AhaBench: Do Agents Learn from Prior Experience? A Benchmark for Long-Horizon Continual Learning

Anonymous authors
Paper under double-blind review

Abstract

Modern language agents are expected to operate over long horizons: they ask follow-up questions, reuse worked examples, handle tool feedback, and adapt to delayed consequences. Most evaluations still reset the agent after a prompt or score only the final state of one trajectory. AhaBench asks a more operational question: when a fixed model receives useful experience, does its later behavior improve under a related evaluation condition where the obvious support has been removed, changed, or delayed? The suite contains three components. Aha-Puzzle tests no-hint exploration after solved hidden-state puzzles; Aha-Euler turns Project-Euler-style mathematical ideas into generated taught/held-out tasks with exact validators; and Aha-Vending, an open-source implementation inspired by Vending-Bench, tests whether a simulated vending agent remains profitable while handling delayed feedback and operational incidents. AhaBench reports a three-part scorecard: Initial Score measures starting competence, Post-Experience Score measures the later empirical outcome, and Learning Lift is their difference. This decomposition is the main empirical message: models that use visible support well, models that reach high post-experience scores, and models that improve most during a run are not always the same. On the common eight-model panel, Claude Opus 4.6 leads aggregate Post-Experience Score at 64.3 and aggregate Learning Lift at +25.8, with Gemini 3.1 Pro close behind at 63.4. The component results explain the split: puzzle traces raise supported scores but often fail to become no-hint exploration behavior; Aha-Euler full teaching reaches 78.6–100.0% while answer-only transfer ranges from 0.0 to 73.9%; and Aha-Vending separates profitable incident handling from bankruptcy and no-order failure. We release benchmark tasks, rubrics, generators, validators, simulator code, and model-evaluation interfaces for evaluating new agents.

1 Introduction

Many applications need an agent that improves within a run, not only an agent that is strong on the first prompt. A troubleshooting agent should avoid a failed query after reading logs; a mathematical agent should reuse an idea after one worked instance; a business agent should change stocking and pricing after delayed evidence about demand. In this paper, an "aha" moment is defined behaviorally: the same fixed model performs better later because earlier experience made a useful lesson available.

That question is mostly invisible in standard evaluations. Single-prompt benchmarks measure stored knowledge or immediate reasoning. One-trajectory agent benchmarks usually score the final state. Both are useful, but neither separates three cases that matter in practice: strong initial ability, temporary use of a visible hint, and durable improvement after the hint, code, or immediate feedback is removed. We study the fixed-weight setting, with no fine-tuning or gradient update; any improvement must come from context, memory, tool state, or inference-time reasoning.

AhaBench makes this setting measurable. Each component supplies an experience source and then tests a related evaluation condition that cannot be solved by copying the source. The support change is deliberate and domain-specific: Aha-Puzzle removes the trace prefix, Aha-Euler changes the input while preserving

the computational idea, and AHA-VENDING observes future business states caused by earlier decisions. The benchmark question is therefore concrete: does the model use earlier experience as a reusable strategy?

Throughout the paper, Initial Score B denotes the cold or early score, Post-Experience Score P denotes the later score after relevant experience, and Learning Lift is $\Delta = P - B$, the change on the same scale.

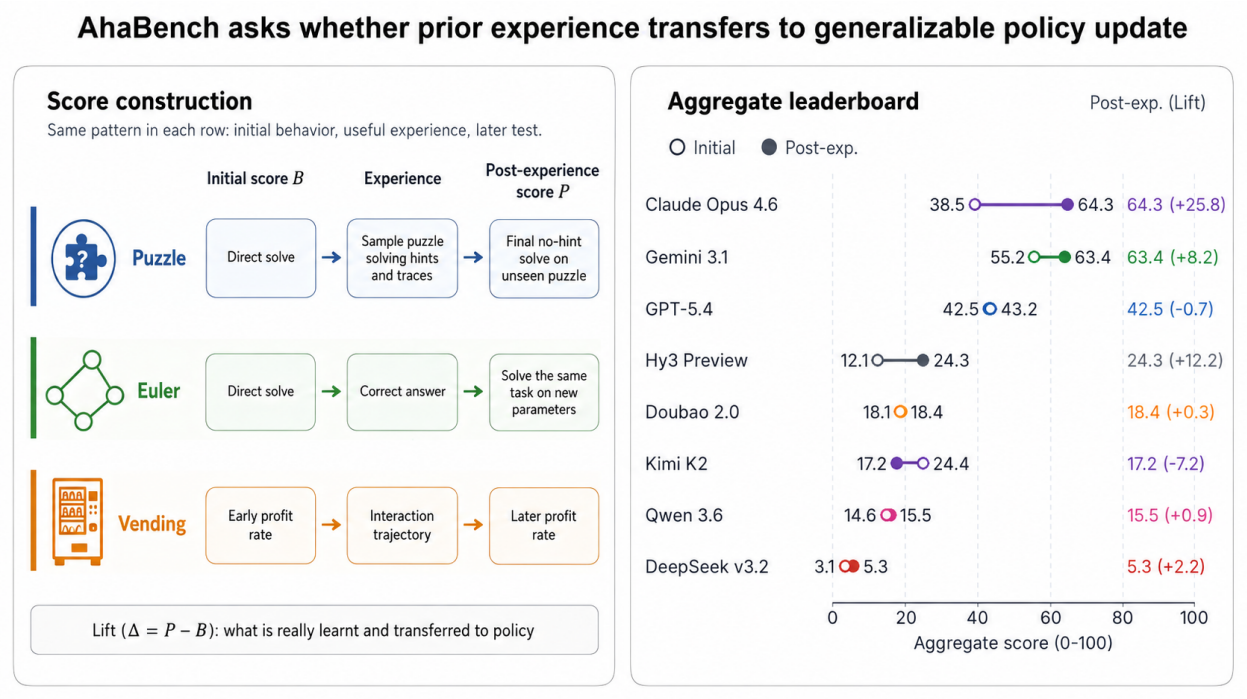

Figure 1: What AHABENCH measures. The left panel expands the common before–experience–after construction: Initial Score B is direct puzzle solving, cold mathematical solving, or early vending profit; experience is solved traces, a related taught task, or a business trajectory; Post-Experience Score P is the no-hint puzzle endpoint, answer-only held-out task, or incident-setting profit. The right panel reports the aggregate leaderboard sorted by P, with Learning Lift $\Delta = P - B$ in parentheses.

The three components are meant to be read as different forms of the same problem. AHA-PUZZLE measures exploration under hidden state: the agent asks up to 20 questions about a situation puzzle, receives minimal host answers, studies solved episodes in the curriculum, and is finally tested without a trace prefix. AHA-EULER measures reusable mathematical reasoning: Project-Euler-style problems are turned into executable generated tasks, and a taught instance must transfer to a held-out input. AHA-VENDING measures operational control: the agent runs a simulated vending business by reading email, ordering products, stocking slots, setting prices, handling weather and incidents, and preserving profit over hundreds of simulated days.

The main leaderboard shows this separation directly. Claude Opus 4.6 leads aggregate Post-Experience Score at 64.3, followed by Gemini 3.1 Pro at 63.4 and GPT-5.4 at 42.5. Lift gives a different view: Claude improves by +25.8, Hy3 Preview by +12.2, and Gemini by +8.2. The component scorecards explain why: GPT-5.4 leads Aha-Euler Post-Experience Score, Qwen has the strongest Aha-Puzzle Lift, and Claude is strongest on the incident-focused Aha-Vending evaluation. The benchmark therefore ranks models, but it also says which kind of experience each model uses in later behavior.

The contribution is a measurement pattern for long-horizon agent evaluation: preserve the structure of experience, define a related post-experience evaluation, compare Initial and Post-Experience scores, keep trajectories available, and report the failure mode. A strong model on this suite should ask better questions in AHA-PUZZLE, reconstruct methods with less procedural help in AHA-EULER, and maintain profitable, resilient operation in AHA-VENDING.

Our contributions are:

1. We formalize how to measure whether inference-time experience changes later policy in fixed-parameter language agents.
2. We introduce AhaBench, a three-part suite covering interactive puzzle exploration, generated mathematical reasoning, and simulated vending operation.
3. We provide fine-grained analyses showing that visible support, full worked procedures, and final reward measure different forms of experience use.
4. We instantiate the suite with deterministic generators, simulator code, scoring rubrics, model-evaluation interfaces, scripted baselines, uncertainty estimates, and reproducible benchmark data.

2 Long-Horizon Continual Learning

AhaBench compares the same fixed-parameter agent before and after relevant experience is available. Let h be the current interaction state and $\mathcal{D}_{<t}$ the prior curriculum, feedback, or environment trajectory. The measured object is the policy shift

$$\pi_\theta(a \mid h; \mathcal{D}_{<t}) \quad \text{versus} \quad \pi_\theta(a \mid h; \varnothing),$$

with fixed parameters θ. For an initial score S_{init} and a post-experience score S_{post}, the basic transfer estimate is

$$\text{gain} = S_{\text{post}} - S_{\text{init}}.$$

We call this quantity the *experience-use gain*. It asks whether prior interaction changes later behavior after the evaluation condition changes. Since no single scalar captures this setting, the paper reports a small metric surface. If S_{sup} is performance while trace support remains visible, the *support-reliance gap* is $S_{\text{sup}} - S_{\text{post}}$; in Aha-Puzzle this is the gap between trace-supported averages and the final no-hint challenge. In Aha-Euler, the analogous *procedure-reliance gap* is $S_{\text{full}} - S_{\text{partial}}$, separating answer-plus-code teaching from answer-only teaching. We also report *support-gain retention*, the ratio $(S_{\text{post}} - S_{\text{init}})/(S_{\text{sup}} - S_{\text{init}})$, whenever the support condition has a natural score. This ratio measures how much of the visible-support gain survives after support is reduced. Aha-Vending uses trajectories rather than one support condition, so the parallel diagnostic is a late-minus-early profit-efficiency slope over days or messages. For compact comparison, the main scorecard reports Initial Score B, Post-Experience Score P, and Learning Lift $\Delta = P - B$. Initial Score answers "where did the model start?"; Post-Experience Score answers "how strong is it after experience under the evaluation condition?"; Lift answers "how much did its aggregate score change after experience?" For clarity, conditions are named by the support available at evaluation time. In Aha-Puzzle, the *final no-hint challenge* is the last curriculum slot, where the agent receives no direct trace prefix for the final puzzle. In Aha-Euler, Q_{teach} denotes the taught instance and Q_{test} denotes the structurally related held-out instance. A graph-composition task decomposes Q_{teach} into dependent subcomputations: the *single-step* subset has one node, while the *composition* sweep uses two to five nodes. In Aha-Vending, the public evaluation is an incident-focused 365-day vending simulation with supplier delays, refunds, phishing, machine issues, weather, stockouts, and bankruptcy risk. Additional simulator variants are used as diagnostics, but the main scorecard reports this incident setting.

When both full and partial teaching are available, we also use

$$\text{retention} = \frac{S_{\text{partial}} - S_{\text{base}}}{S_{\text{full}} - S_{\text{base}}},$$

as a normalized comparison. The primary evidence remains the direct partial-teaching accuracy and the full–partial procedure gap; retention helps compare how much full-teaching gain survives when code and explanation are withheld.

AhaBench differs from ordinary in-context learning in the role played by the evaluation condition. In a standard few-shot prompt, the demonstration remains visible and can be locally copied. Here, the post-experience evaluation is related but non-identical: a new hidden-state puzzle with no trace prefix, a new Q_{test}

input whose Q_{teach} code is hidden in the partial condition, or a future simulator state produced by previous decisions. It also differs from parameter-update continual learning: there is no replay buffer, optimizer, or gradient update. The question is whether deployment-time history changes the behavior of the same fixed agent.

The three evaluations operationalize different forms of transfer. Aha-Puzzle withholds the trace prefix at the final challenge; Aha-Euler withholds explanation and code in partial teaching; and Aha-Vending evaluates the consequences of decisions over later simulated days with operational incidents.

Design principles. The benchmark design is organized around the evaluation shift. Post-experience evaluations are related to the support signal but not identical to it. Demonstration curricula end with a no-help evaluation: the final no-hint challenge in Aha-Puzzle and the held-out Q_{test} instance in Aha-Euler. Teaching ablations vary what the teacher reveals, separating answer-only transfer from answer-plus-code adaptation. Long-horizon runs preserve trajectories because many failures are temporal: repeated invalid actions, late passivity, or premature stopping. Released runs also include explicit coverage and scoring rules, and the three families are reported as related but distinct axes rather than collapsed into a single task type.

