across 1,328 random kills, and under a crash after every step they preserved 96.6% of Claude Haiku 4.5 episodes, where bare, told-of-the-crash, and plain write-ahead-log arms completed 0 of 58.
- **Less context, more speed.** The technique library served 18,000 characters per episode against ACE’s 305,000 and reached $3.67\times$ against $1.95\times$. The claim learner matched label-free ACE with a quarter of the auxiliary calls.
- **Hours, not weeks.** The shared archive moved the C_1 record 12.9 times as far in under four hours as the Einstein Arena’s agents had in the previous seven weeks. One agent passed Liang’s five-day automated packing campaign at all 306 sizes in two runs of 6 and 5.3 hours.

J Formal Results and Proofs

J.1 Exact certificates

Lemma J.1 (Knot reduction). *Let h be a step function on n cells of width w . Then $O_h(t)$ is linear between consecutive multiples of w , so $\sup_t O_h(t) = \max_k C_k$ with $C_k = w \sum_i h_i(1 - h_{i+k})$, the sum over $1 \leq i, i+k \leq n$. Likewise, for f with heights a_i on n cells of $[-\frac{1}{4}, \frac{1}{4}]$, $(f * f)(t)$ is linear between the knots $t_k = -\frac{1}{2} + (k+1)/(2n)$, where it equals $c_k/(2n)$ with $c_k = \sum_{i+j=k} a_i a_j$. With $S = \sum_i a_i$,*

$$\frac{\|f * f\|_\infty}{\|f\|_1^2} = \frac{2n \max_k c_k}{S^2}.$$

Proof. For t between consecutive knots, each pair of cells overlaps on an interval whose length is affine in t , and the integrands are constant on cells. Both functions are therefore affine on each such interval, and a piecewise-linear function attains its supremum at a breakpoint. The identity $\int f = S/(2n)$ gives the ratio. $\square$

Lemma J.2 (Carry-free squaring). *Let $A_0, \dots, A_{n-1} \geq 0$ be integers and B an integer with $B > \max_i A_i \cdot \sum_i A_i$. Each coefficient of X^2 , for $X = \sum_i A_i B^i$, lies in $[0, B)$. The base- B digits of X^2 are therefore exactly the convolution coefficients $c_k = \sum_{i+j=k} A_i A_j$.*

Proof. $c_k = \sum_{i+j=k} A_i A_j \leq \max_i A_i \cdot \sum_j A_j < B$, so no coefficient carries into the next digit. $\square$

For the C_1 construction the condition holds with $B = 2^{216}$. Three moment identities check the squaring globally:

$$\sum_k c_k = S^2, \quad \sum_k k c_k = 2S \sum_i i A_i, \quad \sum_k k^2 c_k = 2S \sum_i i^2 A_i + 2 \left(\sum_i i A_i \right)^2.$$

A second path uses number-theoretic transforms modulo seven primes below 2^{31} , whose 217-bit product exceeds the coefficient bound of 2^{211} , so the Chinese remainder theorem reconstructs each coefficient exactly. For M , put $B_j = 2S - nA_j$. Feasibility $0 \leq h_j \leq 1$ becomes $A_j \geq 0$ and $B_j \geq 0$, and Lemma J.1 gives $\max_t O_h(t) = \max_k \sum_i A_i B_{i+k} / (2S^2)$. The C_3^+ checker squares a carry-free encoding of the signed integer heights, with denominator 10^{21} , and checks the coefficient-sum identity and eight direct sums, including both extrema.

Lemma J.3 (Grid completeness). *Assign each circle to the cell $(\lfloor X_i/g \rfloor, \lfloor Y_i/g \rfloor)$ of a square grid of integer width $g = 2 \max_i R_i$. Two circles whose cells differ by at least two in either coordinate do not overlap. Checking each circle against its own cell and the eight adjacent cells therefore checks each possibly overlapping pair.*

Proof. A cell difference of at least two along an axis forces a coordinate separation greater than $g \geq R_i + R_j$. $\square$

The lemma holds for unequal radii and assumes nothing about the arrangement. At $N = 10,000$ it leaves 113,179 candidate pairs of 49,995,000. Above $N = 2500$ a faster checker sorts the circles by X and sweeps each one forward until $X_j - X_i \geq R_i + \max R$; an overlapping pair has $X_j - X_i < R_i + R_j$, so the sweep cannot stop before it. The fast checker matched the pinned evaluator on all 778 tests that evaluator completed and on 174 adversarial cases, and at $N = 2401$ it takes 2.5 seconds against 116. The six Packomania coordinate files rechecked in detail all overlap slightly in their 12-decimal presentation (at $N = 137$ the worst squared slack is -3.25×10^{-13}); the files released here are feasible as written, with smallest wall slack $1081 \cdot 10^{-18}$ across the first run's files.

Lemma J.4 (Mirror identity). *For admissible h and $0 < k < 2$, $O_h(k) - O_h(-k) = \int_0^k h - \int_{2-k}^2 h$. If $O_h(k) = O_h(-k)$ for all k in an interval I , then $h(x) = h(2-x)$ for almost every $x \in I$.*

Proof. Write $O_h(t) = \int_{[0,2] \cap [-t,2-t]} h - \int h(x)h(x+t) dx$. The autocorrelation term is even in t . The first term is $\int_0^{2-k} h$ at $t = k$ and $\int_k^2 h$ at $t = -k$, and subtracting the common $\int_k^{2-k} h$ gives the identity. Differentiating $\int_0^k h = \int_{2-k}^2 h$ in k gives the symmetry. $\square$

J.2 Transfer of carried experience

Proof of Proposition E.1. With the two approach classes a_S and $\bar{a}$, $\rho_R^z(\bar{a} | p) = 1 - \rho_R^z(a_S | p)$, so the adoption sum in Equation (6) equals $\Delta_R(p) [\mu_R^0(a_S, p) - \mu_R^0(\bar{a}, p)] = \Delta_R(p) G_R(p)$. Setting the within-approach term to zero gives $\tau_R(p) = \Delta_R(p) G_R(p)$, which has the sign of $G_R(p)$ when $\Delta_R(p) > 0$. $\square$

Proof of Corollary E.2. Substituting $\mu_R^0(a_S, p) = v_Q(p)$ and $\mu_R^0(\bar{a}, p) = v_R(p)$ into G_R gives $G_R(p) = v_Q(p) - v_R(p)$, which is positive for a reader below its writer and negative for a reader above it; Proposition E.1 then gives the sign, and $|\tau_R(p)| = \Delta_R(p)(v_R(p) - v_Q(p))$ for a reader above its writer. $\square$

Proof of Corollary E.3. For a subset T containing an approach-carrying item, serving T raises adoption of a_S , so the adoption shift under T is positive, and the within-approach term is zero by hypothesis. Proposition E.1 applied to the store T gives an effect with the sign of $G_R(p)$, which does not depend on T . A subset with no approach-carrying item does not raise adoption, so the proposition does not apply to it. $\square$

Proof of Corollary E.4. Where $G_R(p) \leq 0$ the reader’s own comparison does not favor a_S , so by hypothesis it does not adopt a_S and $\Delta_R(p) = 0$. Where $G_R(p) > 0$, adoption can only rise, so $\Delta_R(p) \geq 0$. In both cases $\Delta_R(p)G_R(p) \geq 0$. $\square$

J.3 Supervision and verification

Proof of Proposition F.3. By Equation (8), a discard requires some check with $c(a) = \text{fail}$. If every check returns pass or unknown on a , no such check exists, so the first clause fails and the attempt is deployed, whatever the outcomes on b . $\square$

Proof of Proposition F.4. A restore to HEAD sets the tracked tree to the most recent commit. If R_{t-1} was committed at the start of ticket t , that commit is R_{t-1} , and only edits made during ticket t are lost. Otherwise the most recent commit is R_h , the last tree the agent committed, and the restore returns the tree to R_h , reverting every deployment made after it, those of tickets $h + 1$ to $t - 1$. $\square$

Proof of Proposition H.3. (i) If no peer reached $s(a)$, then $n_P(s(a)) = 0$ while $\max_x n_P(x) \geq 1$ for a non-empty panel, so a is flagged. (ii) If $s(a)$ is a most common state, $n_P(s(a)) = \max_x n_P(x)$, and a passes. (iii) A correct attempt has $s(a) = s^*$, and it is flagged only if some state x has $n_P(x) > n_P(s^*)$; since $x \neq s^*$, x is a wrong state. (iv) A failure with end state s^* is indistinguishable by its end state from a correct attempt, so every function of the end state treats it as one. $\square$

Proof of Proposition H.4. A check is a function of the instruction, the supervisor’s identity, the start and end databases, and the submitted answer. Two attempts at the same instance share the instruction, the supervisor, and the start database; if they also share the end state and the answer, every argument of every check is equal, so every verdict is equal. $\square$

Proposition J.5 (Accrual keeps recorded coverage). *On the gate’s samples, an admitted revision of c_τ flags at least as many past failures of τ as c_τ did, and fires on none of the past successes of τ .*

Proof. These are the gate’s admission conditions in Definition H.2, read as properties of the admitted check. $\square$

K Algorithms

Algorithm 2: One session of the society on the shared archive. Three Claude Opus 5.5 agents per problem; sessions last at most 4 hours.

State: an append-only, HMAC-chained archive of verified attempts (score, parent, cited items, content-addressed source, producing tool), insights, failure constraints, and tools.

- 1: The harness assembles a briefing of at most 9,000 characters: the team best; a parent drawn by Thompson sampling over clade credit (each candidate’s descendants’ verified gains, normalized by the median of the last 12 parent-based verified gains, with a bounded penalty for distance from the island best); a direction rotated each session; at most 10 insights, 4 recent results, 6 constraints, 8 tools, and 4 offers from the other problems.
 - 2: The agent writes and runs optimizers in its sandbox, copies other agents’ constructions with `kx source`, runs promoted tools from the shared directory, and submits candidates through a command-line client.
 - 3: The evaluator scores each candidate in a separate process; the archive appends the attempt, its score, parent, and cited items, chained to the previous event.
 - 4: The harness re-scores new bests, invalidates attempts that rest on a refuted attempt, promotes a tool or insight when an attempt citing it verifies, and announces a teammate’s new best at the agent’s next submission.
 - 5: The session ends on exit or after 4 hours; the next session starts at step 1. The harness declares a plateau after 15 attempts without gain, integrates every 6th session, and consolidates every 40 attempts; Claude Opus 5.5 compiles insights and Claude Haiku 4.5 distills failures.
-

Algorithm 3: **The lead-thread society.** One lead and two explorers per problem, with a compute pool.

- 1: *Lead.* One agent per problem keeps one conversation for the run: its session cap is the remaining time plus one hour, resumes are unlimited, and a crash resumes the same session. It receives no rotated direction, and its notebook is mirrored read-only to the explorers every 5 minutes.
 - 2: *Explorers.* Two agents hold a direction for two sessions. Each takes its parent from its own line or from the lead’s recent attempts (the lead’s best and its attempts of the last 3 hours), never the lead’s best itself; two consecutive submissions on the lead’s recent attempts draw a warning.
 - 3: *Compute pool.* Long jobs run outside sessions in their own cgroup. A job’s CPU weight is 100, multiplied by 4 if the lead submitted it and by 3 if it descends from the team best, re-evaluated every 30 seconds. The pool parent and the agents’ cgroup are each capped at twice the problem’s fair share, and a problem’s agents and pool together are capped at what the single agent or three society agents could take.
 - 4: *Routing.* An explorer’s new team best goes in full to the lead only. Explorers receive a digest at most once per 30 minutes, worded ‘keep your own line, do not stop your jobs’; the lead receives an explorer update at most once per explorer per hour.
-

Algorithm 4: **Defect-move basin hopping with exact finalization.** The packing agent’s pipeline at each size.

Start from the best public packing at N (Packomania or Liang), or from a neighboring size for the cascade. Return a packing only if its exact sum exceeds each published value at N .

- 1: Apply L-BFGS-B penalty relaxation, increasing the penalty weight from 10^1 to 10^6 , and shrink uniformly to feasibility. Use sequential LP on linearized distances, an inner approximation with feasible iterates and a monotone objective, then Newton’s method on the active-set KKT system. The examined packings are isostatic with $3N$ active constraints; Newton converges in one or two steps to 10^{-16} .
 - 2: Select a circle whose radius is far from the median and re-randomize the disc of circles around it. Relax the whole packing with the penalty method and solve the KKT system on the new overlap set directly; this solve, used in the second run, is three times faster than LP polishing. Keep improvements and repeat.
 - 3: Transfer between sizes: delete each circle of the best $(N+1)$ -packing in turn, or insert a circle into the 16–24 largest holes of the best $(N-1)$ -packing. Resume the defect search from these starts.
 - 4: Refine the KKT solution and shrink radii by 10^{-15} , doubling the shrink until exactly feasible. Write 17-significant-digit decimals and score with the hash-pinned exact evaluator. Claim only if the exact sum exceeds the published value.
-

Algorithm 5: **Per-ticket procedure of guard.**

Input: ticket τ_t ; deployed base $b = R_{t-1}$ and production copy D_{t-1} ; check set C (the visible examples of τ_t , run with and without restarts; the visible examples of earlier tickets; start-up and compatibility on a copy of D_{t-1}).

