EXPLORATIONBENCH: MEASURING AI SYSTEMS' EXPLORATION IN VERIFIABLE ALIEN WORLDS

Anonymous authors
Paper under double-blind review

ABSTRACT

Scientific discovery begins where known problems end. There, AI systems must engage in exploration: framing hypotheses, designing experiments, and iterating on the results. However, evaluating this ability is difficult: (1) how to verify whether a genuinely new hypothesis holds, and (2) how to determine whether a system has discovered it through exploration or merely recalled related knowledge from pre-training data. To this end, we introduce EXPLORATIONBENCH, which turns the wicked problem of evaluating scientific exploration into a concrete and tractable framework built on verifiable Alien Worlds: their rules are executable, so every answer can be checked exactly, and they conflict with familiar knowledge, so recall alone cannot solve the tasks. The benchmark contains two sandboxes, ALIENCODE (31 discovery targets, 70 tasks) and ALIENLOGIC (24 discovery targets, 70 tasks). Each sandbox provides a flawed manual, task-specific environmental feedback, and a dedicated tool-call schema. Systems use these resources to explore the sandbox, then solve held-out tasks. We evaluate 10 AI systems and find that the strongest systems can acquire and apply unfamiliar rules, while performance varies substantially across trajectories and continued exploration can stall or reverse earlier gains. EXPLORATIONBENCH represents a step towards AI systems that can acquire and apply genuinely new knowledge through exploration in unknown environments.

1 INTRODUCTION

Scientific research advances through exploration. Researchers begin with a tentative understanding, choose what evidence to collect, revise their beliefs in light of the outcomes, and apply what they learn to new problems. Current large language models (LLMs) perform strongly on static evaluations of knowledge, mathematics, and professional tasks (Hendrycks et al., 2021; Mathematical Association of America, 2026; Patwardhan et al., 2025), but these evaluations mainly test whether a model can retrieve and reason with knowledge it already has. Exploration asks for something different. A system must decide what evidence to collect, accumulate that evidence across its interaction history, distill new knowledge from the outcomes, and apply that knowledge to new problems. As AI systems enter scientific and engineering workflows (Wang et al., 2023b; Chan et al., 2025), measuring this ability separately from recall and one-shot reasoning becomes increasingly important.

Evaluating exploration ability is difficult because two requirements are in tension. The tasks must be new to the system, so that success cannot come from knowledge acquired during pre-training, yet their answers must be fully known to the evaluator, so that success can be verified (Chollet, 2019). Established domains such as mathematics, coding, and factual question answering meet the second requirement but not the first. A model may reproduce what it memorized, and contamination is hard to exclude for black-box systems (Oren et al., 2024). Genuinely novel outputs, such as a new mathematical result or scientific hypothesis, meet the first requirement but not the second. Verifying them may require expert proof checking, specialized experiments, or years of observation (Wang et al., 2023b), and without such verification an evaluator cannot tell a genuine discovery from a plausible but incorrect claim.

Existing benchmarks cover parts of this setting (section 2). Context-learning benchmarks place the new knowledge in the prompt (Dou et al., 2026b; Asawa et al., 2026; Agarwal et al., 2024), so the model reads the evidence rather than collecting it. Interactive discovery benchmarks let agents gather

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

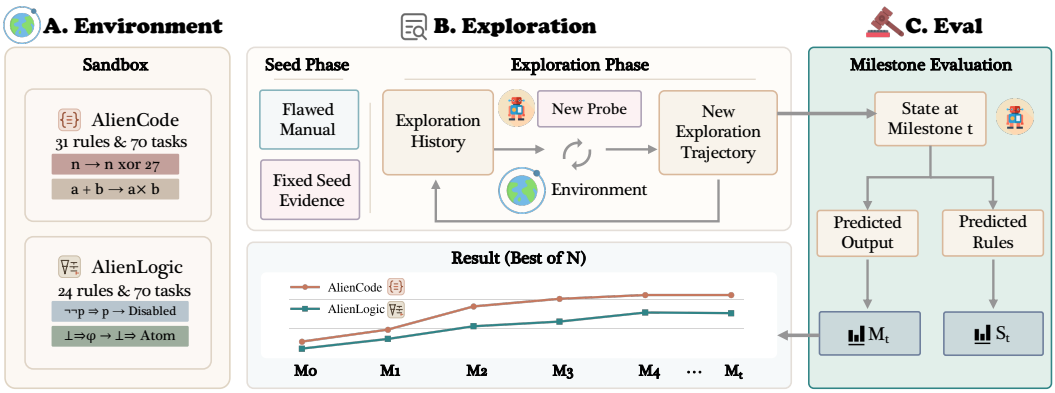


Figure 1: **ExplorationBench: what a system is given, what it may do about it, and how it is scored. (A)** Two executable sandboxes. AlienCode has 31 discovery targets and AlienLogic 24, so the manual each ships with is wrong in ways no amount of prior knowledge recovers. **(B)** Every system starts from the same flawed manual and the same fixed worked examples, then runs four rounds in which it chooses its own probes, runs them in the environment, and adds the results to its exploration history. **(C)** At each milestone M_t, the system is tested without tools. It answers the 70 held-out tasks and separately reports the rules it believes it has found.

evidence in fictional worlds or through chosen experiments (Tang et al., 2024; Jansen et al., 2024; Zheng et al., 2026), but they score the state of that world or the inferred law itself. None of them asks whether a system can collect the evidence it needs and then apply what it learned to unseen tasks after the interaction has ended.

In this work, we introduce ExplorationBench, a benchmark that evaluates exploration in verifiable alien worlds (fig. 1). Each world is a deterministic, executable environment whose hidden rules conflict with familiar semantics. AlienCode is a small programming language with 31 discovery targets, and AlienLogic is a natural-deduction system with 24 discovery targets. In AlienCode, for example, integer literals are silently XOR-ed with 27, so `EMIT(100)` prints `127`, and `PLUCK` counts positions from one although the manual says zero. A system starts from this flawed manual and a few worked examples, then explores for four rounds by submitting programs or proofs and reading the results. After each round it is tested without tool access. It states the rules it believes hold and solves 70 held-out tasks, which an interpreter or a proof-checker grades exactly.

ExplorationBench offers several properties that make exploration measurable. (1) **Resistant to recall.** The hidden rules contradict both the manual and pre-training priors, so recalled knowledge misleads rather than helps. Before exploring, no AlienCode trajectory exceeds 15.7%. (2) **Exactly verifiable.** Every answer is checked by executing it, so grading needs no LLM judge, as in test-based agent evaluation (Jimenez et al., 2024; Merrill et al., 2026). (3) **Resolved over the process.** Beyond the final score, the benchmark records the probes a system chooses and, at every milestone, the rules it reports and its held-out accuracy. This separates discovering a rule from using it. (4) **Controlled.** Matched conditions remove environment feedback, replace the system's probes with a fixed sequence or with its own best sequence, or supply the complete rule set (sections 3.1 and 3.3).

We evaluate ten frontier systems, each with three independent exploration trajectories per sandbox, and rank them by Best@3, the best final accuracy among the three. Exploration produces the knowledge the tasks require. After four rounds the best AlienCode trajectory reaches 89.0%, whereas the same number of model turns without environment feedback leaves systems at 0.5–11.0%. It also matters who designs the experiments. In AlienCode, handing a system back its own best probes without letting it choose them lowers accuracy for 9 of 10 systems, and randomized probes barely help. A system's exploration ability varies across task settings, and its rank in one sandbox barely predicts its rank in the other (Spearman 0.35). Discovering a rule and using it also come apart. Two off-by-one rules enter 51 of the 70 AlienCode tasks and the largest accuracy jumps coincide with their discovery, yet tasks whose required rules a system states correctly are still solved only 73.4% of the time. In AlienLogic, being told the rules (93–97%) beats every system's own

exploration. Finally, exploration is unreliable. Trajectories of one system under one budget end up to 73.3 points apart, far more than repeated answers to the same questions vary, and 6 of the 30 ALIENCODE trajectories end at least 3 points below an earlier milestone.

More findings and case studies are presented in section 4 and appendix H. Frontier systems can acquire unfamiliar rules through exploration, but they do so unreliably and do not always use what they find. EXPLORATIONBENCH provides a testbed for measuring how AI systems acquire and apply new knowledge, and for developing more reliable exploration methods.

2 RELATED WORK

Scientific discovery and rule induction. Interactive benchmarks ask agents to infer hidden or altered rules from evidence (Lake et al., 2017). MARS and DiscoveryWorld study investigation in fictional worlds (Tang et al., 2024; Jansen et al., 2024), and NewtonBench targets scientific law discovery (Zheng et al., 2026). EXPLORATIONBENCH measures the complete path from selecting probes, through reporting discoveries, to using them on unseen tasks after the interaction ends.

Context learning and test-time adaptation. CL-bench and CL-bench Life study learning from complex provided contexts (Dou et al., 2026b;a), EvaLearn studies experience across sequential problems (Dou et al., 2025), and SE-Bench and EdgeBench move adaptation into weights or long-horizon real environments (Yuan et al., 2026; Zhu et al., 2026). Related methods learn from feedback on a system's own attempts (Shinn et al., 2023; Madaan et al., 2023; Chen et al., 2024) or from stored trajectories (Zhao et al., 2024). EXPLORATIONBENCH holds parameters fixed and separates who selects the evidence from whether evidence arrives.

Verifiable interactive evaluation. SWE-bench grades repository patches with tests (Jimenez et al., 2024), WebArena scores website end states (Zhou et al., 2024), and τ-bench compares database states with annotated goals (Yao et al., 2025). EXPLORATIONBENCH shares this preference for executable outcomes, but its governing rules must be discovered before they are used. Appendix E extends this comparison, and table 2 summarizes it.

3 EXPLORATIONBENCH

3.1 THE TASK

Setup. An episode is a tuple $(\mathcal{E}, \mathcal{R}, \mathcal{M}, \mathcal{D}, \mathcal{T})$. $\mathcal{E}$ is a hidden, deterministic sandbox governed by a perturbed rule set $\mathcal{R}$. $\mathcal{M}$ is a public, flawed manual. It describes the *standard* semantics, which are false for the perturbed parts of $\mathcal{E}$. $\mathcal{D}$ is a fixed set of worked examples shared by every system. $\mathcal{T}$ is an unseen task set held out from exploration. The system sees $\mathcal{M}$ and $\mathcal{D}$ and may interact with $\mathcal{E}$, but never sees $\mathcal{R}$. A system that trusts the manual or its pre-training priors is therefore wrong on exactly the parts of $\mathcal{E}$ that matter.

The exploration policy. Let $h_t = (\mathcal{M}, \mathcal{D}, x_{<t}, f_{<t})$ denote the manual, the worked examples, and the interaction history so far. At step t, the system submits a tool input according to

$$x_t \sim \pi(\cdot \mid h_t), \qquad f_t = \mathcal{E}(x_t). \tag{1}$$

We call each executable tool input x_t a *probe*. In ALIENCODE a probe is a candidate program and its feedback f_t is the exact program output. In ALIENLOGIC it is a candidate proof and f_t is a compact verifier result. The system uses this feedback to revise a hypothesis state H_t about $\mathcal{R}$ within a common maximum budget $B_{\max}$. H_t and the evidence in h_t remain inside one context window, with no weight updates, external or persistent memory, or scalar reward.

Protocol. Algorithm 1 gives the protocol. Before M_0, every system receives the same worked examples, each a task with a reference program or proof and the output the environment computes for it, and makes no tool calls. M_0 therefore follows identical evidence, not merely an identical opportunity to collect it. The system then explores for four rounds. In each round it issues tool calls whose arguments are programs or proofs, and the environment executes them and returns

deterministic feedback. At each milestone the system is tested in a separate copy of the conversation with tools disabled. It reports the rules it believes hold, S_t, and answers the held-out tasks. The copy is then discarded, so testing never adds evidence to the exploration.