The measurements are deliberately plural. Aha-Puzzle separates direct score, trace-supported score, and final no-hint transfer gain; Aha-Euler separates full-teaching accuracy from answer-only partial-teaching accuracy; and Aha-Vending separates final net profit from curve shape, survival, and termination mode.

3 Benchmark Suite

Table 1: Benchmark settings and headline signals. The common pattern is experience followed by a changed evaluation: no-hint puzzle exploration, held-out mathematical transfer, and later operational control.

Benchmark	Experience source	Changed evaluation	Public scale	Headline result
Aha-Puzzle	Solved hidden-story episodes with trace prefixes, answer reveal, and reflection	Final hidden-story puzzle with no trace prefix	7 identities; 4 languages; 3 repeats; up to 20 questions	Qwen no-hint gain +12.15 pp; Hy3 Preview −1.50 pp
Aha-Euler	Taught instance Q_{teach}, with answer, explanation, and code in full teaching	Held-out Q_{test}; answer-only teaching hides the procedure	50 single-step tasks; 90 graph-composition tasks; 1–5 nodes; exact validators	Full teaching 78.6–100.0%; answer-only 0.0–73.9%
Aha-Vending	Simulated business trajectory: sales, stockouts, suppliers, cash, weather, incidents	Later days in the incident-focused vending simulation	Seeded 365-day runs; profit curves and termination modes	Claude +$2.8k; Gemini +$2.4k; weaker policies hit bankruptcy/no-order stops

Table 1 treats the three benchmark families as settings rather than as leaderboards. Each family makes a different kind of experience observable. Aha-Puzzle is about exploration under hidden state: the agent must decide which questions to ask in a new interactive environment. Aha-Euler is about mathematical reasoning: the agent must infer or reconstruct a computation from a related taught instance. Aha-Vending is about operational planning: the agent must use delayed consequences from a simulated business to make money without collapsing later in the horizon.

Source provenance and new construction. The data strategy is intentionally mixed. Aha-Puzzle is sourced and curated from classic lateral-thinking situation puzzles: short yes/no hidden-story puzzles, including the classic albatross puzzle in which the solver must uncover an unstated explanation through questions (Sloane, 1991; Wikipedia contributors, 2026). Human annotators verified the fine-grained hidden solution, allowable host answers, scoring rubric, and best-trace summaries before release. Aha-Euler follows the executable-specification view of mathematical reasoning in VeRA, where static math problems are turned into executable specifications with automatic verification (Cheng et al., 2026), and uses Project Euler as the source domain for mathematical-programming ideas (Project Euler, 2026). The benchmark instances are newly implemented by us: we convert each problem family into executable generators with a large input space, create paired taught and held-out instances with exact validators, and connect subcomputations in dependency graphs to control compositional hardness. Aha-Vending is inspired by the Vending-Bench line (Backlund & Petersson, 2025; Andon Labs, 2025); it contributes an open-source simulator, incident setting,

scripted baselines, trajectory summaries, and evaluation interfaces. In short, Aha-Puzzle is sourced and human-verified, Aha-Euler is Project-Euler-inspired but newly generated, and Aha-Vending is produced by our simulator.

Aha-Puzzle. Each sourced situation puzzle hides an explanation chain. The agent asks yes/no or short-answer questions, then gives a final explanation scored for correctness and efficiency. The curriculum reveals progressively shorter prefixes of human-verified best traces; after each puzzle, the correct answer is revealed and the model reflects. The final no-hint challenge receives no direct trace prefix, so it tests whether prior solved episodes changed the exploration policy. Aha-Puzzle contains seven puzzle types in four languages, with fine-grained rubrics that check explanation coverage rather than title recognition. Seven identities are used because the curriculum has seven positions: within each language-and-repeat schedule, every identity appears exactly once in each position and exactly once as the final no-hint evaluation. This balanced schedule controls for puzzle identity and position difficulty instead of letting an easier puzzle dominate a particular curriculum slot. The measurement unit is a long interactive trajectory, not a puzzle title: each puzzle unfolds across languages, repeats, curriculum slots, host responses, reflections, judged explanations, and up to 20 question-answer turns per episode. We report endpoint bootstrap intervals and a puzzle-cluster bootstrap to show uncertainty at the same interaction granularity.

The hidden-state setting is intentionally interactive. A non-interactive multiple-choice version would mainly test recognition of a twist. Here the agent must decide what to ask, when a hypothesis has failed, and whether the final explanation is sufficient to stop. Best traces are therefore not just answer keys: they demonstrate information-gathering strategies such as testing identity reversals, locating time shifts, pinning down ambiguous objects, and closing explanation gaps.

Aha-Puzzle scale and scoring. The Aha-Puzzle unit is a long interaction, not a one-shot title. Per language, a model runs a direct baseline plus seven curriculum slots over the seven identities, yielding $7 + 7 \times 7 = 56$ episodes before repeats; with four languages, three repeats, host responses, reflections, and a 20-question cap, each model evaluation contains hundreds of back-and-forth interaction rounds. The statistical unit is therefore the balanced endpoint trajectory and the puzzle identity, not a single prompt. The scoring pipeline separates ground-truth construction from rubric execution. Human annotators built and checked the hidden solution, allowable host answers, fine-grained rubrics, and best-trace summaries. For scoring, GPT-5.4 executes the rule-based rubric on final explanations. Three human annotators checked 100 sampled graded trajectories from the larger evaluation pool, covering baseline and curriculum episodes across models and languages, and the sampled grades matched the rubric-executor outputs in all cases.

Aha-Euler. Each Aha-Euler family is a generated mathematical-reasoning problem with two related instances. The taught instance Q_{teach} provides experience; the held-out instance Q_{test} tests whether the model can apply the same computation to new inputs. In full teaching, Q_{teach} reveals the answer, explanation, and code, so success can come from finding, retaining, and executing a worked procedure in a long context. In partial teaching, the model sees the Q_{teach} problem statement and answer only, so success requires reconstructing the method. The contrast creates a staged measurement: long-context procedure extraction, then answer-only method learning.

The graph-composition abstraction is the non-trivial design choice. Project Euler supplies the motivating genre: compact mathematical problems whose solutions usually require both insight and code. Our contribution is to turn that genre into a controllable benchmark rather than a fixed answer set. A hard mathematical problem is rarely a monolith; it is a composition of subcomputations. Decomposing Q_{teach} into a dependency graph lets the benchmark reveal feedback locally and then test whether the model composes those local computations on Q_{test}. Single-step tasks test one computational step; composition tasks stress dependency management, intermediate variables, and longer recombination. Exact-match validation is strict but auditable, and Python is allowed because the benchmark evaluates method inference and execution rather than mental arithmetic.

Aha-Vending. The vending agent operates a simulated business: it orders products, stocks slots, sets prices, reads email, searches suppliers, waits, and collects cash. Feedback is implicit in sales, stockouts, failed

orders, weather, incidents, and bankruptcy. Aha-Vending builds on the long-horizon vending-simulator idea introduced by Vending-Bench and Vending-Bench 2, while contributing an open-source simulator tailored to experience-use analysis. The main benchmark setting is the incident-focused single-agent environment: the agent must stay profitable while handling delayed supplier responses, refunds, phishing, machine issues, weather, cash-flow pressure, and other operational events.

The value of Aha-Vending is that the feedback signal is delayed and operational rather than labeled. A bad slot-size hypothesis can waste weeks, overpricing can silently suppress demand, ignoring supplier messages can create cascading stockouts, and over-conserving messages can look rational locally while destroying late-horizon profit. The incident setting makes these failures visible through profit curves, survival, and termination reasons rather than through a single final answer.

Design process and value. We built the suite around failure modes seen in deployed agents rather than by enlarging a static task set. Each component is useful as a standalone agent benchmark: Aha-Puzzle measures exploration and long-horizon planning under hidden state; Aha-Euler measures agentic solution of difficult STEM-style generated tasks; and Aha-Vending measures practical operational control in a money-making simulation. AhaBench turns these components into a continual-learning testbed by adding a source of experience: prior puzzle traces and revealed answers, teaching on a related mathematical assignment, or environment consequences in a vending simulator. It then defines a related but non-identical evaluation: a no-hint puzzle, a new held-out math input from the same algorithmic family, or later simulated business states under incidents. These comparisons separate trace use from exploration transfer, code-supported adaptation from answer-only learning, and short-term profit from durable operation.

Evaluation summaries. Uncertainty is reported at the level where each benchmark varies. Aha-Puzzle uses endpoint-unit and puzzle-cluster bootstrap intervals over final no-hint outcomes. Aha-Euler reports exact-match rates with Wilson intervals for answer-only teaching and the full–partial procedure gap. Aha-Vending reports incident-setting seed intervals, trajectories, and termination modes. The output is a capability surface plus a compact scorecard for readers who want a quick model comparison.

4 Experimental Setup

All numeric claims are derived from benchmark summaries, CSVs, plots, traces, and analysis scripts. Model names denote dated snapshot labels where available; the intended comparison is between learning behaviors under the same benchmark protocol.

Models and aggregation. Aha-Puzzle and Aha-Euler use the same eight-model panel as the cross-suite leaderboard. Language-level Aha-Puzzle means are averaged to keep the four languages balanced, and endpoint bootstrap intervals are computed over scored trajectories. Aha-Euler reports exact-match accuracy on deterministic generated tasks after applying the public scoring rules. Aha-Vending reports average net profit over seeded incident-setting runs and includes additional Aha-Vending-only models as anchors. Display names are normalized in tables, while raw provider identifiers are preserved in the released evaluation records.

Table 2: Evaluation scale and scoring. The columns indicate which evidence each benchmark preserves: human-checked rubric rules, exact validators, trajectories, executable code, and released objects.

Benchmark	Scale	Score	Human	Exact	Traj.	Code	Released objects
Aha-Puzzle	$7 \times 4 \times 3$; baseline + 7 slots	rubric total	✓	×	×	✓	rubrics, specs, runners
Aha-Euler	50 single + 90 graph-composition tasks; 1–5 nodes	exact answer	×	✓	×	✓	generators, validators, tasks
Aha-Vending	incident setting; seeded 365 d	profit/survival	×	△	✓	✓	simulator, configs, runners

5 Main Leaderboard and Metrics

The scorecard reports the state of a model at the start and after relevant experience. Initial Score B is the cold or early score, Post-Experience Score P is the later score under the changed evaluation condition, and Learning Lift $\Delta = P - B$ is the change on the same 0–100 scale. The official leaderboard is sorted by aggregate Post-Experience Score P, because later competence is the quantity users ultimately rely on; Lift is reported beside it to show how much the measured score changed during the run. Table 3 gives the same triplet for the aggregate suite and for each component, so the source of each aggregate rank remains visible.

Table 3: Initial–Post-Experience–Lift leaderboards on the common eight-model panel. The first-glance takeaway is axis disagreement: Claude leads aggregate Post-Experience Score and aggregate Lift, GPT-5.4 leads Aha-Euler Post-Experience Score, and Qwen leads Aha-Puzzle Lift.