Output: the tree R_t to deploy.

- 1: Commit b to the agent’s git (I4).
 - 2: Run the model on the bare system prompt and τ_t , byte for byte (I1); call its result a .
 - 3: Run C on a in one sandboxed process; earlier tickets’ examples count only if they pass on b .
 - 4: **if** every outcome is pass or unknown **then** deploy and commit a ; **return** a (I3).
 - 5: Return each failure (input, expected and actual output, or stack trace) to the agent’s own session for one repair round; rerun C on the repaired a (I2).
 - 6: **if** Equation (8) holds for (b, a) **then** $R_t \leftarrow b$ **else** $R_t \leftarrow a$; ties keep a .
 - 7: Deploy and commit R_t (I4). Nothing written for later is shown to the trusted model (I5).
-

Algorithm 6: Crash recovery with write-ahead intents and receipts.

State persisted outside the model: a record of committed steps; a write-ahead intent for the step in flight; a checkpoint of the Python session; receipts from the tool server’s execution journal.

- 1: Before an external call, write an intent naming the call and its identifier, marked as an external effect and uncertain.
 - 2: Execute the call; the tool server’s journal records a receipt of the executed action and its response.
 - 3: On success, commit the step to the record and clear the intent.
 - 4: **After a crash:** restore the session checkpoint and resume from the last committed step. Settle the uncertain intent from the receipts: if its identifier has a receipt, the effect happened, so commit it once and return the recorded response without running the call again; otherwise the effect did not happen, so propose the call again.
-

Algorithm 7: Prequential replay of the check library. Verdicts in each window are written before its outcomes are revealed.

Input: graded attempts ordered by start time and cut into windows $W_1, \dots, W_{38}$ that close every 50 failures up to 200 and every 100 after; the writer model Claude Opus 5.5; the gate of Definition H.2 with samples of up to 400 past successes of the template, 150 past successes of other templates, and 300 past failures of the template.

State: a library $\mathcal{L}$ holding at most one gated check per template.

- 1: $\mathcal{L} \leftarrow \emptyset$.
- 2: **for** $t = 1, \dots, 38$ **do**
- 3: **CHECK:** for each attempt $a \in W_t$, flag a if the check in $\mathcal{L}$ for a ’s template fires on its end state; record the verdict.
- 4: **REVEAL** the official outcomes $y(a)$ for $a \in W_t$; update each check’s catches, misses, and false fires; revoke a check on its third false fire after admission.
- 5: **LEARN:** for each template with a failure in W_t that no check caught, Claude Opus 5.5 writes a new check (*mint*) or rewrites the template’s check (*maturation*) from past outcomes, trajectories, and end states, with no grader text; a false fire triggers a *repair*; a rejected candidate is retried with the gate’s feedback.
- 6: **GATE:** admit a candidate only if, on past data alone, it fires on none of the template’s past successes and on at most 2% of other templates’ past successes, and catches at least as many of the template’s past failures as the check it replaces; admitted checks serve from W_{t+1} .

A *frozen copy* is $\mathcal{L}$ after window t_0 , never updated again. The anticipation arm also mints from up to two correct attempts of a template before its first failure.

Algorithm 8: End-state agreement done-check. No label and no model call is used.

Input: attempt a at task T ; a panel P of four other recorded attempts at T ; the end-state map s (on τ^2 -bench, the hash of the domain database after a strict replay).

Output: flag or pass.

- 1: For each end state x , let $n(x) = |\{b \in P : s(b) = x\}|$; the attempt a is never in its own panel.
 - 2: Let $n^* = \max_x n(x)$.
 - 3: **if** $n(s(a)) < n^*$ **then** flag a , **else** pass a . A state no peer reached is always flagged; an attempt tied for the most common state passes; an empty panel passes.
-

L Protocols and Fixed Texts

L.1 Fixed control instructions

The four-sentence AppWorld diligence instruction, appended to the standard prompt where a playbook would go:

Working principles (general, not specific to any app): be thorough and verify against the live app state. When an API returns an empty, missing or surprising result where you expected data, do not take it at face value: investigate before concluding, check the arguments you passed, and try reasonable alternatives. Make sure you have collected all of the data the task needs. Before calling `complete_task`, re-check that your answer and every required action are correct and complete.

The HORIZON diligent paragraph, fixed before evaluation and appended to the system prompt on every ticket for every model:

Working principles (general, not specific to this project). Be thorough and verify against the real system before you finish. Run the project’s existing tests and the ticket’s examples, and make sure earlier behaviour still works and the service still starts with its existing data. Never discard or revert changes already in the working tree that you did not make in this session. When the ticket is done and verified, commit your changes with git.

The crash notice given to the bare model told of a crash states that the agent process crashed and was restarted, that the conversation could not be restored, how many steps had run before the crash, that the app state persists and every API call that ran before the crash took effect, and that the Python session was restarted. The metamorphic instruction is a frozen text of 533 words whose content is summarized in Section 5.8.

L.2 Judge and check-writer prompts

The zero-shot τ^2 -bench judge receives the following system prompt:

‘You audit the work of an AI customer-service agent after a chat has ended. The agent had to follow the domain policy and do what the customer validly asked for, using tools that read and write the database. You see the policy, the customer’s messages (not the agent’s), every database change the agent made, and the relevant database records as they were before the chat. Decide whether the agent’s work is **WRONG**: the end state violates the policy, misses or gets wrong something the customer validly requested, or contains a change the customer did not request or confirm. If the work is consistent with the policy and the customer’s requests, it is **OK**.’

The peer judge uses the same prompt with this passage before the answer request, followed by each peer’s changes, capped at 4,000 characters:

‘Other attempts at the same customer request (unverified). The same customer request was also handled in k other independent chats, by other agents. Below are the database changes each of those attempts made. Their correctness is **UNKNOWN**: any or all of them may be wrong. They are evidence about what the request involves, not ground truth. The customer’s wording may differ between chats.’

The check writer’s system prompt begins:

‘You write verification probes for AppWorld, a simulated world of apps . . . A probe is a small deterministic Python function that inspects the world’s database **AFTER** the agent finished a task and decides whether the agent’s work is **WRONG**. . . It never sees the agent’s reasoning and never has the answer key.’

L.3 What the successor read: the rendered store

Figure 13 shows what the successor read on L2_074.

```

## Kernel memory (items verified by the protected evaluator on training problems; may not all apply)
### Lessons about this environment (from official checker verdicts and verified outcomes)
- Checker rule: At least one of your own kernels must actually launch during forward; a pure torch path
(even with a compiled but unused kernel) is rejected as no_custom_kernel.
- Process: never SUBMIT code that has not passed ACTION: CHECK in this episode. [...]
[12 further lessons]
### Pitfalls seen on earlier conv_transpose problems (wrong results or slower than eager)
- Pitfall: Don't rewrite conv_transpose as flipped-weight F.conv2d plus a separate elementwise kernel; the
per-call weight flip and extra full-output pass make it slower. Fuse bias into a real kernel, or skip
pointless copies.
[4 further pitfalls]
### Verified recipes for conv_transpose problems (speedups measured by the protected evaluator)
- [1.32x on ConvTranspose3d_BatchNorm_Subtract] Keep ConvTranspose in cuDNN via torch. Fuse everything
after it into two Triton passes over the contiguous (NxC, S) view. [...]
- [1.26x on ConvTranspose3d_Sum_LayerNorm_AvgPool_GELU] Keep ConvTranspose in cuDNN via torch and fuse only
the epilogue in Triton. [...]
[3 further recipes: 2.29x, 1.68x, 1.06x]
### A verified kernel for a related conv_transpose problem (L2_078 ConvTranspose3d_Max_Max_Sum, 1.00x):
adapt what applies
@triton.jit def fused_kernel(x_ptr, out_ptr, N, C, D, H, W, OD, OH, OW, total, BLOCK: tl.constexpr): [...]
    c = self.conv_transpose(x).contiguous() [...]
[1 further kernel: L2_089, 1.06x]

```

Figure 13: **What the successor read on L2_074.** The rendered store, 10,364 characters: 14 lessons, 5 transposed-convolution pitfalls, 5 recipes, and 2 verified kernels, in the harness’s own headers. Items are verbatim; bracketed text marks omissions. The recipe for L2_074 itself was removed before serving. All five recipes route the convolution through cuDNN, and both kernels call the library convolution before their Triton code.

The removed recipe for L2_074, which the successor rebuilt clause by clause, reads in full:

```

id: rec-7d67d21ded    kind: recipe
label: ### Verified recipes for conv_transpose problems (speedups measured by the
protected evaluator)
text: [1.26x on ConvTranspose3d_LeakyReLU_Multiply_LeakyReLU_Max] Keep the
ConvTranspose3d itself in cuDNN via nn.ConvTranspose3d and fuse the entire epilogue
into one Triton kernel that reads the conv output once: LeakyReLU, per-channel scale,
LeakyReLU, MaxPool. Launch one thread per pooled output element over a flat index
(BLOCK=256, 1D grid). Decode n*c/od/oh/ow with div/mod, load the per-channel
multiplier once, and unroll the 2x2x2 window with tl.static_range. Use tl.where for
the slopes and a -1e30 init for max. Keep all math fp32, which stays within 1e-4.

```

L.4 Serving policies for the predecessor’s store

Table 30 lists the KernelBench experimental designs and their preregistrations.

Each episode runs four steps. A serving policy decides whether the reader sees the store and which items; items naming the current problem are removed. The selected items are rendered under headers by kind and placed in the opening prompt or, for late delivery, after the reader’s first local check with an instruction to keep its own code if faster. The agent writes, checks, and submits a kernel within five turns, and the protected evaluator scores it. Besides bare and show-all, the policies fall into three groups. *Selection* policies choose the items: provenance filtering hides recipes from readers that did not produce the underlying solution; a trained selector serves the items a ridge regression predicts will help; Opus+tables and Opus+digest let Claude Opus 5.5 choose items from per-item outcome tables or their digest; item gating serves only items whose verified speedup exceeds the reader’s own level. *Presentation* policies change how the items read: annotation tags each item with its measured effect for the reader; a reader-relative rendering frames the store as an older model’s floors; a numberless rendering removes the speedups. *Adoption* policies decide whether and when the reader sees the store: zero-shot self-gating, reader-level rules fitted to randomized outcomes, and late delivery. All selections were frozen before running.

Table 30: **KernelBench experimental designs and preregistrations.** All episodes used the same agent loop and evaluator. X: statistics recomputed independently from the recorded episodes. Times are UTC on October 9, 2026.

Design	Randomized	Problems	Episodes	Readers	Check	Preregistration (time UTC)
Randomized	whole store on/off	92	1,288	5.5, 4.8	X	a4aa790efe (06:23:43); run from 06:24:02
Confirmation	5 arms, 3 gate arms	28	1,120	5.5, 4.8	X	amendment f49ad3ccfe, before any confirmation episode
Head-to-head	9 arms, 6 of them forms of the store	10	720	5.5, 4.8	X	0eeef6288 + amendment 6cc4d0fa9e; run from 05:08
Store age	frozen snapshots	10	per snapshot	5.5, 4.8		amendments to the store-age preregistration
Technique library	10 arms	26	720	5.5	X	28c3d1c01e; amendment e2713dcb
Calibration	9 arms	37	317	5.5		db3e84226f; run from 13:23

L.5 HORIZON harness definitions

Table 31 defines the HORIZON harnesses.

Table 31: **HORIZON harnesses.** Model calls are those the harness makes itself. `commit`, `guard`, and `full` give the trusted model a bare-identical session; `diligent` adds its fixed paragraph.

Harness	What it adds to bare	Model calls	Discards work
<code>diligent</code>	a fixed generic diligence paragraph in the system prompt (control)	no	never
<code>commit</code>	commits the deployed tree to the agent’s git at the start and end of each ticket	no	never
<code>guard (v4)</code>	<code>commit</code> plus deterministic checks returned as evidence	no	hard evidence
<code>guard v5</code>	<code>guard</code> with a changed production answer as hard evidence that a ticket mention cannot waive	no	hard evidence
<code>full v1</code>	hard checks; on a successor: handover brief, mandatory check instructions, enforced checks written by the trusted model; keeps the best attempt by a weighted failure count	check writing	weighted count
<code>full v2</code>	<code>full v1</code> with check migration and the discard rule of Equation (8) on a successor too	check writing	hard evidence
<code>verdict harness</code>	requirements list and judge calls to the acting model, compiled tests, rollback, keep-best, settle	critical path	any red

L.6 Shared-archive and port parameters

Table 32 gives the public starting values of the open problems.