Exploration conditions. Five conditions vary whether environment feedback arrives and who directs it. *Autonomous exploration* (a) selects probes from the current history. *Hindsight exploration* (h) replays the probes of the same system's Best@3 trajectory, chosen after the fact. *Fixed-probe exploration* (f) issues one model-independent probe sequence, the same for every system. *Without-tool answering* (w) adds model turns without environment feedback, and *direct answering* (d) adds none. A sixth condition, *open-book answering* (o), supplies the complete rule set, either before exploration (O@M_0) or after autonomous exploration (A4+O). The three feedback conditions use the same tool-calling interface, so they differ only in who directs the probes. Conditions are compared by their M_4 endpoints, Best@n for autonomous exploration and the single trajectory each control runs per system. Set against M_4, O@M_0 and A4+O separate finding the rules from using them. Full definitions appear in appendix F.4.

3.2 The Alien Worlds

AlienCode and AlienLogic share the task and closed-book evaluation above. One is built on program semantics, the other on formal inference. Both are deterministic and executable, and both deliberately conflict with familiar priors. Some rules remain unchanged as red herrings, and every altered rule has at least one held-out task. An interpreter checks AlienCode programs and a proof-checker checks AlienLogic proofs, so scoring is deterministic, exact, and free of LLM judges. Construction and verifier details are in appendices F and F.6.

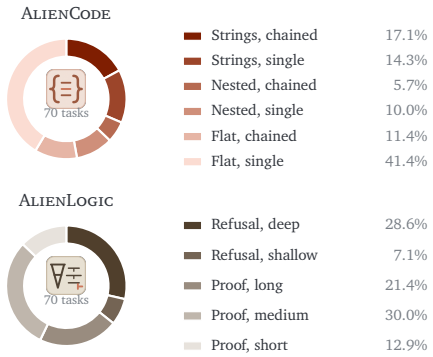


Figure 2: **Held-out task composition.** The 70 tasks of each sandbox by family.

AlienCode. This sandbox is a small calculation language whose familiar-looking operators follow hidden semantics. It contains 31 discovery targets and 70 held-out tasks, namely 37 base tasks, 15 nested composition tasks, 8 ceiling tasks, and 10 rule-coverage tasks. Together they exercise all 31 discovery targets. Before evaluation, each task is classified by representation (flat values, nested containers, or multi-character strings) and by compositional depth (single-step, single-algorithm, or multi-stage, fig. 2). Programs are graded by an interpreter on private evaluator inputs, so success requires rule-aware executable behavior rather than memorizing displayed examples. During exploration, the environment tool executes exactly the candidate program submitted and returns its output.

AlienLogic. This sandbox is a first-order natural-deduction system with 24 discovery targets, each a patched inference rule, and 70 held-out tasks. A proof-checker verifies every submitted proof, and designated unprovable tasks receive credit only when the system correctly declines to prove them. Certifier bounds and rule-side conditions are given in appendix F. During exploration, the environment tool checks candidate proofs and returns a compact verifier result.

3.3 Metrics

Held-out accuracy. At each milestone every held-out question is answered three times, each in a fresh tool-disabled copy of the conversation. $M_{r,t}$ is the mean verdict of trajectory r at milestone t over tasks and answers (eq. (3)), and a missing answer scores zero. The *answering noise* $\sigma_{r,t}$ is the standard deviation of the three single-pass scores. It measures how much the score moves when the same knowledge answers again.

Table 1: **Held-out accuracy under autonomous exploration.** M_0 and M_4 of each system's Best@3 trajectory (eq. (2)), the mean and the lowest M_4 over its three trajectories, and accuracy under open-book answering, with the complete rule set supplied before exploration (O@M_0) or after the Best@3 trajectory's exploration (A4+O). All values are percentages, each the mean of three answers per question; rows are ordered by Best@3 M_4.

		Best@3 trajectory		Three trajectories		Open-book		Diagnostic
System	Reasoning	M_0	M_4 ↑	Mean@3	Worst	O@M_0	A4+O	Aux.
ALIENCODE *program synthesis*								Aux. Rules (/31)
Claude Opus 5	max	3.8	89.0	73.5	49.0	85.2	92.9	27
GPT-5.6 Sol	max	11.0	88.1	79.5	73.8	69.5	91.9	28
Gemini 3.8 Flash	high	1.4	79.0	39.7	6.2	79.5	88.1	28
Kimi K3	max	2.4	79.0	38.7	5.7	53.8	82.9	28
Grok 4.6	xhigh	4.8	69.0	53.2	37.1	52.4	79.0	27
Qwen3.8-Max	max	2.4	65.7	64.0	61.4	42.4	69.5	23
Hy4 preview	high	0.5	63.3	30.2	1.4	28.6	66.2	26
DeepSeek-V4.1-Flash	max	1.4	59.5	46.3	23.3	40.5	68.6	25
Seed2.1 Pro	high	2.4	40.0	24.8	4.8	62.9	56.7	26
DeepSeek-V4-Pro	max	1.9	12.9	7.3	2.4	39.0	52.9	22
ALIENLOGIC *formal proof*								Aux. UnprovRec (%)
Grok 4.6	xhigh	42.4	83.8	72.1	64.3	97.1	97.1	90.7
Claude Opus 5	max	42.9	77.6	73.3	66.7	93.8	94.8	40.0
Qwen3.8-Max	max	43.8	76.2	72.7	70.0	96.7	94.3	53.3
GPT-5.6 Sol	max	51.0	75.2	73.0	70.5	92.9	94.8	44.0
Hy4 preview	high	33.8	74.8	67.8	60.5	95.7	94.8	54.7
DeepSeek-V4-Pro	max	32.9	73.8	65.1	60.0	94.8	93.8	54.7
Kimi K3	max	44.8	72.9	67.9	60.0	95.2	96.2	60.0
DeepSeek-V4.1-Flash	max	50.0	72.4	59.2	51.4	95.2	92.9	41.3
Seed2.1 Pro	high	37.1	67.6	59.5	53.8	93.3	93.3	48.0
Gemini 3.8 Flash	high	47.6	58.1	57.6	57.1	97.1	97.1	48.0

Reasoning labels are API-specific settings, not a common compute scale. Rules counts the hidden rules the Best@3 trajectory states correctly at M_4, and UnprovRec is its rate of correct refusal on unprovable theorems. Neither enters M_4.

Model score. Each system runs n independent trajectories. The primary score is the best endpoint reached by one complete trajectory,

$$\text{Best@}\,n = \max_r M_{r,4}, \qquad r^\star = \min \arg\max_r M_{r,4}. \tag{2}$$

Everything reported with the headline score comes from $r^\star$. Beside it we report $\text{Mean@}\,n$, the lowest endpoint, and every trajectory.

Rule reports. At every milestone the system also states the rule set S_t it believes. In ALIENCODE, each of the 31 rules is judged stated correctly or not (eq. (10)), which splits held-out accuracy by what the report says the system knows. ALIENLOGIC instead reports correct refusal on unprovable theorems. Rule reports never enter M_t. The retained gain $G_r = M_{r,4} - M_{r,0}$ (eq. (9)) and the exploration budget are defined in appendices B and F.4.

4 RESULTS AND FINDINGS

We evaluate ten frontier systems, each at the highest reasoning setting its API offers (appendix F.3). Every system runs n=3 independent trajectories per sandbox, each scored at milestones M_0 through M_4, and each control condition is run once per system. Table 1 lists each system's Best@3 trajectory beside the open-book references. Eight findings address three questions. RQ1 asks whether systems can acquire an alien world by exploring it, RQ2 how discovering a rule relates to using it, and RQ3 how reliably exploration succeeds.

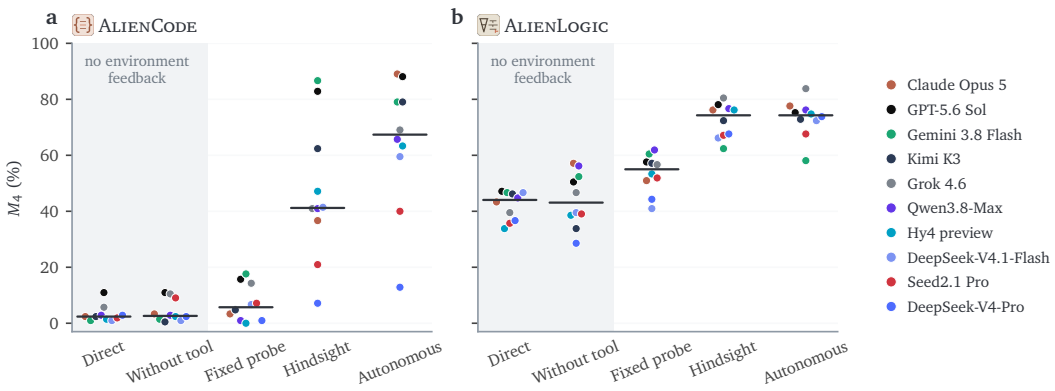


Figure 3: **Held-out accuracy at M_4 under five exploration conditions.** Each dot is one system and the line in each column marks the median. Autonomous exploration is shown at the Best@3 trajectory, whose probes hindsight exploration replays. Shaded columns receive no environment feedback, and from left to right the system takes a larger part in designing the experiments.

4.1 RQ1. Can AI systems acquire an alien world through exploration?

Finding 1. Exploration, not recall or thinking alone, produces the knowledge the tasks require.

Before exploration, recalled knowledge solves almost nothing in AlienCode, and no trajectory exceeds 15.7% at M_0. After four rounds, Best@3 reaches 89.0%, and 7 of 10 systems exceed 60%. AlienLogic starts higher, at 32.9–51.9%, because its rule changes leave part of standard natural deduction intact, and Best@3 rises to 58.1–83.8%. Additional model turns do not substitute for probing (fig. 3 and case 4). Without-tool answering adds the same model turns without environment feedback. It leaves AlienCode at 0.5–11.0%, and in AlienLogic it changes accuracy by -12.4 to $+13.8$ points relative to direct answering, lowering it for three systems.

Finding 2. Exploration works best when the system designs its own experiments.

The three feedback conditions use the same tool-calling interface and differ only in who designs the probes (fig. 3). Under autonomous exploration the system designs each experiment from its own history. Hindsight exploration hands back exactly the experiments of its Best@3 trajectory. The system still reads every result and records its hypotheses, but it no longer decides what to test next. Fixed-probe exploration samples experiments from grammar-valid templates with a fixed seed, the same sequence for every system. In AlienCode, the median falls from 67.4% under autonomous exploration to 41.2% under hindsight and 5.7% under fixed probes. Autonomous exploration beats hindsight for 9 of 10 systems, by a median of 17.4 points, although hindsight replays the probes of the best of the three trajectories. The sandbox is deterministic, so the evidence is identical, and the gap reflects designing the experiments rather than receiving their results. Gemini 3.8 Flash is the exception (86.7% under hindsight against 79.0%). In AlienLogic, designing the experiments adds nothing. Autonomous and hindsight exploration tie (median difference 0.5 points), and hindsight exceeds fixed probes by a median of 21.7 points, so what matters there is which proofs are tried rather than who chose them. Each hindsight run is a single trajectory and varies on its own. Three replays of Hy4 preview's AlienCode probes end at 47.1%, 10.0%, and 46.7%.