Aggregate					Aha-Puzzle				
Rank	Model	Initial	Post-exp.	Lift	Rank	Model	Initial	Post-exp.	Lift
1	Claude 4.6	38.5	**64.3**	+25.8	1	Gemini 3.1	41.2	**44.3**	+3.1
2	Gemini 3.1	55.2	**63.4**	+8.2	2	Qwen 3.6	24.1	**36.3**	+12.2
3	GPT-5.4	43.2	**42.5**	-0.7	3	Doubao 2.0	31.7	**35.3**	+3.6
4	Hy3 Preview	12.1	24.3	+12.2	4	Claude 4.6	27.5	29.6	+2.1
5	Doubao 2.0	18.1	18.4	+0.3	5	GPT-5.4	28.9	28.7	-0.2
6	Kimi K2	24.4	17.2	-7.2	6	DeepSeek v3.2	9.4	15.3	+5.9
7	Qwen 3.6	14.6	15.5	+0.9	7	Kimi K2	16.2	14.7	-1.5
8	DeepSeek v3.2	3.1	5.3	+2.2	8	Hy3 Preview	13.7	12.2	-1.5

Aha-Euler					Aha-Vending				
Rank	Model	Initial	Post-exp.	Lift	Rank	Model	Initial	Post-exp.	Lift
1	GPT-5.4	45.9	**73.9**	+28.0	1	Claude 4.6	48.1	**98.5**	+50.4
2	Claude 4.6	39.9	**64.9**	+25.0	2	Gemini 3.1	100.0	**86.7**	-13.3
3	Gemini 3.1	24.4	**59.3**	+34.9	3	Hy3 Preview	22.6	**60.7**	+38.1
4	Doubao 2.0	9.5	20.0	+10.5	4	Kimi K2	56.9	36.9	-20.0
5	Qwen 3.6	0.0	8.1	+8.1	5	GPT-5.4	54.7	24.8	-29.9
6	Kimi K2	0.0	0.0	+0.0	6	Qwen 3.6	19.6	2.2	-17.4
7	Hy3 Preview	0.0	0.0	+0.0	7	DeepSeek v3.2	0.0	0.7	+0.7
8	DeepSeek v3.2	0.0	0.0	+0.0	8	Doubao 2.0	13.2	0.0	-13.2

Table 3 makes the scorecard explicit. Claude leads aggregate Post-Experience Score and aggregate Lift, GPT-5.4 leads Aha-Euler Post-Experience Score, and Qwen leads Aha-Puzzle Lift. These disagreements are the point of the benchmark: experience use is not a single axis, and a leaderboard should expose the axis on which a model improves.

Reading the scorecard. A model report should read Initial Score, Post-Experience Score, and Lift beside the component axes. If a model ranks well because of Aha-Vending but weakly on Aha-Puzzle, the evidence points toward operational control rather than hidden-state exploration. If it succeeds under answer-plus-code Aha-Euler teaching but fails under answer-only teaching, the bottleneck is reconstructing the computational idea when the procedure is no longer visible. If it stays profitable in Aha-Vending, the issue is not ordinary vending arithmetic but resilience to procurement failures, adversarial incidents, and delayed state. These are different model-improvement directions, and the benchmark keeps them visible beside the scorecard.

Using the artifact. The released code and data let users run the components, inspect task definitions, regenerate Aha-Euler answers, execute Aha-Vending policies, apply the Aha-Puzzle rubrics, and compute the same Initial/Post-Experience/Lift scorecards for new model endpoints. A new model can be compared at leaderboard level and then inspected at the level where behavior changed: a question asked too late, a recurrence derived incorrectly, a supplier constraint forgotten, or a profitable early strategy abandoned before the horizon ends.

6 Results

The results are organized by the support signal that changes. Aha-Puzzle removes the trace prefix, Aha-Euler removes the worked procedure, and Aha-Vending delays feedback through future simulator states. Each table is therefore a comparison between an experience source and a harder evaluation condition, rather than a collection of unrelated scores.

The same pattern appears in three forms. Supported context is much easier than support-free transfer: every reported Aha-Puzzle model scores far higher with trace support than at the final no-hint endpoint, and only Qwen remains clearly positive under puzzle-cluster uncertainty. Aha-Euler separates procedure use from answer-only transfer: full teaching makes every reported model high-accuracy on single-step tasks, but four of eight models fall to 20% or below when code and explanations are withheld. Aha-Vending separates robust operation from brittle business behavior: Claude and Gemini stay strongly profitable under adversarial incidents, while Doubao and DeepSeek approach the bankruptcy floor.

The companion diagnostics capture a practical tension in long-context experience. More history gives the model demonstrations, answers, tool feedback, and environmental consequences; it also gives the model more irrelevant state to filter. A high supported score with low support-gain retention means the model can use visible context without carrying the strategy into the harder evaluation. A positive late-vs-early vending slope is the Aha-Vending form of implicit learning: the agent turns operational consequences into a better later earning rate. A negative profit/day slope, as in Kimi, indicates late-horizon passivity. Appendix E gives representative diagnostic rows.

6.1 Aha-Puzzle: Supported Context Is Not Transfer

Trace-supported curriculum scores are far above final no-hint scores for every reported model. Table 4 reports the direct baseline, trace-supported average, final no-hint endpoint, and no-hint gain on the eight-model panel. Qwen has the clearest transfer gain; DeepSeek and Doubao improve from lower baselines; GPT-5.4, Claude, Gemini, Kimi, and Hy3 Preview show that high supported scores do not automatically become no-hint exploration policy.

Table 4: Aha-Puzzle eight-model panel. The key pattern is support dependence: trace-supported scores are high for every model, but durable no-hint gain is model-dependent.

Model	Baseline	Supported avg.	Final no-hint	Gain [95% CI]
Gemini 3.1	41.24	**75.07**	**44.29**	+3.05 [-2.52,+10.00]
Qwen 3.6	24.12	69.59	36.27	**+12.15** [+7.46,+19.97]
Doubao 2.0	31.67	69.72	35.32	+3.66 [-0.84,+9.75]
Claude 4.6	27.52	67.01	29.57	+2.05 [-6.00,+13.20]
GPT-5.4	28.93	66.86	28.67	-0.26 [-9.00,+22.33]
DeepSeek v3.2	9.40	58.70	15.33	+5.93 [-1.33,+14.24]
Kimi K2	16.17	64.69	14.68	-1.49 [-22.50,+4.27]
Hy3 Preview	13.70	62.20	12.20	-1.50 [-24.00,+3.00]

The released aggregate and the bootstrap table answer complementary questions. The aggregate averages language-level means, while the bootstrap table resamples scored trajectories or puzzle identities. In the main table we report the language-balanced scores together with puzzle-cluster uncertainty for the final no-hint gain. Both views show the same structure: supported context is much easier than no-hint transfer, and positive transfer is a measurable, model-dependent capability.

6.2 Aha-Euler: Full Solutions Mask the Hard Part

On the single-step subset, full teaching makes accuracy high for nearly every model, but answer-only teaching separates models sharply (Table 5). GPT-5.4, Claude Opus 4.6, and Gemini solve many Q_{test} instances after seeing only the Q_{teach} problem and answer; Qwen, DeepSeek, Kimi, and Hy3 Preview mostly require explicit procedure. The point of the ablation is this separation: Aha-Euler distinguishes procedure use under full worked support from transfer when the executable procedure must be reconstructed.

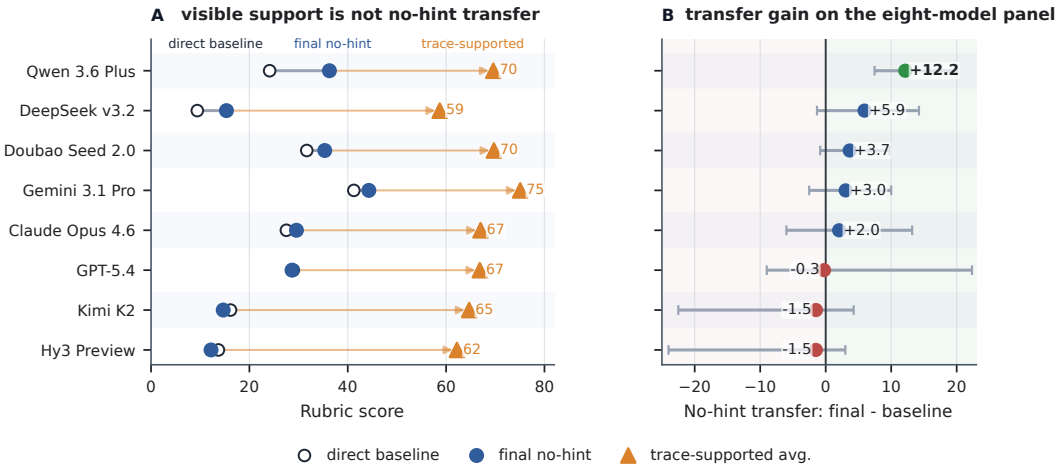

Figure 2: Aha-Puzzle support use versus no-hint transfer. Visible traces raise curriculum scores, but the final no-hint endpoint tests whether exploration behavior changed after support is removed.

Table 5: Aha-Euler single-step teaching-signal ablation. The key pattern is procedure dependence: answer+code teaching is high for nearly every model, while answer-only transfer reveals whether the computation was reconstructed rather than copied.

Model	Cold	Answer+code	Answer-only [95% CI]	Retained gain	Procedure gap ↓
GPT-5.4	45.9	97.5	**73.9** [53.5,87.5]	0.54	**23.6 pp**
Claude Opus 4.6	39.9	**100.0**	**64.9** [49.5,77.9]	0.42	35.1 pp
Gemini 3.1	24.4	84.2	**59.3** [40.7,75.5]	**0.58**	**24.9 pp**
Doubao 2.0	9.5	91.4	20.0 [8.9,39.1]	0.13	71.4 pp
Qwen 3.6	0.0	95.9	8.1 [2.8,21.3]	0.08	87.8 pp
Kimi K2	0.0	**100.0**	0.0 [0.0,39.0]	0.00	100.0 pp
DeepSeek v3.2	0.0	91.8	0.0 [0.0,9.2]	0.00	91.8 pp
Hy3 Preview	0.0	78.6	0.0 [0.0,24.3]	0.00	78.6 pp

Lower full–partial gap means less reliance on full procedural support. The smaller gaps for GPT-5.4, Gemini, and Claude indicate substantially stronger answer-only transfer than rows whose full-teaching success mostly disappears when code and explanation are removed. The staged interpretation is useful. If a model fails under full teaching, the bottleneck is already long-context extraction and execution. If it succeeds under full teaching but fails under answer-only teaching, the bottleneck is method reconstruction. Support-gain retention summarizes how much of the full-teaching gain survives this harder second step.

We inspected partial-teaching traces to connect the aggregate rates to behavior. Successful GPT-5.4 traces include fresh Python programs, tool-feedback repair, and reparameterization for Q_{test}, rather than copying exposed reference code. Failed Qwen and DeepSeek traces often contain substantive executable attempts, locating the remaining gap in algorithm design and implementation. An original-versus-hardened baseline probe gives 15.6% versus 11.7% micro-average accuracy, a small +3.9 point original advantage, supporting the interpretation that generated and hardened task variants substantially reduce direct recall of canonical Project Euler instances.

Graph depth provides an additional stressor. The appendix reports a compositional $\geq$ 2-node roll-up (Table 13) that summarizes how the generator moves from single-step transfer to multi-step composition across the public sweep.

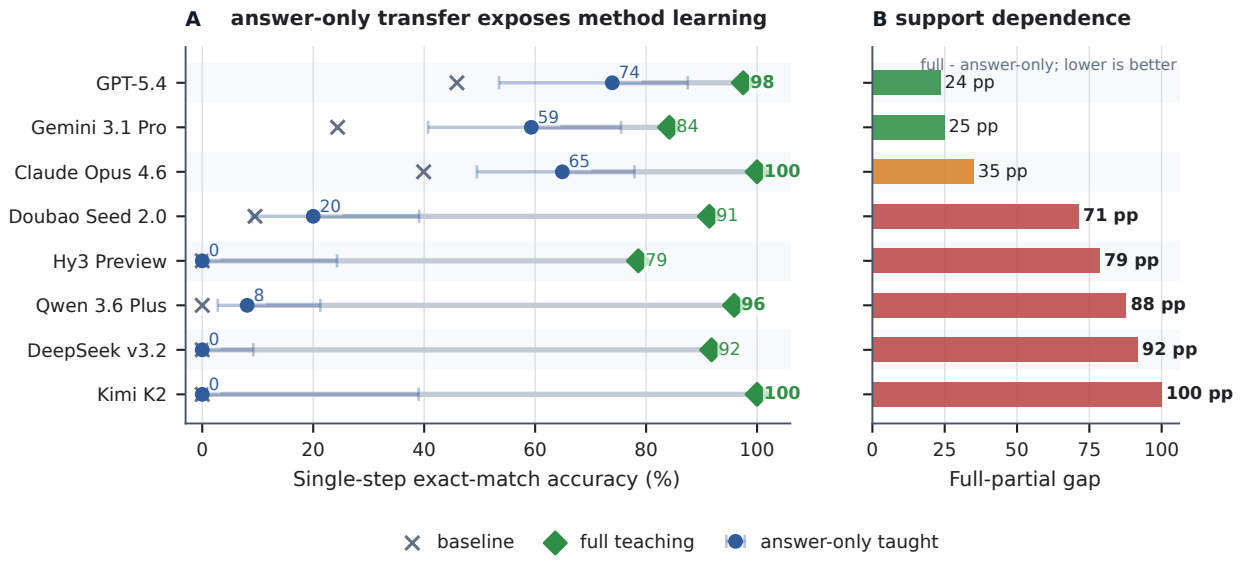

Figure 3: Aha-Euler partial versus full teaching on single-step tasks. Full solutions can mask the hard part: Panel A separates baseline, answer-only transfer, and answer-plus-code teaching; Panel B shows the full–partial gap.