The analytic scorer used on the open problems is the analytic evaluator whose SHA-256 hash is listed in Appendix M.3. The island count is $\max(1, \min(4, \lfloor \text{agents}/2 \rfloor))$, which is one for three agents. Each open-problem arm used a 96-vCPU AMD EPYC Milan machine, with one systemd unit per problem and equal CPU weight; under saturation CPU follows the weights, so each of the four problems received about a quarter of the 96 cores. The isolated arm was followed to 24 working hours by design, the other arms to 39–45 hours, and the society to 41.7 hours. CORAL-faithful used upstream commit `cce3101`; outside the take-home, its configuration contains the problem statement and factual tips only.

Table 32: **Public starting values** at launch (October 7, 2026), rechecked against the public listings. The Einstein Arena’s C_1 record changed to 1.502743486709583 on October 13, 2026.

Problem	Goal	Venue threshold	Public reference at launch
C_1 (first autocorrelation)	min	10^{-8}	1.5027435319539597 (Einstein Arena, October 3, 2026); MILO reported 1.5027435984, above the record
Erdős minimum overlap	min	10^{-7}	0.3808585748578584 (Einstein Arena, August 15, 2026); MILO reported 0.3808567744
AHC026	max	–	official cases only (human winner 1,423,183); development scores reported here

M Records, Certificates, and Provenance

M.1 Record chains for M and C_1

Table 33 lists the published upper bounds for M and C_1 in order; ours are the lowest in each chain.

Table 33: **Published upper bounds for M and C_1 ; ours are the lowest in each chain.** Values as printed by their sources; ‘Arena #’ identifies the Einstein Arena entry. Haugland’s paper prints 0.380926 . . . , which AlphaEvolve and TTT-Discover quote as 0.380927. Exact rational certificates for Hyra’s Erdős entry and its C_1 entry of June 23, 2026, were given by Russell (2026a,b).

Value	Holder	Date
<i>Erdős minimum overlap M</i>		
0.380926 . . .	Haugland (Haugland, 2016)	2016
0.380924	AlphaEvolve (Novikov et al., 2025)	2025
0.380913	LoongFlow (Wan et al., 2025)	2025-12
0.380876	TTT-Discover (Yuksekgonul et al., 2026)	2026-01
0.380874	Aster (Bicker, 2026)	2026-02
0.380871	Together AI (Together AI, 2026) (Arena #9: 0.3808703105862199)	2026-03-19
0.380868	SimpleTES (Ye et al., 2026)	2026-04
0.3808590568145606	lnzwz_AI4M_Agent (Arena #2407)	2026-07-12
0.38085867721583955	Hyra (Arena #2440)	2026-08-03
0.3808585748578584	CodexProLong (Arena #2507)	2026-08-15
0.3808567744	MILO (Jana et al., 2026)	2026-09-29
0.380856738883317420 . . .	this paper, shared archive	2026-10-08
0.380856547875580499 . . .	this paper, single agent	2026-10
<i>First autocorrelation C_1</i>		
1.5098	Matolcsi–Vinuesa (Matolcsi & Vinuesa, 2010)	2010
1.5053	AlphaEvolve (Novikov et al., 2025)	2025
1.5032	AlphaEvolve (Georgiev et al., 2025)	2025-11
1.503133	ThetaEvolve (Wang et al., 2025)	2025-11
1.50286	TTT-Discover (Yuksekgonul et al., 2026)	2026-01
1.50286286	Together AI (Together AI, 2026) (Arena: 1.5028628587053106)	2026-03
1.5027452918597408	Hyra (Arena)	2026-08-11
1.5027437719761116	ExoMind-TTS (Arena #2494)	2026-08-12
1.5027436492	CodexProLong (Arena #2504)	2026-08-14
1.5027435984	MILO (Jana et al., 2026)	2026-09-29
1.5027435319539597	AgentDiscover (Arena #2603)	2026-10-03
1.502740435922766976 . . .	this paper	2026-10-08
1.502743486709583	Atris (Arena #2615)	2026-10-13

M.2 The Erdős lineage, step by step

Table 34: **Nineteen steps from the public Einstein Arena entry to the certified Erdős construction.** The declared lineage in the archive, with the working hour at which each attempt entered, the agent that posted it, its resolution, and its distance below Arena entry #2507. Archive scores are the protected evaluator’s conservative floating-point values; the last is exact.

Step	Hours	Agent	Cells	Below #2507	Move
1	0.32	2	7,168	1.2×10^{-8}	public entry repeated to 7,168 cells; log-sum-exp continuation, β from 10^7 to 3×10^9
2	0.49	2	3,584	2.0×10^{-7}	continuation from the soft $\beta = 10^5$, doubled per stage to 6.5×10^9
3	0.84	2	3,584	3.4×10^{-7}	coarse-to-fine: a population of anneals at 1,792 cells, upsampled
4	1.11	2	3,584	3.7×10^{-7}	another member of that population, upsampled and re-annealed
5	1.63	1	7,168	4.3×10^{-7}	agent 3’s recipe (two-fold refinement, soft re-melt) applied to the best attempt so far
6	3.89	3	1,792	6.4×10^{-7}	symmetrize, then break: antisymmetric noise at the center of the symmetric optimum
7	4.59	3	3,584	7.5×10^{-7}	a second symmetrize-then-break lineage with focused noise settings
8	5.09	3	7,168	7.7×10^{-7}	two-fold refinement, continuation from $\beta = 10^5$
9	5.78	3	3,584	1.04×10^{-6}	symmetrize-then-break from a new symmetric parent
10	6.09	2	1,792	1.25×10^{-6}	eigen-direction symmetry breaking at the symmetric parent of step 9
11	6.43	2	3,584	1.43×10^{-6}	an eigen-mode draw refined two-fold and re-annealed
12	7.09	2	3,584	1.55×10^{-6}	second round of eigen-direction breaking
13	7.42	2	1,792	1.56×10^{-6}	third round
14	8.71	3	3,584	1.70×10^{-6}	memetic genetic algorithm at 1,792 cells with partial-symmetrization restarts; two-fold refinement
15	8.87	3	7,168	1.72×10^{-6}	the same child refined four-fold with slow cooling ($\beta \times \sqrt{2}$ per stage)
16	9.31	3	3,584	1.79×10^{-6}	a blend child of the genetic algorithm
17	9.45	3	7,168	1.80×10^{-6}	that child refined four-fold; first value below MILO’s
18	9.77	3	3,584	1.82×10^{-6}	next generation
19	10.82	3	3,584	1.836×10^{-6}	genetic algorithm at 3,584 cells: uniform-block crossover of a refined best and the mirrored step-16 function

M.3 Provenance and verification scope

Table 35 lists the verification each claimed value passed.

The Erdős construction descends from the run’s first attempt, an upsampling of CodexProLong’s public #2507, through the declared chain of Table 34. Its final step is a uniform-block crossover between one agent’s unsubmitted two-fold refinement of its genetic algorithm’s best and a mirrored earlier attempt, re-melted inside a memetic genetic algorithm that another agent wrote and that ran unmodified. The C_1 construction descends from the run’s second attempt, which started from AgentDiscover’s public #2603. Both analytic constructions build on public basins. For C_3^+ , MILO’s start equals the 8-cell means of BasinHopper’s Arena entry #2492, and credit for the basin belongs to MILO and BasinHopper. Packing searches began from the best public packing at the same size or from a neighboring size through the $N \pm 1$ cascade. Matched to Liang’s packing under the best of the eight square symmetries and the best assignment of centers, the $N = 137$ packing moves its centers by 0.013 RMS, and even the smallest first-run gain, 3.1×10^{-7} at $N = 455$, comes with an RMS displacement of 0.0018. In the shared-archive arm, a scan of the 7,110 tool calls made before the certified Erdős construction found no external URL, network call, or access to another arm.

Table 35: **Each claimed value passed verification.** Scope and outcome of the checks for each claim.

Claim	Checks	Outcome
Erdős and C_1 bounds	second exact implementation; independent exact checker written from the definitions; Einstein Arena verifier	value confirmed
csqv first run (193)	independent checks: feasibility, previous bests, 719 identity comparisons; full-history check: 766 commits, 3,841 coordinate files, 339,696 values	all pass
csqv second run (113 new, 108 improved)	independent checks: all 414 claim files, Liang’s full history rebuilt from a verified mirror, 3,731 identity comparisons	all pass
C_3^+	independent exact checker, each value on the path	value confirmed
Erdős, single agent	independent exact checker	value confirmed
14 points	independent exact checker	value confirmed

The uncompressed construction files and the evaluators they were checked with have the following SHA-256 hashes.

Erdős construction (shared archive)	4183eed832ff8e656f7db5d0457568f9b39f5c52ed9495b8172021ec92a7618f
C_1 construction	88401bde2a0f165832b12ac3be593633e2fbb7221c0f9d516170a0b701118a8c
C_3^+ ($n = 102,400$)	129b14c8dea20beb68b62249e799ec8bdc702633b348be16b226838882b91f34
14 points	c650b95499123f5b801a3a9da50260c3b89091cd8b3e98e4143c47b791bcfbcb6
analytic evaluator	61fe9cdb31be8fb2970f690f8c815638a69201a20ef6de847428d2cbc587dfc8
geometry evaluator	ba4d420a1508c265e268cc1eb417c8e1d046f6e6c04b0d04103fcc4dff5d6c3
fast packing checker ($N > 2500$)	dfdee7664c5a6af86b572f0f5b3696e609f4938e70bf5c77ba7fa73386b0242d

M.4 All 306 packings

Table 36 selects new packings from the smallest margin to the largest, and Table 37 lists all 306.

Table 36: **Selected new packings, from the smallest margin to the largest.** Liang held the previous best at each size. $N = 185$ has the smallest margin of the 306 and $N = 9801$ the largest; $N = 9216$ has the smallest margin among the large sizes.

N	Ours: $\sum r_i$ (exact)	Previous best (Liang)	Margin	Over Packomania
137	6.180252863393018908	6.1800746296240865	1.78×10^{-4}	9.81×10^{-4}
185	7.198651847610267365	7.198651480801373	3.67×10^{-7}	1.03×10^{-3}
455	11.352665116737861479	11.352056015179	6.09×10^{-4}	1.76×10^{-3}
529	12.246054012215808674	12.235811635027076	1.02×10^{-2}	1.67×10^{-2}
1521	20.8261647376347404726	20.823038385363	3.13×10^{-3}	1.77×10^{-2}
1681	21.9127643931576471207	21.908331856608118	4.43×10^{-3}	3.19×10^{-2}
2500	26.7487342580083326146	26.744636270403	4.10×10^{-3}	4.17×10^{-2}
4096	34.2699541749965092688	34.256584228498	1.34×10^{-2}	5.56×10^{-2}
5041	38.0254261999067748255	37.975219102001	5.02×10^{-2}	5.75×10^{-2}
7396	46.0815945983454372089	46.03596799696702	4.56×10^{-2}	6.99×10^{-2}
9216	51.4731348926085804798	51.47297148204148	1.63×10^{-4}	9.90×10^{-2}
9801	53.0664060006081925272	53.0156248155581578993	5.08×10^{-2}	8.36×10^{-2}
10000	53.6095660065611926022	53.60334309311	6.22×10^{-3}	9.04×10^{-2}

Table 37: **Exact sums of radii for all 306 packings.** Unequal circles in the unit square. Previous best is the largest published value as of October 14, 2026; Liang held it at every size. Sources and history coverage are described in Section 4.7. Run 1 is the first run; 2 is the second; 2* replaces a first-run value that also beat the previous best.

