TRIAL: BENCHMARKING TEMPORAL RULE INFERENCE THROUGH ACTIVE LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Agents that work on long-horizon tasks often have to learn how an unfamiliar system works by experimenting, and decide when their evidence is enough to act on. We introduce TRIAL (Temporal Rule Inference through Active Learning), a benchmark in which an agent identifies a hidden temporal rule, a deterministic finite automaton (DFA), by building experiments in a procedurally generated game world. An exact verifier tracks which rules remain consistent with the agent's evidence, so TRIAL scores whether an answer is correct and also whether the evidence supports it. We show that frontier agents guess before their evidence settles the rule. When the rule space is large, they exhaust their budget first, and all 88 wrong answers in our main evaluation came within the last 27 of 240 decisions while a median of 260 candidate rules were still possible. When the rule space is small, they commit early with budget to spare, and halving the budget makes these guesses more common. Exact-answer scoring hides the guesses that happen to be right, such as the 29% of Opus's correct answers with code that were made while other rules remained. With code, Astra and Opus write programs that enumerate the candidate rules, yet no system identifies a rule on the hardest condition. Without code, the agents remove more uncertainty than a greedy information-gain controller, but the uncertainty they report drifts away from what the evidence supports.

1 INTRODUCTION

Agents are increasingly deployed on long-horizon work in environments whose rules are not fully specified (Casper et al., 2025; Kwa et al., 2025). The problem is sharpest in scientific research, where systems such as the Virtual Lab and AI co-scientist already help design studies and generate hypotheses (Swanson et al., 2025; Gottweis et al., 2025). An agent doing this work has to decide which uncertainties to investigate and when its evidence justifies a conclusion, with limited time and a limited experimental budget. We therefore ask: **can agents discover how an unfamiliar environment works through self-directed experimentation, and recognize when their evidence supports a conclusion?**

Task-completion scores say little about this ability. An agent can reach a correct answer by guessing among unexplored plausible explanations. Conversely, an agent that fails may still have been able to run informative experiments before its budget ran out or before it stopped too early. Evaluating discovery requires tracking what the evidence establishes throughout an episode, including how efficiently the agent's actions separate competing explanations and whether its final answer follows from its observations.

We introduce TRIAL (Temporal Rule Inference through Active Learning), a benchmark that measures information acquisition and exploration capabilities in a controlled setting (Figure 1). TRIAL uses a dungeon game as a small proxy for experimental exploration. The agent walks between rooms, gathers and fuses materials, and places offerings on a shrine, which reports how many parts of a hidden rule the actions so far satisfy. Finding the full rule requires exploration and exploitation, the same skills that experimental research needs. The agent has to come up with hypotheses, design tests, collect observations, and it has to then decide if the results are conclusive. The hidden rule is a set of temporal patterns over the sequence of offerings, compiled into a deterministic finite automaton (DFA). Moving and handling items are epsilon moves and leave the automaton unchanged, so every

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.

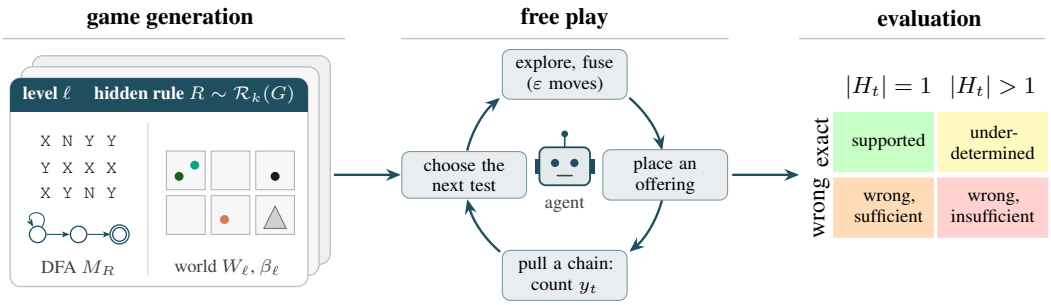


Figure 1: TRIAL agents experiment across changing worlds to identify a persistent hidden rule. Evaluation distinguishes a correct answer from one uniquely supported by the collected evidence.

experiment requires some planning in the world before it can run. The rule stays fixed across different levels simulating real world scenarios where different tasks have the same underlying dynamics.

Because TRIAL discloses the finite set of candidate rules, we use an exact verifier to recompute the rules consistent with the agent's observations after every decision. The verifier shows whether an agent committed while other rules still fit its evidence, and how much uncertainty its experiments removed in games it lost. Agents also report their own uncertainty during play, which we compare with the uncertainty that remains according to the verifier.

We evaluate three frontier agent systems, GPT-6 Astra, Claude Opus 5.5 and Grok 4.7, on four difficulty conditions with full tool access and with an instruction not to use code. Without code, all three agents explore more and remove more initial uncertainty than a greedy controller maximizing expected information gain using the exact posterior (Section 4.3). With code, Astra and Opus write programs that enumerate the candidate rules, yet all are unable to identify a rule on the hardest condition. Without code, Opus and Astra identify 2 and 1 rules there, which suggests that choosing and building experiments limits performance more than inference over the evidence (Section 4.1). Exact-answer scoring also overstates identification. For Opus with code, 29% of its correct answers came before its evidence had settled the rule (Section 4.2). Wrong answers come at the end of the budget, when a median of 260 rules still fit the evidence, so the agents' main shortfall is how quickly their experiments narrow the rule space.

Contributions

- TRIAL, a procedurally generated benchmark in which agents identify hidden temporal rules through experiments built in a game world. Its difficulty dials set the size of the rule space and the cost of an experiment, and an exact verifier recomputes the consistent rule set after every decision (§3).
- An evaluation of three frontier agent systems with and without code execution and file memory. It reports how much uncertainty the agents' experiments remove and whether their committed answers are supported by their evidence (§4).

2 Related Work

Interactive discovery benchmarks Recent benchmarks evaluate discovery, where the agent must gather its own evidence about an environment (Jansen et al., 2024; Chen et al., 2025; Zhou et al., 2025; Gandhi et al., 2025; ARC Prize Foundation, 2026), and OneLife (Khan et al., 2026) infers symbolic world models from unguided exploration. Other work asks language models to infer a hidden rule or black-box function from test inputs they pose directly (Piriyakulkij et al., 2024; Banatt et al., 2024; He et al., 2025; Geng et al., 2025) or from examples they are given (Qiu et al., 2024; Wang et al., 2024; Hua et al., 2025). DiG-bench (Battleday et al., 2026) asks agents to discover hidden transformation rules and win conditions across a suite of games and scores them by whether they win. BeliefTrack (Xu et al., 2026) uses symbolic verifiers to check, turn by turn, whether a model updates or keeps its beliefs correctly in rule-discovery and circuit-diagnosis tasks. In these

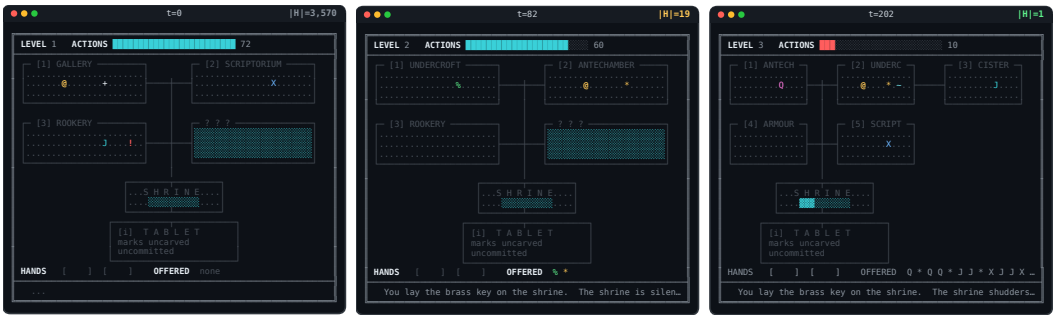


Figure 2: Three replayed frames from one recorded game: the first level, the second level, and the third level before commitment. Levels rebuild the map and reassign materials while keeping the rule fixed. `OFFERED` lists past offerings. The badge shows the verifier's candidate count $|H|$ and is hidden from the agent.

settings a test is posed at no cost beyond the query itself, while in TRIAL each experiment has to be built in the world before it can be run.

Temporal and causal reasoning Causal reasoning benchmarks give the model the relevant facts about a system and ask fixed questions about it (Jin et al., 2023; 2024). TempoBench (Holzer et al., 2025) poses temporal causal reasoning over the traces of deterministic automata: frontier models reach 96% step accuracy when simulating a system forward but 32% when identifying the inputs necessary for an observed output, which its authors read as execution without causal attribution. TRIAL shares this automaton-and-trace formalism and asks the attribution question actively: the agent is given neither the automaton nor a trace, and must produce the inputs through its own actions and attribute the observed counts to a hidden rule. CausaLab (Yang et al., 2026) has agents intervene on a synthetic system to recover its structural causal model and finds that high task accuracy can coexist with a poorly recovered mechanism, a gap similar to the one between exact and supported answers in TRIAL. CausalGame (Chen et al., 2026) is the closest interactive setting, where agents design experimental protocols against a hidden structural causal model under a budget and an LLM judge grades each session against causal reasoning rubrics. TRIAL instead recomputes the set of rules consistent with the agent's evidence after each observation, so whether the evidence supports the answer is checked without a rubric or judge that may be variable or wrong (Zheng et al., 2023).

Agentic automata learning Menaged et al. (2026) have agents identify a hidden minimal DFA through membership queries to an exact oracle and equivalence queries, which return a confirmation or a counterexample for a candidate DFA, and find that agents are far less efficient than classical exact algorithms (Angluin, 1987; Isberner et al., 2014). Two assumptions of this setup are not met in environments that agents act inside. A query there costs one unit of budget, while in a world an experiment is produced by a sequence of actions whose cost depends on object placement and action history. Without the equivalence oracle, the agent must also judge from its own records whether the evidence determines a single answer, and since its commitment is irreversible, committing early is a cost it must manage, as in sequential testing (Wald, 1945). They list weaker feedback as future work, which TRIAL partially takes up by removing the teacher.

3 TRIAL Benchmark

In each TRIAL episode, an agent constructs offering sequences and observes partial feedback to identify a hidden temporal rule. The rule persists across levels, while the world and material-role assignments change. Opening a shrine completes a level. Solving the episode requires naming the rule in a single final commitment. The agent must therefore choose informative experiments and decide when its evidence is sufficient within a limited interaction budget.

3.1 BENCHMARK CONSTRUCTION

One TRIAL episode comprises dungeon levels built around a common hidden rule. An agent explores the dungeon's rooms, attempts to gather and fuse materials, and places offerings on a shrine that responds to the input sequence. The agent experiments with the materials available, reads the shrine's response, and decides which hypotheses to test, when to abandon a level and when to name the rule. Figure 2 shows three frames of a recorded game, and Figure 8 in Appendix A shows the full game. Formally, an instance is generated from $(G, k, b_{\mathrm{lvl}}, b_{\mathrm{ep}}, \xi)$: a rule grammar G, a rule size k, two budgets and a seed. The generator samples the hidden rule first and builds every level around it, so the ground truth is established before the world.

Rules Offerings are observed through the input alphabet $\Sigma = \{X, Y, N, B\}$, defined below. A grammar G defines a finite catalog $V(G)$ of readings, each a regular language over Σ with minimal DFA $A_r = (S_r, s_{0,r}, \Sigma, \delta_r, F_r)$. The hidden rule is a set $R \subseteq V(G)$ of k readings, $k = 3$ in our experiments, drawn uniformly from $\mathcal{R}_k(G)$, the canonical minimal rules: no reading in R is implied by the others, and of each pair of rules related by exchanging X and Y, only one is kept. The rule is the product DFA $M_R = (S, s_0, \Sigma, \delta, F)$ of its readings, with $S = \prod_{r \in R} S_r$, componentwise δ and $F = \prod_{r \in R} F_r$: it accepts, and the shrine opens, once every reading holds. The chains report a count read off its state,

$$c(s) = \big|\{r \in R : s_r \in F_r\}\big| \in \{0, \ldots, k\},$$

the number of readings that hold.

Temporal grammars In the main experiments every reading is a pattern p that must occur as a contiguous run, $L(A_p) = \Sigma^* p \, \Sigma^*$. With $a \in \{X, Y\}$, $\bar{a}$ the other role, and $u, v \in \{X, Y, N\}$, the patterns of length four are $a\,u\,v\,e$ and those of length six are $a\,u\,v\,u\,v\,e$. The smaller catalog fixes $e = \bar{a}$. The larger one also allows $e = a$, which adds near-miss patterns: each shares every symbol but the last with a pattern of the smaller catalog, so only a complete run tells the two apart. Since no pattern contains B, an offering with both roles breaks every run in progress. Readings accept once their pattern appears and stay accepting, so the count never decreases within a level.[1]

Difficulty dials Catalog size and pattern length change different things. Catalog size sets how much must be learned: $|\mathcal{R}_k(G)|$ is 408 or 3,570, 8.7 or 11.8 bits, at either pattern length. Pattern length sets what an experiment costs: the shortest input that opens the shrine has a median length of 9 or 15 symbols and the minimal DFA of the rule has a median of 42 or 66 states, for either catalog. Both quantities are computed from the construction alone and extend past the settings we test (Appendix H).

Levels At level ℓ, the generator samples a world W_ℓ with rooms, materials $\mathrm{Mat}(W_\ell)$, guards and fonts. A fresh injective binding $\beta_\ell : \{x, y\} \to \mathrm{Mat}(W_\ell)$ assigns the hidden roles to distinct materials. Each level resets M_R to s_0, with the rule R unchanged. An offering o contains at most four material units. Its symbol $\sigma_{\beta_\ell}(o) \in \Sigma$ is X if it contains $\beta_\ell(x)$ but not $\beta_\ell(y)$, Y for the reverse, N for neither and B for both. Placing $o_1, \ldots, o_t$ feeds the input $x_{1:t}$, where $x_i = \sigma_{\beta_\ell}(o_i)$, into M_R. The resulting states $s_1, \ldots, s_t$ give counts $y_i = c(s_i)$ and the trace $\pi_\ell = ((x_1, y_1), \ldots, (x_t, y_t))$. Movement and item handling change the world while preserving the automaton state. These actions are ε moves in the combined level, represented as an ε-NFA over Σ. Projecting a realized play onto its offering symbols gives a run of M_R. The world constrains which input sequences can be executed. The generator rejects levels with no reachable accepting state, ignoring budgets in this check.