6.3 Aha-Vending: Incidents Reveal Operational Failure

Table 6 reports Aha-Vending under its incident setting: the agent runs a seeded 365-day vending business while handling supplier delays, refunds, phishing, machine issues, and other adversarial operational events. This is the main Aha-Vending evaluation. Claude and Gemini are profitable across all seeds; Hy3 Preview often preserves value but terminates early without successful orders; Doubao and DeepSeek collapse to the bankruptcy floor. The result is a business-control benchmark rather than a pure profit leaderboard: profit, completed horizon count, mean simulated days, and termination mode together reveal whether a policy is robust or merely lucky.

Table 6: Aha-Vending incident-setting results. The evaluation combines delayed feedback with adversarial operational events; profit should be read with horizon completion and termination mode. Values are averaged over seeds and rounded for display.

Model	Profit ↑	95% interval	Completed	Days	Main termination
Claude 4.6	**+$2.8k**	[+$0.8k,+$4.8k]	5/5	365	completed
Gemini 3.1	+$2.4k	[+$1.5k,+$3.4k]	5/5	365	completed
Hy3 Preview	+$1.6k	[+$0.7k,+$2.4k]	1/5	288	no-order stop
Kimi K2	+$0.7k	[-$0.3k,+$1.8k]	5/5	365	completed
GPT-5.4	+$0.3k	[-$0.1k,+$0.7k]	5/5	365	completed
Qwen 3.6	-$0.4k	[-$0.6k,-$0.3k]	2/5	247	bankruptcy
DeepSeek v3.2	-$0.5k	[-$0.5k,-$0.5k]	0/5	130	bankruptcy
Doubao 2.0	-$0.5k	[-$0.5k,-$0.5k]	0/5	162	bankruptcy
GLM 5.1	**+$2.9k**	[+$2.4k,+$3.3k]	5/5	365	completed
GPT-4o	-$0.5k	[-$0.5k,-$0.4k]	0/5	150	bankruptcy

The Aha-Vending plot shows the same pattern visually. Claude and Gemini remain well above zero despite incidents; GPT-5.4 and Kimi survive but with much smaller margins; several weaker policies are clustered near the bankruptcy floor. The seed ranges matter because an apparently positive mean can hide brittle termination behavior.

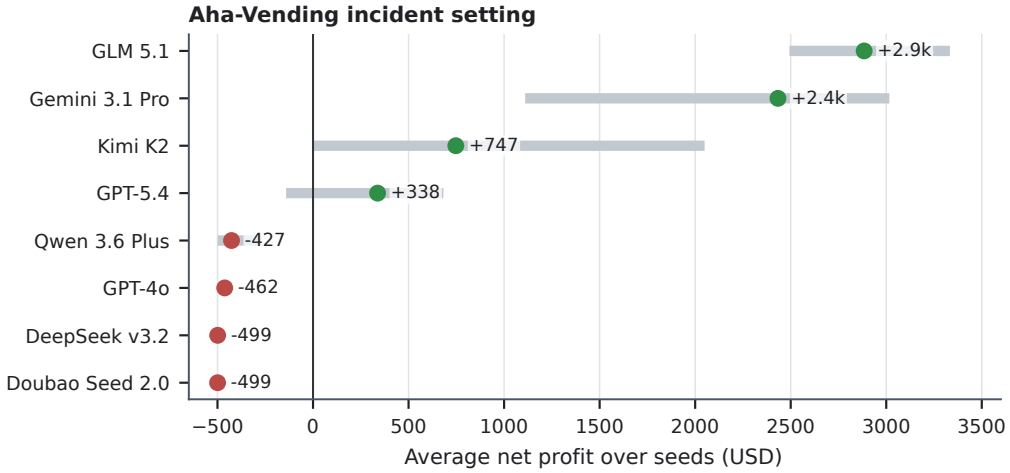

Figure 4: Aha-Vending incident-setting profit. The key message is operational robustness: profitable models remain above zero under adversarial events, while brittle policies collapse toward the bankruptcy floor. Points show seed means and horizontal bars show seed ranges.

Table 7: Descriptive overlap analysis across AhaBench axes. The correlations are not a new leaderboard; they show that puzzle exploration, Euler-style mathematical transfer, and vending operation are not interchangeable skills.

Axis pair	Models	ρ	Overlap	Takeaway
Puzzle gain vs. Euler post-exp. score	8	0.05	near zero	Exploration transfer and method reconstruction move separately.
Puzzle gain vs. Aha-Vending profit	8	−0.63	negative	Hidden-state learning does not imply robust business operation.
Euler post-exp. score vs. Aha-Vending profit	8	0.28	weak	Mathematical transfer and incident handling are distinct axes.

7 Cross-Benchmark Analysis

The three benchmarks reveal different axes of experience use rather than one universally best model. Aha-Puzzle asks whether a model learns how to ask. Aha-Euler asks whether it can reconstruct a computational method without exposed code. Aha-Vending asks whether discovered constraints, supplier reliability, cash flow, and message budget become durable operational policy.

The axis disagreement is the main empirical signal. A model can have high direct hidden-state score but modest transfer gain, high full-teaching math accuracy but near-zero answer-only transfer, or positive vending profit with fragile incident handling. Table 7 is a descriptive overlap analysis: it shows how the same eight-model panel moves across axes. We use these correlations as compact evidence that the axes are not interchangeable, not as stable population estimates of capability relationships. The disagreements suggest that "learning from experience" decomposes into separable skills.

8 Related Work

Benchmark papers as measurement artifacts. Influential benchmarks do more than collect tasks: they define what should be measured and make comparison easy to audit. ImageNet made scale, public protocol, and leaderboard central to vision (Deng et al., 2009); GLUE and SuperGLUE did the same for language understanding (Wang et al., 2019b;a); MMLU, BIG-bench, and HELM extended the pattern to broad language-model evaluation with multi-metric reporting (Hendrycks et al., 2021a; Srivastava et al., 2023;

Liang et al., 2023). AhaBench keeps the useful parts of that tradition, a clear leaderboard and reproducible artifacts, while changing the unit of measurement from an isolated item to an experience-evaluation pair.

Continual learning, meta-learning, and in-context learning. Classical continual learning studies parameter adaptation without catastrophic forgetting (Ratcliff, 1990; McCloskey & Cohen, 1989; Thrun, 1998; Kirkpatrick et al., 2017; Lopez-Paz & Ranzato, 2017; Parisi et al., 2019); meta-learning and test-time adaptation study fast use of new data at evaluation time (Vinyals et al., 2016; Snell et al., 2017; Finn et al., 2017; Sun et al., 2020). In-context learning shows that demonstrations can improve immediate answers (Brown et al., 2020; Wei et al., 2022; Kojima et al., 2022), while demonstration analyses show that visible examples need not imply semantic learning (Min et al., 2022). AhaBench takes that ambiguity as the object of study: the model is fixed, experience is deployment-time history, and the evaluation changes so that copying the support signal is not enough.

Component lineages. Aha-Euler draws on math, code, and executable-validation benchmarks: GSM8K, MATH, Minerva, BBH, Codex, AlphaCode, PAL, VeRA, and Project Euler (Cobbe et al., 2021; Hendrycks et al., 2021b; Lewkowycz et al., 2022; Suzgun et al., 2023; Chen et al., 2021; Li et al., 2022; Gao et al., 2023; Cheng et al., 2026; Project Euler, 2026). Aha-Puzzle connects to lateral-thinking and incomplete-information puzzle evaluation (Sloane, 1991; Wikipedia contributors, 2026; Jiang et al., 2023; Huang et al., 2024; Chen et al., 2024). Aha-Vending connects to long-horizon agent benchmarks and most directly to Vending-Bench's business-simulation setting (Yao et al., 2022; Shridhar et al., 2021; Yao et al., 2023b; Shinn et al., 2023; Wang et al., 2023; Zhou et al., 2023; Jimenez et al., 2024; Xie et al., 2024; Chan et al., 2024; Mialon et al., 2023; Backlund & Petersson, 2025; Andon Labs, 2025). AhaBench connects these threads by asking what behavior survives after interaction when the support signal is reduced, delayed, or changed. Appendix A gives the longer positioning survey.

Where AhaBench differs. The closest existing benchmarks either broaden coverage over static items or evaluate agents in a single environment trajectory. AhaBench instead makes the experience-to-evaluation relation the unit of measurement: a solved interactive episode should improve later no-hint exploration, a taught computation should transfer when the worked procedure is withheld, and delayed operational consequences should improve later decisions. This is why the paper reports support conditions, post-experience outcomes, retained gain, trajectory shape, and failure mode together rather than only a final scalar.

9 Conclusion

AhaBench asks a measurable question: after interaction, does a fixed language agent behave better under a related future evaluation? The answer cannot be read from an endpoint alone. A trace-supported Aha-Puzzle score is not the same as asking the right questions without a trace; a full-teaching Aha-Euler success is not the same as inferring the method from an answer alone; and a profitable Aha-Vending run can still hide bankruptcy risk, delayed recovery, or late passivity.

The benchmark is therefore organized around experience-and-evaluation pairs. Each component supplies experience, then changes the evaluation condition so copying visible support is insufficient. The leaderboard ranks models by Post-Experience Score, while Initial Score and Learning Lift show whether the result came from prior competence, adaptation during the run, or both. This gives AhaBench the intended shape of an agent benchmark: a public scorecard for new model endpoints, plus enough structure to diagnose which form of experience use failed.

For model development, the useful signal is the disagreement between axes. Improving Aha-Puzzle means better exploration under uncertainty; improving Aha-Euler means recovering a reusable computation from sparse teaching; improving Aha-Vending means maintaining profitable operational routines under delayed and adversarial feedback. AhaBench is designed so these improvements can be reported separately and then aggregated transparently, rather than hidden inside a single opaque score.

References

Andon Labs. Vending-bench 2. `https://andonlabs.com/evals/vending-bench-2`, 2025. Accessed 2026-05-29.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Axel Backlund and Lukas Petersson. Vending-bench: A benchmark for long-term coherence of autonomous agents. *arXiv preprint arXiv:2502.15840*, 2025.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, 2020.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, et al. MLE-bench: Evaluating machine learning agents on machine learning engineering. *arXiv preprint arXiv:2410.07095*, 2024.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yura Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Qi Chen, Bowen Zhang, Gang Wang, and Qi Wu. Weak-eval-strong: Evaluating and eliciting lateral thinking of LLMs with situation puzzles. In *Advances in Neural Information Processing Systems*, 2024.

Zerui Cheng, Jiashuo Liu, Chunjie Wu, Jianzhu Yao, Pramod Viswanath, Ge Zhang, and Wenhao Huang. VeRA: Verified reasoning data augmentation at scale. *arXiv preprint arXiv:2602.13217*, 2026.

François Chollet. On the measure of intelligence. *arXiv preprint arXiv:1911.01547*, 2019.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2Web: Towards a generalist agent for the web. In *International Conference on Learning Representations*, 2024.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, 2017.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. PAL: Program-aided language models. In *International Conference on Machine Learning*, 2023.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021a.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Advances in Neural Information Processing Systems*, 2021b.

Shulin Huang, Shirong Ma, Yinghui Li, Mengzuo Huang, Wuhe Zou, Weidong Zhang, and Haitao Zheng. LatEval: An interactive LLMs evaluation benchmark with incomplete information from lateral thinking puzzles. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation*, pp. 10144–10160, 2024.

Yifan Jiang, Filip Ilievski, Kaixin Ma, and Zhivar Sourati. BRAINTEASER: Lateral thinking puzzles for large language models. *arXiv preprint arXiv:2310.05057*, 2023.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world github issues? In *International Conference on Learning Representations*, 2024.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, 2022.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models. In *Advances in Neural Information Processing Systems*, 2022.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Remi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. Competition-level code generation with AlphaCode. *Science*, 378(6624):1092–1097, 2022.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. Holistic evaluation of language models. *Transactions on Machine Learning Research*, 2023.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

Stephanie Lin, Jacob Hilton, and Owain Evans. TruthfulQA: Measuring how models mimic human falsehoods. In *Annual Meeting of the Association for Computational Linguistics*, 2022.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, et al. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*, 2023.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, 2017.

Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of Learning and Motivation*, volume 24, pp. 109–165. Academic Press, 1989.

Gregoire Mialon, Clementine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: A benchmark for general ai assistants. *arXiv preprint arXiv:2311.12983*, 2023.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work? In *Empirical Methods in Natural Language Processing*, 2022.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. WebGPT: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *ACM Symposium on User Interface Software and Technology*, 2023.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive APIs. *arXiv preprint arXiv:2305.15334*, 2023.

Project Euler. About project euler. `https://projecteuler.net/about`, 2026. Accessed 2026-05-29.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *International Conference on Learning Representations*, 2024.

Roger Ratcliff. Connectionist models of recognition memory: Constraints imposed by learning and forgetting functions. *Psychological Review*, 97(2):285–308, 1990.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. iCaRL: Incremental classifier and representation learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, 2023.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *arXiv preprint arXiv:2303.11366*, 2023.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Cote, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*, 2021.

Paul Sloane. *Lateral Thinking Puzzlers*. Sterling Pub. Co., New York, 1991. ISBN 0-8069-8226-8.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, 2017.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*, 2023.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *International Conference on Machine Learning*, 2020.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, et al. Challenging BIG-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL*, 2023.

Sebastian Thrun. Lifelong learning algorithms. In *Learning to Learn*, pp. 181–209. Kluwer Academic Publishers, 1998.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, and Daan Wierstra. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, 2016.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. SuperGLUE: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems*, 2019a.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*, 2019b.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, 2022.

Wikipedia contributors. Situation puzzle. `https://en.wikipedia.org/wiki/Situation_puzzle`, 2026. Accessed 2026-05-29.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, et al. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *Advances in Neural Information Processing Systems*, 2024.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems*, 2022.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, 2023a.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023b.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. STaR: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems*, 2022.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Annual Meeting of the Association for Computational Linguistics*, 2019.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, 2017.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, et al. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023.

A Extended Related Work

AhaBench connects continual learning, test-time adaptation, and in-context learning, but changes the object being measured. Classical continual learning studies adaptation across tasks without forgetting (Ratcliff, 1990; McCloskey & Cohen, 1989; Thrun, 1998; Kirkpatrick et al., 2017; Rusu et al., 2016; Zenke et al., 2017; Lopez-Paz & Ranzato, 2017; Rebuffi et al., 2017; Parisi et al., 2019); our setting instead fixes the parameters and studies deployment-time history. Meta-learning and test-time adaptation ask how systems use new experience at evaluation time (Vinyals et al., 2016; Snell et al., 2017; Finn et al., 2017; Sun et al., 2020); AhaBench differs because the experience is often conversational, tool-mediated, or environmental rather than a batch of labeled examples. In-context learning and reasoning prompts show that demonstrations can improve immediate answers (Brown et al., 2020; Wei et al., 2022; Kojima et al., 2022; Yao et al., 2023a; Zelikman et al., 2022), but demonstrations need not provide semantic learning (Min et al., 2022); AhaBench is designed around this ambiguity by changing or removing the support signal at evaluation time.

General-purpose and reasoning benchmarks establish the reporting norms that AhaBench adopts, but their unit is usually an isolated item. MMLU, BIG-bench, and HELM argue for broad coverage, diverse difficult tasks, and transparent multi-metric evaluation (Hendrycks et al., 2021a; Srivastava et al., 2023; Liang et al., 2023). ARC, HellaSwag, TruthfulQA, and the Abstraction and Reasoning Corpus expose complementary gaps in commonsense, truthfulness, and abstraction (Clark et al., 2018; Zellers et al., 2019; Lin et al., 2022; Chollet, 2019). Mathematical and process-supervised benchmarks such as GSM8K, MATH, Minerva, BBH, and process-verifier work measure reasoning quality and solution checking under structured evaluations (Cobbe et al., 2021; Hendrycks et al., 2021b; Lewkowycz et al., 2022; Suzgun et al., 2023; Lightman et al., 2023). AhaBench follows the same diagnostic tradition, but asks whether earlier interaction changes a later answer or action when direct support is reduced, withheld, or delayed.

Aha-Euler is closest to work on mathematical programming, code generation, executable specifications, and program-aided reasoning. Codex-style code benchmarks, program synthesis studies, AlphaCode, and program-aided language models show that executable reasoning can dramatically change what LLMs can solve (Chen et al., 2021; Austin et al., 2021; Li et al., 2022; Gao et al., 2023). VeRA shows how static mathematical reasoning data can be turned into executable specifications and scaled with automatic verification (Cheng et al., 2026). Project Euler is a natural source domain because its problems are explicitly mathematical/computer-programming challenges (Project Euler, 2026). Aha-Euler builds on this executable-specification lineage while changing the measured object: it turns the source genre into newly generated taught/held-out families with exact validators and graph-composed subcomputations, so the measured quantity is transfer from a taught instance rather than recall of a famous solution.

Aha-Puzzle connects to lateral-thinking and situation-puzzle evaluation, but keeps the interactive exploration policy central. Situation puzzles are yes/no hidden-story games popularized in lateral-thinking puzzle collections (Sloane, 1991; Wikipedia contributors, 2026). Recent LLM evaluations such as BRAINTEASER, LatEval, and SPLAT show that lateral thinking and incomplete-information interaction remain difficult for language models (Jiang et al., 2023; Huang et al., 2024; Chen et al., 2024). Puzzle differs in emphasis: it uses sourced, human-verified situation puzzles as a compact probe of whether prior solved episodes improve the no-hint endpoint, rather than treating the visible interaction itself as the final score.

Agent benchmarks evaluate tool use, web tasks, software repair, embodied planning, desktop control, user interaction, ML engineering, and general-assistant behavior (Nakano et al., 2021; Yao et al., 2022; Shridhar et al., 2021; Yao et al., 2023b; Shinn et al., 2023; Wang et al., 2023; Liu et al., 2023; Zhou et al., 2023; Deng et al., 2024; Jimenez et al., 2024; Yao et al., 2024; Xie et al., 2024; Chan et al., 2024; Mialon et al., 2023). Tool-use training and tool-calling benchmarks further stress that acting through tools is a distinct capability from answering in text (Schick et al., 2023; Qin et al., 2024; Patil et al., 2023). Generative-agent simulations show another route to studying persistent behavior in social or environmental loops (Park et al., 2023). These benchmarks motivate our data design: keep trajectories, report failure modes, and separate final scalar reward from the process that produced it.

Aha-Vending is closest to the Vending-Bench line (Backlund & Petersson, 2025; Andon Labs, 2025): all use a simulated vending business to expose long-horizon coherence failures. Aha-Vending contributes an

open-source incident-setting simulator with scripted policy anchors, trajectory and failure-mode reporting, reproduction scripts, and placement inside a three-family suite where operational feedback is compared with explicit Puzzle feedback and Aha-Euler teaching. The broader contribution is narrower but sharper than general agent benchmarking: make the *feedback regime* and post-experience evaluation the organizing variables across task families.

B Additional Benchmark Details

B.1 Benchmark Design Checks

Table 8: Design checks built into the released benchmark.

Benchmark	Released data	What the design separates
Aha-Puzzle	Decreasing trace budget; final no-hint challenge; endpoint balancing; language-level aggregation; coverage accounting	Visible trace use versus no-hint exploration transfer
Aha-Euler	Paired taught/held-out instances; full versus answer-only teaching; deterministic generators; exact validators; direct-recall probe; trace audit	Procedure following versus answer-only computational transfer
Aha-Vending	Seeded tracks; high-balance ablation; weather and shock tracks; termination reasons; curves; scripted single-agent baselines	Final profit versus operational robustness and temporal failure modes

B.2 Puzzle Aggregation and Bootstrap

The released Puzzle aggregate averages first within language and then across languages, so each language contributes equally to the cross-lingual score. The endpoint bootstrap resamples valid model–language–repeat–puzzle units with paired final-challenge minus baseline differences. The puzzle-cluster bootstrap resamples puzzle identities and averages all valid endpoint units for sampled puzzles. The two intervals expose endpoint-level and puzzle-identity-level variation; the cluster interval is the main-text summary.

Table 9: Aha-Puzzle eight-model aggregate. Scores are language-level means aggregated across available languages; the final no-hint column is the last curriculum slot.

Model	Endpoint languages	Baseline	Final no-hint	Supported avg.	No-hint gain
Gemini 3.1 Pro Preview	4	41.24	44.29	75.07	+3.05
Qwen 3.6 Plus	4	24.12	36.27	69.59	+12.15
Doubao Seed 2.0 Pro	4	31.67	35.32	69.72	+3.66
Claude Opus 4.6	4	27.52	29.57	67.01	+2.05
GPT-5.4	4	28.93	28.67	66.86	−0.26
Hy3 Preview	4	13.70	12.20	62.20	−1.50
DeepSeek v3.2 Thinking	4	9.40	15.33	58.70	+5.93
Kimi K2 Thinking	4	16.17	14.68	64.69	−1.49

B.3 Puzzle Rubric Example

Each puzzle has required explanation-chain elements and optional details. For example, *Waterweed* requires that the model recover a prior failed rescue, the mistaken object, the fisherman's statement that the river has no waterweed, and the realization that the object was the drowning person's hair. The rubric executor marks each required and optional element, accepts paraphrases and language differences, and marks solved only when all required elements are substantially covered. The final score decomposes into correctness and efficiency: correctness contributes up to 70 points in proportion to covered rubric elements, while efficiency contributes up to 30 points, starting at full credit within five turns and decreasing by two points for each additional turn. This rubric structure distinguishes a complete hidden-state explanation from a plausible twist that misses required story links.

Table 10: Aha-Puzzle scoring and validation checks. The rubric executor applies explicit rules; human checks validate the construction and sampled grading outputs.

Object	Quantity	Check
Puzzle identities	7	Ground-truth bank present in every language curriculum spec
Languages	4	Chinese, English, Russian, Japanese coverage in detailed table
Human construction checks	all identities	Hidden solutions, allowable host answers, rubrics, and best-trace summaries checked during benchmark construction
Rubric executor	GPT-5.4	Executes explicit required/optional-element rules on the final explanation
Human grading validation	100 sampled grades; 3 annotators	100% agreement with rubric-executor grades on the sampled trajectories
Per-model run scale	$7 + 7 \times 7 = 56$ episodes per language	Baseline plus seven curriculum slots over seven puzzle identities
Score formula	70 correctness + 30 efficiency	Correctness checks rubric coverage; efficiency rewards shorter successful interactions within the 20-question cap

B.4 Detailed Bootstrap Procedure

For the endpoint bootstrap, an endpoint unit is a scored model–language–repeat–puzzle trajectory with both baseline and final no-hint scores. We sample endpoint trajectories with replacement within a model, compute the paired final-challenge minus baseline difference for every sampled trajectory, and average. For the puzzle-cluster bootstrap, we sample puzzle identities with replacement and average all scored endpoint trajectories belonging to sampled puzzles. This treats puzzle identity, rather than translation or repeat, as the effective independent unit.

Table 11: Aha-Puzzle language-level aggregate across selected models.

Language	Baseline	Final no-hint	Supported avg.	No-hint gain
Chinese	31.58 ± 12.51	30.79 ± 13.09	65.05 ± 5.23	$+0.36 \pm 8.38$
English	20.13 ± 10.60	25.26 ± 11.61	66.41 ± 5.36	$+4.79 \pm 5.68$
Russian	21.80 ± 8.90	22.14 ± 11.82	71.05 ± 4.49	$+1.15 \pm 6.71$
Japanese	23.54 ± 8.39	27.57 ± 11.88	64.35 ± 5.09	$+4.03 \pm 6.20$

The language breakdown is a useful diagnostic rather than the main score. Chinese has the highest direct baseline score (31.58), consistent with the fact that the source puzzles are Chinese, but the final no-hint gains are larger in English and Japanese, and the highest supported average is Russian. The main aggregation therefore keeps languages balanced and treats source-language familiarity as a secondary analysis rather than the explanation of the transfer pattern.

B.5 Aha-Euler Trace Audit

Task identifiers such as `DAG1_644` denote single-step generated Aha-Euler tasks; the numeric suffix is the released task identifier used to link prompts, traces, and validators.