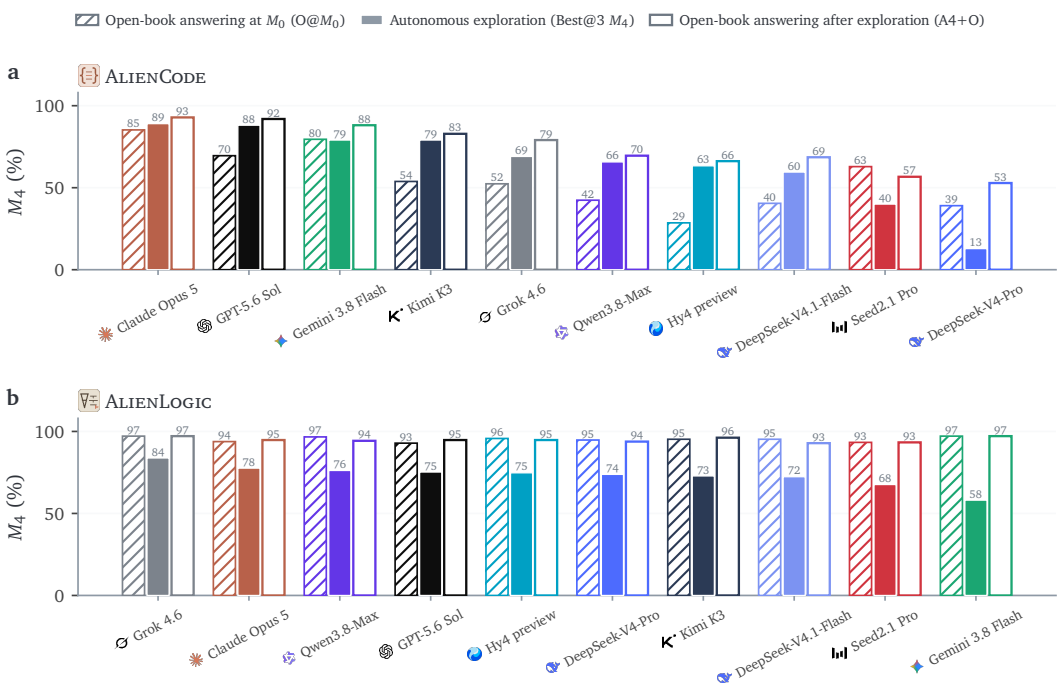


Figure 4: **Finding out versus being told.** Held-out accuracy at M_4 under open-book answering before exploration (O@M_0, hatched), after autonomous exploration alone (Best@3, solid), and under open-book answering after that exploration (A4+O, outlined). Systems are ordered by Best@3 M_4 within each sandbox.

Finding 3. A system's exploration ability varies across task settings.

The Spearman correlation between the two Best@3 rankings is 0.35 (fig. 8). Grok 4.6 ranks fifth in AlienCode and first in AlienLogic, Gemini 3.8 Flash ties for third and ranks last, and DeepSeek-V4-Pro moves from last to sixth. The sandboxes also separate systems differently. AlienCode spreads Best@3 over 12.9–89.0%, whereas eight of 10 systems fall within 72.4–83.8% in AlienLogic.

4.2 RQ2. How does discovering a rule relate to using it?

Finding 4. Discovering the rules beats being told them in AlienCode, but not in AlienLogic.

In AlienCode (fig. 4), the Best@3 trajectory outscores O@M_0 for 7 of 10 systems, and Mean@3 does so for 5. GPT-5.6 Sol reaches 88.1% against 69.5% under O@M_0, and Hy4 preview 63.3% against 28.6%. A4+O beats O@M_0 in 26 of 30 trajectories, by a median of 14.5 points, so exploration supplies practice in using the rules and not only the rules themselves. AlienLogic inverts the pattern. O@M_0 alone reaches 93–97%, no Best@3 trajectory matches it, and exploring first adds nothing (A4+O minus O@M_0 has a median of 0.0 points). AlienLogic is limited by discovery, and AlienCode by use. Exploration can also interfere with use. Seed2.1 Pro scores lower under A4+O than under O@M_0 (56.7% against 62.9%).

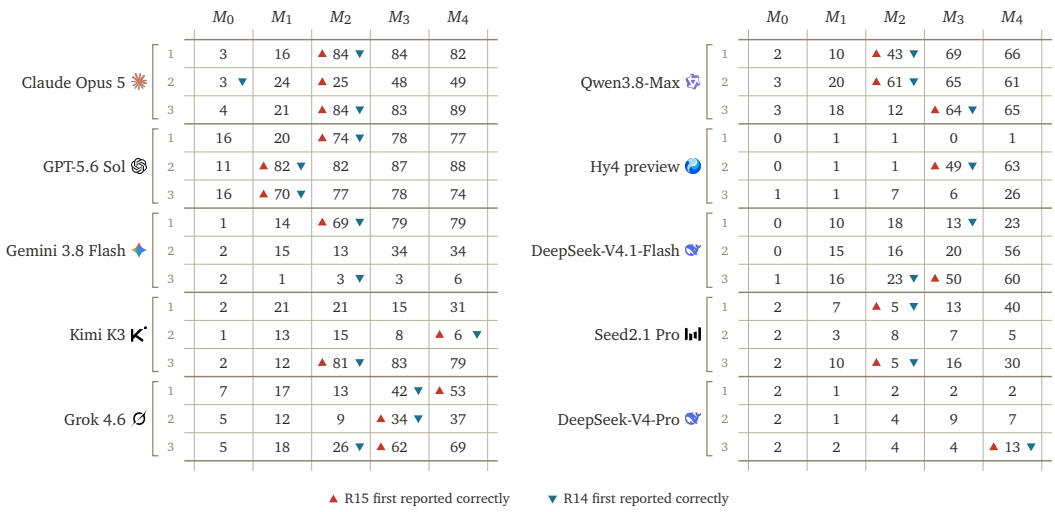


Figure 5: **Accuracy jumps when the keystone rules are found.** Each block is one system and each row one of its three ALIENCODE trajectories; columns are milestones, and each cell gives held-out accuracy (%). Triangles mark the milestone at which the trajectory's rule report first states R15 (CARVE slice shift, pointing up) and R14 (PLUCK index shift, pointing down) correctly.

Finding 5. ALIENCODE hinges on two keystone rules: systems that discover both correctly perform far better on subsequent held-out tasks.

Both keystone rules are positional, and together they enter 51 of the 70 tasks. They are the index shift of PLUCK (R14) and the slice shift of CARVE (R15). Of the 15 trajectories that end at or above 50%, 13 state both correctly in their rule reports, and of the 8 that end below 25%, 5 state neither. Large accuracy jumps coincide with their discovery (fig. 5 and case 1). Of the 13 jumps of at least 30 points between consecutive milestones, 12 occur in the round in which the rule report newly states R15 correctly, and 11 in the round in which it newly states R14. Across all 30 trajectories, the number of rules stated correctly at M_4 correlates with M_4 at $r = 0.85$. This evidence is correlational, and the rule report never enters the score.

Finding 6. Knowing a rule does not guarantee using it correctly.

When a trajectory's M_4 rule report states every rule a task requires correctly, the task is still solved only 73.4% of the time (647 task–trajectory pairs, fig. 9). Two trajectories state both keystone rules correctly yet end at 5.7% and 12.9%. The dissociation also runs the other way. In 26 of 30 trajectories, a rule stated correctly at one milestone later drops out of the rule report, yet accuracy on the tasks that require it still rises, from 11.4% to 16.9%. A task that requires a rule the report states incorrectly is still solved 34.6% of the time. The rule report is therefore a lossy readout of what a system can do and no substitute for held-out accuracy. Case 2 shows both directions within one system.

4.3 RQ3. HOW RELIABLE IS EXPLORATION?

Finding 7. Same-system, same-budget trajectories vary by tens of points, far beyond the noise from repeated answering.

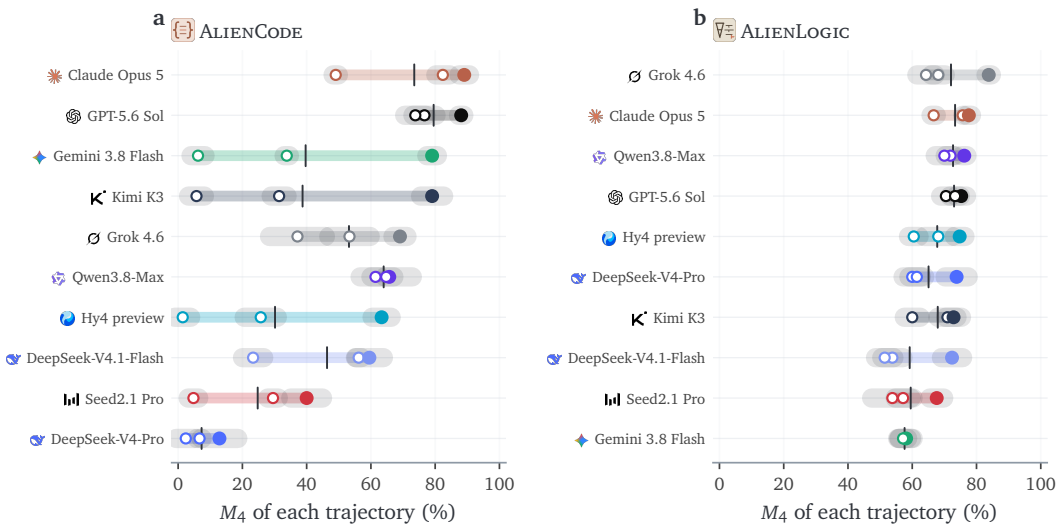


Figure 6: **Outcomes vary between trajectories, not between answers.** Each row is one system. Dots are its three trajectories' M_4 (filled for Best@3), the coloured band spans them, and the short vertical line marks Mean@3. The grey halo around each dot extends one standard deviation of that trajectory's three answers per question to either side.

Answering noise is small (fig. 6). Its standard deviation over the three answering passes at the same milestone is at most 4.7 points in AlienCode and 2.9 in AlienLogic. Trajectories of the same system differ far more, by up to 73.3 points in AlienCode and 21.0 in AlienLogic. In AlienCode, Kimi K3 ends between 5.7% and 79.0% (case 2), and Gemini 3.8 Flash between 6.2% and 79.0%. Every AlienCode trajectory spends 45–48 of its 48 tool calls, so the spread does not reflect a difference in effort. With outcomes this dispersed, the choice of aggregate changes the ranking. Qwen3.8-Max ranks sixth in AlienCode by Best@3 and third by Mean@3, while Kimi K3 falls from a tie for third to seventh.

Finding 8. Exploration gains arrive in leaps, at different times, and can reverse.

In each Best@3 trajectory in AlienCode, the largest single step contributes 47–92% of the retained gain (fig. 10). The round in which that step arrives differs by system. It is the first round for GPT-5.6 Sol, the second for Claude Opus 5, Kimi K3, and Gemini 3.8 Flash, and the third for Grok 4.6 and Hy4 preview. Seed2.1 Pro and DeepSeek-V4-Pro take it only in the last round and are still rising when the budget ends. Continued exploration can also undo progress. Of the 30 AlienCode trajectories, 6 end at least 3 points below an earlier milestone, as do 3 of the 30 AlienLogic trajectories. Gemini 3.8 Flash's Best@3 trajectory in AlienLogic peaks at 63.8% after the first round and ends at 58.1% (case 3).

5 Conclusion

We present ExplorationBench, a benchmark for how AI systems explore worlds whose rules contradict familiar knowledge. Its two sandboxes contain 55 discovery targets and 140 held-out tasks, and a program checks every answer. Across ten frontier systems, four rounds of exploration lift the best AlienCode trajectory to 89.0%, while the same turns without environment feedback stay at or below 11.0%. Exploration works best when the system designs its own experiments, but it is still far from reliable. A system's exploration ability differs across tasks, stating a rule correctly does not ensure using it, and trajectories of one system can end far apart. ExplorationBench provides a testbed for studying exploration as a capability in its own right and for developing systems that learn from their environments. More broadly, the same design can evaluate exploration wherever unfamiliar rules can be made executable and their consequences verified exactly, offering insights for developing future AI systems for scientific discovery. Limitations are discussed in appendix D.

AI USE STATEMENT

We used generative AI tools only to assist the presentation of this paper, including text editing, translation of quoted model outputs, and plotting and typesetting code. The authors reviewed all such material and take full responsibility for the content of this work.