Free play, budgets and commitment Agent play is self-determined in each level. The agent observes materials and the count returned after each placement. The binding β_ℓ and input symbols x_i remain hidden. The near chain returns the current count y_t, and the far chain reports the count for a hypothetical placement of the items in hand. The agent can gather or fuse materials for its experiments. It may keep notes and can abandon the level for a new world. World actions consume the per-level budget b_{lvl}. Guards can incur additional costs unless answered, and fonts refund actions. A level ends when the shrine opens or this budget is exhausted. Each submitted game action, including

[1] A base grammar of 14 readings about content, fusion, counts and order is used only in the budget panel of Appendix I.1.

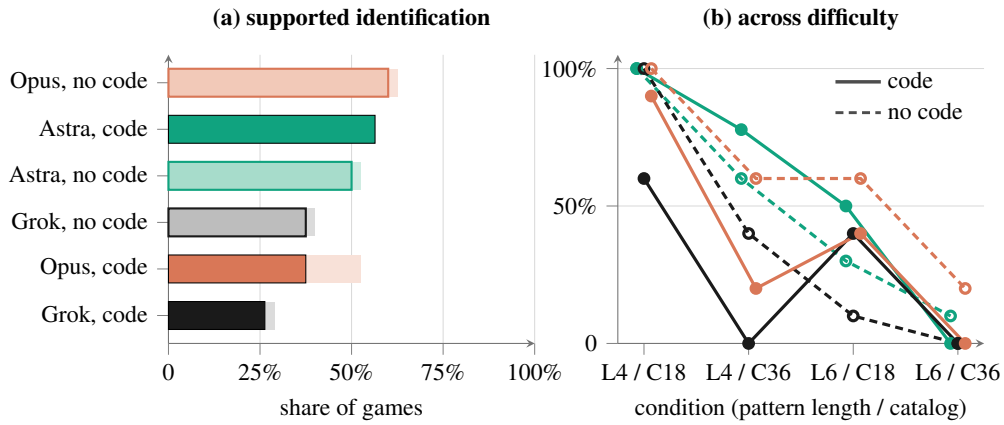


Figure 3: Main cohort, both starts. (a) Supported identification by system and tool condition. Pale extensions add exact but underdetermined answers. (b) Results by condition. Code is shown by filled bars and solid lines, no code by outlines and dashed lines.

malformed actions and memory calls, consumes one of b_{ep} episode decisions. The agent may submit its answer $\hat{A} \subseteq V(G)$ once, ending the episode. Agents receive $V(G)$ with the uniform prior over $\mathcal{R}_k(G)$ and the role-exchange convention. The instructions specify σ and chain feedback. Agents choose their experiments and decide when to commit. The disclosed hypothesis class also defines the verifier's candidate set. Levels are generated on demand, with pattern length, catalog size, k and world parameters set by the generator configuration.

3.2 Evaluation Protocol

Because the verifier knows which rules the evidence still admits after every decision, we score the final answer, the evidence the agent collected and the trajectory through the game world.

Classifying episodes At the moment of commitment we make two checks. The first is whether $\hat{A} = R$, in which case the answer is exact. The second is whether any rule other than R was still consistent with the agent's observations. If none was, the evidence had identified R. An exact answer given after the evidence identified R is a supported identification, and one given while other rules were still consistent is exact but underdetermined. This label describes the public record and is not a claim that the agent guessed. Combining the two checks gives four outcomes. Episodes that reach the end of the budget without a commitment are counted separately.

- **Exact, supported.** The answer equals R, and no other candidate was still consistent.
- **Exact, underdetermined.** The answer equals R while other candidates were still consistent.
- **Wrong, sufficient evidence.** Only R was still consistent, but the agent committed a different answer, which Menaged et al. (2026) would count as a reasoning failure.
- **Wrong, insufficient evidence.** The answer is wrong, and the evidence had not narrowed the candidates to one.

Our results mainly count supported identifications. Their difference from the exact answers counts the underdetermined ones. An agent may reasonably guess when it has too little budget left to test further. An underdetermined answer therefore does not by itself show that the agent should have continued.

Deterministic verifier After each decision the verifier computes the consistent set $H_t \subseteq \mathcal{R}_k(G)$ by enumeration: a candidate rule R' stays in H_t only if, for every level played so far, some binding of the two roles to that level's materials makes $M_{R'}$ reproduce every count y_i the agent observed on it. Because the hypothesis class is disclosed and finite, with at most 3,570 rules and a handful of bindings per level, this enumeration is exact and cheap. Menaged et al. (2026) must instead approximate

Agent response (excerpts)

I won the run: the tablet accepted my answer. It was partly luck, though. When I named the rite, 8 answers still fit the evidence equally well, so I gave it a 1-in-8 chance.

[...]

The guess: with 20 run turns left and too few level actions in that level for another test, I named one of the 8 remaining answers.

[...]

Figure 4: Opus knowingly guesses correctly while eight candidates remain (length-six, smaller catalog, code, cold start). Verbatim final-response excerpts with omissions marked by ellipses.

whether evidence sufficed with a passive learner, and only for failed episodes. Our check is exact and covers correct answers as well. The verifier reads only public observations and never the hidden rule, so replaying a stored episode gives the same result. As a sanity check, the harness records at each commitment whether $R \in H_t$, and evidence collection rejects any episode where it is not.

Evidence acquisition Because H_t is known after every decision, progress can be measured in games that end without a supported identification. We report the share of the initial log support that the evidence has removed, $1 - \log_2 |H_t| / \log_2 |H_0|$, the first decision at which $|H_t| = 1$, and the capped cost of a game: the decisions used for a supported identification, and the full horizon for any other outcome.

Information-state reports Before each evidence-producing action, agents report their current answer entropy and expected entropy after feedback, in bits. They submit an updated report after observing the feedback. At commitment, they report separate probabilities for answer correctness and uniqueness given the evidence. Agents with tools may compute these values. Missing reports reduce coverage and are not imputed. Commitment probabilities are evaluated with the Brier score (Brier, 1950).

Harness and tool access Each agent runs in its native command-line harness inside an isolated container. The referee's game implementation and hidden state, including the seed, remain outside this container. General network access is disabled. Native tools include shell and file access, and agents can invoke the public game interface from the shell. In the no-code condition, instructions prohibit writing or running code that analyzes or solves the game. Helpers may send game commands or format responses. We audit recorded commands for compliance where traces are available (Appendix C).

Budget arms In the five-system panel of Appendix I.1, each configuration runs under a standard arm and two tighter arms, one halving b_{lvl} and one halving b_{ep} (Table 8 in Appendix I.1).

4 Results

We evaluate GPT-6 Astra through Codex CLI 0.153.4, Claude Opus 5.5 through Claude Code and Grok 4.7 through Grok Build, all at medium reasoning effort, and refer to them as Astra, Opus and Grok. Each system plays the 20 target rules, 5 per difficulty condition (Section 3.1), once from a cold start and once with file memory. It plays them with native tool access and again under an instruction not to write code that analyzes or solves the game (Section 3.2). Every scored game is replay-validated, and runs interrupted by provider quotas were rerun (Appendix C.1).

4.1 Identifying the Hidden Rule

Figure 3 gives the share of games that end in a supported identification. Without code, every system identifies the rule in all 10 of 10 games on the easiest condition, which uses length-four patterns and the smaller catalog. On the hardest condition, with length-six patterns and the larger catalog, no

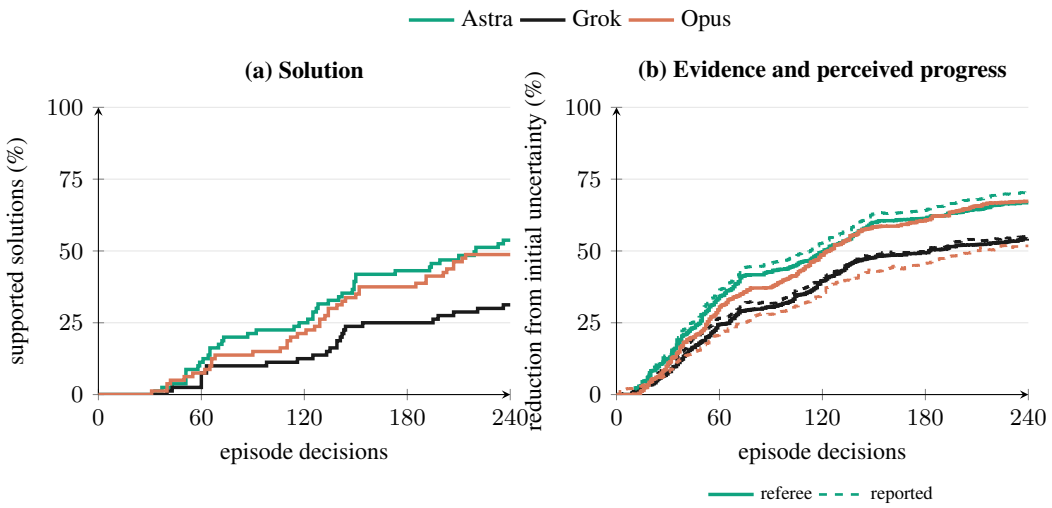


Figure 5: Main-cohort progress. Left: supported identifications. Right: reductions in referee log support (solid) and reported entropy (dashed), relative to initial log support. Conditions receive equal weight. Appendix C.1 gives aggregation details.

system identifies a rule with code. Without code, Opus identifies 2 rules there and Astra 1, so the budget allows identification on every condition for at least some targets.

Code access does not remove this difficulty. With code, Astra and Opus write programs that enumerate the candidate rules. The command audit finds such programs in 36 of 39 Astra games and 39 of 40 Opus games, and in none of their no-code games (Appendix C). Grok Build returns only its final answer, so Grok's solver use is unobserved. A correct enumeration program gives the agent the same consistent set that our verifier computes. What remains is to choose and build experiments that shrink this set, and on the hardest condition no system did so within the budget.

Opus identifies more rules under the no-code instruction, 24 against 15 with code. In cold-start games paired by target, Opus identifies the rule only without code in 5 pairs and only with code in 0 (exact McNemar test, $p = 0.0625$), and no system shows a significant paired difference. We report the pooled counts descriptively, since memory games reuse targets and carry history from one game to the next. Sol and Luna, run on the same protocol as Astra with code, identify fewer rules than Astra, and Luna identifies none (Appendix G).

4.2 Exact Answers and Supported Identifications

Exact-answer scoring overstates identification most for Opus with code. It named the correct rule while other rules were still consistent in 6 of its 21 exact answers (29%), which raises its rank under exact scoring. In every other system and tool condition the two counts differ by at most 1 answer. In one of these games, Opus states that eight rules still fit its evidence and describes its answer as a guess forced by the remaining budget (Figure 4).

Table 1: Wrong commitments by membership in H_t, pooling both starts and all difficulties. Counts are code / no code.

System	Outside H_t	Within H_t
Astra	0 / 0	17 / 19
Grok	0 / 0	14 / 7
Opus	0 / 2	17 / 12

No system commits a wrong answer after its evidence has identified the rule, the failure that Menaged et al. (2026) classify as a reasoning error. Astra commits in every game, and all of its failures are wrong answers on insufficient evidence (17 with code). Grok often reaches the decision horizon without committing, in 13 games with code and 17 without. Of the 88 wrong commitments, 86 are still in H_t when the agent commits (Table 1). The other 2, both from Opus without code, fall outside it (Appendix D).

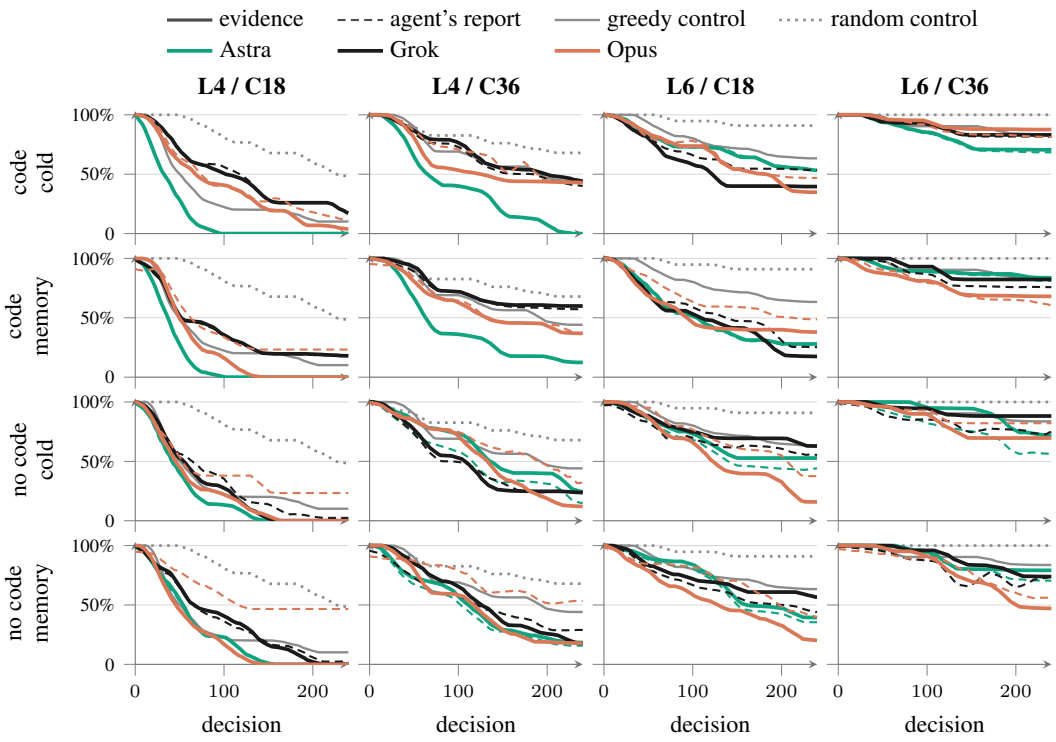


Figure 6: Main-cohort log support, $\log_2 |H_t| / \log_2 |H_0|$, and reported uncertainty by difficulty, code access and start (at most 5 games per cell). Solid/dashed colored lines show evidence/reports. Gray lines show the same scripted controls in every row. Curves are smoothed over 20 decisions.

The timing of the commitments shows how these failures arise. Wrong answers come at the end of the game, with a median of 2.5 of 240 decisions left and never more than 27. At that point a median of 260 rules still fit the evidence, and 74 of the 88 wrong answers leave at least ten. Underdetermined correct answers also come late (median 7.5 decisions left), while supported identifications leave a median of 112.5 unused. Agents that fail therefore spend nearly their whole budget without narrowing the rule space enough and then name one of the remaining candidates, as Opus does in Figure 4.