B.6 Aha-Euler Construction and Direct-Recall Probe

Each Aha-Euler task exposes two problem instances of the same family. Q_{teach} is decomposed into node-level subquestions, and Q_{test} is reserved for post-teaching evaluation. The single-step subset has one node; the deepest public composition tasks require composing five taught substeps. Gold answers are generated by deterministic task solutions and validators. The original-versus-hardened probe in Table 15 separates single-step baseline attempts using task metadata in the public migration bundle and gives a targeted measure of direct-recall sensitivity.

Table 12: Representative partial-teaching trace audit covering successful and failed trajectories.

Model	Task	Correct	New code	Repair	Note
GPT-5.4	`DAG1_644`	Yes	Yes	Yes	Rebuilds a dynamic program and reparameterizes for Q_{test}.
GPT-5.4	`DAG1_962`	Yes	Yes	Yes	Uses iterative tool debugging before exact output.
Gemini 3.1 Pro	`DAG1_880`	Yes	Yes	Yes	Repairs a tool-detected issue.
Qwen 3.6 Plus	`DAG1_696`	No	Yes	No	Matrix-polynomial code; wrong Q_{teach}, wrong Q_{test}.
DeepSeek v3.2 Thinking	`DAG1_923H`	No	Yes	Yes	Repairs syntax/runtime issues but not the algorithm.
Kimi K2 Thinking	`DAG1_970`	No	Yes	Yes	Attempts scaled computation but returns wrong extrapolation.

Table 13: Aha-Euler post-teaching pass rates for single-step tasks and a compositional roll-up over two- to five-node tasks. Values are percentages.

Model	Single-step	2–5 node roll-up
GPT-5.4	97.5	92.3
DeepSeek v3.2 Thinking	91.8	81.5
Doubao Seed 2.0 Pro	91.4	94.4
Claude Opus 4.6	100.0	100.0
Gemini 3.1 Pro Preview	84.2	89.5
Qwen 3.6 Plus	95.9	79.3
Kimi K2 Thinking	100.0	76.9
Hy3 Preview	78.6	35.3

Table 14: Aha-Euler aligned-task view on single-step tasks. Values are pass rates in percent for tasks scored in all three conditions, complementing the phase-level estimator in Table 5.

Model	Baseline	Full	Partial
GPT-5.4	56.5	95.7	73.9
Gemini 3.1 Pro Preview	37.5	83.3	62.5
Claude Opus 4.6	39.9	100.0	64.9
Doubao Seed 2.0 Pro	15.0	85.0	25.0
Qwen 3.6 Plus	0.0	97.1	8.6
Kimi K2 Thinking	0.0	100.0	0.0
DeepSeek v3.2 Thinking	0.0	94.6	0.0
Hy3 Preview	0.0	100.0	0.0

B.7 Aha-Vending Scripted Baselines

The profiles in Table 16 are built-in rule policies executed through the simulator's `scripted` backend. The labels are profile names preserved from the simulator configuration. They use the same observation and action interfaces as model agents. `naive_fixed` restocks a fixed product set and serves as a simple floor. `human` is a conservative hand-coded profile. `claude35_sonnet` is a stronger hand-coded profile calibrated from prior development traces. Watchdog termination after "no successful order for 20 days" is part of the same simulator termination logic applied to model runs; mean days in the main table therefore matters for interpreting profit.

C Additional Aha-Vending Results

D Full Protocol Notes

D.1 Puzzle Interaction Protocol

The Puzzle setting separates three roles. The tested model is the *tester*: it chooses questions and decides when to stop. The host environment answers from the hidden solution and is instructed to provide minimal yes/no or short-answer information. The judge evaluates only the final explanation against a rubric. This

Table 15: Aha-Euler single-step baseline accuracy (%) on original-inspired tasks versus hardened variants after infrastructure-error exclusion.

Model	Original-inspired	Hardened	Original advantage
Qwen 3.6 Plus	0.0	6.2	-6.2
GPT-5.4	53.3	57.1	-3.8
Doubao Seed 2.0 Pro	22.2	0.0	+22.2
Kimi K2 Thinking	0.0	0.0	0.0
DeepSeek v3.2 Thinking	0.0	0.0	0.0
Gemini 3.1 Pro Preview	35.0	15.8	+19.2
Micro-average	15.6	11.7	+3.9

Table 16: Aha-Vending scripted single-agent baselines run with the built-in `scripted` backend. Mean days reports simulated days completed before horizon completion or watchdog termination.

Profile	Setting	Days	Profit	Units	Termination
`naive_fixed`	nominal	365.0	-256.80	502	completed horizon
`naive_fixed`	incident	50.0	-86.82	36	no successful order for 20 days
`human`	nominal	33.8	+17.50	74	no successful order for 20 days
`human`	incident	34.0	-25.86	50	no successful order for 20 days
`claude35_sonnet`	nominal	101.0	+1364.94	1198	no successful order for 20 days
`claude35_sonnet`	incident	73.2	+468.05	548	no successful order for 20 days

Table 17: Spearman rank correlations across Aha-Vending single-agent diagnostic settings over overlapping public-summary models.

Setting	Nominal	High-bal.	Weather	Incident
Nominal	1.00	0.78	0.94	0.85
High-bal.	0.78	1.00	0.87	0.60
Weather	0.94	0.87	1.00	0.84
Incident	0.85	0.60	0.84	1.00

separation keeps the measurement focused on exploration policy: the environment answers the questions asked, and the tested model earns credit by recovering the hidden explanation rather than by recognizing a familiar title.

Each puzzle episode proceeds as follows. The model receives a short story and may ask up to 20 questions. At any point it may stop and provide a final explanation. The host answers minimally and does not volunteer unasked hidden links. The final explanation is scored for required elements, optional details, and efficiency. After the episode, the correct answer is revealed and the model is asked to reflect on an optimal questioning path. The next curriculum slot may include a trace prefix for a different puzzle, but the final no-hint challenge includes zero direct trace prefix for the evaluated puzzle.

The curriculum is intentionally not a static few-shot prompt. Earlier slots provide heavy trace support; later slots progressively remove it. The agent therefore sees examples of useful question sequences, but the final no-hint evaluation requires using the learned exploration strategy without direct support. This design distinguishes at least three outcomes: cold hidden-state competence, trace-supported competence, and no-hint transfer after experience.

D.2 Aha-Euler Teaching Protocol

Aha-Euler treats each problem family as a generator-backed validation task. The same underlying solution can be instantiated at multiple inputs, making it possible to teach Q_{teach} and test Q_{test} without reusing the same answer. In the single-step subset, Q_{teach} is one taught subproblem. In the graph-composition sweep,

Table 18: Shared-market arena order from the vending summary. All models lose money under the coupled peer-demand condition, showing a harder operational regime than the single-agent settings.

Arena order	Model	Net profit	Units sold
1	GLM 5.1	-135	701
2	Gemini 3.1 Pro	-174	130
3	Hy3 Preview	-251	339
4	GPT-5.4	-367	330
5	Qwen 3.6 Plus	-399	190
6	DeepSeek v3.2 Thinking	-411	67
7	Claude Opus 4.6	-424	0
8	Doubao Seed 2.0 Pro	-452	22

Q_{teach} is decomposed into a small graph of dependent subquestions. This creates a controlled bridge between single-step algorithm transfer and compositional transfer.

In full teaching, after each Q_{teach} node the model receives the answer, an explanation, and reference code. This condition is deliberately generous: it measures whether the agent can use a worked solution and adapt it to Q_{test}. In partial teaching, the model receives only the Q_{teach} answer. It still sees the Q_{teach} question, so the signal is better described as question-plus-answer teaching rather than answer alone. The partial condition is the sharper probe because there is no exposed procedure to port. Success requires deriving, recalling, or reconstructing a method that works for the new Q_{test} input.

All attempts can use Python because the evaluated ability is executable mathematical reasoning. The benchmark measures whether the agent can infer the right computation, implement it, debug it, and return the exact normalized answer. Exact-match scoring is strict and auditable for generated numerical tasks.

D.3 Aha-Vending Simulator Protocol

Aha-Vending exposes the agent to a stateful simulated business. The agent receives observations through tools and acts by sending supplier messages, placing orders, stocking machine slots, changing prices, collecting cash, searching, reading email, or waiting for the next day. The environment tracks inventory, slot capacities, product costs, customer demand, weather, supplier delays, cash flow, operating fees, incidents, and bankruptcy. The result is not a labeled solution but a trajectory of consequences.

The simulator is useful because mistakes compound. If the agent misunderstands slot sizes, it may buy inventory that cannot be stocked. If it waits too often, inventory decays into stockouts and lost demand. If it ignores cash collection, procurement becomes impossible. If it treats every supplier message as reliable, phishing and delays can dominate the year. These failures are not visible in a one-shot task. They are visible as curves, termination reasons, and repeated action patterns.

E Metric Definitions and Reporting

For the reported scorecard, let B_m^{Pu}, B_m^{Eu}, and B_m^{Ve} be the Puzzle direct-baseline score, Aha-Euler cold pass rate, and normalized early-window Aha-Vending profit-rate score for model m. Let P_m^{Pu}, P_m^{Eu}, and P_m^{Ve} be the corresponding Post-Experience scores: final no-hint Puzzle score, answer-only Aha-Euler pass rate, and normalized Aha-Vending incident-setting profit. Aha-Vending Post-Experience Score is normalized as:

$$P_m^{\text{Ve}} = 100\,\text{clip}_{[0,1]}\left(\frac{\Pi_m - \Pi_{\min}}{\Pi^* - \Pi_{\min}}\right),$$

where Π_m is mean net profit in dollars under the incident setting, $\Pi_{\min} = -500$ is the bankruptcy floor, and Π^* is the best reported mean profit in that setting. The aggregate scorecard is

$$B_m = \frac{B_m^{\text{Pu}} + B_m^{\text{Eu}} + B_m^{\text{Ve}}}{3}, \qquad P_m = \frac{P_m^{\text{Pu}} + P_m^{\text{Eu}} + P_m^{\text{Ve}}}{3}.$$

Learning Lift is the difference between the two aggregate scores:

$$\Delta_m = P_m - B_m.$$

Thus $B+\Delta = P$ up to table rounding. Support-gain retention in Puzzle and Aha-Euler, and late-minus-early Aha-Vending profit efficiency, remain diagnostic quantities for explaining where the lift came from.

Table 19: Representative support-retention and temporal-efficiency diagnostics. For Aha-Puzzle and Aha-Euler, higher support-gain retention, $(S_{\text{post}}-S_{\text{base}})/(S_{\text{sup}}-S_{\text{base}})$, means more visible-support gain remains after support is reduced. For Aha-Vending, Δ is late-minus-early profit per day.

Setting	Model	Initial	Supported	Post-exp.	Retention/trend
Aha-Puzzle score	Qwen	24.12	69.59	36.27	0.27
Aha-Puzzle score	Kimi	16.17	64.69	14.68	-0.03
Aha-Euler accuracy	Gemini	24.4	84.2	59.3	0.58
Aha-Euler accuracy	Qwen	0.0	95.9	8.1	0.08
Aha-Vending \$/day	Doubao	-1.95	–	+2.45	$\Delta = +4.40$
Aha-Vending \$/day	Kimi	+6.74	–	+5.21	$\Delta = -1.53$

Table 20: Metric definitions used across the suite.