N	Ours (exact)	Previous best	Run	Margin
137	6.180252863393018908	6.1800746296240865	1	1.78×10^{-4}
142	6.295677951246816425	6.29544590229006	2	2.32×10^{-4}

N	Ours (exact)	Previous best	Run	Margin
146	6.387824814777793825	6.387741228719	1	8.36×10^{-5}
148	6.432813837725447782	6.432731989105226	2	8.18×10^{-5}
156	6.603639017462671874	6.603620180147317003	1	1.88×10^{-5}
157	6.624666534036148959	6.624047849350299358	1	6.19×10^{-4}
158	6.644814970138116362	6.644712963825	1	1.02×10^{-4}
182	7.139690800460969286	7.139578643773063	1	1.12×10^{-4}
185	7.198651847610267365	7.198651480801373	2	3.67×10^{-7}
188	7.258048914776490874	7.257784502262	2	2.64×10^{-4}
191	7.318151495335000215	7.317941398677	1	2.10×10^{-4}
197	7.437019447033417126	7.436991067624	1	2.84×10^{-5}
205	7.586370214315707481	7.58593152635	1	4.39×10^{-4}
206	7.604029579995051711	7.603903612129	1	1.26×10^{-4}
207	7.622256298439724534	7.622201198658	1	5.51×10^{-5}
208	7.640351003724895499	7.639954768449	2*	3.96×10^{-4}
209	7.658277084748046658	7.657963657677	1	3.13×10^{-4}
210	7.675905467605473087	7.675809055326	1	9.64×10^{-5}
211	7.694537274684769008	7.694453725207	1	8.35×10^{-5}
212	7.712608628407755615	7.712565088592	2	4.35×10^{-5}
217	7.80679666119002324	7.806394277851230227	1	4.02×10^{-4}
219	7.844778866628022924	7.844486169173	2	2.93×10^{-4}
230	8.043720551997481443	8.043355500284	1	3.65×10^{-4}
231	8.060446369119714437	8.060436244514	1	1.01×10^{-5}
235	8.127288800049589413	8.127280870897	1	7.93×10^{-6}
236	8.143769883851493495	8.143218415657	2*	5.51×10^{-4}
237	8.161337722639579324	8.160815684546	2*	5.22×10^{-4}
238	8.178481169206609564	8.177732544592	2*	7.49×10^{-4}
239	8.19552185189848535	8.195152547633	2	3.69×10^{-4}
247	8.337695735934358299	8.337657453498	2	3.83×10^{-5}
248	8.356051847278934624	8.356030683330472	1	2.12×10^{-5}
249	8.373358570759210365	8.373264801181	2	9.38×10^{-5}
251	8.408141749169110775	8.407669216587	2	4.73×10^{-4}
252	8.425246854790941687	8.425177921053259047	1	6.89×10^{-5}
253	8.442989172878737881	8.442907861345068	1	8.13×10^{-5}
265	8.638010415812943158	8.638002402392415	1	8.01×10^{-6}
270	8.718536927553344382	8.718530269452	1	6.66×10^{-6}
275	8.800682728448504174	8.800226795221416	2	4.56×10^{-4}
276	8.817957668499285686	8.81703289155219142	2*	9.25×10^{-4}
277	8.834999349367095091	8.833012846276	1	1.99×10^{-3}
278	8.852205791172315364	8.849078849651	1	3.13×10^{-3}
279	8.868518437897543248	8.868057555173	2*	4.61×10^{-4}
280	8.885030250053348558	8.884272111120044598	1	7.58×10^{-4}
282	8.918493325379532771	8.917797248353	2*	6.96×10^{-4}
285	8.96791620811850299	8.967891423896	1	2.48×10^{-5}
290	9.043321702076018638	9.04325923378808564	2	6.25×10^{-5}
293	9.089325767255382313	9.088930498053	2	3.95×10^{-4}
294	9.104272968673472539	9.104113342019	1	1.60×10^{-4}
295	9.119665026277559479	9.119378513625	1	2.87×10^{-4}
297	9.149914661992509187	9.149420862049253	2*	4.94×10^{-4}
298	9.165542978165433146	9.164640675262	1	9.02×10^{-4}
299	9.181211255170868153	9.179904120375110247	1	1.31×10^{-3}
301	9.210605939119740701	9.209844096512	1	7.62×10^{-4}
302	9.22700188390250834	9.22573648925747	1	1.27×10^{-3}
303	9.242592646262930874	9.242272991889	1	3.20×10^{-4}
304	9.258437915144478758	9.25726107191	2*	1.18×10^{-3}
305	9.274141800935633951	9.273668547572258825	1	4.73×10^{-4}
306	9.289406225596147138	9.288950901832218	2*	4.55×10^{-4}
307	9.305966463450673529	9.305804368812488	2	1.62×10^{-4}
308	9.321712880738834391	9.321332860923203474	1	3.80×10^{-4}
309	9.337522992279633285	9.336984289826	1	5.39×10^{-4}
310	9.353544233519262189	9.35347181516	2	7.24×10^{-5}
312	9.385339629618463925	9.385012613536	2	3.27×10^{-4}
313	9.400829588631472821	9.400819251021	2	1.03×10^{-5}
314	9.415755705722215604	9.415455693751	1	3.00×10^{-4}
317	9.463025175677656707	9.462942784057088	2	8.24×10^{-5}
319	9.492589332754020739	9.49257936595	2	9.97×10^{-6}
322	9.536854656248981044	9.536742445151	1	1.12×10^{-4}

<i>N</i>	Ours (exact)	Previous best	Run	Margin
323	9.55007468961524108	9.550057569242	2	1.71×10^{-5}
324	9.564320590060022936	9.564266944475238	2	5.36×10^{-5}
326	9.592508448082535822	9.592121131691	1	3.87×10^{-4}
327	9.607065015508765455	9.606720244607	2*	3.45×10^{-4}
328	9.621690585460442514	9.620923544343102	2*	7.67×10^{-4}
330	9.650833311758388682	9.65044098968523	2*	3.92×10^{-4}
331	9.665811403516624348	9.665118699705321	2*	6.93×10^{-4}
332	9.680828738536478292	9.680090467272466	2*	7.38×10^{-4}
333	9.695701083590445449	9.694588243127	1	1.11×10^{-3}
334	9.710331621276520065	9.709972693508	1	3.59×10^{-4}
335	9.725018848379329933	9.721816063906	2*	3.20×10^{-3}
336	9.738739854456882468	9.73653445863842	1	2.21×10^{-3}
337	9.753233327246692098	9.75210127893	1	1.13×10^{-3}
338	9.768180431741219585	9.767449549800071	1	7.31×10^{-4}
339	9.783441153023109691	9.782725516609080259	1	7.16×10^{-4}
340	9.799084684767045642	9.798959804691	2	1.25×10^{-4}
341	9.813888661890454879	9.812931944945	1	9.57×10^{-4}
342	9.829547330330974519	9.8295144572022	1	3.29×10^{-5}
343	9.844099982131961271	9.843018954046475	1	1.08×10^{-3}
344	9.859515561168946915	9.859295071497252457	1	2.20×10^{-4}
345	9.87358552086328381	9.871546836350314299	1	2.04×10^{-3}
346	9.888383325104795847	9.887974822624	2*	4.09×10^{-4}
347	9.902696554149374503	9.902123538691	1	5.73×10^{-4}
348	9.918143382694258488	9.91796879350822	1	1.75×10^{-4}
349	9.932279281546956182	9.932003707993196804	2*	2.76×10^{-4}
350	9.947361055015214958	9.946196231108	1	1.16×10^{-3}
351	9.961904602931062361	9.961599925030004095	1	3.05×10^{-4}
352	9.975865175992803554	9.97581660387	2	4.86×10^{-5}
353	9.990153986516426724	9.990123404486	2	3.06×10^{-5}
355	10.018386622881226899	10.017348811819142	2*	1.04×10^{-3}
356	10.031525871972499036	10.028837895494	1	2.69×10^{-3}
357	10.043155996959930828	10.042391218778	2*	7.65×10^{-4}
358	10.057435413129955764	10.056424852513222791	2*	1.01×10^{-3}
359	10.071582829648081267	10.069701157534	2*	1.88×10^{-3}
360	10.085823006988295958	10.084055771106213388	2*	1.77×10^{-3}
361	10.10000208397209231	10.099355683993142262	2*	6.46×10^{-4}
362	10.114018040278457605	10.113675817734	2	3.42×10^{-4}
363	10.12750840269437469	10.12629987944	1	1.21×10^{-3}
365	10.155217304591001733	10.154922603302	2	2.95×10^{-4}
366	10.168397181761067536	10.16811450529	1	2.83×10^{-4}
367	10.182648829166982135	10.182614617914	2	3.42×10^{-5}
368	10.196857425007338991	10.1963592751	2*	4.98×10^{-4}
369	10.210713274007651438	10.210656019959	2	5.73×10^{-5}
370	10.224720905620632762	10.224249820483148	2	4.71×10^{-4}
371	10.239004135209290486	10.238525853697	2	4.78×10^{-4}
372	10.253076166349626014	10.252311504903277	2*	7.65×10^{-4}
373	10.267091557747521027	10.266214821013454	2*	8.77×10^{-4}
374	10.281299169870545924	10.28066989177917	1	6.29×10^{-4}
375	10.295080178778702707	10.295012338309	2	6.78×10^{-5}
377	10.323772258880832415	10.323534933290476	2	2.37×10^{-4}
378	10.338898189785464554	10.337798454625	2*	1.10×10^{-3}
379	10.353188704949128584	10.352780705647	1	4.08×10^{-4}
380	10.36725773848013405	10.365846051273	2*	1.41×10^{-3}
382	10.39612612311933295	10.395847164838	2*	2.79×10^{-4}
385	10.437860471197656089	10.437625654595	1	2.35×10^{-4}
386	10.452039102134645595	10.451308287138255456	2*	7.31×10^{-4}
387	10.465876421007100064	10.463628278476463	1	2.25×10^{-3}
388	10.479132042252730757	10.476431485678	2*	2.70×10^{-3}
389	10.491437817694915549	10.488500253096	2*	2.94×10^{-3}
390	10.50300789636752495	10.502128814062322122	2*	8.79×10^{-4}
391	10.51642997300092863	10.515311384285056	2*	1.12×10^{-3}
392	10.529610580101346057	10.528692881753	2*	9.18×10^{-4}
393	10.542195825826727665	10.540910774497	1	1.29×10^{-3}
394	10.555538160481650851	10.554218362585011	1	1.32×10^{-3}
395	10.569110554063826485	10.567762415409	2*	1.35×10^{-3}
396	10.582374620028917666	10.58099542035442	2*	1.38×10^{-3}

N	Ours (exact)	Previous best	Run	Margin
397	10.595687917110271892	10.595108503166227	2	5.79×10^{-4}
398	10.608312418820262668	10.606684686129	2*	1.63×10^{-3}
399	10.621626878821647983	10.620528360865329324	2*	1.10×10^{-3}
400	10.634726119716070452	10.631978493560295	1	2.75×10^{-3}
401	10.647716037964392313	10.646889813174	1	8.26×10^{-4}
402	10.661209221981892196	10.659068621689	2*	2.14×10^{-3}
403	10.674724153898649818	10.674019157454056839	2	7.05×10^{-4}
404	10.688375617963671047	10.687354545195152	2*	1.02×10^{-3}
405	10.701715493219220473	10.700808830268	2*	9.07×10^{-4}
406	10.715267351039029484	10.714986963599	1	2.80×10^{-4}
407	10.729245680329723122	10.728493628098237	1	7.52×10^{-4}
408	10.743122836965742822	10.742051825622447	1	1.07×10^{-3}
410	10.769978677997050915	10.768985209394291	2*	9.93×10^{-4}
411	10.783634149286997851	10.78257853628	2*	1.06×10^{-3}
414	10.824247248009148147	10.824057489898	1	1.90×10^{-4}
415	10.838054903166930373	10.83767261400815267	2	3.82×10^{-4}
416	10.851392838276020644	10.85092468347	2*	4.68×10^{-4}
417	10.865028494085665832	10.864815186674136	2*	2.13×10^{-4}
418	10.879200713107233879	10.878523629038142328	2*	6.77×10^{-4}
419	10.892923217745895404	10.892665080332	2	2.58×10^{-4}
420	10.905746469756890149	10.905704806942	2	4.17×10^{-5}
421	10.919271671117306093	10.919253797972	2	1.79×10^{-5}
422	10.933153784290083566	10.933019712196083	2	1.34×10^{-4}
429	11.023620199547893579	11.0235188005	2	1.01×10^{-4}
431	11.047869192158593843	11.047800143152	2	6.90×10^{-5}
432	11.059686509241607732	11.05950471636	2	1.82×10^{-4}
433	11.07180813453815854	11.071788062005	2	2.01×10^{-5}
434	11.08469235627254661	11.084566793068	2	1.26×10^{-4}
435	11.096980672263004299	11.09674075478	2*	2.40×10^{-4}
436	11.110025353489768893	11.109013784905	2*	1.01×10^{-3}
437	11.122695806372661652	11.122021758613112	2*	6.74×10^{-4}
438	11.134839101675239046	11.13457027483	1	2.69×10^{-4}
439	11.147629354619641796	11.147344979031	2	2.84×10^{-4}
440	11.159615153423291399	11.159484571692378	2*	1.31×10^{-4}
441	11.171982651663833435	11.171344962483127014	1	6.38×10^{-4}
442	11.184490987463337887	11.183744585430095	1	7.46×10^{-4}
443	11.196513494210879254	11.19641700082	2*	9.65×10^{-5}
446	11.234974044276115496	11.234571141333	1	4.03×10^{-4}
447	11.24830239337127629	11.247594138506289377	2*	7.08×10^{-4}
448	11.261427073638572046	11.260595834042038	1	8.31×10^{-4}
449	11.274439247889263104	11.273153984076	2*	1.29×10^{-3}
450	11.288382110876846532	11.286856223680338153	2*	1.53×10^{-3}
451	11.301645614570400843	11.299866162827167	2*	1.78×10^{-3}
452	11.314438958825364132	11.313692402308046	2*	7.47×10^{-4}
453	11.326936777094785668	11.326450332304	2*	4.86×10^{-4}
454	11.340365936017801164	11.339991901574384328	2	3.74×10^{-4}
455	11.352665116737861479	11.352056015179	2*	6.09×10^{-4}
456	11.365532134589393053	11.36498937378	2*	5.43×10^{-4}
457	11.378521899153775402	11.378259041302492	2*	2.63×10^{-4}
458	11.391405692236707436	11.39067932275504	2	7.26×10^{-4}
459	11.405016199113972102	11.404111026276	2*	9.05×10^{-4}
460	11.418032166461032314	11.416851702401	2*	1.18×10^{-3}
461	11.43047172379733585	11.429618959065	2*	8.53×10^{-4}
462	11.443482108404651252	11.440362320075362442	2*	3.12×10^{-3}
463	11.456320226995420563	11.453418970267057036	2*	2.90×10^{-3}
464	11.469002151498845839	11.46425431102548	2*	4.75×10^{-3}
465	11.481048139187243047	11.47665682458905	2*	4.39×10^{-3}
466	11.493805662681604063	11.489213060903426	2*	4.59×10^{-3}
467	11.506304631730447725	11.501332372758387	2*	4.97×10^{-3}
468	11.517371765714565939	11.514331674698486	2	3.04×10^{-3}
469	11.52785977071742016	11.526435282894049185	2*	1.42×10^{-3}
470	11.53928527759627643	11.538072626079	2*	1.21×10^{-3}
471	11.551889012026367603	11.549771140737	2*	2.12×10^{-3}
472	11.563424342061380308	11.562200541317	2	1.22×10^{-3}
473	11.575146830922995816	11.57392384541202	2*	1.22×10^{-3}
474	11.587061088836031854	11.585691131256	2*	1.37×10^{-3}