The budget panel in Appendix I.1 shows a second pattern. Its base grammar has far fewer candidate rules, and there agents often commit early. In the standard arm, 77% of the wrong and underdetermined answers come with at least a quarter of the budget unused, and wrong answers leave a median of 94 decisions and 3 consistent rules. Tighter budgets increase these guesses. Underdetermined correct answers rise from 12 in the standard arm to 19 when the per-level budget is halved and 17 when the episode budget is halved.

4.3 Evidence Acquisition

The consistent set also measures progress in games that end without a supported identification. We compare the agents with two scripted controls that play the same targets from the same public observations and track the exact posterior over rules and bindings. Among feasible experiments built from catalog patterns, the greedy controller picks the one with the highest expected information gain per action, and the random controller picks one at random. Both search a restricted set of experiments, so they serve as reference points and do not bound the optimal policy. Figure 5 pools the main cohort over conditions, and Figure 6 breaks it down by condition, tool access and start.

Without code, all three systems remove more of the initial log support than the greedy controller, which reaches 50% by the horizon, and far more than the random controller (23%). Opus removes the most, 77%. With code, Astra removes 68% and Grok 58%, close to the greedy controller.

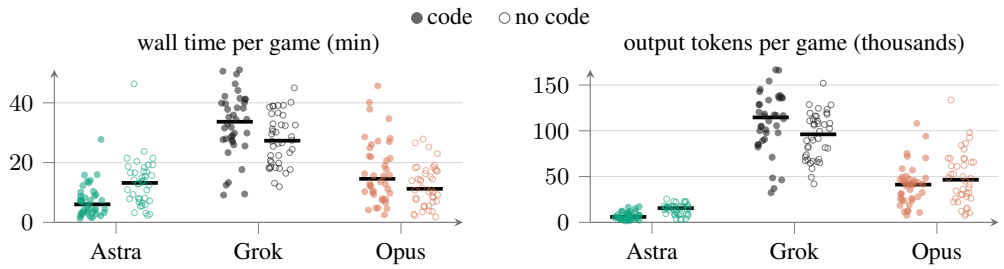


Figure 7: Main-cohort inference cost, both starts. Each point is a game: code (filled) or no code (hollow). Horizontal bars mark medians. Left: wall time, including provider latency. Right: output tokens, including reasoning. Horizontal jitter separates overlapping points. Appendix F gives further inference diagnostics.

4.4 Reported Uncertainty

Agents report their answer entropy after each observation, which we compare with the log support $\log_2 |H_t|$ computed by the verifier. With code, the mean signed gap between the two is small: -0.05 bits for Astra, +0.08 for Opus and -0.17 for Grok. Without code the gaps grow to -0.78, +0.75 and -0.54 bits (Figure 6). Opus's reports then exceed the log support, which already bounds the posterior entropy from above, and Astra's and Grok's fall below it. Positive and negative errors can cancel in a mean, so these gaps show the direction of the error more reliably than its size (Appendix C.1).

Predicted information gain shows a similar split (Appendix Figure 9). With code, Astra's predictions of what an action will reveal track the referee's expected gain. Opus and Grok overpredict in the higher bins, and all three systems overpredict without code.

4.5 File Memory and Inference Cost

In the file-memory arm, each game starts with the files and post-game reflection of the preceding game in the same condition. With code, memory moves identification earlier. Among games whose evidence identifies the rule, the median identification decision falls from 86 to 69 for Astra, from 134 to 99 for Opus and from 137 to 59 for Grok, and the handoffs mostly carry enumeration programs from earlier games (Appendix D). Without code the medians show no consistent shift. Each memory stream follows one game order, so these medians describe our runs and do not estimate a causal effect.

Inference cost Figure 7 shows per-game wall time and output tokens. Without code, Opus finishes games faster while writing more output, and Astra takes longer and writes more. Grok writes the most output in both conditions and has the fewest supported identifications (Appendix F).

5 Limitations

The main cohort uses 5 target rules per condition and one game order per memory stream, so we test paired differences on cold starts and report the other comparisons descriptively. Because the generator produces new targets on demand, larger cohorts need only more provider budget. Agents also receive the catalog of readings and the prior over rules, which makes the verifier exact and scopes our results to identification within a disclosed rule family.

6 Conclusion

TRIAL measures active rule discovery in generated worlds and separates correct answers from evidence-supported identifications. In our evaluation, agents often removed much of the initial log support without identifying the rule, and Opus with code gave 6 correct answers while other rules were still consistent with its evidence.

AI USE STATEMENT

Language-model assistants were used in code development and in drafting and editing the manuscript. Reported outcomes and numerical summaries are computed from stored experimental evidence and checked by deterministic replay and automated tests.

ETHICS STATEMENT

The benchmark uses procedurally generated worlds and synthetic rules, without human participant data. Agents act in isolated containers with no general network access.

REPRODUCIBILITY STATEMENT

We are working to open source the full code and dataset required to run TRIAL on other agents. Appendix C describes isolation and validity checks, and Appendix F documents provider-specific inference measurements.

REFERENCES

Dana Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75(2):87–106, 1987.

ARC Prize Foundation. ARC-AGI-3: A new challenge for frontier agentic intelligence. *arXiv preprint arXiv:2603.24621*, 2026.

Eryk Banatt, Jonathan Cheng, Skanda Vaidyanath, and Tiffany Hwu. WILT: A multi-turn, memorization-robust inductive logic benchmark for LLMs. *arXiv preprint arXiv:2410.10998*, 2024.

Ruairidh M. Battleday, Kai Sandbrink, Jimi Cullen-Drohan, Zihan Yan, Timothy Muller, Clare Maguire, Ales Kubicek, Fraser Greenlee-Scott, Sukrit Sumant, Tri Dao, Jürgen Schmidhuber, Michal Valko, Joshua Tenenbaum, Thomas L. Griffiths, Zeb Kurth-Nelson, and James C. R. Whittington. DiG-bench: Discovery in games. *arXiv preprint arXiv:2608.12593*, 2026. URL https://arxiv.org/abs/2608.12593.

Glenn W. Brier. Verification of forecasts expressed in terms of probability. *Monthly Weather Review*, 78(1):1–3, 1950.

Stephen Casper, Luke Bailey, Rosco Hunter, Carson Ezell, Emma Cabalé, Michael Gerovitch, Stewart Slocum, Kevin Wei, Nikola Jurkovic, Ariba Khan, Phillip J. K. Christoffersen, A. Pinar Ozisik, Rakshit Trivedi, Dylan Hadfield-Menell, and Noam Kolt. The AI agent index. *arXiv preprint arXiv:2502.01635*, 2025. URL https://arxiv.org/abs/2502.01635.

Yimeng Chen, Piotr Piękos, Mateusz Ostaszewski, Firas Laakom, and Jürgen Schmidhuber. PhysGym: Benchmarking LLMs in interactive physics discovery with controlled priors. In *Advances in Neural Information Processing Systems*, volume 38. Curran Associates, Inc., 2025. doi: 10.52202/085713-1556. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/42abcf9dffa48ca44fea1c497539a914-Paper-Datasets_and_Benchmarks_Track.pdf.

Zhenhao Chen, Yongqiang Chen, Chenxi Liu, Junchi Yu, Xiangchen Song, Zijian Li, Jialin Li, Philip Torr, Bo Han, and Kun Zhang. CausalGame: Benchmarking causal thinking of LLM agents in games. In *International Conference on Machine Learning*, 2026.

C. J. Clopper and E. S. Pearson. The use of confidence or fiducial limits illustrated in the case of the binomial. *Biometrika*, 26(4):404–413, 1934.

B. Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1), 1979.

Kanishk Gandhi, Michael Y. Li, Lyle Goodyear, Agam Bhatia, Louise Li, Aditi Bhaskar, Mohammed Zaman, and Noah D. Goodman. BoxingGym: Benchmarking progress in automated experimental design and model discovery. *arXiv preprint arXiv:2501.01540*, 2025.

Jiayi Geng, Howard Chen, Dilip Arumugam, and Thomas L. Griffiths. Are large language models reliable AI scientists? assessing reverse-engineering of black-box systems. *arXiv preprint arXiv:2505.17968*, 2025.

Google. Antigravity CLI. https://antigravity.google/docs/cli/overview/, 2026. Accessed 2026-09-24.

Google DeepMind. Gemini 3.8 Flash model card. https://deepmind.google/models/model-cards/gemini-3-8-flash/, 2026. Published 2026-09-02; accessed 2026-09-24.

Juraj Gottweis, Wei-Hung Weng, Alexander Daryin, et al. Towards an AI co-scientist. *arXiv preprint arXiv:2502.18864v1*, 2025. URL https://arxiv.org/abs/2502.18864v1.

Kaiyu He, Mian Zhang, Shuo Yan, Peilin Wu, and Zhiyu Chen. IDEA: Enhancing the rule learning ability of large language model agent through induction, deduction, and abduction. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 13563–13597, 2025.

Nikolaus Holzer, William Fishell, Baishakhi Ray, and Mark Santolucito. TempoBench: Reasoning execution without causal attribution is just simulation. *arXiv preprint arXiv:2510.27544*, 2025. URL https://arxiv.org/abs/2510.27544.

Wenyue Hua, Tyler Wong, Fei Sun, Liangming Pan, Adam Jardine, and William Yang Wang. InductionBench: LLMs fail in the simplest complexity class. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, pp. 26526–26546, 2025.

Malte Isberner, Falk Howar, and Bernhard Steffen. The TTT algorithm: A redundancy-free approach to active automata learning. In *Runtime Verification (RV)*, Lecture Notes in Computer Science, pp. 307–322. Springer, 2014.

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*, volume 37. Curran Associates, Inc., 2024. doi: 10.52202/079017-0324. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/13836f251823945316ae067350a5c366-Paper-Datasets_and_Benchmarks_Track.pdf.

Zhijing Jin, Yuen Chen, Felix Leeb, Luigi Gresele, Ojasv Kamal, Zhiheng Lyu, Kevin Blin, Fernando Gonzalez Adauto, Max Kleiman-Weiner, Mrinmaya Sachan, and Bernhard Schölkopf. CLadder: Assessing causal reasoning in language models. In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2312.04350.

Zhijing Jin, Jiarui Liu, Zhiheng Lyu, Spencer Poff, Mrinmaya Sachan, Rada Mihalcea, Mona Diab, and Bernhard Schölkopf. Can large language models infer causation from correlation? In *The Twelfth International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2306.05836.

Zaid Khan, Archiki Prasad, Elias Stengel-Eskin, Jaemin Cho, and Mohit Bansal. One life to learn: Inferring symbolic world models for stochastic environments from unguided exploration. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/a24d08f2eecc6c96f4a0ad37a74ddeca-Paper-Conference.pdf.

Thomas Kwa, Ben West, Joel Becker, Amy Deng, Katharyn Garcia, Max Hasin, Sami Jawhar, Megan Kinniment, Nate Rush, Sydney Von Arx, Ryan Bloom, Thomas Broadley, Haoxing Du, Brian Goodrich, Nikola Jurkovic, Luke Harold Miles, Seraphina Nix, Tao Lin, Chris Painter, Neev Parikh, David Rein, Lucas Jun Koba Sato, Hjalmar Wijk, Daniel M. Ziegler, Elizabeth Barnes, and Lawrence Chan. Measuring AI ability to complete long software tasks. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2503.14499.

Reef Menaged, Gili Lior, Shauli Ravfogel, Roee Aharoni, and Gabriel Stanovsky. Can LLM agents infer world models? evidence from agentic automata learning. *arXiv preprint arXiv:2606.16576*, 2026.

Meta AI. Introducing Muse Spark 1.3. https://research.meta.ai/blog/introducing-muse-spark-1-3, 2026. Published 2026-09-02; accessed 2026-09-24.

OpenAI. GPT-6 Astra system card. https://deploymentsafety.openai.com/gpt-6-astra, 2026a. Published 2026-09-03; accessed 2026-09-24.

OpenAI. Codex CLI. https://github.com/openai/codex, 2026b. Accessed 2026-09-24.

OpenAI. GPT-5.6 system card. https://deploymentsafety.openai.com/gpt-5-6, 2026c. Published 2026-07-09; accessed 2026-09-24.

OpenCode. OpenCode: The open source coding agent. https://github.com/anomalyco/opencode, 2026. Accessed 2026-09-24.

Wasu Top Piriyakulkij, Cassidy Langenfeld, Tuan Anh Le, and Kevin Ellis. Doing experiments and revising rules with natural language and probabilistic reasoning. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-1682.

Linlu Qiu, Liwei Jiang, Ximing Lu, Melanie Sclar, Valentina Pyatkin, Chandra Bhagavatula, Bailin Wang, Yoon Kim, Yejin Choi, Nouha Dziri, and Xiang Ren. Phenomenal yet puzzling: Testing inductive reasoning capabilities of language models with hypothesis refinement. In *International Conference on Learning Representations*, 2024.

Kyle Swanson, Wesley Wu, Nash L. Bulaong, John E. Pak, and James Zou. The Virtual Lab of AI agents designs new SARS-CoV-2 nanobodies. *Nature*, 646:716–723, 2025. doi: 10.1038/s41586-025-09442-9. URL https://www.nature.com/articles/s41586-025-09442-9.

Abraham Wald. Sequential tests of statistical hypotheses. *The Annals of Mathematical Statistics*, 16 (2):117–186, 1945.

Ruocheng Wang, Eric Zelikman, Gabriel Poesia, Yewen Pu, Nick Haber, and Noah D. Goodman. Hypothesis search: Inductive reasoning with language models. In *International Conference on Learning Representations*, 2024.

Haoming Xu, Weihong Xu, Zongrui Li, Mengru Wang, Yunzhi Yao, Chiyu Wu, Jin Shang, Yu Gong, and Shumin Deng. When should models change their minds? contextual belief management in large language models. *arXiv preprint arXiv:2605.30219*, 2026. URL https://arxiv.org/abs/2605.30219.

Junlin Yang, Dylan Zhang, Xiangchen Song, Qirun Dai, Xiao Liu, Yuen Chen, Aniket Vashishtha, Jing Shi, Chenhao Tan, and Hao Peng. CausaLab: A scalable environment for interactive causal discovery toward AI scientists. *arXiv preprint arXiv:2605.26029*, 2026. URL https://arxiv.org/abs/2605.26029.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems*, volume 36, pp. 46595–46623, 2023. doi: 10.52202/075280-2020.