Metric	Definition	Interpretation
Initial Score B	Mean of normalized starting scores: Puzzle direct baseline, Aha-Euler cold pass rate, and normalized early-window Aha-Vending earning rate	Starting competence before the relevant curriculum, teaching, or later-horizon experience is used
Post-Experience Score P	Mean of normalized later scores: final no-hint Puzzle, answer-only Aha-Euler, and normalized Aha-Vending incident-setting profit	Aggregate benchmark result; ranks measured performance after experience
Learning Lift Δ	$P - B$ on the aggregate scorecard scale	Aggregate score change from Initial to Post-Experience; reported beside P rather than used to override later competence
Experience-use gain	$S_{\text{post}} - S_{\text{init}}$, where the post-experience evaluation reduces, changes, or delays direct support	Main transfer quantity; report with the support condition that was removed
Supported average	Average score over trace-supported Puzzle curriculum slots	Measures performance while trace support remains visible
No-hint gain	Final no-hint Puzzle score minus baseline	Endpoint and cluster bootstrap over seven long-interaction puzzle identities
Support-reliance gap	$S_{\text{sup}} - S_{\text{post}}$, where S_{sup} is performance while trace support remains visible	Smaller is better when the goal is durable exploration rather than trace use
Support-gain retention	$(S_{\text{post}} - S_{\text{init}})/(S_{\text{sup}} - S_{\text{init}})$ for Puzzle and Aha-Euler support conditions	Fraction of visible-support gain that survives after support is reduced
Full-teaching accuracy	Q_{test} exact-match accuracy after answer, explanation, and code for Q_{teach}	Measures use of a worked procedure
Partial-teaching accuracy	Q_{test} exact-match accuracy after Q_{teach} question and answer only	Measures method reconstruction from sparse teaching
Procedure-reliance gap	$S_{\text{full}} - S_{\text{partial}}$	Smaller is better when the goal is algorithm reconstruction from sparse teaching
Retention ratio	$(S_{\text{partial}} - S_{\text{base}})/(S_{\text{full}} - S_{\text{base}})$	Unstable near floors or small full-teaching gain
Net profit	Final net worth minus starting balance	Can hide bankruptcy floors and late passivity
Curve shape	Profit or net worth over time	Required for temporal failure diagnosis
Late–early profit efficiency	$r_{\text{late}} - r_{\text{early}}$, with r measured per day or per message	Positive values indicate that operational experience improves later earning rate
Temporal failure surface	Joint report of final profit, survival, drawdown or bankruptcy, seed bifurcation, and termination mode	Keeps final score tied to the path that produced it

F Long-Horizon Cost Model

Task count is a poor proxy for long-horizon evaluation cost. In a one-shot benchmark, doubling the number of items roughly doubles the number of prompts. In a long-horizon benchmark, later turns often resend the entire accumulated context, call tools or environments, and produce outputs that must be audited. If a curriculum has T stages and each stage contains H_t turns with context length C_t, token cost scales with $\sum_t H_t C_t$, not simply T. Adding a puzzle, an Aha-Euler composition node, or a simulator seed can multiply cost because it also expands judging, tool execution, retry handling, and failure analysis.

Accordingly, AhaBench treats an item as a trajectory, not a static row. The high-value evidence is the measurement surface per item: support removal, taught/held-out separation, scripted anchors, trajectory curves, and audit samples.

G Benchmark Design Rationale

G.1 Why These Three Families

The suite is intentionally heterogeneous but not arbitrary. Puzzle, Aha-Euler, and Aha-Vending each isolate a primitive interface through which an agent can improve from experience.

Epistemic exploration. Puzzle asks whether the agent learns how to interrogate an unknown state. The evaluated behavior is not merely a correct final answer; it is the policy that selects informative questions. This matters because many deployed agents fail before final reasoning begins: they search the wrong space, overcommit to early hypotheses, ignore negative evidence, or stop with an incomplete explanation chain. Situation puzzles are compact but demanding probes of that failure mode because the surface story is short while the hidden state must be uncovered through interaction.

Algorithmic recomposition. Aha-Euler asks whether a model can infer a reusable computation from a related example. The design deliberately separates code-supported adaptation from answer-only transfer. In real agent workflows, a model often receives partial logs, examples, or answers without a full reference implementation. A benchmark that only tests whether the model can adapt exposed code would overstate this ability. The taught/held-out pairing makes the teaching signal concrete while preventing direct answer reuse.

Operational feedback control. Aha-Vending asks whether environmental consequences become durable policy. A good action must be repeated, revised, and coordinated over hundreds of simulated days. This captures a different kind of learning: maintaining discovered constraints, using delayed rewards, adjusting under shocks, and avoiding repeated operational mistakes. The simulator turns vague ideas such as "agent memory" and "reflection" into measurable profit, survival, and trajectory shape.

G.2 Capability Surface With Two Leaderboards

A leaderboard alone would hide the central scientific point. One model can have high final no-hint Puzzle score but no transfer gain, another can have low cold performance but large answer-only Aha-Euler retention, and a third can be profitable early but collapse under incident-heavy operation. Ranking them only by a normalized average would imply a false equivalence among exploration, algorithmic transfer, and operational robustness. AhaBench therefore reports the capability surface together with Initial Score, Post-Experience Score, and Learning Lift. The disagreement between axes is the evidence that long-horizon continual learning is not one monolithic skill.

G.3 What Counts as Learning in This Suite

The benchmark keeps model weights fixed and measures operational learning: a later action distribution or final answer improves because of prior experience. The clearest evidence appears when the evaluation differs

from the support signal. A Puzzle gain on the final no-hint challenge is stronger than a high score while a trace prefix is visible. An Aha-Euler partial-teaching success is stronger than a full-teaching success with exposed code. A vending recovery after repeated slot-fit failures is stronger than early profit from a lucky initial order.

The ablations separate support-dependent behavior from reusable experience use. Copying a visible trace, re-running supplied code with a changed constant, relying on a memorized canonical Project Euler solution, or exploiting an environment shortcut are measurable behaviors, and the suite records which condition produced them.

H Failure Taxonomies

H.1 Puzzle Failure Modes

Table 21: Puzzle failure taxonomy used for interpreting hidden-state trajectories.

Failure mode	Observable behavior	Why it matters
Surface literalism	Treats the opening story as complete and literal	Fails to search hidden state
Narrow hypothesis lock-in	Asks many variants of one theory after negative answers	Poor exploration policy
Late identity reversal	Delays checking whether a person/object is not what it seems	Misses common situation-puzzle structure
Temporal confusion	Fails to distinguish current event from past cause	Produces plausible but incomplete explanation
Premature stopping	Stops before required explanation-chain elements are established	Inflates apparent efficiency at cost of correctness
Trace dependence	Performs well with prefixes but not on the final no-hint challenge	Uses support without transfer
Context distraction	Prior episodes reduce focus on the hidden state being evaluated	Long context can hurt policy selection

H.2 Aha-Euler Failure Modes

Table 22: Aha-Euler failure taxonomy. Exact-match failure can arise from distinct causes, so aggregate accuracy is read alongside trace audits.

Failure mode	Observable behavior	Interpretation
Algorithm selection error	Chooses wrong recurrence, search, number-theoretic identity, or dynamic program	Did not infer the latent method
Implementation bug	Correct idea but code has indexing, overflow, truncation, or boundary error	Method partially inferred, execution failed
Tool-debug failure	Runtime or syntax error persists across tool calls	Low self-repair loop
Arithmetic/computation error	Code runs but returns wrong numerical value due to precision or scale	Execution issue under computational pressure
Formatting error	Correct value embedded in invalid or extra output	Output-format failure distinct from reasoning failure
Template retrieval	Solves by recognizing canonical family without using teaching signal	Prior familiarity with the source genre
Code-porting only	Full-teaching success through direct adaptation of exposed code	Procedure following, not answer-only transfer

Table 23: Aha-Vending failure taxonomy. These failures are temporal and are often invisible in final net profit alone.

Failure mode	Observable behavior	Consequence
Slot-fit misconception	Buys products too large for available machine slots	Cash trapped in unusable inventory
Understocking	Waits or delays reorders despite demand	Stockouts and lost sales
Overstocking	Buys too much inventory before demand is known	Cash-flow pressure and bankruptcy risk
Price miscalibration	Sets prices too high or too low for demand/cost	Low sales or low margins
Supplier mismanagement	Sends incomplete orders or fails to follow up	Inventory starvation
Incident vulnerability	Falls for phishing or ignores repairs/refunds	incident-setting value loss
Message-budget passivity	Waits repeatedly to conserve messages	Late-horizon profit collapse
Arena miscoordination	Fails to model shared demand or peers	shared-market loss

H.3 Aha-Vending Failure Modes

I Aha-Vending State and Action Details

Table 24: Representative Aha-Vending state variables exposed through observations and tools.

State category	Examples	Learning challenge
Cash and balance	Starting balance, daily fees, collected cash, procurement costs	Avoid bankruptcy while investing in inventory
Inventory	Product counts in storage and machine slots	Maintain stock without overbuying
Slot layout	Slot sizes, product dimensions, duplicate placement	Infer feasible stocking policy
Demand	Product sales, weather effects, price sensitivity	Adjust prices and product mix from delayed signals
Suppliers	Search results, email responses, delivery delays, minimum quantities	Build reliable procurement routines
Incidents	Shocks, refunds, phishing, machine issues	Distinguish real operational tasks from adversarial messages
Message budget	Remaining turns/messages and watchdog behavior	Preserve enough action bandwidth for late horizon

Table 25: Representative Aha-Vending actions and common failure patterns.

Action	Purpose	Common failure
Search suppliers	Discover purchasable products and costs	Repeated broad search without conversion to orders
Send supplier email/order	Procure inventory	Missing SKU, wrong quantity, ignored supplier constraint
Inspect storage/machine	Check inventory, slot fit, and sales state	Failing to verify before buying or stocking
Stock machine	Move inventory into sellable slots	Product too large, wrong slot, inefficient duplication
Change prices	Tune margin and demand	Overpricing suppresses sales; underpricing loses margin
Collect cash	Convert machine cash to available funds	Delayed collection causes procurement failure
Wait	Advance time to observe demand and deliveries	Repeated waiting becomes passivity

J Formal Estimators

J.1 Puzzle Transfer Estimator

Let m index models, p puzzle identities, ℓ languages, and r repeats. Let $B_{m,p,\ell,r}$ be the baseline score and $T_{m,p,\ell,r}$ be the final no-hint score when both are valid. The endpoint-unit estimator is

$$\widehat{\Delta}_m^{\text{unit}} = \frac{1}{|\mathcal{I}_m|} \sum_{(p,\ell,r)\in\mathcal{I}_m} \left(T_{m,p,\ell,r} - B_{m,p,\ell,r}\right),$$

where $\mathcal{I}_m$ is the set of valid paired endpoint units. The endpoint bootstrap resamples elements of $\mathcal{I}_m$. The cluster estimator first averages within puzzle identity,

$$\bar{\Delta}_{m,p} = \frac{1}{|\mathcal{I}_{m,p}|} \sum_{(\ell,r)\in\mathcal{I}_{m,p}} \left(T_{m,p,\ell,r} - B_{m,p,\ell,r}\right),$$

and bootstraps over puzzle identities p. This gives a puzzle-identity-level interval in addition to the endpoint-unit interval.

J.2 Aha-Euler Teaching Estimator

For each model m and teaching condition $c \in \{\text{base}, \text{full}, \text{partial}\}$, let $Y_{m,i,c} \in \{0,1\}$ denote exact-match success on scored task i. The reported phase-level accuracy is

$$\widehat{S}_{m,c} = \frac{\sum_i \mathbf{1}\{i \in \mathcal{V}_{m,c}\} Y_{m,i,c}}{|\mathcal{V}_{m,c}|},$$

where $\mathcal{V}_{m,c}$ is the scored-task set for the condition. An aligned-task estimator uses

$$\mathcal{V}_m^{\cap} = \mathcal{V}_{m,\text{base}} \cap \mathcal{V}_{m,\text{full}} \cap \mathcal{V}_{m,\text{partial}},$$

as the shared scored-task set. The main tables use the phase-level estimator, and the aligned-task table gives the paired view. The primary Aha-Euler evidence is the direct partial-teaching rate and full–partial gap; paired intersections give a compact check of the same contrast, and the retention ratio is a normalized view of it.

J.3 Aha-Vending Curve Estimators

For a model m, seed s, and simulated day d, let $W_{m,s,d}$ be net worth and W_0 starting balance. Final net profit is

$$P_{m,s} = W_{m,s,D_s} - W_0,$$

where D_s is the final simulated day before horizon completion or termination. The reported single-agent mean is $\frac{1}{S}\sum_s P_{m,s}$. We interpret this scalar together with the curve $W_{m,s,d} - W_0$, termination reason, and survival day because identical final values can arise from very different paths: early bankruptcy, slow decline, late passivity, or recovery after initial mistakes.

K Trace Case Studies

K.1 Aha-Euler: Successful Answer-Only Reconstruction

In the audited GPT-5.4 partial-teaching trajectory on task `DAG1_644`, the model receives the Q_{teach} statement and answer but no reference implementation. It proposes a dynamic-programming computation, writes Python, receives tool feedback, repairs a syntax problem, and then changes the input parameters for Q_{test}. The trajectory shows the distinction targeted by the benchmark: behaviorally it is fresh executable reconstruction under sparse feedback, not direct code copying.

K.2 Aha-Euler: Substantive Failure Rather Than Refusal

The audited Qwen partial-teaching failure on task `DAG1_696` is informative for the opposite reason. The model writes nontrivial matrix-polynomial code and uses the calculator, but the Q_{teach} prediction is wrong. After receiving the correct Q_{teach} answer, it changes state representation and truncation choices for Q_{test}, then returns another wrong value. The failure is algorithmic and implementation-level under answer-only feedback, illustrating why exact validators and trace inspection are both useful.