REPRODUCIBILITY STATEMENT

Section 3 and appendix F specify the sandboxes, the worked examples, the interaction protocol (algorithm 1), the budget accounting, and every control condition. Appendix F.6 defines the verifiers and every reported metric. Appendix F.3 lists the evaluated model versions and reasoning settings, and appendix H reproduces logged probes, programs, proofs, and rule reports. The supplementary material contains the evaluation code as it ran, including the harnesses, the protocol, the scoring, and the control conditions. It also contains the ALIENCODE interpreter and the ALIENLOGIC proof checker with their rules factored out into rule packs, and two small public demo worlds built and checked with the same procedure. Loaded with the private rule packs, the released engines reproduce every verdict in our logs (7,497 proofs and 7,651 programs).

To prevent leakage, we do not release the rules, held-out tasks, answer keys, or interaction logs of the two evaluation worlds. Every task depends on the same small set of rules, so publishing any of them would let a future system reach the ceiling by recall rather than exploration, and its scores would no longer be comparable with those reported here. After publication, we will accept requests to evaluate new systems on the private worlds and will return aggregate scores only.

REFERENCES

Rishabh Agarwal, Avi Singh, Lei Zhang, Bernd Bohnet, Luis Rosias, Stephanie Chan, Biao Zhang, Ankesh Anand, Zaheer Abbas, Azade Nova, et al. Many-shot in-context learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pages 76930–76966, 2024.

Parth Asawa, Christopher M. Glaze, Gabriel Orlanski, Ramya Ramakrishnan, Benji Xu, Asim Biswal, Vincent Sunn Chen, Frederic Sala, Matei Zaharia, and Joseph E. Gonzalez. Continual learning bench: Evaluating frontier ai systems in real-world stateful environments. *arXiv preprint arXiv:2606.05661*, 2026.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-bench: Evaluating conversational agents in a dual-control environment, 2025.

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

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejas Patwardhan, Lilian Weng, and Aleksander Madry. MLE-bench: Evaluating machine learning agents on machine learning engineering. *International Conference on Learning Representations (ICLR)*, 2025.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. Teaching large language models to self-debug. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=KuPixIqPiq.

François Chollet. On the measure of intelligence. *arXiv preprint arXiv:1911.01547*, 2019. URL https://arxiv.org/abs/1911.01547.

Shihan Dou, Ming Zhang, Chenhao Huang, Jiayi Chen, Feng Chen, Shichun Liu, Yan Liu, Chenxiao Liu, Cheng Zhong, Zongzhang Zhang, Tao Gui, Chao Xin, Chengzhi Wei, Lin Yan, Qi Zhang, and Xuanjing Huang. Evalearn: Quantifying the learning capability and efficiency of llms via sequential problem solving. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Shihan Dou, Yujiong Shen, Chenhao Huang, Junjie Ye, Jiayi Chen, Junzhe Wang, Qianyu He, Shichun Liu, Changze Lv, Jiahang Lin, Jiazheng Zhang, Ming Zhang, Shaofan Liu, Tao Ji, Zhangyue Yin, Cheng Zhang, Huaibing Xie, Jianglu Hu, Jingcheng Deng, Lincheng Li, Minda Hu, Shaolei Wang, Syrus Zhao, Weichao Wang, Yan Lei, Yang Liu, Yanling Xiao, Yiting Liu, Zenan Xu, Zhen Guo, Ziliang Zhao, Pluto Zhou, Tao Gui, Qi Zhang, Xuanjing Huang, Yu-Gang Jiang, Di Wang, and Shunyu Yao. CL-bench Life: Can language models learn from real-life context?, 2026a.

Shihan Dou, Ming Zhang, Zhangyue Yin, Chenhao Huang, Yujiong Shen, Junzhe Wang, Jiayi Chen, Yuchen Ni, Junjie Ye, Cheng Zhang, Huaibing Xie, Jianglu Hu, Shaolei Wang, Weichao Wang, Yanling Xiao, Yiting Liu, Zenan Xu, Zhen Guo, Pluto Zhou, Tao Gui, Zuxuan Wu, Xipeng Qiu, Qi Zhang, Xuanjing Huang, Yu-Gang Jiang, Di Wang, and Shunyu Yao. CL-bench: A benchmark for context learning. *arXiv preprint arXiv:2602.03587*, 2026b.

Jinyuan Fang, Yanwen Peng, Xi Zhang, Yingxu Wang, Xinhao Yi, Guibin Zhang, Yi Xu, Bin Wu, Siwei Liu, Zihao Li, et al. A comprehensive survey of self-evolving ai agents: A new paradigm bridging foundation models and lifelong agentic systems. *arXiv preprint arXiv:2508.07407*, 2025.

Huan-ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Yiran Wu, Hongru Wang, Han Xiao, Yuhang Zhou, Shaokun Zhang, Jiayi Zhang, Jinyu Xiang, Yixiong Fang, Qiwen Zhao, Dongrui Liu, Qihan Ren, Cheng Qian, Zhenhailong Wang, Minda Hu, Huazheng Wang, Qingyun Wu, Heng Ji, and Mengdi Wang. A survey of self-evolving agents: What, when, how, and where to evolve on the path to artificial super intelligence. *Transactions on Machine Learning Research (TMLR)*, 2026. Preprint at arXiv:2507.21046.

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

Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. MLAgentBench: Evaluating language agents on machine learning experimentation. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, pages 20271–20309, 2024.

Peter Jansen, Marc-Alexandre Côté, Tushar Khot, Erin Bransom, Bhavana Dalvi Mishra, Bodhisattwa Prasad Majumder, Oyvind Tafjord, and Peter Clark. DiscoveryWorld: A virtual environment for developing and evaluating automated scientific discovery agents. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=cDYqckEt6d.

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

Brenden M. Lake, Tomer D. Ullman, Joshua B. Tenenbaum, and Samuel J. Gershman. Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40:e253, 2017.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pages 46534–46594, 2023.

Mathematical Association of America. AIME: American invitational mathematics examination. https://maa.org/math-competitions/aime, 2026. Accessed 2026-07-15.

Lingrui Mei, Jiayu Yao, Yuyao Ge, Yiwei Wang, Baolong Bi, Yujun Cai, Jiazhi Liu, Mingyu Li, Zhong-Zhi Li, Duzhen Zhang, et al. A survey of context engineering for large language models. *arXiv preprint arXiv:2507.13334*, 2025.

Mike A. Merrill et al. Terminal-Bench 2.0: Evaluating language models on real terminal workflows. *arXiv preprint arXiv:2606.terminalbench*, 2026.

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 *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 11048–11064, 2022. URL https://aclanthology.org/2022.emnlp-main.759/.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, et al. In-context learning and induction heads. *Transformer Circuits Thread*, 2022. URL https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html. Also arXiv:2209.11895.

Yonatan Oren, Nicole Meister, Niladri S. Chatterji, Faisal Ladhak, and Tatsunori Hashimoto. Proving test set contamination in black-box language models. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=KS8mIvetg2.

Tejas Patwardhan et al. GDPval: A benchmark for evaluating frontier model performance on professional deliverables. *arXiv preprint arXiv:2509.gdpval*, 2025.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pages 8634–8652, 2023.

David Silver and Richard S. Sutton. Welcome to the era of experience. In George Konidaris, editor, *Designing an Intelligence*. MIT Press, 2025. URL https://storage.googleapis.com/deepmind-media/Era-of-Experience/The%20Era%20of%20Experience%20Paper.pdf. In press.

Theodore R. Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas L. Griffiths. Cognitive architectures for language agents. *Transactions on Machine Learning Research (TMLR)*, 2024.

Xiaojuan Tang, Jiaqi Li, Yitao Liang, Song-Chun Zhu, Muhan Zhang, and Zilong Zheng. MARS: Situated inductive reasoning in an open-world environment. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=3qoQ6AolAz.

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*, 2023a. URL https://arxiv.org/abs/2305.16291.

Hanchen Wang, Tianfan Fu, Yuanqi Du, Wenhao Gao, Kexin Huang, Ziming Liu, Payal Chandak, Shengchao Liu, Peter Van Katwyk, Andreea Deac, Anima Anandkumar, Karianne Bergen, Carla P. Gomes, Shirley Ho, Pushmeet Kohli, Joan Lasenby, Jure Leskovec, Tie-Yan Liu, Arjun Manrai, Debora Marks, Bharath Ramsundar, Le Song, Jimeng Sun, Jian Tang, Petar Veličković, Max Welling, Linfeng Zhang, Connor W. Coley, Yoshua Bengio, and Marinka Zitnik. Scientific discovery in the age of artificial intelligence. *Nature*, 620(7972):47–60, 2023b. doi: 10.1038/s41586-023-06221-2.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, 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 (NeurIPS)*, volume 35, pages 24824–24837, 2022.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An explanation of in-context learning as implicit bayesian inference. In *International Conference on Learning Representations (ICLR)*, 2022.

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

Shunyu Yao, Howard Chen, Austin W. Hanjie, Runzhe Yang, and Karthik Narasimhan. COLLIE: Systematic construction of constrained text generation tasks. In *International Conference on Learning Representations (ICLR)*, 2024.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=roNSXZpUDN.

Jiarui Yuan, Tailin Jin, Weize Chen, Zeyuan Liu, Zhiyuan Liu, and Maosong Sun. SE-Bench: Benchmarking self-evolution with knowledge internalization. *arXiv preprint arXiv:2602.04811*, 2026. URL https://arxiv.org/abs/2602.04811.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 19632–19642, 2024.

Tianshi Zheng, Kelvin Kiu Wai Tam, Newt Nguyen Kim Hue Nam, Baixuan Xu, Zhaowei Wang, Jiayang Cheng, Hong Ting Tsang, Weiqi Wang, Jiaxin Bai, Tianqing Fang, Yangqiu Song, Ginny Wong, and Simon See. NewtonBench: Benchmarking generalizable scientific law discovery in LLM agents. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://openreview.net/forum?id=Gk6umqW74m. Preprint at arXiv:2510.07172.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations (ICLR)*, 2024.

Deyao Zhu, Xin Zhou, Shengling Qin, Xuekai Zhu, Hangliang Ding, Shu Zhong, Zixin Wen, Zhonglin Xie, Chenhui Gou, Linxuan Ren, et al. EdgeBench: Unveiling scaling laws of learning from real-world environments. *arXiv preprint arXiv:2607.05155*, 2026. URL https://arxiv.org/abs/2607.05155.

Yuxin Zuo, Zikai Xiao, Li Sheng, Fei Huang, Jianhong Tu, Yuxuan Liu, Tianyi Tang, Xiaomeng Hu, Yang Su, Qingfeng Lan, et al. Qwen-AgentWorld: Language world models for general agents. *arXiv preprint arXiv:2606.24597*, 2026. URL https://arxiv.org/abs/2606.24597.

A ADDITIONAL FIGURES

Figure 7 shows how held-out accuracy changes over the milestones of every system's Best@3 trajectory. The other figures support section 4 and are referenced from the findings they illustrate.