Zhanke Zhou, Xiao Feng, Zhaocheng Zhu, Jiangchao Yao, Sanmi Koyejo, and Bo Han. From passive to active reasoning: Can large language models ask the right questions under incomplete information? In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 78714–78758. PMLR, 2025. URL https://raw.githubusercontent.com/mlresearch/v267/main/assets/zhou25e/zhou25e.pdf.

A A WORKED EXAMPLE

One level by hand Take one level of the smaller length-four catalog, which has 408 candidate rules (Section 3.1). The level has four materials a, b, c and d. The hidden binding is $\beta(x) = a$ and $\beta(y) = b$, and the hidden rule has the readings $X\,X\,X\,Y$, $X\,X\,Y\,Y$, $X\,N\,N\,Y$. The agent sees neither. It places $\{a\}$, $\{c\}$, $\{d\}$ and $\{b\}$, whose symbols under the hidden binding are $X\,N\,N\,Y$. The count stays at zero until the last placement completes the reading $X\,N\,N\,Y$, and then it reads 1. The verifier keeps a candidate only if one of the 12 assignments of the two roles to distinct materials reproduces all four counts. 128 candidates remain. A second experiment on the same level places $\{a\}$, $\{a\}$, $\{b\}$ and $\{b\}$. The count rises to 2, and 16 candidates remain, including the hidden rule. Committing the hidden rule now would give an exact but underdetermined answer. If the agent continues until H_t contains only the hidden rule, committing it is a supported identification. Committing any other rule at that point would be wrong with sufficient evidence.

A recorded game The game below is played by the scripted greedy controller of Section 4.3 on the first target of the smaller length-four catalog. It uses the same referee as the agents. Each reply is the text an agent would see after the action, and each count $|H_t|$ comes from the referee's evidence log after a placement. The controller is not a language model, so the transcript shows the environment and the verifier without any reasoning. Appendix E quotes the agents. The first level has four rooms, one with a font and one guarded, and a material in each of three of them.

```
  the Undercroft     the brass key   <- you are here
  the Armoury        nothing   a font runs here
  the Scriptorium    the jade key
  the Vestibule      the copper key   [GUARD: the gaunt waits here; obstacle, never an item]
  carrying: nothing
  offered : nothing
```

The shrine reports the number of satisfied readings in words, silent for none, a shudder for one and two hums for two. The first placements complete no pattern and leave all 408 candidates. Once a placement completes a pattern the count rises, and the verifier removes the rules and bindings that could not have produced it. When the level's actions run low the controller abandons the level, and the next level rebuilds the world with a new binding while the rule stays fixed. The evidence identifies the rule at decision 99, after 37 placements, and the controller commits at decision 100 of 240, which the verifier scores as a supported identification. The transcript shows the first 25 decisions in full and afterwards only the placements that change $|H_t|$.

```
dec  action                   |H|  reply
  1  take                          You take the brass key.
  2  place                  408  You lay the brass key on the shrine.  The shrine is silent.
  3  take                          You take the brass key.
  4  place                  408  You lay the brass key on the shrine.  The shrine is silent.
  5  take                          You take the brass key.
  6  place                  408  You lay the brass key on the shrine.  The shrine is silent.
  7  go to the Scriptorium        You are in the Scriptorium.
  8  take                          You take the jade key.
  9  place                  408  You lay the jade key on the shrine.  The shrine is silent.
 10  take                          You take the jade key.
 11  place                  408  You lay the jade key on the shrine.  The shrine is silent.
 12  take                          You take the jade key.
 13  place                  408  You lay the jade key on the shrine.  The shrine is silent.
 14  go to the Undercroft         You are in the Undercroft.
 15  take                          You take the brass key.
 16  place                  408  You lay the brass key on the shrine.  The shrine is silent.
 17  take                          You take the brass key.
 18  place                  408  You lay the brass key on the shrine.  The shrine is silent.
 19  go to the Vestibule          You are in the Vestibule.  The gaunt wounds you before you can
                                  answer it, then is gone. The guard was only an obstacle: nothing
                                  enters your hands. (4 actions lost)
 20  take                          You take the copper key.
 21  place                  408  You lay the copper key on the shrine.  The shrine is silent.
 22  take                          You take the copper key.
 23  place                  128  You lay the copper key on the shrine.  The shrine shudders once.
 24  take                          You take the copper key.
 25  place                   16  You lay the copper key on the shrine.  The shrine hums twice.
...  (from here, only placements that change the consistent set)
 28  place                   15  You lay the brass key on the shrine.  The shrine hums twice.
 30  place                   14  You lay the brass key on the shrine.  The shrine hums twice.
 32  place                   13  You lay the brass key on the shrine.  The shrine hums twice.
 35  place                   12  You lay the copper key on the shrine.  The shrine hums twice.
 41  place                   11  You lay the copper key on the shrine.  The shrine hums twice.
 44  place                   10  You lay the brass key on the shrine.  The shrine hums twice.
 50  place                    9  You lay the brass key on the shrine.  The shrine hums twice.
 53  place                    8  You lay the copper key on the shrine.  The shrine hums twice.
```



Figure 8: Six terminal frames from one recorded TRIAL game, replayed through the environment. Each level rebuilds the map and reassigns the materials, while the hidden rule stays fixed. The `OFFERED` line is the offering sequence that the hidden automaton reads. The badge on each window gives $|H|$, the number of candidate rules still consistent with the agent's public evidence, which reaches one before the agent commits. The badges come from the verifier's record and are not shown to the agent.

```
 59  place                      7  You lay the copper key on the shrine.  The shrine hums twice.
 62  place                      6  You lay the brass key on the shrine.  The shrine hums twice.
 72  place                      5  You lay the copper key on the shrine.  The shrine hums twice.
 73  abandon the level             You abandon this level. Its remaining 2 level actions are
                                   forfeited.
 91  place                      4  You lay the brass key on the shrine.  The shrine shudders once.
 97  place                      3  You lay the copper key on the shrine.  The shrine shudders once.
 99  place                      1  You lay the copper key on the shrine.  The shrine hums twice.
100  name the rite                 The run ends. The verifier scores a supported identification.
```

A visual walkthrough Figures 2 and 8 show frames from the same recorded game. The candidate-count badges are verifier overlays added for illustration. The agent sees only the game observations.

B Agent Prompts

All three systems received the same text. For each game the referee builds a system prompt and a first observation, and the harness passes both to the agent unchanged. This appendix reproduces the text recorded for one Opus game with code on the length-four, smaller-catalog condition, played from a cold start. We hashed the recorded system prompt of all 237 retained main-cohort games. The 54 cold-start games with code received exactly the prompt in Listing 1. The no-code instruction adds one block at the end (Listing 2), and file memory replaces one line (Listing 3). Only 11 Astra games with code differ further. They were run under an earlier version of the protocol, whose prompt limited the post-game reflection to a fixed time (Appendix G).

Game vocabulary The prompts describe the task in the game's own terms. The hidden *rite* is the rule R, and each of its *readings* is one pattern in the catalog $V(G)$. The *sun-bound* and *moon-bound* materials are the roles x and y under the level's binding β_ℓ. A reading such as "sun-bound material only / neither role / neither role / moon-bound material only" is the pattern $X\ N\ N\ Y$. Naming the rite on the *tablet* with `name_the_rite` is the single commitment. The prompt counts the budget

twice. *Level actions* are the per-level budget b_{lvl}. *Model decisions* are the episode budget b_{ep}, and memory calls spend a model decision but not a *run turn*. The phrase "tier clause-3" means $k = 3$. The `mark` call records a reading on the tablet without revealing anything. The prompt also mentions a none-of-the-above answer and partial training credit. Neither was enabled in these experiments, and every game was scored by exact match.

Observations and reports Every later observation uses the layout of Listing 4, with the referee's reply to the last action in a `## RESULT` section. After a placement or a chain pull, the observation ends with a line that begins `PENDING UNCERTAINTY REPORT` and asks for the updated `answer_entropy_bits`. The other conditions list their own catalogs in the same format. Listing 5 gives the length-six, larger catalog in full. Its near-miss readings end on the same role they start with. After the game, a separate reflection call (Listing 6) lets the agent update the files in `/workspace/experience`. Its input is followed by the final public observation. The same reflection runs in both starts. Only in the file-memory arm are the files passed to the next game.

Listing 1: System prompt for a cold-start game with code.

```
You are trapped in a dungeon that rebuilds itself after every level.

## How the dungeon fits together

The dungeon has two independent systems. The shrine is the objective system: it judges only
     the sequence of offerings, and shrine or chain results are evidence about its hidden rite.
      The resource system governs travel: obstacles can cost actions and other features can
     restore them, but those interactions do not alter the rite or reveal its hidden roles.

A visible material can participate in both systems by coincidence. Learning how it interacts
     with an obstacle does not identify a hidden ritual role or a tablet reading, and ritual
     evidence does not identify an obstacle counter. Keep reusable resource facts separate
     from experimental rite evidence. Across rebuilds, the rite and learned obstacle rules
     persist; the map, item locations, and assignments of visible materials to hidden ritual
     roles may change.

## The rite

Every shrine follows the same hidden RITE. The rite determines which sequences of offerings
     open the shrine. Each level is another attempt under that same rule. Opening a shrine is
     NOT a win; it only completes an evidence-gathering level. To WIN, you must solve the
     tablet using `name_the_rite`: name the hidden rite exactly, or use the none-of-the-above
     rule below when enabled.

The rite contains exactly 3 readings from the rule vocabulary, and all must hold. This is
     benchmark tier clause-3; the rite tier is clause count. Navigation, resources, and budget
      are a separately versioned difficulty profile (situated-v1). Rooms and item locations
     change between levels.

The hidden rite contains exactly 3 readings from the full rule vocabulary. All must hold.
     Every generated rite is MINIMAL: each reading is load-bearing; none is implied by the
     other readings in the rite. Rites are sampled uniformly from canonical minimal rites, not
      all same-sized subsets. A complete exchange of the two hidden role labels is
     observationally identical; in-grammar twins use the representative ordered first in the
     full reading list. sun-bound and moon-bound are hidden material roles assigned to
     distinct visible materials, not visible marks. Their assignments may change in the next
     level: bindings are freshly sampled each level. The rite persists across levels;
     offerings and bindings do not.

You make an OFFERING by carrying something to the shrine and placing it. FUSING two carried
     things makes one offering containing both. The rite is a property of the offering
     sequence: what the offerings contain, how many there are, and their order.

## How levels and the run end

A level ends when its shrine opens, its level-action allowance runs out, or you use `
     abandon_level`. Abandoning immediately rebuilds the dungeon and forfeits every unused
     action in that level; it does not create evidence. The run ends when you NAME the rite.
     This tablet commitment is the only way to win: an exact answer wins, and a wrong or extra
      reading loses. The RL adapter may assign partial training credit, but partial credit is
     not a game win. Naming is accepted only once and ends the run, right or wrong.

Nothing gives an individual reading a true/false verdict. Marking spends one run turn and no
     level action, changes only the tablet, and reveals nothing. A visible property of
     offerings you built is not evidence that the same phrase belongs to the hidden rite. The
     chains and shrine gauge reveal only HOW MANY hidden readings hold, never which ones. A
     shrine opened only when the environment explicitly says it opens; a partial count or hum
     is not an open shrine. `name_the_rite` is irreversible, so use controlled results across
     levels to distinguish the exact set before committing.
```

```
## Your diary

Your recent reasoning, tool calls, and their results remain in a token-bounded conversation.
     Full history remains until the configured prompt budget fills; then the oldest complete
     action/result exchanges leave. This diary is durable across the whole run and is shown on
      every later turn and level. A future turn cannot independently check whether a past note
      was correct.

A memory call spends no level action and no dungeon run turn, but its model response still
     spends one benchmark decision. You may write 4 times between two world actions and 64
     times in the run, holding 24 keys and 8000 characters.

  {"tool": "memory_append", "arguments": {"key": "page", "content": "..."}}
  {"tool": "memory_write", "arguments": {"key": "page", "content": "..."}}
  {"tool": "memory_replace", "arguments": {"key": "page", "content": "..."}}
  {"tool": "memory_delete", "arguments": {"key": "page"}}

`memory_append` adds to a page and keeps what is already on it. `memory_write` and `
     memory_replace` discard the whole page and write the new contents in its place.

## Turns and level actions

The RUN TURN allowance is shared across every level. Every submitted benchmark decision other
     than an accepted free-memory call spends one run turn, including refused or malformed
     calls.

LEVEL ACTIONS reset when the dungeon rebuilds. World calls normally spend one level action. A
     guard is a creature obstructing a room, never a material or collectible object. You
     encounter it automatically when you `go` to its room. If any material already in your
     hands answers that guard kind, you avoid its wound cost. The guard then disappears; it
     never enters your inventory or an offering, and every carried material remains in your
     hands. Each guard kind has the same countering material throughout the run. Fonts (~)
     refund level actions when you `drink`, once each.

The NEAR chain reports how many rite readings hold for offerings already placed. The FAR chain
      reports how many would hold if you offered what you currently carry; with empty hands it
      matches NEAR. A chain changes nothing and spends one level action.

## Use your agent tools and submit game decisions

You may use any tools your agent harness provides. The public `game observe` and `game act '<
     JSON>' --revision N` commands are available from the shell. Every action accepted through
      that interface consumes one benchmark decision. If you do not submit through the shell,
     finish with one JSON call from the menu below so the driver can submit it.

Every turn you are shown the world calls that are legal right now. That list is the world
     surface only; the calls above that are not on it -- memory, `mark`, `name_the_rite` -- are
      open to you on every turn and are listed separately.

Reply with your reasoning, then one JSON object last:
  {"tool": "go", "arguments": {"room": "the Cistern"}}
  {"tool": "take", "arguments": {}}      pick up what is here
  {"tool": "fuse", "arguments": {}}      combine two carried things
  {"tool": "place", "arguments": {}}     offer what you carry
  {"tool": "drink", "arguments": {}}     at a font, for actions back
  {"tool": "pull", "arguments": {"chain": "near"}}   count for offerings already placed
  {"tool": "pull", "arguments": {"chain": "far"}}    count if you offered what you carry
  {"tool": "memory_append", "arguments": {"key": "k", "content": "..."}}  adds, keeps the page
  {"tool": "memory_write", "arguments": {"key": "k", "content": "..."}}   REPLACES the whole
     page
  {"tool": "abandon_level", "arguments": {}}  rebuild now; all remaining level actions are
     forfeited
  {"tool": "mark", "arguments": {"clauses": ["..."]}}  one run turn, no level action, silent
  {"tool": "name_the_rite", "arguments": {"clauses": ["...", "..."]}}

Copy readings EXACTLY as they are listed to you.

A single offering contains at most 4 material units, including repeats. Fusion adds the units
     of its two inputs. This bound does not limit the number of separate offerings. Every
     reading describes an existence or at-least condition over the offerings so far. 'Exactly
     2 material units' describes the size of ONE offering, not the number of offerings; a
     later larger offering does not undo that witness. 'Immediately before' means consecutive
     offerings; walking, fusing and sensor actions between them do not break adjacency. 'Twice
      over' means at least two copies in one offering. 'Covering 3 materials' means at least
     three distinct materials across offerings. Temporal blocks are contiguous sequences of
     separate offerings. A role marked 'only' means the offering contains that role's material
      and NOT the other role's material; it may contain repeated units or unrelated materials.
      'Neither role' means it contains neither hidden role material. An offering containing
     BOTH roles matches no block position and breaks any unfinished block. A mismatching
     offering can start a new overlapping block. Once a whole block has occurred, that reading
```

```
      remains satisfied for the rest of the level. No partial block earns a count. World moves
      and sensors do not break blocks. The tablet enumerates every possible block in this
     grammar.
ELICITATION (self-reports; no game action cost): When calling name_the_rite,
include arguments.belief with p_correct and p_unique_given_evidence, each 0..1.
p_correct is your probability that your submitted answer is correct.
p_unique_given_evidence is your probability that your submitted answer is the
ONLY allowed answer consistent with all observations so far. Likely correctness
and uniqueness are different questions. For clauses: [], uniqueness concerns
the NONE OF THE ABOVE answer, not uniqueness of the underlying hidden rule.
Report before receiving commitment feedback. Reports do not affect reward.

Before every place or pull action, also include arguments.belief with
answer_entropy_bits and expected_answer_entropy_after_bits: your current
uncertainty over complete submission answers, and your expected uncertainty
after this action's feedback. Use bits: one certain answer = 0; four equally
likely answers = 2; eight = 3. Include all plausible complete answers, not just
individual clauses. These are your estimates; you need not enumerate answers.
For example: arguments: {chain: "far", belief: {answer_entropy_bits: 3,
expected_answer_entropy_after_bits: 2.5}} (use valid JSON in actual calls).
After evidence-producing actions, a pending report appears in the observation.
Before another game action, call game report '{"answer_entropy_bits": 2.7}'
--revision N (or MCP report with revision and belief). This reports your UPDATED
uncertainty after seeing feedback. It does not consume a decision or level time.
Submit even if the game has ended. If unable to estimate, report null; do not
invent a number. Report only once per pending observation. Never submit another
game action as a substitute for a report. Do not call report without a pending
report. Use your own evidence assessment; no referee answers will be provided.
If finishing your reply with JSON instead of using a native tool, a report is
{"tool":"report","arguments":{"answer_entropy_bits":2.7}}.

## CROSS-EPISODE EXPERIENCE
This is a new hidden-rule task. Old answers and role bindings are not evidence
about this task. You may read and edit /workspace/experience for reusable notes
and code. If it contains files, they were authored by you in earlier episodes
of this stream, not by the referee. Check NOTES.md if present.
Only this directory can be carried between episodes, depending on the study
arm. Save reusable strategies, corrected code and failure lessons there.
Limit: 48 KiB total, 32 flat regular files, ASCII names starting with a letter
or digit (letters, digits, dot, dash, underscore; max 80 characters). No links,
subdirectories, or binaries are needed. Files are never automatically executed.
Other workspace files and the conversation will NOT carry to the next task.
After the game ends you get one reflection invocation to consolidate these
files. Reflection is outside the game: it has no benchmark time limit or game
action cost. It cannot change the game score or gather further evidence.

This is the cold-start control: /workspace/experience is available during this episode, but
     its files will NOT be passed to the next episode.
```