N	Ours (exact)	Previous best	Run	Margin
475	11.599219194526562834	11.598296060694024	2	9.23×10^{-4}
476	11.61155852453233632	11.610694328765323	2*	8.64×10^{-4}
477	11.624452310597051141	11.623024467761	2*	1.43×10^{-3}
478	11.636914830544697899	11.635281049091	2	1.63×10^{-3}
479	11.649389015997359249	11.64769164874	2*	1.70×10^{-3}
480	11.661661158415590252	11.65953501583	2*	2.13×10^{-3}
481	11.673763300518623944	11.671384096329	2*	2.38×10^{-3}
482	11.68598075827936964	11.685360636960487	2*	6.20×10^{-4}
483	11.698136683837465316	11.695599609112100993	2*	2.54×10^{-3}
484	11.710268820056188339	11.708318294287	1	1.95×10^{-3}
485	11.722203751520105601	11.721377250648	2*	8.27×10^{-4}
486	11.734219479816948341	11.733992103902	1	2.27×10^{-4}
487	11.746160535468672659	11.743513784756045	1	2.65×10^{-3}
488	11.758033928429962301	11.75773729545031	2*	2.97×10^{-4}
489	11.770418177018486664	11.769610100652270179	1	8.08×10^{-4}
491	11.795357516976009804	11.794376896928	1	9.81×10^{-4}
492	11.807911061361513592	11.807466669778	2*	4.44×10^{-4}
494	11.833126626504711719	11.832363607649	1	7.63×10^{-4}
495	11.84567636635822177	11.843409517905	1	2.27×10^{-3}
496	11.857947090500060434	11.854375105080131155	1	3.57×10^{-3}
497	11.869935767222369992	11.866743178315495	2*	3.19×10^{-3}
498	11.881828015949956634	11.878206095286	2*	3.62×10^{-3}
499	11.893702525652968591	11.890330089707353182	2*	3.37×10^{-3}
500	11.904692790839477703	11.903170336229	2	1.52×10^{-3}
529	12.246054012215808674	12.235811635027076	2*	1.02×10^{-2}
545	12.438370773369283606	12.436533265502467	2*	1.84×10^{-3}
576	12.783287280809794978	12.775152325972053	2*	8.13×10^{-3}
613	13.189103030713546166	13.185931386373403	2*	3.17×10^{-3}
625	13.319991865908629933	13.31520620527706	2*	4.79×10^{-3}
676	13.856049222648425738	13.853532292832384	2*	2.52×10^{-3}
685	13.955172747399860736	13.947043137408414568	2*	8.13×10^{-3}
729	14.400812568074163977	14.393408217492	2*	7.40×10^{-3}
761	14.707492637208948155	14.70204839839	2*	5.44×10^{-3}
784	14.9373697687773226	14.933651793985	1	3.72×10^{-3}
841	15.473997494816899313	15.471199967493	2*	2.80×10^{-3}
900	16.013142231633788875	16.011101957183	2*	2.04×10^{-3}
925	16.227491472071067454	16.220391311081	2*	7.10×10^{-3}
961	16.547694231105267125	16.546730901966101565	2*	9.63×10^{-4}
1013	16.983517564206329008	16.981739092791	2*	1.78×10^{-3}
1024	17.06085052872915334	17.05935881728	2*	1.49×10^{-3}
1089	17.621467119689173457	17.621287603134	1	1.80×10^{-4}
1156	18.1547164235663760493	18.154691434658	2	2.50×10^{-5}
1225	18.692401713825186423	18.68534103330226597	2*	7.06×10^{-3}
1296	19.2118093599524339584	19.210318925583	2	1.49×10^{-3}
1369	19.7457995076503394942	19.743771260852284001	2*	2.03×10^{-3}
1444	20.3075855902416606423	20.3034779015522608607	2*	4.11×10^{-3}
1521	20.8261647376347404726	20.823038385363	2*	3.13×10^{-3}
1600	21.382176450346681265	21.379516344136200715	2	2.66×10^{-3}
1681	21.9127643931576471207	21.908331856608118	2	4.43×10^{-3}
1764	22.4578968732403947246	22.4520051255372620683	2	5.89×10^{-3}
1849	22.9898051034169263807	22.985298580705	2	4.51×10^{-3}
1936	23.5300922258535365191	23.5267489263813161838	2	3.34×10^{-3}
2025	24.0676680013099326851	24.063783503649375	2	3.88×10^{-3}
2116	24.5942010474018950755	24.5897477732620810592	2	4.45×10^{-3}
2209	25.1413895101643834221	25.140593125125	2	7.96×10^{-4}
2304	25.6775861979922932624	25.67601721119147	2	1.57×10^{-3}
2401	26.2121996227255827636	26.204628504855	2	7.57×10^{-3}
2500	26.7487342580083326146	26.744636270403	2	4.10×10^{-3}
2601	27.2877616710413712979	27.277167007842422	2	1.06×10^{-2}
2704	27.8273469755102592661	27.823137574183	2	4.21×10^{-3}
2809	28.3670156025491741709	28.366020412230327	2	9.95×10^{-4}
2916	28.902573124336435516	28.900709505172	2	1.86×10^{-3}
3025	29.4378743580168108693	29.434933513113315	2	2.94×10^{-3}
3136	29.9655094127494333891	29.960279941385227	2	5.23×10^{-3}
3249	30.5126109390480752217	30.511081629717214	2	1.53×10^{-3}
3364	31.0520113445074166928	31.043887002321	2	8.12×10^{-3}

N	Ours (exact)	Previous best	Run	Margin
3481	31.587913644699342109	31.5855752548044300599	2	2.34×10^{-3}
3600	32.126934220549428686	32.12538745069016	2	1.55×10^{-3}
3721	32.663267852677569786	32.66089172144	2	2.38×10^{-3}
3844	33.193823210866161958	33.180384313552	2	1.34×10^{-2}
3969	33.7366223131194701731	33.724449969923285	2	1.22×10^{-2}
4096	34.2699541749965092688	34.256584228498	2	1.34×10^{-2}
4225	34.8104993058235709634	34.7987129787464730395	2	1.18×10^{-2}
4356	35.352981803283012785	35.344143157206375	2	8.84×10^{-3}
4489	35.8881843850942609884	35.88786423495221127	2	3.20×10^{-4}
4624	36.4273201321753786192	36.4262906927244911081	2	1.03×10^{-3}
4761	36.9621950207806265272	36.9524893011524330359	2	9.71×10^{-3}
4900	37.4924318418774285388	37.4737223188770673348	2	1.87×10^{-2}
5041	38.0254261999067748255	37.975219102001	2	5.02×10^{-2}
5184	38.5659582142139985128	38.5606233090760161769	2	5.33×10^{-3}
5329	39.1100156109129384405	39.10078913399521	2	9.23×10^{-3}
5476	39.6492888005970851704	39.6484565852801326352	2	8.32×10^{-4}
5625	40.1734262130076659303	40.155096145848184	2	1.83×10^{-2}
5776	40.7107950380194638342	40.6992702134202371675	2	1.15×10^{-2}
5929	41.252631687843275651	41.243099663071	2	9.53×10^{-3}
6084	41.7892393930579420645	41.78262494966233	2	6.61×10^{-3}
6241	42.334382883547468397	42.317650758177	2	1.67×10^{-2}
6400	42.8706472713367775853	42.8624330274154088561	2	8.21×10^{-3}
6561	43.4124697449814533816	43.4077685338294833929	2	4.70×10^{-3}
6889	44.4850008242160214055	44.4792579069230272248	2	5.74×10^{-3}
7056	45.0153156776231525877	45.003245497698	2	1.21×10^{-2}
7225	45.5507848462751684862	45.544167858726	2	6.62×10^{-3}
7396	46.0815945983454372089	46.03596799696702	2	4.56×10^{-2}
7569	46.6226542867372155364	46.60251442977	2	2.01×10^{-2}
7744	47.1539558491425514707	47.130009797431	2	2.39×10^{-2}
8100	48.2363866771907498001	48.212395982255	2	2.40×10^{-2}
8281	48.7760475329245679329	48.749808455343	2	2.62×10^{-2}
8464	49.316789802446577971	49.312183797404	2	4.61×10^{-3}
8649	49.852961777958462785	49.837391057263375	2	1.56×10^{-2}
8836	50.392955349089191156	50.39153822784036	2	1.42×10^{-3}
9025	50.9340588223221777488	50.928027389488	2	6.03×10^{-3}
9216	51.4731348926085804798	51.47297148204148	2	1.63×10^{-4}
9409	52.0089673982626189371	52.0062291183621934845	2	2.74×10^{-3}
9604	52.5407108431373618664	52.530893570438	2	9.82×10^{-3}
9801	53.0664060006081925272	53.0156248155581578993	2	5.08×10^{-2}
10000	53.6095660065611926022	53.60334309311	2	6.22×10^{-3}

N Additional Results

N.1 Summary of headline results and their verification

P: preregistered; R: recomputed from the raw artifacts with independent analysis code; E: exact arithmetic.

Table 38: **Summary of headline results** (Claude Opus 5.5 unless noted). Intervals are 95% unless marked.