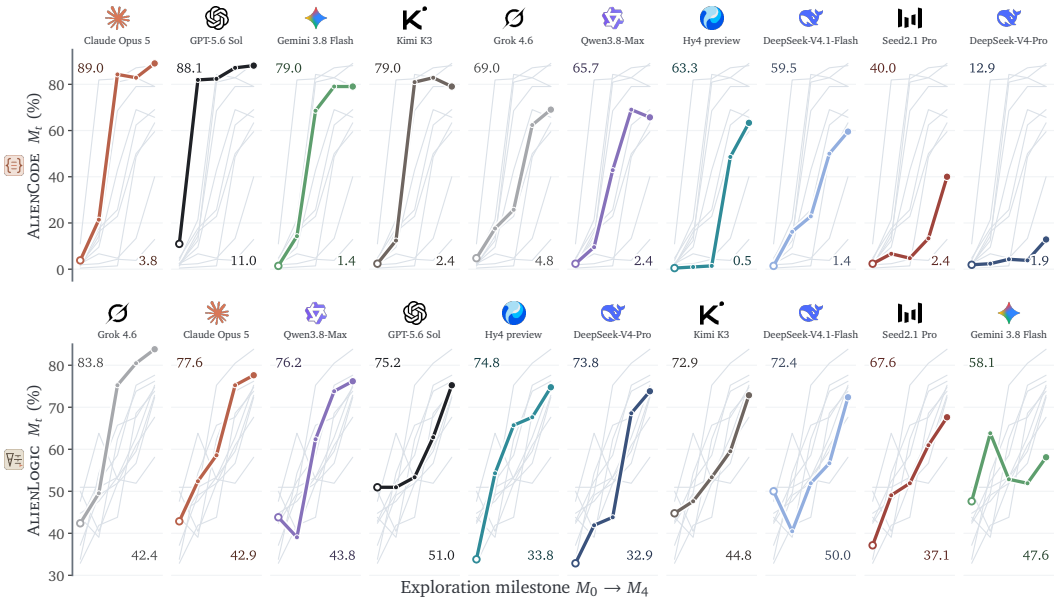


Figure 7: **Held-out accuracy over autonomous exploration.** Top: ALIENCODE; bottom: ALIENLOGIC. Each cell highlights one system's M_0–M_4 curve; the other nine are grey. The upper-left and lower-right values are M_4 and M_0. Cells are ordered by M_4 within each sandbox. Each curve is the Best@3 trajectory, and each milestone is the mean of three answers per held-out question.

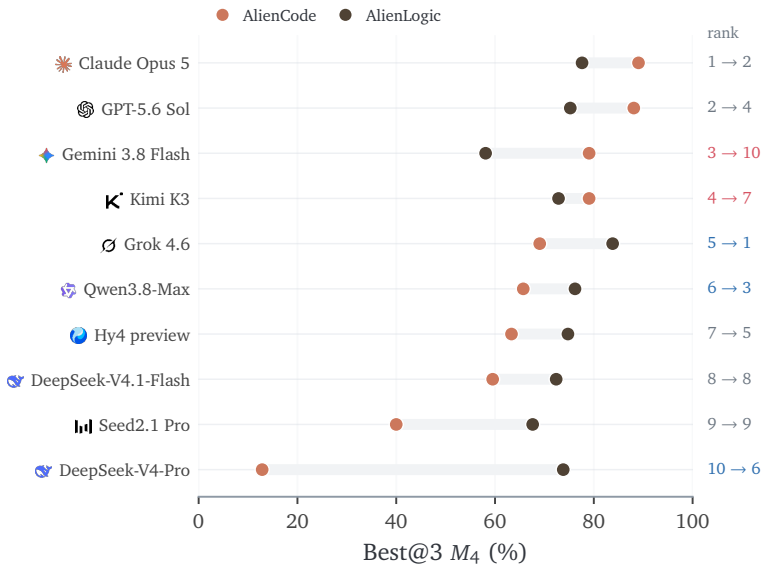


Figure 8: **Best@3 M_4 in the two sandboxes.** Each row is one system, ordered by its ALIENCODE score; the right column gives its rank in ALIENCODE and then in ALIENLOGIC. Red marks a fall of three or more places, blue a rise of three or more.



Figure 9: **Knowing a rule is not being able to use it.** For each system, pooled over its three ALIENCODE trajectories, filled dots give accuracy at M_4 on tasks whose required rules the M_4 rule report states correctly, and open dots on tasks with at least one required rule missing or stated incorrectly. The dashed segment up to 100% is the remaining execution gap; n counts task–trajectory pairs in the first group.

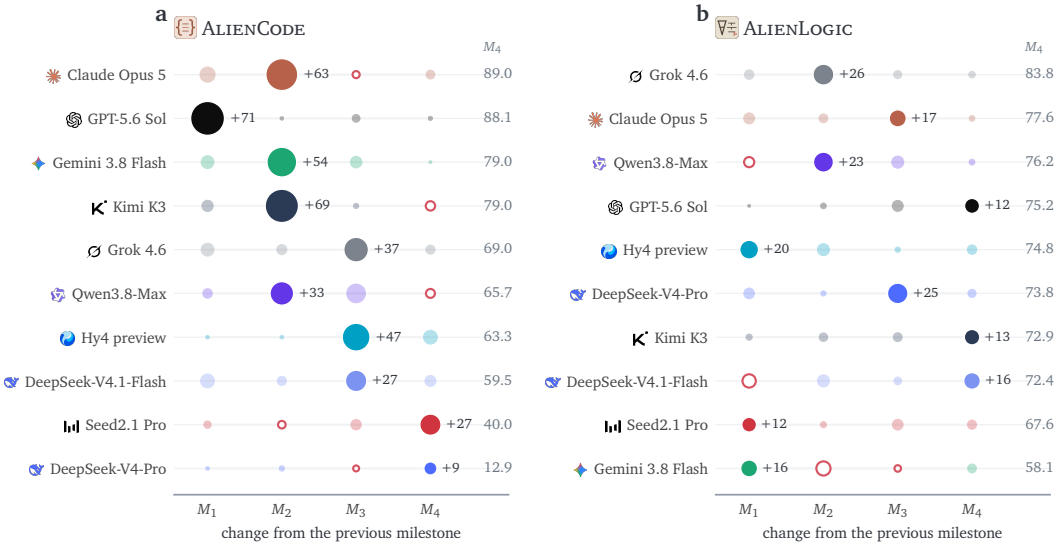


Figure 10: **When the gain arrives.** For each system's Best@3 trajectory, bubble area is the change in held-out accuracy from the previous milestone ($M_t - M_{t-1}$). The solid bubble marks the largest step and is labelled with its size; a red ring marks a milestone at which accuracy fell. The right column gives M_4.

B METRIC DETAILS

This section gives the full metric definitions summarized in section 3.3.

What we measure. Held-out task performance is $Q(H_t) = \Pr_{\tau\sim\mathcal{T}}[\text{the system solves } \tau \text{ given } H_t]$. The exploration curve $(Q(H_t))_{t=0}^{N}$ is reported as $(M_0, \ldots, M_N)$, with M_0 measured after the worked examples and M_t after round t. The benchmark records three connected parts of the process separately. *Probe selection* is which probes π selects, read from the recorded probes and budget. *Reported discovery* is the rule set S_t the system reports from H_t. *Rule use* is whether the system can apply H_t to unseen tasks, measured by held-out accuracy. The three diverge in our experiments (section 4), so they are reported separately rather than as one score.

Held-out accuracy. Every held-out question is answered three times, each time in a fresh tool-disabled copy of the conversation at that milestone. For trajectory r at milestone t, with $y^{(k)}_{r,\tau,t} \in \{0,1\}$ the verdict on the k-th answer to task τ,

$$M_{r,t} = \frac{100}{|\mathcal{T}|} \sum_{\tau \in \mathcal{T}} \frac{1}{3} \sum_{k=1}^{3} y^{(k)}_{r,\tau,t}. \tag{3}$$

An answer that is missing or exhausts its time budget scores zero. Writing $M^{(k)}_{r,t}$ for the score of the k-th pass alone, the *answering noise* $\sigma_{r,t}$ is the standard deviation of $M^{(1)}_{r,t}$, $M^{(2)}_{r,t}$, and $M^{(3)}_{r,t}$. It measures how much the score moves when the same knowledge answers again, with exploration held fixed.

Model score. Each system runs n independent trajectories under the same protocol. The primary score is the best endpoint reached by one complete trajectory,

$$\mathrm{Best@}\, n = \max_{r \in \{1,\dots,n\}} M_{r,4}, \qquad r^\star = \min \arg\max_{r} M_{r,4}, \tag{4}$$

where the lowest trajectory index breaks an exact tie. The milestone curve, rule reports, and budget reported with the headline score all come from $r^\star$, and no synthetic trajectory is assembled from different trajectories at different milestones or tasks. Beside Best@n we report $\mathrm{Mean@}\, n = n^{-1} \sum_r M_{r,4}$, the lowest endpoint, and every trajectory, and rankings compare systems only at the same n.

Rule reports. At every milestone the system also states the rule set S_t it currently believes. In ALIENCODE, each of the 31 evaluator-side rules is judged stated correctly or not, and we report the number stated correctly (eq. (10)). Each task τ exercises a known set of rules $\mathcal{R}(\tau)$. The task is *covered* at milestone t when every rule in $\mathcal{R}(\tau)$ is stated correctly, which splits held-out accuracy by what the report says the system knows. ALIENLOGIC has no rule-report score, and its auxiliary diagnostic is correct refusal on unprovable theorems. Rule reports are diagnostic and never enter M_t.

Dynamics and budget. We describe how a trajectory reaches its endpoint by its per-round steps and its retained gain $G_r = M_{r,4} - M_{r,0}$ (eq. (9)). The benchmark also records the exploration budget $\mathbf{B} = (C, P, T)$ of tool calls, probe units, and exploration tokens. The budget is reported but not scored, and its accounting is given in appendix F.4.

C DISCUSSION

C.1 WHAT BEST@n MEASURES

Best@n asks what endpoint a system can reach within n independent attempts. It answers whether a system can build an effective exploration trajectory at all, but it is not a reliability measure: a system with one strong trajectory and $n-1$ failures can outrank one whose every trajectory is moderately strong. We therefore compare systems only at the same n and report Mean@n, every endpoint, and the best–worst range beside the ranking.

The best trajectory is chosen once, by its complete M_4 accuracy, and every analysis of that system uses the same trajectory. Taking the best milestone from one trajectory and the best answer to each task from another would construct an outcome that no agent produced. For the same reason, Best@n is not pass@n, which takes the union of task-level successes and can exceed every observed trajectory.

C.2 WHY THE SEED PHASE PROVIDES TEN EXAMPLES

Before M_0, ALIENCODE gives every system ten fixed examples covering the basic program forms used later: literals, operator calls, sequences, nesting, and function bodies. This establishes a shared minimum fluency with the language and prevents basic syntax learning from being conflated with exploration, while leaving most hidden rules to be discovered in the four scored rounds.

D LIMITATIONS

EXPLORATIONBENCH deliberately reduces exploration to two deterministic, executable worlds so that every experiment and held-out answer has an exact outcome. That control is also its boundary: real scientific exploration involves noisy and incomplete observations, costly or irreversible experiments, open-ended hypothesis spaces, and horizons far longer than four rounds. Our results therefore measure whether systems can acquire unfamiliar rules in verifiable synthetic environments, not whether they can conduct real scientific discovery. Three trajectories per system and three answers per question expose trajectory and answering variability, but are insufficient to estimate either distribution precisely.

E EXTENDED RELATED WORK

This section extends section 2 and positions EXPLORATIONBENCH relative to three lines of work: interactive scientific discovery and rule induction, context learning and test-time adaptation, and verifiable agent evaluation (Tang et al., 2024; Jansen et al., 2024; Dou et al., 2026b; 2025; Jimenez et al., 2024; Yao et al., 2025).

Scientific discovery and interactive rule induction. Interactive benchmarks ask agents to infer hidden or altered rules from evidence, connecting to causal world models and active system identification (Lake et al., 2017). MARS and DiscoveryWorld study investigation in fictional worlds (Tang et al., 2024; Jansen et al., 2024), and NewtonBench targets scientific law discovery (Zheng et al., 2026). Nearby settings measure compliance with stated constraints in COLLIE (Yao et al., 2024) and search over experiment configurations in MLAgentBench (Huang et al., 2024). Voyager instead evaluates open-ended skill accumulation in a sandbox (Wang et al., 2023a). EXPLORATIONBENCH measures the complete path from selecting probes, through reporting discoveries, to using them on unseen tasks. Its unfamiliar executable worlds help separate knowledge acquired during evaluation from pre-trained recall.