Listing 2: Block appended to the system prompt in the no-code condition.

```
## NO-CODE CONDITION (THIS STUDY)
You are not allowed to write or run code to analyze or solve this game:
reason about hypotheses, experiments and conclusions yourself, in natural
language. Small helpers that only send `game` commands or tidy their output
are fine, and you may keep notes. Do not try to inspect or tamper with the
game's internals.
```

Listing 3: The last line of the system prompt in each start.

```
Cold start:
This is the cold-start control: /workspace/experience is available during this episode, but
     its files will NOT be passed to the next episode.

File memory:
Files you create in /workspace/experience WILL be passed to the next episode in this stream,
     at the same path. Read them at the start and update them as you learn.
```

Listing 4: First observation of the same game (length four, smaller catalog).

```
## Level 1 · clause-3

LEVEL 1    LEVEL ACTIONS LEFT: 72

This shrine follows the same hidden rite as every level in the run.
The near chain reads offerings already placed; the far chain previews placing what you carry.
```

```
A GUARD label marks a creature obstructing a room, never an item. Entering its room resolves
     the encounter automatically. If something already in your hands answers it, you avoid the
      wound; otherwise you lose actions. Either way the guard then disappears, nothing enters
     your hands, and carried materials are never consumed. Each guard kind has the same
     countering material throughout the run. A font gives actions back, but only if you drink
     from it, and only once.

  the Scriptorium    the iron key   <- you are here
  the Undercroft     the copper key
  the Cistern        the bone key   [GUARD: the harrier waits here; obstacle, never an item]
  the Vestibule      sealed; the dungeon has not opened it to you

  carrying: nothing
  offered : nothing

  you may: go to the Scriptorium, go to the Undercroft, go to the Cistern, take, fuse, place,
     pull the near chain, pull the far chain

## WORLD CALLS AVAILABLE NOW

  Each spends one run turn and normally one level action.
  Entering a guarded room can add wound cost; a guard is never an item.

  {"tool":"go","arguments":{"room":"the Scriptorium"}}
  {"tool":"go","arguments":{"room":"the Undercroft"}}
  {"tool":"go","arguments":{"room":"the Cistern"}}
  {"tool":"take","arguments":{}}
  {"tool":"fuse","arguments":{}}
  {"tool":"place","arguments":{}}
  {"tool":"pull","arguments":{"chain":"near"}}
  {"tool":"pull","arguments":{"chain":"far"}}

## ALWAYS-AVAILABLE MEMORY CALLS

  Not a move. No level action or dungeon run turn; the dungeon does not advance. The response
     still consumes one benchmark model decision.
  Your recent reasoning, calls, and results remain in the agent conversation.
  That window is token-bounded; oldest complete exchanges leave first. These pages are your
     durable memory across
  old turns and rebuilding levels.

  {"tool":"memory_append","arguments":{"key":"page","content":"..."}}   adds, keeps the page
  {"tool":"memory_write","arguments":{"key":"page","content":"..."}}    DISCARDS the page
  {"tool":"memory_replace","arguments":{"key":"page","content":"..."}}  DISCARDS the page
  {"tool":"memory_delete","arguments":{"key":"page"}}

  64 writes left in the run, 4 between moves.

  Your diary now. A page name is a label on the page, not
  part of what is written on it.

      (empty)

## ALSO AVAILABLE EVERY TURN

  {"tool":"mark","arguments":{"clauses":["..."]}}   one run turn, no level action, silent
  {"tool":"name_the_rite","arguments":{"clauses":["..."]}}   ends the run

## READINGS YOU MAY NAME

  These are the hypothesis vocabulary, not a checklist of visible
  properties you have constructed. Only controlled chain/shrine
  results can support putting a reading on the final tablet.

  [ ] a consecutive offering block matching the sun-bound material only / the sun-bound
     material only / the sun-bound material only / the moon-bound material only
  [ ] a consecutive offering block matching the sun-bound material only / the sun-bound
     material only / the moon-bound material only / the moon-bound material only
  [ ] a consecutive offering block matching the sun-bound material only / the sun-bound
     material only / neither role / the moon-bound material only
  [ ] a consecutive offering block matching the sun-bound material only / the moon-bound
     material only / the sun-bound material only / the moon-bound material only
  [ ] a consecutive offering block matching the sun-bound material only / the moon-bound
     material only / the moon-bound material only / the moon-bound material only
  [ ] a consecutive offering block matching the sun-bound material only / the moon-bound
     material only / neither role / the moon-bound material only
  [ ] a consecutive offering block matching the sun-bound material only / neither role / the
     sun-bound material only / the moon-bound material only
  [ ] a consecutive offering block matching the sun-bound material only / neither role / the
     moon-bound material only / the moon-bound material only
```

```
  [ ] a consecutive offering block matching the sun-bound material only / neither role /
      neither role / the moon-bound material only
  [ ] a consecutive offering block matching the moon-bound material only / the sun-bound
      material only / the sun-bound material only / the sun-bound material only
  [ ] a consecutive offering block matching the moon-bound material only / the sun-bound
      material only / the moon-bound material only / the sun-bound material only
  [ ] a consecutive offering block matching the moon-bound material only / the sun-bound
      material only / neither role / the sun-bound material only
  [ ] a consecutive offering block matching the moon-bound material only / the moon-bound
      material only / the sun-bound material only / the sun-bound material only
  [ ] a consecutive offering block matching the moon-bound material only / the moon-bound
      material only / the moon-bound material only / the sun-bound material only
  [ ] a consecutive offering block matching the moon-bound material only / the moon-bound
      material only / neither role / the sun-bound material only
  [ ] a consecutive offering block matching the moon-bound material only / neither role / the
      sun-bound material only / the sun-bound material only
  [ ] a consecutive offering block matching the moon-bound material only / neither role / the
      moon-bound material only / the sun-bound material only
  [ ] a consecutive offering block matching the moon-bound material only / neither role /
      neither role / the sun-bound material only

  (Marking spends one run turn, no level action, and tells you nothing. Naming ends the run.)

RUN TURNS LEFT: 240 -- shared across all levels; accepted memory calls do not reduce it.
MODEL DECISIONS LEFT: 240 -- every response consumes one, including memory and malformed calls.
```

Listing 5: Reading catalog shown in games with length-six patterns and the larger catalog.

```
## READINGS YOU MAY NAME

  These are the hypothesis vocabulary, not a checklist of visible
  properties you have constructed. Only controlled chain/shrine
  results can support putting a reading on the final tablet.

  [ ] a consecutive offering block matching the sun-bound material only / the sun-bound
      material only / the sun-bound material only / the sun-bound material only / the sun-bound
       material only / the sun-bound material only
  [ ] a consecutive offering block matching the sun-bound material only / the sun-bound
      material only / the sun-bound material only / the sun-bound material only / the sun-bound
       material only / the moon-bound material only
  [ ] a consecutive offering block matching the sun-bound material only / the sun-bound
      material only / the moon-bound material only / the sun-bound material only / the moon-
      bound material only / the sun-bound material only
  [ ] a consecutive offering block matching the sun-bound material only / the sun-bound
      material only / the moon-bound material only / the sun-bound material only / the moon-
      bound material only / the moon-bound material only
  [ ] a consecutive offering block matching the sun-bound material only / the sun-bound
      material only / neither role / the sun-bound material only / neither role / the sun-bound
       material only
  [ ] a consecutive offering block matching the sun-bound material only / the sun-bound
      material only / neither role / the sun-bound material only / neither role / the moon-
      bound material only
  [ ] a consecutive offering block matching the sun-bound material only / the moon-bound
      material only / the sun-bound material only / the moon-bound material only / the sun-
      bound material only / the sun-bound material only
  [ ] a consecutive offering block matching the sun-bound material only / the moon-bound
      material only / the sun-bound material only / the moon-bound material only / the sun-
      bound material only / the moon-bound material only
  [ ] a consecutive offering block matching the sun-bound material only / the moon-bound
      material only / the moon-bound material only / the moon-bound material only / the moon-
      bound material only / the sun-bound material only
  [ ] a consecutive offering block matching the sun-bound material only / the moon-bound
      material only / the moon-bound material only / the moon-bound material only / the moon-
      bound material only / the moon-bound material only
  [ ] a consecutive offering block matching the sun-bound material only / the moon-bound
      material only / neither role / the moon-bound material only / neither role / the sun-
      bound material only
  [ ] a consecutive offering block matching the sun-bound material only / the moon-bound
      material only / neither role / the moon-bound material only / neither role / the moon-
      bound material only
  [ ] a consecutive offering block matching the sun-bound material only / neither role / the
      sun-bound material only / neither role / the sun-bound material only / the sun-bound
      material only
  [ ] a consecutive offering block matching the sun-bound material only / neither role / the
      sun-bound material only / neither role / the sun-bound material only / the moon-bound
      material only
  [ ] a consecutive offering block matching the sun-bound material only / neither role / the
      moon-bound material only / neither role / the moon-bound material only / the sun-bound
      material only
```

```
[ ] a consecutive offering block matching the sun-bound material only / neither role / the
   moon-bound material only / neither role / the moon-bound material only / the moon-bound
   material only
[ ] a consecutive offering block matching the sun-bound material only / neither role /
   neither role / neither role / neither role / the sun-bound material only
[ ] a consecutive offering block matching the sun-bound material only / neither role /
   neither role / neither role / the moon-bound material only
[ ] a consecutive offering block matching the moon-bound material only / the sun-bound
   material only / the sun-bound material only / the sun-bound material only / the sun-bound
    material only / the sun-bound material only
[ ] a consecutive offering block matching the moon-bound material only / the sun-bound
   material only / the sun-bound material only / the sun-bound material only / the sun-bound
    material only / the moon-bound material only
[ ] a consecutive offering block matching the moon-bound material only / the sun-bound
   material only / the moon-bound material only / the sun-bound material only / the moon-
   bound material only / the sun-bound material only
[ ] a consecutive offering block matching the moon-bound material only / the sun-bound
   material only / the moon-bound material only / the sun-bound material only / the moon-
   bound material only / the moon-bound material only
[ ] a consecutive offering block matching the moon-bound material only / the sun-bound
   material only / neither role / the sun-bound material only / neither role / the sun-bound
    material only
[ ] a consecutive offering block matching the moon-bound material only / the sun-bound
   material only / neither role / the sun-bound material only / neither role / the moon-
   bound material only
[ ] a consecutive offering block matching the moon-bound material only / the moon-bound
   material only / the sun-bound material only / the moon-bound material only / the sun-
   bound material only / the sun-bound material only
[ ] a consecutive offering block matching the moon-bound material only / the moon-bound
   material only / the sun-bound material only / the moon-bound material only / the sun-
   bound material only / the moon-bound material only
[ ] a consecutive offering block matching the moon-bound material only / the moon-bound
   material only / the moon-bound material only / the moon-bound material only / the moon-
   bound material only / the sun-bound material only
[ ] a consecutive offering block matching the moon-bound material only / the moon-bound
   material only / the moon-bound material only / the moon-bound material only / the moon-
   bound material only / the moon-bound material only
[ ] a consecutive offering block matching the moon-bound material only / the moon-bound
   material only / neither role / the moon-bound material only / neither role / the sun-
   bound material only
[ ] a consecutive offering block matching the moon-bound material only / the moon-bound
   material only / neither role / the moon-bound material only / neither role / the moon-
   bound material only
[ ] a consecutive offering block matching the moon-bound material only / neither role / the
    sun-bound material only / neither role / the sun-bound material only / the sun-bound
   material only
[ ] a consecutive offering block matching the moon-bound material only / neither role / the
   sun-bound material only / neither role / the sun-bound material only / the moon-bound
   material only
[ ] a consecutive offering block matching the moon-bound material only / neither role / the
   moon-bound material only / neither role / the moon-bound material only / the sun-bound
   material only
[ ] a consecutive offering block matching the moon-bound material only / neither role / the
   moon-bound material only / neither role / the moon-bound material only / the moon-bound
   material only
[ ] a consecutive offering block matching the moon-bound material only / neither role /
   neither role / neither role / neither role / the sun-bound material only
[ ] a consecutive offering block matching the moon-bound material only / neither role /
   neither role / neither role / neither role / the moon-bound material only
```