K.3 Aha-Vending: Doubao Recovery

Doubao's nominal trajectory illustrates why final profit alone collapses important behavior. In several seeded runs, the model loses money during the first 30–60 days because it repeatedly buys or attempts to stock large products into small machine slots. This is an exploration failure with real cost. In the majority-recovery pattern, the model eventually infers enough of the slot layout to recover and finish positive; in the remaining runs it reaches bankruptcy. The curve and seed-level outcomes distinguish this pattern from a model that steadily makes small profits.

K.4 Aha-Vending: Kimi Passivity

Kimi exhibits a different failure mode. It can earn early profit, but later increases waiting behavior and reduces stocking activity. This looks like over-conservation of message budget or loss of operational momentum. The important point is that the agent has already discovered profitable actions; the failure is maintaining and repeating them over a long horizon. This is exactly the sort of behavior that a one-shot task cannot expose and a final scalar can underdescribe.

L Additional Arena Figures

M Extended Interpretation Guide

M.1 Interpreting Puzzle Gains

Puzzle gains measure whether prior hidden-state episodes changed the no-hint endpoint. A model with high baseline and modest gain may already possess strong priors for situation puzzles. A model with low baseline and positive gain may be more responsive to curriculum feedback. A model with high trace-supported average but low final-challenge score may be excellent at exploiting visible support but show low no-hint transfer. These distinctions are the reason the benchmark reports all three quantities.

The most important trajectory behavior in Puzzle is question selection. Good trajectories test structural hypotheses early: whether names are literal, whether the protagonist is who the story implies, whether the salient object is misidentified, whether the key event happened before the surface scene, and whether a negative answer rules out an entire family of theories. Less effective trajectories spend many turns on narrow variants of a single story. Aggregate tables emphasize outcomes while the rubrics and summaries preserve the explanation criteria needed to compare exploration policies.

M.2 Interpreting Aha-Euler Retention

The retention ratio is useful because it compresses the partial/full comparison into one number. When baseline is near zero and full teaching is near one, retention is close to partial-teaching accuracy; when full-teaching gain is smaller, the full–partial gap is easier to read directly. We therefore report retention as a normalized view and use partial-teaching rate plus full–partial gap as the main Aha-Euler evidence.

Answer-only success has several possible channels. Seeing a Q_{teach} problem and its answer can reveal the scale of the answer, constrain a latent formula, trigger recognition of a known problem family, or provide a check while the model derives a method. These routes are all relevant to deployed agents. The benchmark makes them measurable by removing exposed code.

M.3 Interpreting Aha-Vending Profit

Aha-Vending profit is one useful scalar; profit curves add the temporal signal. Curves show when a model learned, failed, recovered, or became passive. Termination modes distinguish bankruptcy from watchdog termination or horizon completion. Units sold distinguish high-margin low-volume strategies from broad stocking strategies. Seed-level outcomes reveal whether a mean is driven by robust policy or by a few recoveries. Diagnostic variants with relaxed starting balance expose how severe bad policies become when early bankruptcy does not truncate the run.

The shared-market arena is a stress test. It changes peers, shared demand, strategic messages, and market economics at the same time. The result is a hard deployment condition in which all participants lose money, revealing that the single-agent policies do not directly carry over to competitive shared-demand operation.

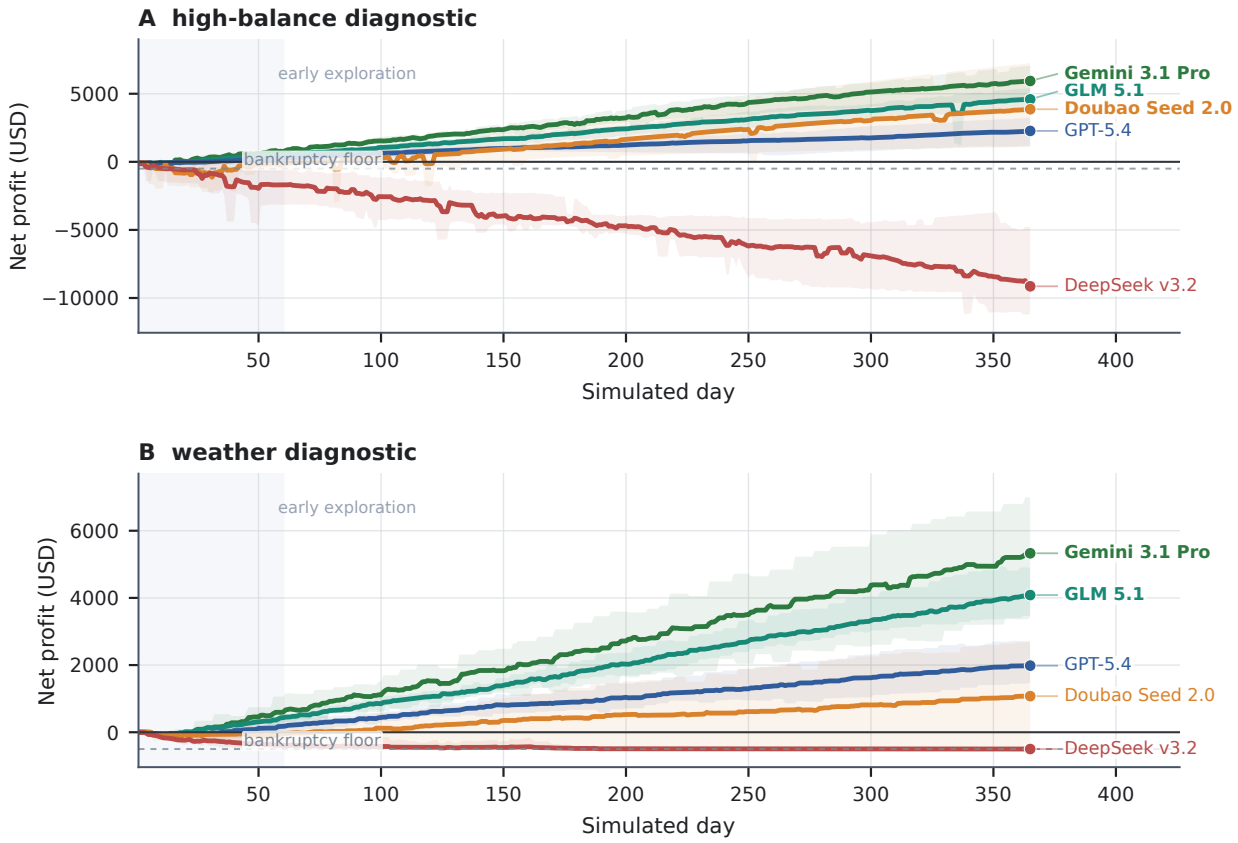

Figure 5: Additional Aha-Vending diagnostic curves. The high-balance setting removes the low-balance bankruptcy cliff, while the weather setting adds real-weather demand variation; the full-width panels keep the trajectory labels and seed bands readable.

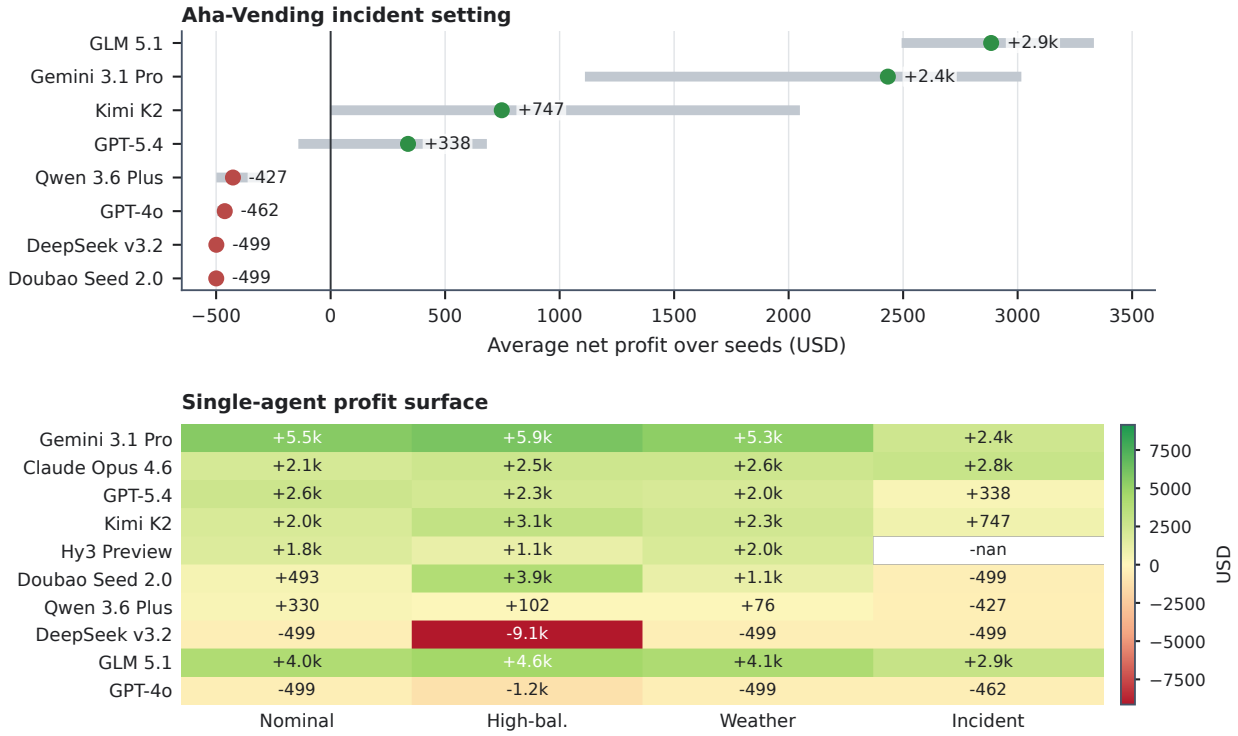

Figure 6: Additional Aha-Vending summary views. The incident-setting dot-and-range plot shows seed min–max ranges with mean dots, and the profit matrix summarizes how nominal survival, high-balance loss exposure, weather variation, and incidents induce different model rankings.

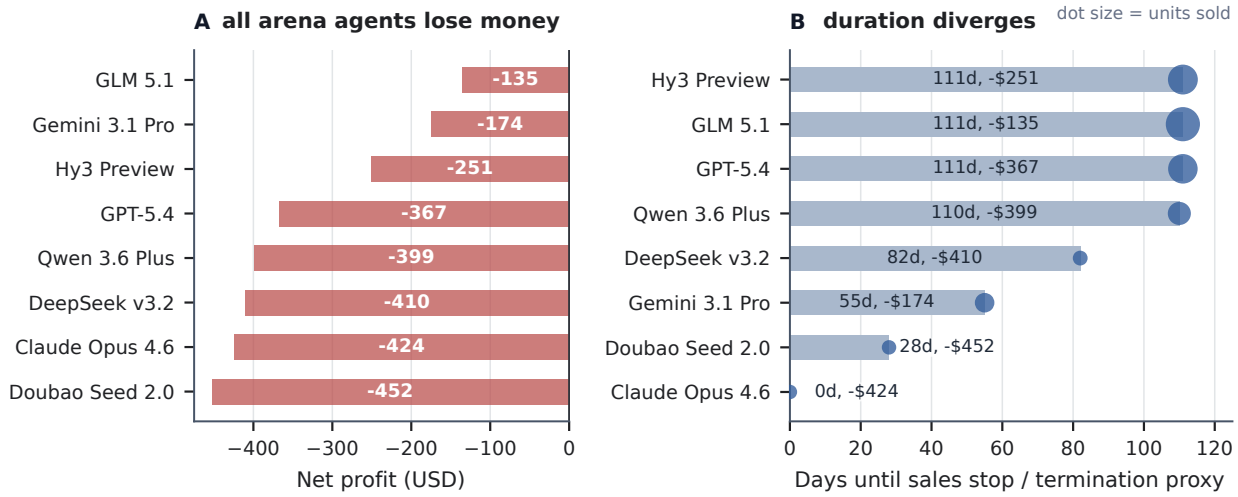

Figure 7: Shared-market arena views. The arena stresses policy under peer behavior, shared demand, strategic messages, and altered market economics.