Context learning and test-time adaptation. A growing position emphasizes experience generated by the system itself (Silver and Sutton, 2025). CL-bench and CL-bench Life study learning from complex provided contexts (Dou et al., 2026b;a), while EvaLearn studies experience across sequential problems (Dou et al., 2025). SE-Bench moves adaptation into model weights (Yuan et al., 2026), and EdgeBench examines longer-horizon learning in real-world environments (Zhu et al., 2026). Surveys organize self-evolving agents across changes to weights, memory, tools, and prompts (Gao et al., 2026; Fang et al., 2025). This setting also connects to context engineering and in-context learning (Mei et al., 2025; Brown et al., 2020), with mechanisms studied through induction circuits and implicit Bayesian inference (Olsson et al., 2022; Xie et al., 2022) and in-context state represented explicitly in language-agent architectures (Sumers et al., 2024).

Existing approaches typically study learning from supplied demonstrations (Min et al., 2022), feedback on a system's own attempts (Shinn et al., 2023; Madaan et al., 2023; Chen et al., 2024), or stored trajectories replayed as context (Zhao et al., 2024). Longer reasoning instead spends inference without adding evidence (Wei et al., 2022). EXPLORATIONBENCH holds parameters fixed and places these routes in one protocol. Autonomous exploration selects probes online, hindsight exploration replays the system's own best trajectory, and without-tool answering adds deliberation without environment feedback. This separates who selects evidence from whether evidence arrives.

Verifiable interactive evaluation. Verifiable agent evaluation grades outcomes rather than descriptions, building on the interleaving of reasoning and tool use in ReAct (Yao et al., 2023). SWE-bench grades repository patches with tests (Jimenez et al., 2024), WebArena scores website end states (Zhou et al., 2024), and τ-bench compares database states with annotated goals (Yao et al., 2025). τ^2-bench extends this setting to environments in which both parties act (Barres et al., 2025). EXPLORATIONBENCH shares this preference for executable outcomes. An interpreter grades ALIENCODE, while a proof-checker and bounded certifier grade ALIENLOGIC. A related line turns the environment itself into the prediction target: Qwen-AgentWorld trains a language world model to simulate how an environment would respond, scored against recorded observations across seven domains (Zuo et al., 2026). That asks how faithfully a system can reproduce known dynamics, whereas EXPLO-

RATIONBENCH asks whether it can uncover dynamics nobody stated. Here, the governing rules must be discovered before they are used on closed-book tasks. Table 2 summarizes the differences in evidence, novelty, progress measurement, and transfer.

F EXPERIMENTAL DETAILS

F.1 ENVIRONMENTS, TASKS, AND WORKED EXAMPLES

EXPLORATIONBENCH contains two environments with the same exploration and evaluation protocol but different executable objects. ALIENCODE exposes a small programming language whose familiar-looking operators follow hidden semantics. Its evaluator contains 31 discovery targets and 70 held-out tasks. ALIENLOGIC exposes a Fitch-style proof checker with 24 discovery targets, each an active rule patch, and 70 held-out tasks, including designated unprovable goals. Programs must pass all private inputs, and proofs must verify. An unprovable goal earns credit only for a correct refusal.

ALIENCODE includes deliberately counter-intuitive keywords and 8 red herrings. For example, `SHATTER` appears to mean destruction but means multiply, while `EMIT(100)` prints `127`. Some rules are scope-dependent, and no primitive performs addition, so the model must construct addition from other operations. Engineering and algorithm tasks that require code are evaluated on 5 private legal inputs. Displayed examples specify task interfaces rather than the private test set.

ALIENLOGIC uses non-contiguous opaque rule IDs and side conditions, including duplicate-premise and use-once guards that prevent a model from discharging an assumption by simply restating a premise. Its designated unprovable tasks are certified only within declared formula, line, indentation, and node bounds.

Both environments use prescribed worked examples before M_0, but the examples are sandbox-specific. ALIENCODE presents ten fixed worked examples, balanced across five task bands (apply, interact, scope, engineer, and algorithm). Each contains a task, a reference AlienCode program, and the output the interpreter computes for it. The model submits no code in this phase, so every trajectory reaches M_0 with identical evidence and no tool calls charged. ALIENLOGIC presents eight accepted worked proofs that show valid proof syntax. M_0 is therefore comparable within a sandbox, but comparisons of $M_4 - M_0$ across sandboxes are only descriptive, because the two sets of worked examples differ.

F.2 COMPARISON WITH RELATED BENCHMARKS

Table 2 places EXPLORATIONBENCH beside the benchmarks of section 2: where each one's evidence comes from, how it controls novelty, and where it finally tests competence.

F.3 MODELS AND REASONING CONTROLS

We evaluate GPT-5.6 Sol, Claude Opus 5, Qwen3.8-Max (0902), Gemini 3.8 Flash, DeepSeek-V4.1-Flash, DeepSeek-V4-Pro, Grok 4.6, Kimi K3, Seed2.1 Pro (0915), and Hy4 preview. Each system uses the highest reasoning setting its API offers, fixed across trajectories and sandboxes; the high/xhigh/max labels name request settings, not comparable compute levels. Dated suffixes identify the exact model versions evaluated.

Each system runs n=3 independent trajectories per sandbox. Interactions use structured tool calls: the environment executes each call and returns a tool result (appendix F.5).

F.4 INTERACTION BUDGET AND CONTROL CONDITIONS

Exploration keeps one continuous history. After each round, the system is tested in a discarded, tool-disabled copy of the conversation, so testing never becomes evidence for later rounds.

Milestone number is a protocol stage, not a resource unit. At each milestone we record cumulative tool calls C, probe units P, and exploration tokens T. Each of the four rounds allows at most 12 tool calls in ALIENCODE and 12 proofs in ALIENLOGIC, the latter capped at 48 across the run. ALIENCODE packs up to five probe units into one tool call, charging one for each top-level `EMIT` argument and three for each such observation inside a loop body; ALIENLOGIC submits one

Table 2: **Where the evidence comes from, and where competence is tested.** Each row states a benchmark's primary protocol rather than ranking design choices. EXPLORATIONBENCH measures exploration by combining autonomous probe selection, milestone measurement, and unseen-task evaluation after interaction ends; no single column alone defines that distinction.

		Learning setup		*What is measured*		
Benchmark	**Target**	**Evidence**	**Novelty control**	**Progress**	**Transfer**	**Ground truth**
CL-bench (Dou et al., 2026b)	Context learning	Provided context	Expert-authored content	Final score	Same context	Expert rubrics, LLM verifier
EvaLearn (Dou et al., 2025)	Sequential learning	Prior solved tasks	Authored task sequences	Learning curve	Later related tasks	Rubrics, LLM verifier
SE-Bench (Yuan et al., 2026)	Weight internalization	Docs, training tasks	Obfuscated APIs	Pre/post score	Closed-book held-out	Tests, AST checks
SWE-bench (Jimenez et al., 2024)	Software repair	Issue, codebase	Real GitHub issues	Final patch	Same repository	Test suites
DiscoveryWorld (Jansen et al., 2024)	Scientific investigation	Agent actions	Fictional worlds	Final score	Same world	World state
NewtonBench (Zheng et al., 2026)	Physical-law discovery	Chosen experiments	Counterfactual laws	Final equation	Inferred law	Symbolic equivalence
EdgeBench (Zhu et al., 2026)	Long-horizon learning	Environment feedback	New real-world tasks	Learning curve	Same task	Task-specific evaluator
EXPLORATION BENCH (ours)	**Exploration**	**Chosen probes and feedback**	**Executable rules that conflict with priors**	**$M_0 \to M_4$ milestones**	**Unseen tasks after interaction**	**Interpreter; proof checker, bounded certifier**

verifier-evaluated proof per call, so $C = P$. We use actual consumption, not the caps, in every budget we report. The loop charge is fixed rather than multiplied by the realized iteration count, and a call over the five-unit cap is rejected in full and executes no code. Feedback is capped separately at 15 lines and 600 characters per call, and returned output never adds probe units. The worked examples are context rather than model calls and are not charged. The budget is recorded and reported but never scored.

Table 3 reports the budget of each system's Best@3 trajectory. Every trajectory spends nearly all of its tool calls, while exploration tokens differ by more than an order of magnitude between systems.

Table 3: **Exploration budget of each system's Best@3 trajectory.** C is tool calls, P probe units, and T exploration tokens in millions, summed over the four rounds. In ALIENLOGIC, $C = P$. Rows follow the ALIENCODE order of table 1.

	ALIENCODE			ALIENLOGIC	
System	C	P	T (M)	$C = P$	T (M)
Claude Opus 5	48	168	11.59	48	0.67
GPT-5.6 Sol	48	131	3.62	48	0.91
Gemini 3.8 Flash	45	124	2.41	48	1.08
Kimi K3	45	216	2.01	48	0.38
Grok 4.6	48	197	0.50	48	0.15
Qwen3.8-Max	48	162	0.99	48	0.68
Hy4 preview	48	210	1.41	48	0.33
DeepSeek-V4.1-Flash	48	143	10.85	48	1.63
Seed2.1 Pro	48	219	0.34	48	0.18
DeepSeek-V4-Pro	48	210	4.55	48	0.37

F.5 FORMAL TOOL-CALL PROTOCOL

Each request continues the API's structured conversation history, keeping tool-call IDs and reasoning-state fields instead of flattening them into text, so later probes condition on the full interaction history.

Algorithm 1 The EXPLORATIONBENCH protocol for structured tool calls. Fixed worked examples establish M_0. After each round the benchmark records C, P, and T, and the system is tested in discarded, tool-disabled copies of the conversation, so only probe feedback updates H.

Require: environment tool $\mathcal{E}$ (hidden rules $\mathcal{R}$); manual $\mathcal{M}$; fixed worked examples $\mathcal{D}$; held-out set $\mathcal{T}$; rounds N; sandbox caps $\bar{\mathbf{B}}$
Ensure: milestone scores $(M_0, \ldots, M_N)$, rule reports $(S_0, \ldots, S_N)$, and actual budgets $(\mathbf{B}_0, \ldots, \mathbf{B}_N)$

$H \leftarrow$ INITHYPOTHESIS$(\mathcal{M}, \mathcal{D})$ ▷ identical evidence; no model tool calls
for $i \leftarrow 0$ **to** N **do**
  $\mathbf{B}_i \leftarrow$ CUMULATIVEUSAGE(H)
  $S_i \leftarrow$ REPORTRULES(TOOLDISABLEDCOPY(H))
  $M_i \leftarrow$ SCORETOOLDISABLEDCOPIES$(H, \mathcal{T})$
  if $i < N$ **then** ▷ **Evidence acquisition**. Run probes and update
    while the model requests a call to $\mathcal{E}$ and budget remains **do**
      $x \leftarrow$ TOOLARGUMENTS(H); $f \leftarrow$ EXECUTE$(\mathcal{E}, x)$
      $H \leftarrow$ RETURNTOOLRESULT(H, x, f)
return $(M_0, \ldots, M_N)$, $(S_0, \ldots, S_N)$, $(\mathbf{B}_0, \ldots, \mathbf{B}_N)$

For trajectory r, the cumulative exploration budget at milestone t is

$$\mathbf{B}_{r,t} = \big(C_{r,t}, P_{r,t}, T_{r,t}\big), \tag{5}$$

where C is executed tool calls and therefore submitted probes, P is charged probe units, and T is API-reported input plus output tokens for model requests made during exploration.