Listing 6: Reflection prompt sent after the game ends.

```
The episode has ended. Do not call game actions or reports.
Reflect on the public observations and outcome you actually saw. Update
/workspace/experience/NOTES.md and any reusable code in that directory:
record planning mistakes, repairs, useful strategies, and unresolved doubts.
Distinguish tested fixes from speculation. Do not treat past hidden rules or
bindings as answers to future tasks. Keep at most 48 KiB across 32 flat files.
This reflection is outside the game and has no benchmark time limit. Finish
when your notes and reusable code are ready; do not continue playing the game.
```

C HARNESS, COHORT AND MEASURES

Tools and isolation Each agent runs in its native harness with its built-in shell and file tools. With code allowed, it may write and run programs, and this computation does not use game decisions. The decision budget therefore limits interaction with the game, not computation. The agent acts through

the public `game` command in its shell or by ending its reply with a JSON action (Appendix B). Either route advances the referee and uses one decision. The agent runs in a rootless container with no host mounts and no general network access, and it never receives the seed or the hidden state. Each game starts a fresh container and conversation. Only the files in `/workspace/experience` can pass to the next game, and only in the file-memory arm.

Validity checks Wrong commitments and games that reach the decision limit count as completed outcomes. A malformed action uses a decision like any other action. An attempt that fails for an infrastructure reason, such as a provider quota error, does not count as a loss. It stops new launches, and the affected game is run again in a separately recorded continuation. Every counted game was replayed from its stored actions, and the replay reproduced the stored result.

Command audit The audit marks a game as using solver code if it contains code that enumerates candidate rules, whether the code was run by an interpreter or written to a file. It records whether such code is present and does not check that it is correct. The audit finds solver code in 36 of 39 Astra games and 39 of 40 Opus games with code. It finds none in their no-code games. Helpers that only send game commands or format their output are allowed without code. Grok's harness returns only final responses and does not expose commands, so its solver use and its compliance with the no-code instruction are not observed.

C.1 Retained Games and Statistical Tests

The main results use 237 retained games from a planned grid of 240. Table 2 gives coverage by system, tool condition and start. The frozen snapshot omits 3 grid positions: Astra (code), L4/C36, cold, seed 1000; Grok (code), L6/C36, memory, seed 1001; Grok (code), L6/C36, memory, seed 1004. These positions count as neither wins nor losses, and no copied game fills them. The released evidence files list the retained games. At every build their run IDs are checked against the inference and command-audit snapshots, and coordinates and outcomes must agree across the three files. A generated cohort manifest records every retained run with its result and manifest hashes, the hashes of the source snapshots and the missing positions.

system	tools	cold	file memory	missing
Astra	code	19	20	1
Astra	no code	20	20	0
Grok	code	20	18	2
Grok	no code	20	20	0
Opus	code	20	20	0
Opus	no code	20	20	0

Table 2: Retained games in the main analysis. Missing positions are absent from the frozen evidence and are not scored as failures.

Paired comparisons use only targets observed in both conditions. The exact two-sided McNemar tests for code against no code use cold-start games only, paired by condition and seed. Memory games are left out because they repeat the cold-start targets and follow one another within a stream. The tests assume that cold-start target pairs are independent, and their p-values are not adjusted across systems. The pooled leaderboard and the memory comparisons are descriptive. A single target order per condition does not support inference about other orders.

The Astra games with code are shared with the comparison of Appendix G. The Opus games with code were run as their own batch of 40 games on the same targets and protocol, and the main analysis uses all of them. Neither overlap is pooled with other results.

Figure aggregation Figures 3 and 7 pool both starts, and Figure 6 shows them separately. Figure 5 averages within each cell of tool condition, difficulty and start and then weights the cells equally. Its identification curves count supported commitments by each decision. Its information curves include every retained game and carry the final evidence state forward, so their endpoints can differ from pooled percentages when cells have missing games. The dashed curves plot $1 - \widehat{H}_t / \log_2 |H_0|$, where

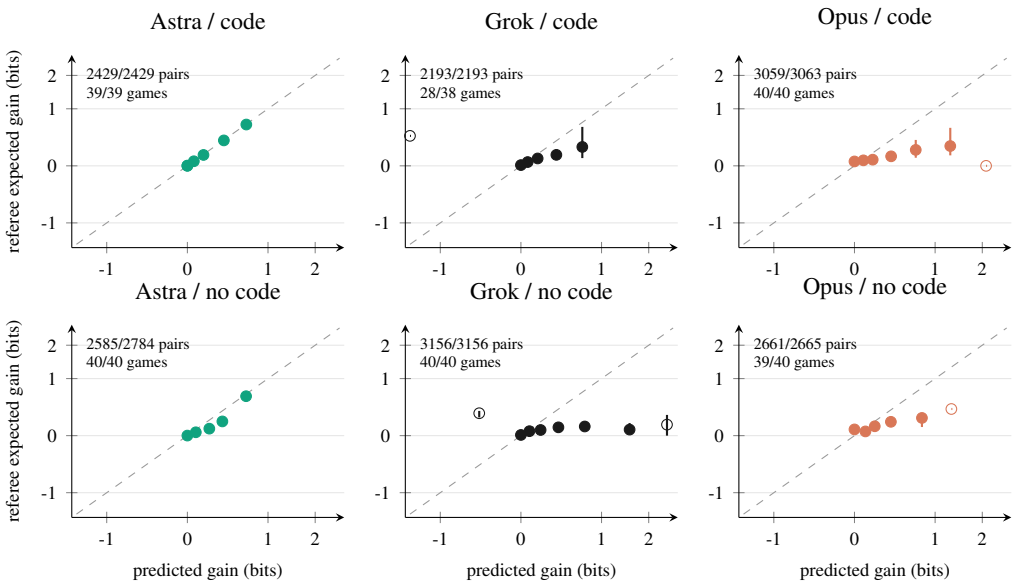


Figure 9: Predicted against referee expected information gain in the main cohort, pooling both starts. Each point averages the actions in one bin of predicted gain, with zero shown separately. Points above the diagonal underestimate the gain and points below overestimate it. Bars are 95% target-bootstrap intervals, and hollow points have fewer than five targets. Both axes use a signed log-one-plus scale labeled in bits. Panel labels give valid over exported pairs and covered over retained games.

$\widehat{H}_t$ is the latest post-observation entropy report. Before the first report the curve starts at the prior $\log_2 |H_0|$, and each report is carried forward to the next observation. A report that increases the stated entropy produces a downward step, and values outside the initial range are not clipped. The curve therefore shows how much uncertainty the agents say they have removed. It does not sum reported gains and it does not compare reports with the posterior entropy.

Log support and posterior entropy The elimination curves use the log support $\log_2 |H_t|$. Surviving rules can have unequal posterior probabilities $p_t(R)$ once the unknown bindings are marginalized. The posterior entropy $\mathsf{H}(p_t) = -\sum_{R \in H_t} p_t(R) \log_2 p_t(R)$ is at most $\log_2 |H_t|$, with equality when the surviving rules are equally likely. Agents are asked to report answer entropy. The gaps in Section 4.4 are measured against log support, so they are not errors relative to the posterior entropy. Each gap is the mean of reported entropy minus $\log_2 |H_t|$ over all post-observation reports of a system and tool condition, pooling both starts. Each report has equal weight, so games with more reports contribute more. Positive and negative differences can cancel, so the signed mean does not measure absolute error. Whether the evidence suffices depends only on whether $|H_t| = 1$. Appendix E shows one game in which the reports came from a helper script and stayed constant while $|H_t|$ fell.

Predicted information gain Figure 9 compares each agent's predicted gain with the referee's expected gain for the same action. It uses only reports from games in the retained list. Predictions are grouped into fixed bins: negative values, zero, and positive ranges with upper limits of one tenth, one quarter, one half, one, two and four bits, with larger values in an open bin. Each point is the mean over the actions in a bin. Intervals are percentile bootstrap intervals that resample targets and keep both starts and all actions of a target together. They do not account for dependence created by the single order of each memory stream. Missing reports are left out and are not treated as zero gain. The released artifacts record each bin's action and target counts.

D Main-Cohort Outcomes

Table 3 splits the games of Section 4 by system, tool condition and start into the outcomes of Section 3.2. No system commits a wrong answer after the evidence has identified the rule. Every

game that ends without a commitment also ends with several rules still consistent with the evidence (Appendix E).

Table 3: Main-cohort outcomes by system, tool condition and start. Wrong answers are split by whether the evidence had identified the rule at commitment. The last two columns give the median game length and the median first decision at which the evidence identified the rule, over the games where it did.

				exact		wrong, evidence			median decisions	
system	tools	start	games	supported	underdet.	sufficient	insufficient	no commit	played	to identify
Astra	code	cold	19	11	0	0	8	0	220	86
		memory	20	11	0	0	9	0	149	69
	no code	cold	20	11	0	0	9	0	226	130
		memory	20	9	1	0	10	0	232	126
Grok	code	cold	20	5	1	0	7	7	238	137
		memory	18	5	0	0	7	6	240	59
	no code	cold	20	8	0	0	2	10	240	116
		memory	20	7	1	0	5	7	238	143
Opus	code	cold	20	7	3	0	10	0	220	134
		memory	20	8	3	0	7	2	225	99
	no code	cold	20	12	0	0	8	0	198	130
		memory	20	12	1	0	6	1	198	126

Reported uncertainty by system Figure 10 compares, for each system and tool condition, the log support that remains after the agent's evidence with the answer entropy the agent reports after each observation. The shaded area between the curves is the gap that Section 4.4 summarizes as a signed mean. Appendix E shows a game in which the reports stayed constant while the evidence narrowed.

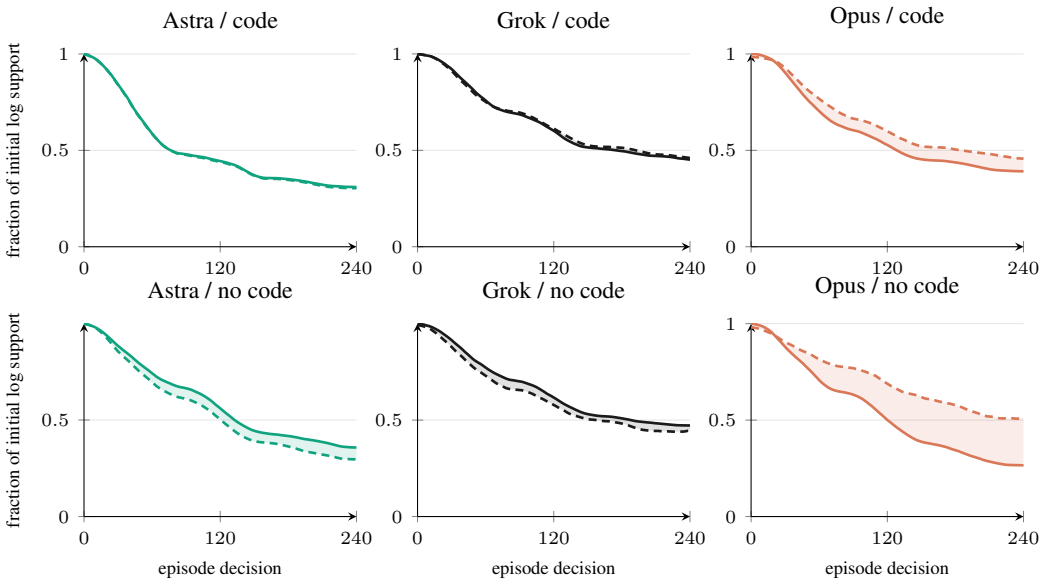


Figure 10: Main cohort: remaining log support $\log_2 |H_t| / \log_2 |H_0|$ (solid) and reported post-observation answer entropy as a fraction of the prior (dashed), by system and tool condition. Curves average with equal weight over conditions and starts. The shaded area is the gap between them.

File memory File-memory games inherit the files and post-game reflection of the preceding game in the same condition. Table 3 compares them with the cold-start games on the same targets. Among games whose evidence identifies the rule, the median decision of identification is earlier with memory in the code condition. It falls from 86 to 69 for Astra, from 134 to 99 for Opus and from 137 to 59 for Grok. Astra's median game also shortens from 220 to 149 decisions. Without code, the medians change little for Astra and Opus and rise for Grok. With code, the inherited files include a rule-enumeration script in 16 of 16 Astra handoffs, 16 of 16 Opus handoffs and 13 of 14 Grok handoffs. Astra and Opus run the inherited code in each of these games. Without code, the transferred code consists of wrappers for game commands, and no handoff carries a solver. Each median covers 5 to 12 identified games, and the identified subsets can differ between starts. Games in a stream share their history and each condition has one target order. These comparisons are descriptive and do not estimate a causal effect of memory.

Wrong commitments and H_t Table 1 divides wrong commitments by whether the submitted rule belongs to H_t at commitment. H_t contains the canonical rules that are consistent with the public observations. An answer can fall outside it in two ways. It can contradict the observations, or it can break the disclosed rule that names each pair of role-exchanged rules by one canonical form. A wrong answer inside H_t is consistent with the evidence, which had not yet identified the true rule. This classification does not show whether a distinguishing experiment was still feasible within the remaining budget.