Result	Instrument and contrast	n	Estimate [CI]	Status
Erdős bound, shared archive	Exact value vs. Einstein Arena record / MILO / HarnessTTT best	—	1.84×10^{-6} / 3.55×10^{-8} / 1.13×10^{-5} below	E, R
Erdős bound, single agent	Exact value vs. Einstein Arena record / MILO	—	2.03×10^{-6} / 2.27×10^{-7} below	E, R
C_1 bound	Exact value vs. Einstein Arena record (October 13) / MILO / HarnessTTT best	—	3.05×10^{-6} / 3.16×10^{-6} / 1.04×10^{-3} below	E, R
C_3^+ bound	Exact value vs. MILO / Arena leading entry	—	1.90×10^{-5} / 2.43×10^{-4} below	E, R
Circle packings	Sizes above every published value	306	margins 3.7×10^{-7} to 5.1×10^{-2}	E, R
Compute-only control	Agents’ descent vs. preregistered classical control from the same entries, M / C_1	—	$3,187 \times / 2,938 \times$ as far	P
Take-home	Shared archive vs. isolated agents / CORAL port, cycles at 8 h	3 seeds per arm	914 vs. 946 / 948 (medians)	P
Open problems	Society ahead of CORAL-faithful and isolated agents on all three problems	crossover	by 6.5 working hours	P
Verified technique	KernelBench, library — ACE playbook, mean capped speedup	26 problems $\times$ 3	+1.71 [1.19, 2.25] (90%)	P, R
Holdout only	Same contrast, 16 holdout problems	16 problems $\times$ 3	+1.68 [1.10, 2.31] (90%)	P, R
Library to ACE	ACE given the library — ACE playbook	26 problems $\times$ 3	+1.34 [0.77, 1.91] (90%)	P, R
Randomized exposure	Store on/off, $\Delta \log(1 + \text{speedup})$, Claude Opus 5.5/ Claude Opus 4.8	92 / 77 problems	−0.10 [−0.16, −0.05] / +0.26 [0.18, 0.34]	P, R
Store vs. no store	Confirmation design, Claude Opus 5.5/ Claude Opus 4.8	28 problems	−0.18 [−0.33, −0.07] / +0.22 [0.15, 0.29]	P, R
Late delivery	Store after first check — no memory	20 problems	+0.106 [0.008, 0.210]	P
No harm	HORIZON, Claude Opus 5.5 throughout; guard, commit rule, and first supervisor version vs. bare	2,880 harness tickets	0 below bare	P, R
Positive control	Verdict harness rerun unchanged, same streams	579 tickets	360 below bare; −0.344 [−0.448, −0.240]	P, R
Handover	HORIZON, Qwen3.5-27B from ticket 40, evidence-only supervisor — bare, mean S_{prod}	6 streams	+0.476 [+0.149, +0.803]	P
Commit rule	Same, commit after every ticket — bare	6 streams	+0.445 [+0.034, +0.856]	P
Crash stream	AppWorld test-challenge, undocumented change, 52 crashes, harness vs. bare, SGC	139 scenarios	92.1 vs. 79.1	P, R
Recovery stream	AppWorld, Claude Opus 4.8 then Claude Opus 5.5, harness vs. fresh-restart bare, SGC	139 scenarios	86.3 vs. 71.2	P, R
Receipts	AppWorld, crash after every step	39 tasks	38 of 39; 97.4% [86.8, 99.5]	P
Silent change	AppWorld, claim learner — diligence instruction	173 tasks $\times$ 3	+10.8 [6.6, 15.2]	P
Metamorphic instruction	Sealed change classes, metamorphic — diligence instruction	216 tasks $\times$ 3	+6.6 [3.7, 9.9]	R
Agreement check	τ^2 -bench test split, ΔJ vs. same-information judge / zero-shot judge (50 panels)	600 attempts	+16.5 [5.4, 32.1] / +40.0 [26.0, 61.0]	P, R
Check library	AppWorld record, novel-state catch after the first 200 failures	1,036 novel failures	90.1%; +24.0 [16.4, 32.8] over frozen	P, R
Transfer	Checks from Claude Haiku 4.5, Qwen, and GPT-6 Astra failures on Opus-family novel failure states	32 states; 7,185 correct attempts	30 of 32 caught; 0 false fires	R
Anticipation	Anticipate — failure-only library, instance first failures	110 first failures	+55.5 [44.6, 65.6]	P, R
Headroom fit	Comparison record, lift over a no-harness control vs. headroom, k	15 comparisons	0.54 [0.43, 0.68]	—

N.2 Sensitivity of the headroom fit

 Table 39: **Headroom fits after removing groups of comparisons** (Claude Opus 5.5 comparisons against a no-harness control; profile-likelihood 95% intervals with τ re-optimized at each point). The four carried-knowledge rows are L010–L012 and L016; L124 is a subset of L126; L057 is the one mid-task handoff; L099 was not independently recomputed and L166 was analyzed after the fact; L174 mixes Claude Opus 4.8 and Claude Opus 5.5.

Row set	n	k [95% CI]	AIC, constant / headroom
All comparisons	15	0.539 [0.427, 0.675]	96.8 / 77.2
Without the four carried-knowledge rows	11	0.563 [0.443, 0.713]	68.1 / 55.5
Without a row that is a subset of another	14	0.514 [0.400, 0.655]	90.1 / 72.0
Without the one mid-task handoff	14	0.537 [0.422, 0.677]	89.3 / 71.5
Without one row not independently recomputed and one post hoc row	13	0.499 [0.383, 0.643]	82.9 / 65.9
Without the mixed-model AppWorld stream	14	0.568 [0.440, 0.718]	91.9 / 73.3
Without all of the above	6	0.503 [0.338, 0.748]	36.5 / 30.1

N.3 HORIZON: per-stream results and forensics

Table 40 shows that the positive control did not recover, and Table 42 traces the duplication in the verdict harness’s protected tests.

Table 40: **The positive control did not recover.** The verdict harness rerun unchanged with Claude Opus 5.5 on the six fresh streams. From its first ticket below bare to the last analyzed ticket, each ticket stayed below bare. Mean S_{prod} is over tickets 41 to the last analyzed ticket; bare is 1.000 throughout.

	Stream 1	Stream 2	Stream 3	Stream 4	Stream 5	Stream 6	All
First ticket below bare	22	48	35	25	41	54	
Last analyzed ticket	62	89	120	116	101	91	579 reached
Tickets below bare	41	42	86	92	61	38	360
Mean S_{prod} from ticket 41	0.808	0.693	0.539	0.561	0.640	0.696	0.656

Table 41: **How an orphaned deployment breaks production.** Bare Claude Opus 5.5 hands over to Claude Haiku 4.5 at ticket 40 on the six older streams. Each row is the stream’s first production break: Claude Haiku 4.5’s git commands at that ticket, the post-handover service lines they removed, and the effect. Claude Haiku 4.5 made no commit, so each command restored the tree of ticket 39.

Stream	Ticket	S_{prod}	Claude Haiku 4.5’s commands at the ticket	Lines lost	Effect
O1	63	1.000 → 0.937	git stash (three times)	348	withdrawal fee 1.00 instead of 1.50
O2	72	1.000 → 0.028	git stash; git checkout app/service.py app/storage.py	420	older interest rule; 107 untouched accounts wrong
O3	44	1.000 → 0.909	git stash; git checkout; rm tests/test_t039.py ...	134	transfer fee 1.55 instead of 2.06
O4	41	1.000 → 0.000	git checkout app/ledger.py; rm tests/test_t040.py	8	outage: unknown event type ‘fee’
O5	76	1.000 → 0.000	git checkout app/money.py app/store.py; git checkout app/commands.py	735	outage: unknown event type ‘fee’
O6	68	1.000 → 0.000	git stash; git checkout -- .; git clean -fd tests/	498	outage: unsupported state version: 15

Table 42: **Duplication in the verdict harness’s protected tests.** One stream, handover run, tickets before the handover. Distinct test sources grow 2.4-fold from ticket 10 to ticket 30; active protected tests grow 167-fold. At ticket 30, 43,812 of the active tests were re-admitted copies.

After ticket	10	20	25	28	30
Active protected tests	263	1,043	4,587	17,639	43,974
Distinct test sources	196	349	420	448	469
Most copies of one source	18	148	1,024	4,300	10,776

N.4 KernelBench: anchoring in detail

Tables 43 to 45, 47 and 48 and Figures 14 and 15 give the anchoring and technique results in detail: correctness and fast kernels by reader, the L2_074 case, where the loss sits, the successor’s readouts, per-problem effects, the effect of store age, and the technique library’s lift.

Table 43: **The store repairs Claude Opus 4.8’s correctness and cuts Claude Opus 5.5’s fast kernels.** fast_p is the fraction of episodes correct and faster than p times eager, averaged over problems. Differences are store minus no store, paired by problem, with 95% problem-bootstrap intervals. Randomized test: 92 problems for Claude Opus 5.5, 77 for Claude Opus 4.8; confirmation: 28 problems.

Design	Metric	Claude Opus 5.5				Claude Opus 4.8			
		off	on	difference		off	on	difference	
Randomized	fast_0 (correct)	0.996	1.000	+0.004	[+0.000, +0.010]	0.673	0.968	+0.294	[+0.203, +0.387]
Randomized	fast_1	0.855	0.843	−0.012	[−0.051, +0.025]	0.357	0.604	+0.247	[+0.145, +0.346]
Randomized	fast_2	0.377	0.262	−0.115	[−0.179, −0.054]	0.056	0.117	+0.061	[+0.015, +0.113]
Randomized	fast_5	0.141	0.112	−0.030	[−0.061, −0.003]	0.026	0.017	−0.009	[−0.026, +0.000]
Confirmation	fast_1	0.893	0.723	−0.170	[−0.304, −0.045]	0.366	0.598	+0.232	[+0.098, +0.366]
Confirmation	fast_2	0.304	0.188	−0.116	[−0.232, −0.018]	0.009	0.036	+0.027	[+0.000, +0.080]

Table 44: **On L2_074 the store turns the successor into its predecessor.** Randomized test, every episode on the problem. Own-kernel share is the fraction of runtime spent in the submission’s own kernels; the rest is library code. Tokens are output tokens per episode.

Reader, store	Episodes	Design	Capped speedup	Own-kernel share	Tokens
Claude Opus 5.5, off	7	one CUDA kernel for the module	6.52–9.68 (mean 8.86)	0.99–1.00	13,276
Claude Opus 5.5, on	3	cuDNN + Triton epilogue	1.263–1.268	0.04–0.05	2,958
Claude Opus 4.8, off	2	cuDNN + Triton or CUDA epilogue	1.274; compile error	0.04	1,854
Claude Opus 4.8, on	2	cuDNN + Triton epilogue	1.260–1.271	0.05	2,304

Table 45: **The loss sits where the successor’s own kernels are far above the store.** The eight problems with the largest store effect on Claude Opus 5.5 in the randomized test. Means are capped speedup; CUDA counts episodes written in CUDA C++; token counts episodes whose own kernels take under 5% of runtime.

Problem	Family	bare		store		$\Delta \log$	Recipes, kernels
		mean	CUDA	mean	CUDA / token		
L2_074 ConvTranspose3d_LeakyReLU_...	conv_transpose	8.86	7/7	1.27	0/3, 1/3	−1.46	5, 2
L2_008 Conv3d_Divide_Max_...	conv	4.09	5/5	1.06	0/5, 5/5	−0.89	6, 2
L1_085 conv_depthwise_2D_...	conv	3.44	5/5	1.08	0/5, 0/5	−0.75	6, 2
L2_056 Matmul_Sigmoid_Sum	matmul	4.34	0/6	1.88	0/4, 3/4	−0.73	6, 2
L3_048 Mamba2ReturnY	attention	4.07	2/5	1.10	0/5, 2/5	−0.69	0, 0
L2_067 Conv2d_GELU_GlobalAvgPool	conv	2.31	5/5	0.87	3/5, 0/5	−0.67	6, 2
L2_071 Conv2d_Divide_LeakyReLU	conv	2.45	2/2	1.00	2/8, 0/8	−0.57	6, 2
L2_057 Conv2d_ReLU_HardSwish	conv	4.74	2/2	2.25	0/8, 0/8	−0.57	6, 2

Table 46: **Only removing the approach items removes the loss.** Calibration run: Claude Opus 5.5 on 37 problems where its bare kernels are mostly CUDA C++, 317 episodes, up to two per arm and problem. Contrasts are paired by problem over the 17–25 problems each pair shares, with 95% problem-bootstrap intervals (2,000 draws); they are not additive.

Arm	What the reader saw	Contrast	Δ log speedup
show-all	the verbatim store in the opening prompt	– none	–0.272 [–0.466, –0.081]
lessons only	the 14 lessons	– none	+0.001 [–0.066, +0.073]
no approach	lessons and pitfalls, no recipes or kernels	– none	–0.001 [–0.119, +0.115]
show-all	recipes and kernels added to the above	– no approach	–0.244 [–0.450, –0.077]
no numbers	the store with all speedups removed	– show-all	+0.002
reader-relative	the store framed as an older model’s floors, with the reader’s own levels	– show-all	+0.106 [+0.008, +0.211]
frontier-gated	items whose speedup exceeds the reader on their source problem	– none	–0.118 [–0.306, +0.045]
target-gated	the above, also exceeding the reader’s predicted level on the target	– none	–0.087 [–0.312, +0.122]
late	the verbatim store after the first local check	– none	+0.106 [+0.008, +0.210]
late	the same	– show-all	+0.333 [+0.125, +0.519]

Table 47: **Shown the store, Claude Opus 5.5 writes less CUDA and fewer tokens, and submits sooner.** Problem-paired differences. ‘Both’ uses 1,256 Claude Opus 5.5 episodes over 120 problems (578 without, 678 with the store); ‘Randomized’ uses 920 episodes over 92 problems.

Readout	Set	No store	Store	Difference
Correct episodes	Both	576/578	678/678	+0.003
Log speedup given correct	Both			–0.126 [–0.185, –0.075]
Output tokens per episode	Both	9,597	7,142	–2,588 [–3,286, –1,940]
Submit right after first check	Both	76%	86%	+10 pts
Episodes in CUDA C++	Randomized	202/466	60/454	43.4% → 13.2%
Token kernels (< 5% own runtime)	Randomized	92/466	145/454	19.7% → 31.9%
Store coefficient, problem fixed effects	Randomized			–0.106 [–0.168, –0.053]
+ CUDA indicator	Randomized			–0.053 [–0.099, –0.013]
+ CUDA and token-kernel indicators	Randomized			–0.024 [–0.065, +0.015]

Table 48: **The whole store lowers Claude Opus 5.5 on nine of ten head-to-head problems and raises it on none.** Mean capped speedup by problem, 4 repetitions, for Claude Opus 5.5 (top) and Claude Opus 4.8 (bottom).