F.6 EXECUTABLE OUTCOMES AND METRIC DEFINITIONS

Let $a_{\tau,t} = A(\tau; H_t)$ be the response produced for unseen task τ in an independent tool-disabled copy of the conversation at milestone t. The benchmark first reduces every response to an executable binary outcome

$$y_{\tau,t} = V_{\mathcal{E}}(\tau, a_{\tau,t}) \in \{0, 1\}, \tag{6}$$

where $V_{\mathcal{E}}$ is the sandbox-specific verifier. ALIENCODE holds two kinds of task. For the 69 synthesis tasks the model submits a program, which passes only if it is correct on every evaluator input; for the 1 prediction task the task supplies the program z_τ and the model submits the output it expects, which passes only on an exact match with what the interpreter actually prints:

$$V_{\mathcal{E}}^{\text{CODE}}(\tau, a) = \begin{cases} \prod_{z \in \mathcal{Z}_\tau} \mathbb{1}\big[\text{Exec}_{\mathcal{E}}(a, z) = g_\tau^{\mathcal{E}}(z)\big], & \text{synthesis,} \\ \mathbb{1}[a = \text{Exec}_{\mathcal{E}}(z_\tau)], & \text{prediction.} \end{cases} \tag{7}$$

Both branches compare normalized text, so trailing whitespace and an equal-valued numeric literal do not decide a verdict. For ALIENLOGIC, let $u_\tau = 1$ denote a theorem designated unprovable within the benchmark's declared certifier bounds. Then

$$V_{\mathcal{R}}^{\text{LOGIC}}(\tau, a) = \begin{cases} \mathbb{1}[\text{ProofCheck}_{\mathcal{R}}(\tau, a) = \text{ACCEPT}], & u_\tau = 0, \\ \mathbb{1}[\text{Decline}(a)], & u_\tau = 1. \end{cases} \tag{8}$$

For trajectory r at milestone t, held-out accuracy $M_{r,t}$ is the three-answer mean of eq. (3) over the full scored set $\mathcal{T}$. The per-pass scores $M_{r,t}^{(k)}$ give the answering noise $\sigma_{r,t}$. The *step* $s_{r,t} = M_{r,t} - M_{r,t-1}$ is the change during round t, and the *retained gain* is their sum,

$$G_r = M_{r,4} - M_{r,0} = \sum_{t=1}^{4} s_{r,t}. \tag{9}$$

We describe how a trajectory reaches its endpoint by the share of G_r carried by its largest step and by whether it ends below an earlier milestone. These quantities describe the selected trajectory and are never used to select it.

The rule report score of section 3.3 scores the reported rule set S_t against the evaluator-side ALIENCODE rule inventory $\mathcal{R}_{\text{eval}}$ as the number of rules it states correctly:

$$\text{RuleReport}_t = \sum_{\rho \in \mathcal{R}_{\text{eval}}} J(\rho, S_t), \tag{10}$$

It is elicited at every milestone; we report its value at M_4.

Over n trajectories, the model score and its selected trajectory are

$$\mathrm{Best@}\, n = \max_r M_{r,4}, \qquad r^\star = \min \arg\max_r M_{r,4}. \tag{11}$$

All trajectory-level quantities paired with the headline score—M_0, G, intermediate milestones, rule reports, budgets, and errors—are read from $r^\star$. We separately report $\mathrm{Mean@}\, n = n^{-1} \sum_r M_{r,4}$, every endpoint, and the best–worst range, and rank systems only at the same n. We attach no confidence interval to Best@n: a maximum depends explicitly on n, which is why n and every trajectory are reported beside it.

The five exploration conditions differ only in how evidence is obtained; the sixth, open-book answering, supplies the complete rule set as the reference used in RQ2.

- Direct answering (d). No model turns occur after M_0.
- Without-tool answering (w). Model turns for hypothesis revision occur without environment feedback.
- Fixed-probe exploration (f). Probes are sampled deterministically from a fixed pool of grammar-valid templates. One sequence serves all ten systems: its seed is fixed, and its per-call volume follows one mid-ranked trajectory, the Best@3 trajectory of Grok 4.6, so every system meets the same probes, independent of its own hypotheses.
- Hindsight exploration (h). The probes of the same system's Best@3 trajectory in that sandbox, chosen after the fact by its M_4, are replayed through the same tool-calling interface.
- Autonomous exploration (a). Probes are selected from the trajectory's own history.
- Open-book answering (o). The complete rule set is supplied before evaluation, either before exploration (O@M_0) or after the four autonomous rounds (A4+O).

Hindsight replay keeps the structured exchange of tool calls and tool results and changes only who chooses the probe. The replayed probes are submitted verbatim but executed live, so their feedback is recomputed rather than copied, and the system still records its own hypotheses between probes. The difference between autonomous and hindsight exploration therefore reflects choosing the probes, not a change of interaction protocol. Because the replayed sequence comes from the system's best trajectory rather than an average over trajectories, the comparison is conditioned on that particular sequence.

G PER-SYSTEM RESULTS

Figures 11 to 14 break each system's gain down by task family and by milestone under the five exploration conditions. Autonomous exploration averages three trajectories; each control condition contributes the one trajectory scored for the control study. Scores are three-answer means (eq. (3)), and gains are measured from each condition's own M_0.



Figure 11: **Task-family gains for every AlienCode system.** Cells show $M_4 - M_0$ within each task family, in percentage points. Tasks are split by compositional depth and by representation, so each task enters one column of each. Autonomous rows average three trajectories; control rows are one trajectory each.

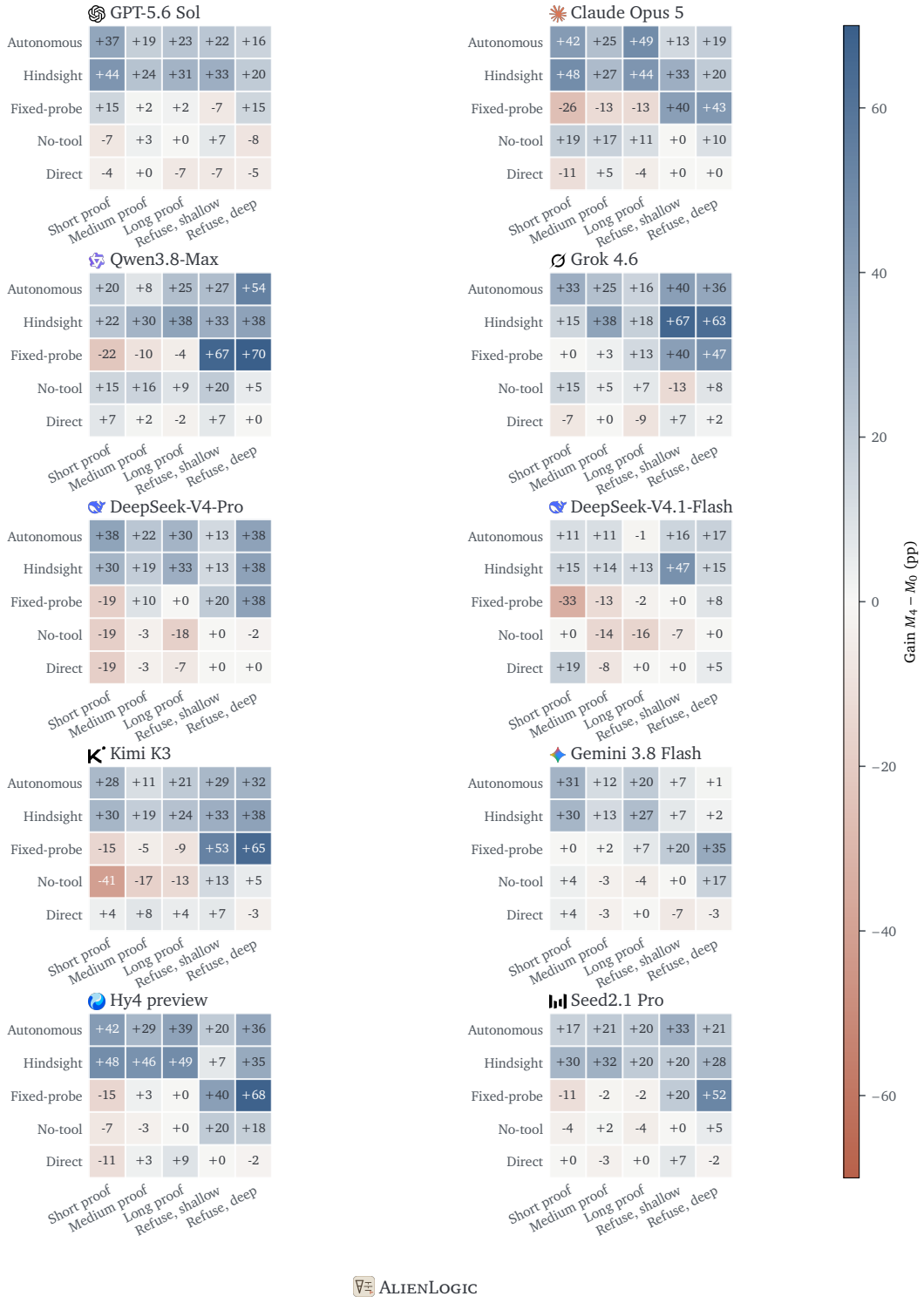


Figure 12: **Task-family gains for every AlienLogic system.** Cells show $M_4 - M_0$ within each task family, in percentage points. Columns are the five families of fig. 2: provable theorems by proof length, unprovable ones by how deeply the goal nests. Autonomous rows average three trajectories; control rows are one trajectory each.

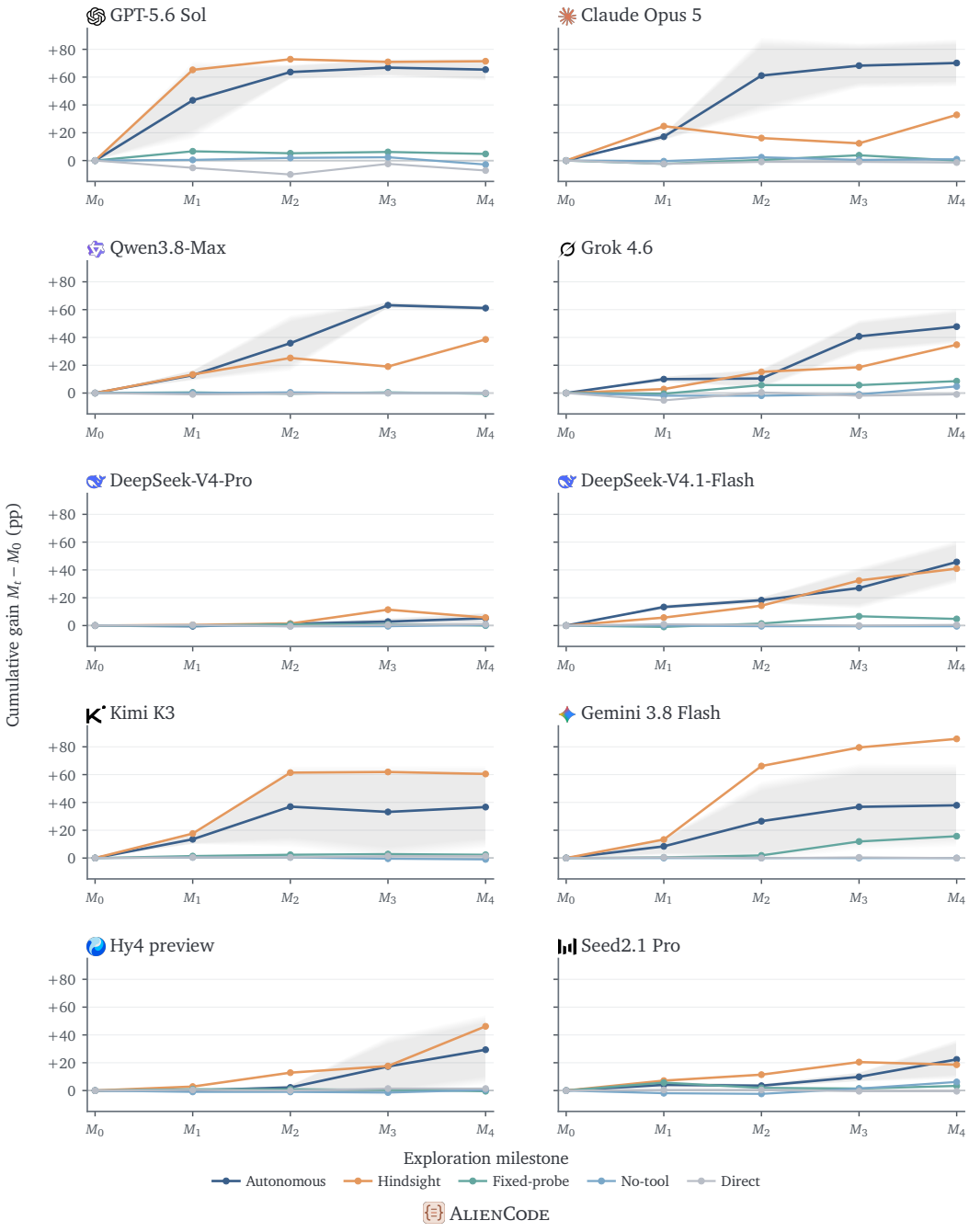


Figure 13: **Exploration trajectories for every AlienCode system.** Curves show cumulative gain $M_t - M_0$. The autonomous curve is the mean of three trajectories, and its shading is $\pm$ population SD across them; each control curve is a single trajectory.