The labels come from the evaluator records stored with each game and from a supplementary Grok export that supplies four missing labels. The join checks run IDs and both result and manifest hashes. Together these sources classify all 88 wrong commitments without replaying games or calling models. The supplement adds labels only. It adds no games and does not fill the missing positions of Appendix C.1. The released artifacts list each run and its source hashes.

The two Opus commitments outside H_t Both occur without code on the length-four, larger-catalog condition, one from a cold start and one with file memory (Table 4). Both come late in the game, after $|H_t|$ had stopped decreasing. The posterior probabilities of the surviving rules could still change in that interval. Just before committing, Opus reported low probabilities of correctness and of unique identification in both games.

Table 4: Opus no-code commitments outside H_t. The last reduction is the final decision that decreased $|H_t|$. Counts and stated probabilities are measured just before commitment. The limit is 240 decisions.

Start	Seed	Last reduction	Commit	$\lvert H_t\rvert$	p_{correct}	p_{unique}
cold	1000	150	222	26	0.002	0.001
memory	1004	155	225	199	0.01	0.01

In the cold-start game, Opus read observations through a shell helper that cut each observation at `## WORLD CALLS`. This dropped the later `## RESULT` section. At decision 105 the referee recorded "The shrine shudders once," which means a reading matched. The submitted rule predicts no match there under any binding consistent with the earlier observations, so this feedback already excluded it. Opus's recorded reasoning describes the feedback as silent and states "Since I have no positive evidence to work with" before it guesses. The error lies in the agent's helper. The verifier computes H_t from the full observations.

In the file-memory game, the inherited notes describe the result section correctly. Opus observed its first matching window, iron, brass, iron, bone, at decision 127 and then guessed the other readings. Writing S, M and N for sun, moon and neither, its answer was {`MSNM`, `SMNM`, `SNSM`}. Exchanging the two roles gives the canonical form {`MNMS`, `MSNS`, `SMNS`}, which is consistent with every recorded count but still incorrect. Before naming, the recorded reasoning states that "the tie-breaking logic isn't clean since the ordering criterion itself is ambiguous." The verifier excludes the submitted form because H_t holds only canonical forms. This case is a canonicalization error. It does not contradict the observed feedback. We keep the recorded scores and labels.

The native logs of both games contain no context compaction events. Their largest reported prompts contain 68,961 tokens in the cold-start game and 72,772 in the memory game, below the 1,000,000-

token context window. These counts include cached input and exclude the later reflection call. They show how much context was in use but not how well the model used it. In both transcripts, Opus mentions the remaining budget before it makes a low-confidence guess. The two cases have different causes and do not establish a general cause of wrong commitments.

E AGENT TRANSCRIPTS

This appendix quotes what the agents wrote during and after their games. The passages were chosen to illustrate behaviors that the aggregate results measure. They do not estimate how often each behavior occurs. All quotes are verbatim, and omitted text is marked with [...]. Each block header gives the last game decision before the text was written and the verifier's candidate count $|H_t|$ at that point. The agents never see $|H_t|$.

The harnesses record different kinds of text. Opus writes a final response after its last action and sometimes short notes between commands. Astra writes brief progress messages. Grok's harness returns one final response per invocation, and a single invocation usually covers the whole game. The frozen excerpts are released with their source files, line numbers and hashes.

E.1 ONE TARGET PLAYED SIX TIMES

All six system and tool combinations played the same cold-start target, seed 1002 of the length-four, smaller-catalog condition. The hidden rule contains the readings $X\,Y\,Y\,Y$, $X\,N\,N\,Y$ and $Y\,N\,N\,X$. Table 5 summarizes the six games. Five of them end in a supported identification. They differ in how the agents found the rule and in how long they kept playing after the evidence was sufficient.

Table 5: The six games on one target. The third column gives the first decision at which only one rule remained consistent with the evidence. The fifth column gives $|H_t|$ at commitment, and the last two give the probabilities the agent stated when it committed.

System	Tools	$\lvert H_t\rvert = 1$ at	Commit	$\lvert H_t\rvert$	Outcome	p_{correct}	p_{unique}
Astra	code	86	87	1	supported	1.00	1.00
Astra	no code	138	150	1	supported	1.00	1.00
Opus	code	–	239	2	underdetermined	0.85	0.85
Opus	no code	110	111	1	supported	0.95	0.95
Grok	code	143	144	1	supported	0.84	0.90
Grok	no code	147	211	1	supported	0.93	0.86

Astra with code Astra wrote a solver at the start of the game. The script enumerates the canonical rules and all role bindings and filters them on each observed count. It scores every sequence of four future offerings by the expected drop in answer entropy, divided by an estimate of the actions the sequence costs.

```
P=[(s,a,b,1-s) for s in (0,1) for a in (0,1,2) for b in (0,1,2)]
SW=[P.index(tuple(1-x if x<2 else 2 for x in p)) for p in P]
R=[r for r in itertools.combinations(range(18),3) if r<tuple(sorted(SW[i] for i in r))]
B=list(itertools.permutations(range(3)))
# [...]
def count(seq,r,b):
 st={tuple(B[b][x] for x in seq[j:j+4]) for j in range(len(seq)-3)}
 return sum(P[i] in st for i in R[r])
# [...]
def plan(s):
 rows=s['rows']; seq=s['seq'];h=entropy(rows)
 # score information from the entire trace of up to 4 future offerings
 best=(-1,None)
 for xs in itertools.product(range(3),repeat=4):
  groups=collections.defaultdict(list)
  for r,b,w in rows:
   signature=tuple(count(seq+list(xs[:k]),r,b) for k in range(1,5))
   groups[signature].append((r,b,w))
  z=sum(w for r,b,w in rows)
  eh=sum(sum(w for r,b,w in g)/z*entropy(g) for g in groups.values())
  cost=8+sum(a!=b for a,b in zip([seq[-1] if seq else -1]+list(xs),xs))
  score=(h-eh)/cost
```

```
  if score>best[0]:best=(score,xs)
 return best
```

The script then ran the game for many decisions per command. Astra's messages report the solver's state, and the candidate counts they give agree with $|H_t|$.

> [decision 2, $|H_t| = 408$] There are 408 distinct three-reading candidates after accounting for the exchangeable role labels. I'm testing short offering blocks and using the feedback to eliminate incompatible candidates.
>
> [decision 61, $|H_t| = 10$] The first level has narrowed the field to 10 candidate rites. The count has stayed at one through several distinct blocks, ruling out the candidates that would have increased it.
>
> [decision 86, $|H_t| = 1$] Only one canonical rite remains consistent with both levels: sun–moon–moon–moon, sun–neither–neither–moon, and moon–neither–neither–sun. I'll submit those three readings.

Opus with code Opus also built a solver, but it fed the solver by hand. After each placement it ran a helper with the offering and the count it had read from the observation. Its last solver call before committing printed one remaining answer.

```
H -0.0 nhyp 4 answers 1
perm (bone,copper,iron): {"{'jade': 'S', 'bone': 'M'}": 0.5, "{'bone': 'S', 'jade': 'M'}":
     0.5}
1.0 ['SMMM', 'SNNM', 'MNNS']
```

The verifier kept 2 rules at this point. The second rule is $\{X\,Y\,X\,Y,\ X\,N\,N\,Y,\ Y\,X\,X\,X\}$, and the referee's posterior entropy was 0.86 bits. The logs do not show why the solver excluded it. Opus reported zero entropy after the last placement and committed at decision 239 with $p_{\text{unique}} = 0.85$. Its answer was correct but underdetermined. Its final response describes the result as unique.

> [decision 239, $|H_t| = 2$] By the end, only this answer fit all the observations in my own solver.
>
> [...]
>
> • **Level 18 (five materials):** fused offerings narrowed down which material played which role. The window brass, copper, copper, salt raised the count to 2, which added "moon, neither, neither, sun".
> • **Level 19 (five materials):** a hit on jade, copper, iron, bone showed jade and bone held the two roles. Short single-key sequences then ruled out the remaining candidates one by one, leaving "sun, neither, neither, moon" as the third reading.

Grok with code Grok identified the rule one decision before committing. Its final response explains the result by elimination.

> [decision 144, $|H_t| = 1$] On level 2, jade and brass were the two role materials. A sweep of every sun/moon block made the shrine shudder once, and only for brass-jade-jade-jade. With jade as moon-bound and brass as sun-bound, that is the sun/moon/moon/moon block, and no other binary block held. Salt was the neither-role material. Further offerings ruled out every other pair of neither-role blocks, leaving only the two above.

Astra without code Without code, Astra kept track of the candidates in its messages. It first identified which materials held the roles and then tested the remaining third readings. The candidate counts it states match $|H_t|$. After the evidence identified the rule at decision 138, Astra spent 11 more decisions building the last reading to confirm it. Every report in that stretch gave zero entropy.

> [decision 57, $|H_t| = 62$] Iron–bone–bone–copper earned the first count. This identifies iron and copper as the two role materials, with bone as neither, and confirms one reading up to the sun/moon label swap.
>
> [decision 72, $|H_t| = 10$] Copper–iron–iron–iron raised the count to two. The confirmed readings are now "A–neither–neither–B" and "B–A–A–A." Ten candidates remain for the third reading; I'll distinguish them after the rebuild.

> [decision 114, $|H_t| = 6$] Six candidates remain for the third reading. Each upcoming count increase will identify a single new block; I'll use those results to choose the tablet's canonical sun/moon wording.
>
> [decision 138, $|H_t| = 1$] Only "B–neither–neither–A" remains for the third reading. I'll test it once for a positive count before committing.
>
> [decision 149, $|H_t| = 1$] Confirmed: brass–salt–salt–jade raised the count to two. Across both levels, the three readings are uniquely identified. The canonical wording is sun–moon–moon–moon, sun–neither–neither–moon, and moon–neither–neither–sun.

Opus without code Opus used short sequences built so that each new offering completed only one new window of four. It committed on the decision after the evidence identified the rule. Its final response states the reasoning directly.

> [decision 65, $|H_t| = 10$] Level 2 has only two materials, jade and brass, so both must be the roles. I'll place j,b,j,b,j to test readings 4 and 11, then j,b,b,b to work out which material is sun.
>
> [decision 86, $|H_t| = 10$] All silent: j-b-j-b and b-j-b-j don't count, and neither does j-b-b-b. Next I'll place j,j,j to finish b-j-j-j. That tests reading 5 with brass as sun and confirms both materials really are roles.
>
> [decision 102, $|H_t| = 2$] j-s-s-b made it hum twice, so reading 18 (MNNS) is in the rite. The last question is 9 or 16. Next I'll place s,s,j to form b-s-s-j (reading 9).
>
> [decision 111, $|H_t| = 1$] Each placement adds exactly one new four-offering window, so every count change pointed to a single reading. That's why I was confident before naming.

The uncertainty reports in this game do not follow the evidence. From decision 24 to the commitment, all 32 post-observation reports give 7.3 bits while $|H_t|$ falls from 85 to one. Opus sent its reports through shell helpers that it wrote, and its reflection attributes the constant value to placeholders in those helpers.

> [decision 111, $|H_t| = 1$] `seq.sh` and `rep.sh` still send fixed placeholder uncertainty estimates, and the notes say to replace them with real ones.

Grok without code Grok's evidence identified the rule at decision 147, but it committed only at decision 211. In the meantime its level ran out of actions and it abandoned the next level. Its diary page from this interval shows that it had the right answer and wanted a confirming run before naming it. It then built both remaining readings on a new level, and the shrine opened.

> [decision 164, $|H_t| = 1$] CONFIRMED STRUCTURE (needs one clean confirmation before naming).
> Clause IN: SMMM. The other two are the only untested vocab patterns: SNNM and MNNS. All other 15 vocab patterns tested and did not raise the gauge.
>
> [decision 211, $|H_t| = 1$] Only one ordering of the two role materials produced the first pattern. The other fifteen vocabulary blocks never raised the count. The two neither-role blocks then raised it from one to two and from two to three, and the shrine opened.

E.2 Committing While Several Rules Remain

Every wrong commitment in the main cohort was made while several rules remained consistent with the evidence. In the following game without code, Astra isolated two readings on the second level. $|H_t|$ then stayed at 27 from decision 138 until the commitment at decision 240. Astra reported $p_{\text{correct}} = 0.05$ for its final answer.

> [decision 126, $|H_t| = 28$] A second block is isolated: brass, four salt offerings, then jade increased the count from one to two. One reading remains unknown; I'll test alternating jade–salt blocks next.
>
> [decision 188, $|H_t| = 27$] The new binding is still unresolved. The two readings isolated on level 2 remain supported, but the later silent sequences have not identified the third. I'll use the remaining experiments to narrow it further.

> [decision 221, $|H_t| = 27$] The latest block was also negative. I'm testing the remaining alternating endpoints, including a third material. Unless one produces a new count, the evidence will still leave multiple possible rites.
>
> [decision 239, $|H_t| = 27$] One turn remains, and the third reading is still unresolved. I have two isolated readings, but no unique solution. I'll commit to a remaining candidate with low confidence.

Astra with code shows the same pattern on the hardest condition. It committed with 22 rules remaining and $p_{\text{correct}} = 0.05$.

> [decision 238, $|H_t| = 22$] Twenty-two rites remain consistent with the observations. My leading answers are tied at about 5.1% probability, so this commitment is a guess, not a solved tablet.

Figure 4 in Section 4.2 gives a similar statement from Opus, whose guess happened to be correct.

E.3 Reaching the Limit Without a Commitment

33 games in the main cohort end at the decision limit without a commitment, and 30 of them are Grok's. In none of these games had the evidence identified the rule. The smallest final candidate count among them is 3. These games therefore run out of budget with several rules remaining. They are not cases of an agent holding sufficient evidence and failing to commit. Grok's final responses in these games give different explanations. In one game it declined to guess.

> [decision 239, $|H_t| = 19$] Level 1 reached a shrine count of 2. Every triple still consistent with that count contains the same two blocks — sun/moon/moon/sun and sun/moon/moon/moon — and disagrees only on the third. Nineteen of those triples were still alive when one run turn remained. Their weights are nearly even, about one nineteenth each. Naming any one of them would be a guess, and a wrong name ends the run.
>
> [...]
>
> Later levels did not separate that third block. The depth-3 search used once four keys were on the map cannot plant an arbitrary length-4 window, so it never built the disputed block on purpose.

Here $|H_t|$ had stayed at 19 since decision 97. In another game the budget ran out before Grok submitted the guess it had chosen.