Problem	bare	show-all	provenance	selector	digest	tables	annotated
Claude Opus 5.5							
L2_014	1.00	1.00	1.00	1.00	1.01	1.00	1.01
L2_022	1.34	1.03	1.04	1.03	1.16	2.65	1.04
L2_030	1.27	1.03	1.02	1.03	1.24	1.03	1.03
L2_033	1.13	1.01	1.01	1.01	1.01	1.01	1.01
L2_050	10.00	10.00	10.00	10.00	10.00	10.00	5.82
L2_061	1.04	0.94	1.03	1.03	0.94	1.03	0.94
L3_010	0.90	0.76	0.85	0.77	0.74	0.77	0.71
L3_023	0.81	0.79	0.59	0.74	0.73	0.90	1.02
L3_032	7.22	6.82	5.24	6.49	5.89	5.31	6.75
L3_040	8.38	6.84	6.47	7.33	6.78	5.90	7.68
Claude Opus 4.8							
L2_014	0.00	0.75	0.75	1.00	0.75	0.75	0.75
L2_022	1.03	1.04	1.03	1.03	1.03	1.03	1.03
L2_030	0.51	1.03	1.02	1.03	1.03	1.02	1.02
L2_033	0.99	0.99	1.00	0.99	1.00	0.99	0.99
L2_050	0.83	1.09	1.10	1.10	1.10	1.10	1.10
L2_061	0.99	1.02	1.02	1.02	1.02	1.02	1.02
L3_010	0.38	0.80	0.81	0.73	0.72	0.87	0.82
L3_023	0.20	0.29	0.30	0.29	0.29	0.30	0.30
L3_032	0.88	0.81	0.64	0.86	0.85	0.88	0.89
L3_040	0.12	0.28	0.12	0.11	0.78	0.79	0.71

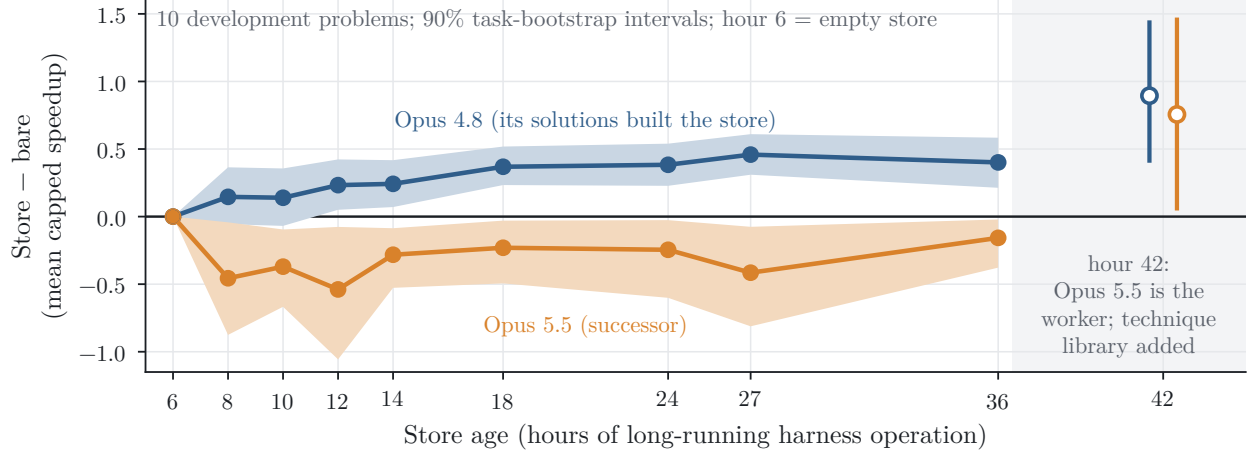


Figure 14: **Each snapshot from hour 8 to hour 36 slows the successor and lifts the predecessor.** Frozen snapshots minus bare in mean capped speedup on ten development problems; store cells $k = 4$, bare $k = 10$, 90% task-bootstrap intervals. Bare means: Claude Opus 5.5 $2.35\times$, Claude Opus 4.8 $0.78\times$. Kernel work began at hour 6.1. Hollow hour-42 points: the store after the upgrade, with the technique library added.

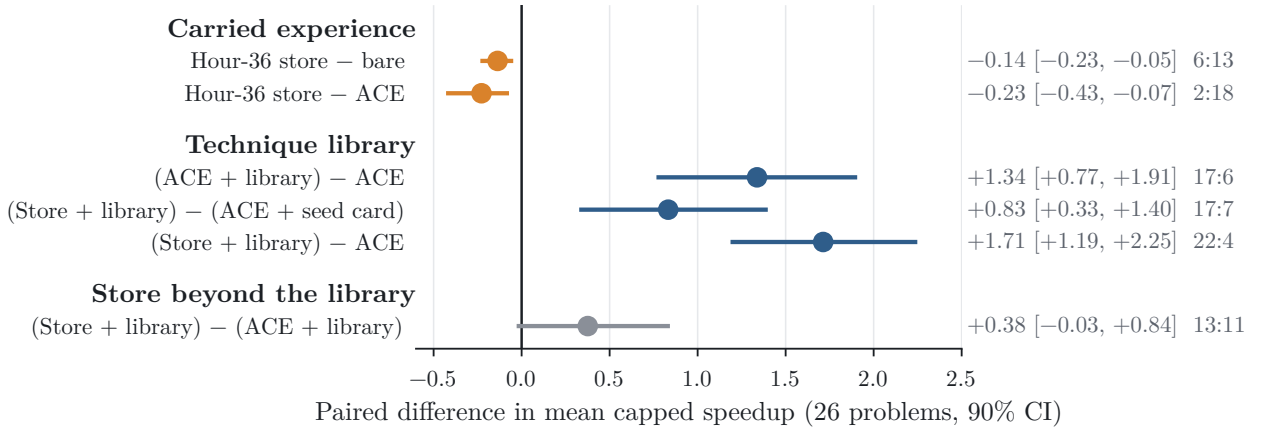


Figure 15: **The technique library lifts the store past ACE’s playbook and lifts ACE’s playbook too.** Claude Opus 5.5; differences in mean capped speedup over 26 problems outside the cards’ engine trials, paired by problem, with 90% task-bootstrap intervals and win:loss counts. The store is the long-running harness’s hour-36 store; ACE uses its own hour-36 playbook.

N.5 Verification capital in detail

Tables 49 and 50 and Figures 16 and 17 give the check library’s accrual by window, the revision history of one check, agreement and its labels on held-out τ^2 -bench, and depreciation at every freeze point.

Table 49: **The stock stops growing in window 8; its value keeps changing.** Pass/fail run. Left: accrual of the learning library by window bin, with failures (per attempt) the library missed, checks held at the end of the bin, and admitted mints, maturations, and other revisions. Right: catch (%) of the bin’s novel failure states by the learning library and by copies frozen after 50, 200, 800, and 1,800 failures; a dash marks a bin before the freeze.

Windows	Accrual						Catch of novel failure states (%)					
	Failures	Missed	Checks	Mints	Matur.	Other	States	Learning	50	200	800	1,800
1–4	200	168	39	30	5	9	150	7.3	–	–	–	–
5–10	600	60	48	5	29	6	277	83.4	22.0	70.8	–	–
11–18	800	9	48	0	6	2	171	94.7	10.5	70.2	88.3	–
19–26	800	8	48	0	7	0	181	95.6	15.5	74.6	87.8	–
27–32	600	8	48	0	7	0	148	94.6	5.4	79.7	84.5	87.8
33–38	607	37	48	0	23	4	259	87.6	10.0	44.4	68.0	68.7

Table 50: **Revision history of one learned check.** Template ‘follow all artists of all *genre* songs in my playlists’ in the pass/fail run: 314 attempts at three instances, 11 failures and 303 correct attempts. At each gate the candidate also fired on none of 150 successes of other templates. No version fired on a correct attempt.

Window	Event	Check and gate
1	Anticipation arm writes a check from two correct Claude Opus 5.5 attempts, before any failure	Missing, removed and extra follows; silent on 6 past successes
4	First failure: a Claude Haiku 4.5 attempt leaves one reggae artist unfollowed. The failure-only library has no check and passes it; the anticipation check flags it	Minted from that failure: the same three conditions; silent on 23 past successes, catches 1 of 1 past failure
6–7	Three Claude Haiku 4.5 failures, each with artists unfollowed	All three caught
8	A Claude Haiku 4.5 attempt follows all five classical artists, then submits <code>answer=5</code> to an action task; passed	Matured: wider instruction parser, and no answer on an action task; silent on 93 past successes, catches 5 of 5 past failures (old check 4)
23–37	Six failures: five by Qwen, one by Claude Haiku 4.5; three Qwen attempts submit answers	All six caught; the copy frozen after 200 failures passes the three answers

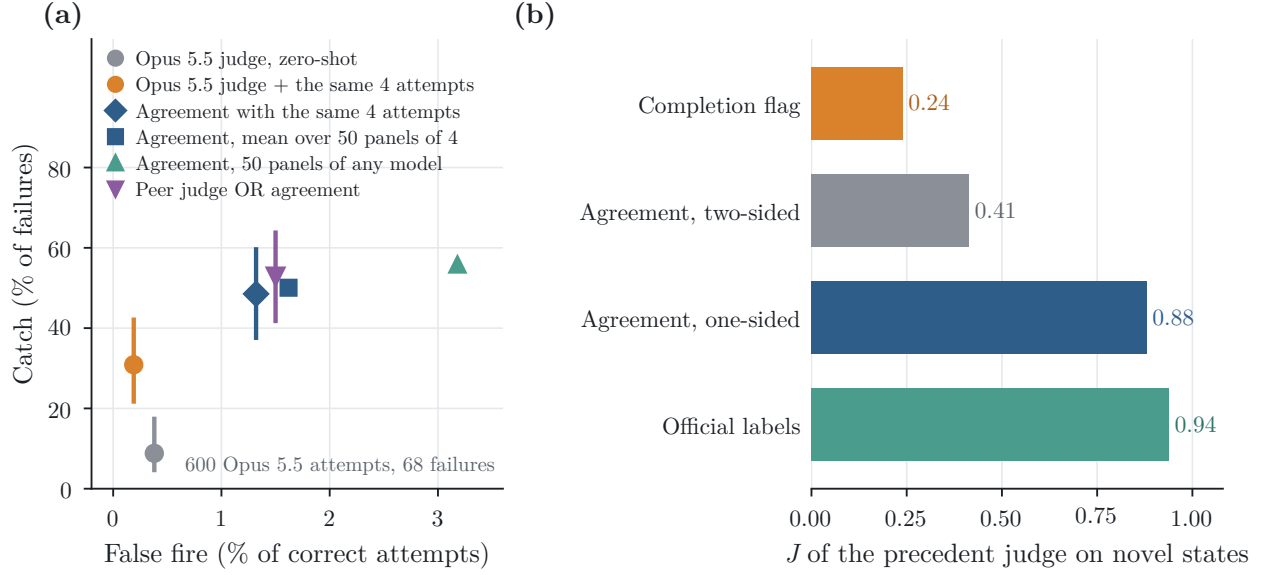


Figure 16: **Agreement catches more than both judges on held-out τ^2 -bench, and agreement labels teach a learner where completion flags do not.** (a) Held-out τ^2 -bench operating points for 600 Claude Opus 5.5 attempts, 68 of them failures; rates are per attempt, and bars on single-panel systems are Wilson 95% intervals on catch. (b) The same prequential Claude Opus 5.5 precedent judge learning from different label sources on AppWorld, evaluated on 1,036 novel failure states and 627 novel correct states. One-sided agreement reports agreement as success; two-sided agreement also calls non-plurality states failures.