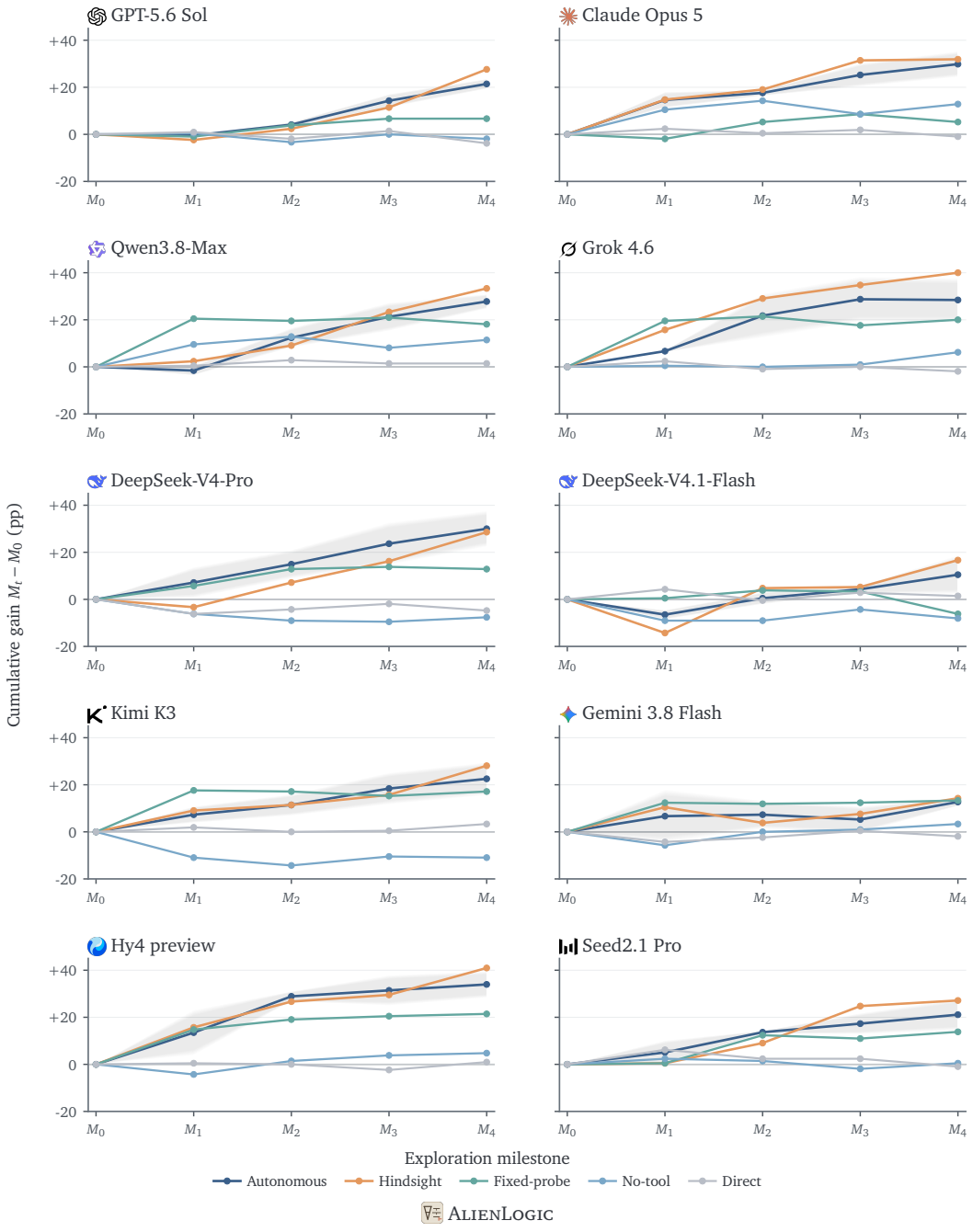


Figure 14: **Exploration trajectories for every AlienLogic system.** Curves show cumulative gain $M_t - M_0$. The autonomous curve is the mean of three trajectories, and its shading is $\pm$ population SD across them; each control curve is a single trajectory.

H CASE STUDIES

Each case follows one trajectory to show a mechanism behind a finding; the cases are illustrations, not further evidence. Programs, proofs, and rule reports are shown as logged. Explanations the systems wrote in Chinese, the language of the prompts, are translated.

Case 1 · One round finds the keystone rules

GPT-5.6 Sol · ALIENCODE

ANSWERING MODEL(S)	TASK / RECORD	EVIDENCE	INTERACTION
GPT-5.6 Sol (max)	Trajectory 2, round 1, call 5	Rule reports and held-out accuracy at M_0 and M_1	12 tool calls in round 1

Probe as submitted, with the sandbox output (two further lines omitted)

```
SET xs AS STRAND(31,24,25,26)
EMIT(xs)                              [1, 2, 3, 4]
EMIT(STRAND(PLUCK(xs,27),PLUCK(xs,26),PLUCK(xs,25),PLUCK(xs,-28)))
                                      [3, 2, 1, 4]
EMIT(CARVE(xs,26,31))                 [1, 2, 3]
```

The same probe read with the two rules the system already reported at M_0

Integer literals are XOR-ed with 27 (R01) and `STRAND` reverses its arguments (R16), so `xs` is `[1, 2, 3, 4]` and the calls ask for `PLUCK` at indices 0, 1, 2, and -1 and for `CARVE(xs, 1, 4)`.

Call	Manual predicts	Sandbox returns
`PLUCK` at 0, 1, 2, -1	`1, 2, 3, 4`	`4, 1, 2, 3`
`CARVE(xs, 1, 4)`	`[2, 3, 4]`	`[1, 2, 3]`

Rule report

	M_0 (8 of 31 rules correct)	M_1 (28 of 31 rules correct)
R14	`UNKNOWN`	`(get seq (- i 1))`
R15	`UNKNOWN`	`(slice seq (- lo 1) (- hi 1))`

Every value the probe returns differs from the manual's, so one call exposes both keystone shifts. After this round the report states 28 of the 31 rules, and held-out accuracy rises from 11.0% to 81.9%; on the 51 tasks that use R14 or R15 it rises from 9.8% to 80.4%.

Case 2 · Same system, same budget, one scope rule apart

Kimi K3 · ALIENCODE

ANSWERING MODEL(S)	TASK / RECORD	EVIDENCE	INTERACTION
Kimi K3 (max)	Trajectories 2 and 3; task A46 at M_4	Rule reports, held-out accuracy, M_4 programs	48 and 45 tool calls

Rules stated correctly (of 31) and held-out accuracy

	Trajectory 2		Trajectory 3	
M_0	7	1.4%	3	2.4%
M_1	10	12.9%	12	12.4%
M_2	14	15.2%	27	81.0%
M_3	19	7.6%	28	82.9%
M_4	22	5.7%	28	79.0%

R14 (`PLUCK` index shift) is stated from M_4 in trajectory 2 and from M_2 in trajectory 3. R01+ (integer literals inside a function body are XOR-ed with 53; 69 of the 70 tasks need it) is stated only at M_0 in trajectory 2 and from M_3 in trajectory 3.

Task A46 at M_4: write `CRAFT last(lst)` that returns the last element

Trajectory 2	`DELIVER PLUCK(lst, 27)`	FAIL `IndexError` on every test
Trajectory 3	`DELIVER PLUCK(lst, 53)`	PASS

Trajectory 2 explains its answer: "`PLUCK(seq, i)` is `seq[i-1]`, so $i = 0$ returns the last element; integer literals are XOR-ed with 27, so 0 is written 27."

Both trajectories state the index shift at M_4, and trajectory 2 states 22 rules. It encodes the literal with the top-level rule inside a function body, where 27 becomes 46, so the index runs out of range. On the 51 tasks that use R14 or R15, trajectory 2 solves 2.6% at M_4 and trajectory 3 75.8%. The report also lags behind use: trajectory 3 reaches 81.0% at M_2 before it states R01+.

Case 3 · A correct refusal overwritten by a later belief

Gemini 3.8 Flash · ALIENLOGIC

ANSWERING MODEL(S)	TASK / RECORD	EVIDENCE	INTERACTION
Gemini 3.8 Flash (high)	Trajectory 2 (Best@3); theorem H13	Answers at M_1 and M_2, rule reports, held-out accuracy	Four exploration rounds

Theorem H13, designated unprovable: a patched rule lets each premise be cited only once

Premises `n`, `IMPL(n, o)`, `IMPL(n, q)`; goal `AND(o, q)`.

Answers

M_1, correct in all three answers PASS: "`ANSWER: UNPROVABLE`. K28 forbids citing a premise more than once, and deriving `AND(o, q)` needs the premise `n` for both implications."

M_2, wrong in all three answers, as at M_3 and M_4 FAIL:

```
1.  n            [premise]      5.  o           [IMPL_E 2, 4]
2.  IMPL(n, o)   [premise]      6.  q           [IMPL_E 3, 4]
3.  IMPL(n, q)   [premise]      7.  AND(o, q)   [AND_I 5, 6]
4.  n            [reit 1]
```

The rule report's entry on premise use

- M_0 "A premise line cannot be directly cited more than once without explicitly reiterating it via reit."
- M_2 "Each declared top-level premise can be directly referenced at most once across the entire proof script."
- M_4 "A declared premise can only be directly referenced at most once (can be bypassed by copying via a single reit into a reusable derived line)."

At M_1 the system refuses correctly. From M_2 it routes around the restriction by reiterating the premise, which the checker rejects, and by M_4 its report states the workaround as a rule. Accuracy falls from 63.8% at M_1 to 52.9% at M_2: six theorems answered correctly in all three answers at M_1 fail in all three at M_2, and two move the other way.

Case 4 · Thinking without evidence withdraws what the examples showed

Kimi K3 · ALIENCODE

ANSWERING MODEL(S)	TASK / RECORD	EVIDENCE	INTERACTION
Kimi K3 (max)	Without-tool answering	Rule reports at M_0 to M_4; round-4 turn	Four rounds of model turns, no tool calls

Rule report

Rules stated correctly: 16, 3, 7, 3, and 1 at M_0 through M_4; entries marked `UNKNOWN` rise from 9 to 27. Held-out accuracy moves from 1.4% to 0.5%.

	M_0	M_4
R01	`(^ n 27)`	`UNKNOWN`
R01+	`(^ n 53)`	`UNKNOWN`
R22	`(+ (len seq) 1)`	`UNKNOWN`

Round-4 turn (excerpt, translated)

"No new evidence; the posterior is not updated relative to the last round. … Integer values are unstable: 32 to 59, 9 to 60, list elements 4/5 to 48/49, length 3 to 4; no reliable uniform offset."

The observations it cites are the ones its M_0 report explained: 32 to 59 is 32 XOR 27, 9 to 60 is 9 XOR 53, and a length of 3 read as 4 is R22. With no new evidence, the extra turns make the report more cautious rather than more accurate, and the worked examples still in context stop being trusted.