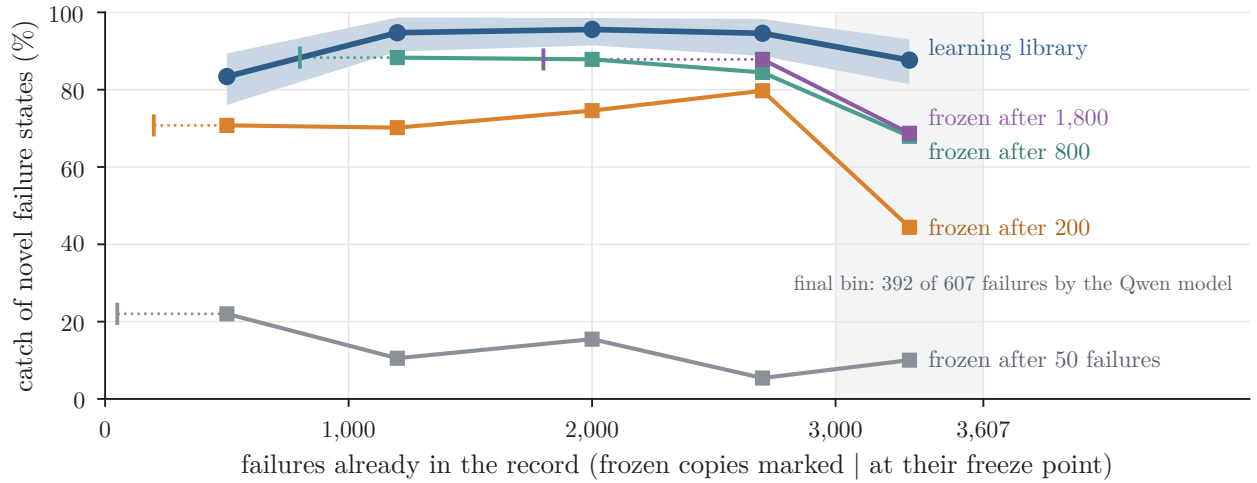


Figure 17: **Frozen copies fall behind at every freeze point.** Catch of novel AppWorld failure states by window bin. The library learns from past official pass/fail outcomes without grader text; its band is a 95% template-cluster bootstrap interval. Vertical marks indicate when copies were frozen. The shaded final bin has 392 Qwen failures out of 607; there, catch falls to 44.4–68.7% for copies frozen after 200 or more failures, while the learning library holds 87.6%.

On held-out τ^2 -bench, cross-harness Claude Opus 5.5 panels catch 51.6% of wrong-database failures, against 28.4% for panels drawn from other trials of the same run. Claude Haiku 4.5 panels catch 71.1% of failures at the Claude Opus 5.5 plurality, with 8.9% false fire on Claude Opus 5.5 successes against 1.7% for Claude Opus 5.5 panels. On the training record, other-model panels exceed same-model panels by $\Delta J = +5.5$ [+2.8, +8.1]. Before the library study, frozen Claude Opus 5.5-written checks caught 66.2% of 367 held-out other-instance failure states and flagged none of 370 distinct correct states. On novel failure states, the graded-run library caught 82.6% of Claude Haiku 4.5's (854), 91.1%

of Qwen’s (247), and 29 of 29 of Claude Opus 5.5’s; per attempt, it caught 100% of Claude Opus 5.5’s 53 failures with no flags on 6,865 correct attempts.

O Verbatim Artifacts

O.1 The two L2_074 kernels

The successor’s bare kernel (episode h1-00609, 9.63×) is a single CUDA C++ kernel compiled with `load_inline`. Weights for four output channels are staged in shared memory as `float4`; each thread accumulates, for two pooled positions, the eight parity outputs of the pooling window from a $2 \times 2 \times 3$ input neighborhood, then applies the epilogue and the maximum in registers. Its `forward` falls back to the reference operators for other shapes; on the benchmark’s shapes the kernel runs. The listing is verbatim with indentation halved.

```
__global__ void fused_kernel(const float* __restrict__ x, const float* __restrict__ w,
    const float* __restrict__ bias, const float* __restrict__ mul, float* __restrict__ out,
    int D, int H, int W)
{
    __shared__ float4 ws[16*27];
    const int ocg = blockIdx.y;
    const int n = blockIdx.z;
    const int tid = threadIdx.y * 16 + threadIdx.x;
    for (int i = tid; i < 16*27; i += 256) {
        int ci = i / 27, tap = i % 27;
        float4 v;
        v.x = w[(ci*32) + ocg*4 + 0]*27 + tap;
        v.y = w[(ci*32) + ocg*4 + 1]*27 + tap;
        v.z = w[(ci*32) + ocg*4 + 2]*27 + tap;
        v.w = w[(ci*32) + ocg*4 + 3]*27 + tap;
        ws[i] = v;
    }
    __syncthreads();
    const int htiles = H / 16;
    const int pd = blockIdx.x / htiles;
    const int ph = (blockIdx.x % htiles) * 16 + threadIdx.y;
    const int w0 = 2 + threadIdx.x;
    const bool vd = (pd + 1) < D;
    const bool vh = (ph + 1) < H;
    const bool vw = (w0 + 2) < W;

    float acc[2][8][4];
    #pragma unroll
    for (int j = 0; j < 2; j++)
        #pragma unroll
        for (int p = 0; p < 8; p++)
            #pragma unroll
            for (int o = 0; o < 4; o++) acc[j][p][o] = 0.f;

    const long long DHW = (long long)D * H * W;
    const float* xb = x + (long long)n * 16 * DHW;
    #pragma unroll 1
    for (int ci = 0; ci < 16; ci++) {
        const float* xc = xb + ci * DHW;
        float xin[2][2][3];
        #pragma unroll
        for (int dd = 0; dd < 2; dd++)
            #pragma unroll
            for (int hh = 0; hh < 2; hh++)
                #pragma unroll
                for (int ww = 0; ww < 3; ww++) {
                    bool valid = (dd == 0 || vd) && (hh == 0 || vh) && (ww < 2 || vw);
                    xin[dd][hh][ww] = valid ? __ldg(xc + ((long long)(pd + dd) * H + ph + hh) * W + w0 + ww) : 0.f;
                }
    }
    const float4* wc = ws + ci * 27;
    #pragma unroll
    for (int kd = 0; kd < 3; kd++) {
        const int a = (kd == 1) ? 0 : 1;
        const int id = (kd == 0) ? 1 : 0;
        #pragma unroll
        for (int kh = 0; kh < 3; kh++) {
            const int b = (kh == 1) ? 0 : 1;
            const int ih = (kh == 0) ? 1 : 0;
            #pragma unroll
            for (int kw = 0; kw < 3; kw++) {
                const int c = (kw == 1) ? 0 : 1;
                const int iw = (kw == 0) ? 1 : 0;
                const int par = a * 4 + b * 2 + c;
                float4 wv = wc[(kd * 3 + kh) * 3 + kw];
                #pragma unroll
                for (int j = 0; j < 2; j++) {
                    float xv = xin[id][ih][iw + j];
                    acc[j][par][0] = fmaf(wv.x, xv, acc[j][par][0]);
                    acc[j][par][1] = fmaf(wv.y, xv, acc[j][par][1]);
                }
            }
        }
    }
}
```

```

        acc[j][par][2] = fmaf(wv.z, xv, acc[j][par][2]);
        acc[j][par][3] = fmaf(wv.w, xv, acc[j][par][3]);
    }
}
}
}
const int PHW = H * W;
#pragma unroll
for (int o = 0; o < 4; o++) {
    const int oc = ocg * 4 + o;
    const float bv = bias[oc];
    const float mv = mul[oc];
    #pragma unroll
    for (int j = 0; j < 2; j++) {
        float mx = -3.402823466e38f;
        #pragma unroll
        for (int p = 0; p < 8; p++) {
            float v = acc[j][p][o] + bv;
            v = v > 0.f ? v : 0.2f * v;
            v = v * mv;
            v = v > 0.f ? v : 0.2f * v;
            mx = fmaxf(mx, v);
        }
        out[((long long)(n * 32 + oc) * D + pd) * PHW + ph * W + w0 + j] = mx;
    }
}
}
}

```

The successor's kernel with the store (episode h1-00814, 1.27×) keeps `nn.ConvTranspose3d` and adds one Triton kernel for the epilogue and pool. The listing is verbatim with indentation halved and the imports omitted.

```

@triton.jit
def fused_pool_kernel(x_ptr, m_ptr, out_ptr, C, D, H, W, OD, OH, OW, total,
    BLOCK: tl.constexpr):
    pid = tl.program_id(0).to(tl.int64)
    idx = pid * BLOCK + tl.arange(0, BLOCK).to(tl.int64)
    mask = idx < total

    ow = idx % OW
    t = idx // OW
    oh = t % OH
    t = t // OH
    od = t % OD
    nc = t // OD
    c = nc % C

    mult = tl.load(m_ptr + c, mask=mask, other=1.0)
    mult2 = mult[:, None]

    j = tl.arange(0, 2).to(tl.int64)
    D64 = D.to(tl.int64)
    H64 = H.to(tl.int64)
    W64 = W.to(tl.int64)
    mask2 = mask[:, None] & (j[None, :] < 2)

    res = tl.full((BLOCK,), float("-inf"), tl.float32)
    for dz in tl.static_range(2):
        iz = od * 2 + dz
        for dy in tl.static_range(2):
            iy = oh * 2 + dy
            rowbase = ((nc * D64 + iz) * H64 + iy) * W64 + ow * 2
            off = rowbase[:, None] + j[None, :]
            v = tl.load(x_ptr + off, mask=mask2, other=0.0)
            v = tl.where(v > 0, v, v * 0.2)
            v = v * mult2
            v = tl.where(v > 0, v, v * 0.2)
            res = tl.maximum(res, tl.max(v, axis=1))
    tl.store(out_ptr + idx, res, mask=mask)

class ModelNew(nn.Module):
    def __init__(self, in_channels, out_channels, kernel_size, stride, padding, output_padding, multiplier_shape):
        super().__init__()
        self.conv_transpose = nn.ConvTranspose3d(in_channels, out_channels, kernel_size, stride=stride,
            padding=padding, output_padding=output_padding)
        self.multiplier = nn.Parameter(torch.randn(multiplier_shape))

    def forward(self, x):
        y = self.conv_transpose(x).contiguous()
        N, C, D, H, W = y.shape
        OD, OH, OW = D // 2, H // 2, W // 2
        out = torch.empty((N, C, OD, OH, OW), device=y.device, dtype=y.dtype)
        total = N * C * OD * OH * OW
        m = self.multiplier.reshape(-1).contiguous()
        BLOCK = 256
        grid = (triton.cdiv(total, BLOCK),)
        fused_pool_kernel[grid](y, m, out, C, D, H, W, OD, OH, OW, total, BLOCK=BLOCK, num_warps=4)
        return out

```

O.2 A learned executable check

This check verifies the template ‘follow all artists of all *genre* songs in my playlists’ (Section 5.9). It is the window-8 maturation. The window-4 version differs in its instruction parser, which required ‘of all’ before the genre and accepted no ‘spotify’ before ‘playlists’, and in lacking the condition on the submitted answer. The code is verbatim, except that blank lines are removed and wrapped lines and adjacent string literals are joined.

```
import re
def probe(ctx):
    m = re.search(r"follow all (?:(the )?artists of (?:(all )?(?:the )?(.+?)[- ]genre songs in (?:(any of )?(?:all )?my (?:spotify )?playlists)", ctx.instruction, re.I)
    if not m:
        return {"applicable": False, "flag": False, "evidence": "instruction does not match this template"}
    genre = m.group(1).strip().lower()
    S = ctx.models.start.spotify
    E = ctx.models.end.spotify
    me = S.User.find_one(email=ctx.supervisor["email"])
    if me is None:
        return {"applicable": False, "flag": False, "evidence": "supervisor spotify user not found"}
    expected = set()
    for pl in S.Playlist.find_all(user_id=me.id):
        for s in pl.songs:
            if (s.genre or "").strip().lower() == genre:
                expected.update(s.artist_ids or [])
    if not expected:
        return {"applicable": False, "flag": False, "evidence": f"no {genre} songs found in the user's playlists in the start state"}
    start_follow = {f.artist_id for f in S.UserArtistFollowing.find_all(user_id=me.id)}
    end_follow = {f.artist_id for f in E.UserArtistFollowing.find_all(user_id=me.id)}
    missing = sorted(expected - end_follow)
    if missing:
        return {"applicable": True, "flag": True, "evidence": f"artists {missing} of {genre} songs in the user's playlists are not followed; expected all of {sorted(expected)} (check all playlist pages)"}
    removed = sorted(start_follow - end_follow)
    if removed:
        return {"applicable": True, "flag": True, "evidence": f"previously followed artists {removed} were unfollowed"}
    extra = sorted((end_follow - start_follow) - expected)
    if extra:
        return {"applicable": True, "flag": True, "evidence": f"newly followed artists {extra} are not artists of any {genre} song in the user's playlists (expected set {sorted(expected)})"}
    # This is an action task; no answer should be submitted.
    ans = ctx.predicted_answer
    if ans is not None:
        txt = str(ans).strip()
        if txt and txt.lower() not in ("none", "null"):
            return {"applicable": True, "flag": True, "evidence": f"the follows are correct, but this task requires no answer and the agent submitted answer {ans!r}; complete the task without an answer (answer=None)"}
    return {"applicable": True, "flag": False, "evidence": f"all {len(expected)} expected artists {sorted(expected)} are followed, no extra follows, no answer submitted"}
```