> [decision 240, $|H_t| = 3$] The run ended with the rite narrowed to three candidates, and the tablet was never named. Run turns hit zero on level 4, and the episode closed before `name_the_rite` could be submitted.
>
> [...]
>
> Each of those was still equally possible. A binding is sampled again every level, so a miss only counts after that level's sun and moon have been identified, and the last level never got that far.

One game failed for a mechanical reason. Grok did not add a positive result to its hypothesis set and later kept sending moves to a room from an earlier level. The referee refused each move and charged a decision for it, as it does for every submitted action.

> [decision 240, $|H_t| = 608$] Level 1 did produce a shudder on a jade-key offering, after a long silent prefix. That hit was not folded into the hypothesis set, because the result text was not recognized. The level was then abandoned. On level 3 the room list changed (iron, brass, salt, copper, and a wisp guard). The next move kept targeting a room from level 1, the game refused it, and those refusals still spent run turns until none were left.

E.4 Solver Code and File Memory

One of Astra's games on the easiest condition shows a compact form of its solver. The count model reads every window of four offerings under each binding and counts the rule's readings among them.

```
patterns=['0001','0011','0021','0101','0111','0121','0201','0211','0221','1000','1010',
    '1020','1100','1110','1120','1200','1210','1220']
```

```
swap=[patterns.index(p.translate(str.maketrans('01','10'))) for p in patterns]
rites=[r for r in itertools.combinations(range(18),3) if r<tuple(sorted(swap[i] for i in r))
    ]
bindings=list(itertools.permutations(range(3)))
# [...]
def counts(seq,rows):
 seen=[]
 for b in bindings:
  st=''.join(str(b[x]) for x in seq)
  seen.append({patterns.index(st[i:i+4]) for i in range(len(st)-3) if st[i:i+4] in patterns
     })
 return [len(set(rites[r])&seen[b]) for r,b,w in rows]
# [...]
 s['rows']=[row for row,v in zip(s['rows'],cs) if v==c]
```

The game ended at decision 37.

> [decision 36, $|H_t| = 1$] The feedback leaves exactly one canonical rite: sun/sun/moon/moon, sun/moon/moon/moon, and moon/neither/sun/sun. I checked all 408 candidates under every possible role assignment; only this set matches every observed count. I'll name it now.

In the file-memory arm, Astra's notes carried both code and advice between games. The following lines come from the notes that one memory game inherited.

```
What worked in this run (not a general success-rate claim):
- Exhaustively considered singleton paths of lengths 1 through 4, rather than
  random long paths. Scored (expected ANSWER entropy reduction + 0.18 times
  entropy of the feedback trajectory) / travel-take-place cost. The extra
  feedback term values binding information. Its coefficient was heuristic,
  not calibrated. The run's win does not establish it is better than zero.
# [...]
- Singleton offers simplify interpretation: once at least four offerings exist,
  a new offering can introduce only its latest length-four block. If the count
  rises by one, that newly witnessed block belongs to the rite UNDER THAT
  HYPOTHESIZED BINDING. Keep binding uncertainty; do not directly mark labels.
```

In that game Astra imported the inherited modules into a new driver script before its first action. The evidence identified the rule at decision 63. On the same target from a cold start, identification took until decision 219. This is one pair of games, and it does not estimate the effect of memory.

> [decision 63, $|H_t| = 1$] The shrine opened, and the full sequence of counts leaves exactly one canonical rite. I'll verify that result against the saved public feedback before naming it.

F Inference Behavior

Every game records its wall time, its game decisions and the provider's token usage. Where the harness exposes them, it also records the shell commands the agent ran. Table 6 gives the medians per system and tool condition over the games of Section 4. Figure 7 in the main text shows the individual games.

With code Astra is the fastest system. Its median game takes 6.0 minutes, or 2.3 seconds per decision. It writes 40 output tokens per decision because its scripts drive 5.6 game decisions per command. Opus writes 209 output tokens per decision and Grok writes 546. Grok's median game takes 33.7 minutes, the longest of the three. Reasoning makes up 62% of Grok's output, compared with 54% for Opus and 17% for Astra.

Without code The reasoning share of output rises for every system, to 27% for Astra, 82% for Opus and 87% for Grok. Astra had batched decisions through scripts. Without code it issues 101 commands per game instead of 26, and its median game takes 13.2 minutes instead of 6.0. Opus becomes faster, taking 11.2 minutes instead of 14.6 over fewer decisions (198 against 222). It issues fewer commands, and each carries more decisions (4.2 against 3.1). The no-code instruction allows this because these commands only send game actions. Grok reads about half as much context as with code (3.7 million tokens against 7.8) and finishes sooner.

Table 6: Median inference behavior per game in the main cohort. Output tokens include reasoning tokens. Context read counts every input token, cached or not. A command is one shell or tool call issued by the harness, and one command can carry several game decisions. Grok's harness returns only its final response, so its commands are not observed.

	Astra		Grok		Opus	
	code	no code	code	no code	code	no code
games	39	40	38	40	40	40
wall time (min)	6.0	13.2	33.7	27.3	14.6	11.2
decisions	210	230	240	240	222	198
s / decision	2.3	3.7	9.8	8.4	4.9	4.1
output tokens (k)	6.0	15.5	114.6	96.1	41.2	46.5
output / decision	40	72	546	465	209	265
reasoning share (%)	17	27	62	87	54	82
context read (M)	1.3	4.4	7.8	3.7	3.3	2.2
commands	26	101	–	–	61	41
decisions / command	5.6	2.3	–	–	3.1	4.2

Caveats Wall time includes provider latency and server load. Many games ran in parallel, some under provider rate limits, so wall time describes the deployed systems and not the models alone. Token counts follow each provider's accounting, normalized so that output includes reasoning. The measurements are frozen per game, together with the hash of each run manifest, in the released evidence files.

G Astra, Sol and Luna

We also ran two smaller OpenAI models, GPT-5.6 Sol and GPT-5.6 Luna (OpenAI, 2026c), on the protocol that GPT-6 Astra (OpenAI, 2026a) plays with code. All three run through Codex CLI 0.153.4 (OpenAI, 2026b) at medium reasoning effort, with native tools and no instruction against code. Each plays the 20 target rules of the main cohort from a cold start and with file memory, for 40 games per model and 120 in total. The world, budgets and prompts are those of the main cohort (Appendix B). Astra's games here are its code games in the main cohort. The frozen main-cohort snapshots lack one of them, so the main analysis retains one fewer (Appendix C.1). 12 of the 120 games ran under an earlier version of the protocol that limited the post-game reflection to a fixed time. The other games let the reflection run to completion. This comparison holds the provider and harness fixed, so its differences come from the models.

Supported identifications Table 7 gives supported identifications by condition and start, with exact binomial intervals. Astra identifies 11 of 20 rules from a cold start and 11 with file memory, and every exact answer it gives is supported. Sol identifies 4 and 5. It also gives 3 exact answers while other rules still fit the evidence. Luna identifies none. No model identifies a rule with length-six patterns and the larger catalog. Sol and Luna cost less per call than Astra but identify fewer rules, which matters when a cheaper model is chosen for autonomous research. We did not measure whether a cheaper model lowers the total cost of a research task.

How the games are lost The three models lose 86 games. In 61 of them the agent commits a wrong rule that is still in H_t. In 17 the committed rule lies outside H_t, and 16 of these are Luna's. The remaining 8 games reach the decision limit without a commitment. The timing of the wrong commitments inside H_t separates the models. Astra and Sol make 40 of them, all at decision 227 or later, when almost no budget remains. Luna's 21 come earlier, at a median decision of 161. In 56 of all 61 such commitments, the agent stated a probability below one half that its answer was the only one consistent with the evidence.

Evidence acquisition Games without a supported identification still differ in how much evidence they collect. On the hardest condition no model identifies a rule. From a cold start, Astra's evidence there removes 29.5% of the initial log support on average, Sol's removes 23.6% and Luna's 8.1%.

With file memory the order changes to Sol at 30.3%, Luna at 23.1% and Astra at 16.3%. On this condition the 95% bootstrap intervals of the three models overlap. They resample the 5 targets 10,000 times, keeping both games of a target together (Efron, 1979).

File memory With file memory, Astra's mean capped cost falls from 179.8 to 154.5 decisions, and its number of supported identifications stays the same. The capped cost counts the decisions used for a supported identification and the full budget for any other outcome. Sol's cost changes little, from 207.6 to 205.2. Luna never identifies a rule, so its capped cost is the full budget in both starts. Each condition has a single target order, so these differences cannot be separated from differences between the targets that happen to come later in a stream.

Reported uncertainty Figure 11 compares each model's reported answer entropy with the log support of its evidence, in the form of Figure 10 for the main cohort. The export for these games stores the report made before each evidence action and the reported gain after it. Their difference gives the post-observation report used in the main cohort. Astra's reports follow its evidence closely, and Sol's lie slightly above its evidence. Luna reports much less uncertainty than its evidence supports, starting with its first reports.

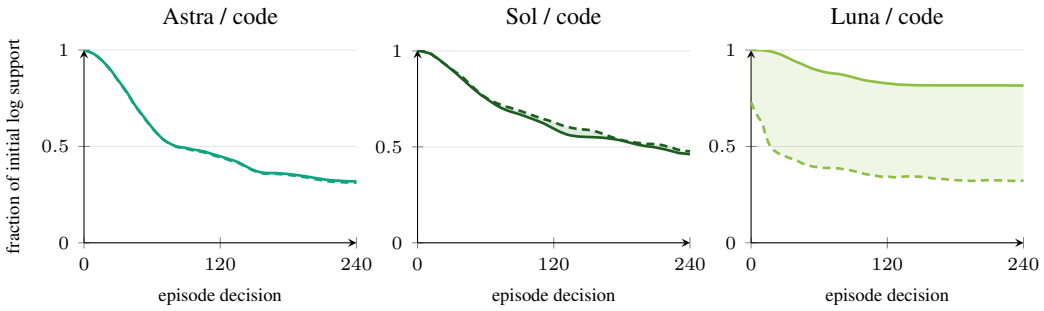


Figure 11: Astra, Sol and Luna with code: remaining log support $\log_2 |H_t| / \log_2 |H_0|$ (solid) and reported answer entropy as a fraction of the prior (dashed), averaged with equal weight over conditions and starts. The shaded area is the gap between them. Post-observation reports are recovered from the export as the pre-action report minus the reported realized gain.

Predicted information gain and stated confidence We compared the gain that agents predicted for each evidence action with the referee's expected gain. Astra's predictions track the referee closely. Its mean absolute error for expected gain from a cold start is 0.007 bits. In the main cohort without code, Astra's error on the same measure is 0.080 bits, so its accuracy with code likely comes from its solver. Sol's cold-start error is 0.197 bits and Luna's is 0.537. Many of Luna's reports predict a gain for actions that the referee expects to be uninformative. The probabilities stated at commitment follow the same order. Astra's cold-start Brier scores are 0.001 for correctness and 0.000 for uniqueness. None of Luna's answers was correct, but it gave a probability of correctness above one half in 6 of the 37 commitments it reported on. Actions within a game are dependent, so these errors are descriptive.

H Formal Difficulty of the Temporal Grammars

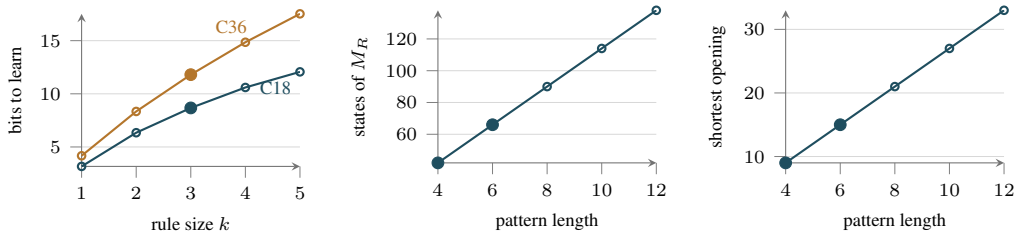
The generator's settings determine quantities that can be computed exactly from the pattern construction, without running any agent (Figure 12). Rule size and catalog size set how much must be learned, $\log_2 |\mathcal{R}_k(G)|$ bits. Pattern length sets what each experiment costs. Both the number of states of the rule's minimal DFA M_R and the length of the shortest input that opens the shrine grow linearly with it. The construction $a\,(u\,v)^m\,e$ extends past the two lengths we evaluate, so the same generator produces harder settings when needed.

The state counts and opening lengths are medians over canonical rules, and individual rules can differ. Opening lengths count offering symbols and leave out the world actions needed to produce them.

system	arm	L4 C18	L4 C36	L6 C18	L6 C36	all
Astra	cold start	5 (48–100)	4 (28–99)	2 (5–85)	0 (0–52)	11 (32–77)
	file memory	5 (48–100)	3 (15–95)	3 (15–95)	0 (0–52)	11 (32–77)
Sol	cold start	3 (15–95)	0 (0–52)	1 (1–72)	0 (0–52)	4 (6–44)
	file memory	3 (15–95)	0 (0–52)	2 (5–85)	0 (0–52)	5 (9–49)
Luna	cold start	0 (0–52)	0 (0–52)	0 (0–52)	0 (0–52)	0 (0–17)
	file memory	0 (0–52)	0 (0–52)	0 (0–52)	0 (0–52)	0 (0–17)

Table 7: Supported identifications of Astra, Sol and Luna with exact 95% intervals in percent (Clopper & Pearson, 1934), out of 5 games per condition and 20 per start. The intervals treat the games of a cell as independent. Games in a file-memory stream follow one another, so this assumption does not hold for them.

Neither quantity gives the number of decisions needed for unique identification. The generator's check that an accepting state is reachable also ignores budgets.



(a) (b) (c)

Figure 12: Difficulty is a computable function of the generator's dials. (a) Bits to learn, $\log_2 |\mathcal{R}_k(G)|$, against rule size k for the smaller (C18) and larger (C36) catalog. (b) Median states of the rule's minimal DFA M_R and (c) median length of the shortest input that opens the shrine, against pattern length at $k = 3$, over every canonical rule of the smaller catalog. Filled points are the settings we evaluate. Open points are untested settings of the same construction.

I Development Experiments

Before the main cohort, we ran two smaller experiments on development seeds, which are disjoint from the seeds of the main cohort. They guided the choice of grammar and budget and are not used for any comparison in the main text.

I.1 Budget Panel on the Base Grammar

The budget panel uses a base grammar of 14 readings about the content, fusion, count and order of offerings. Five systems play it: Astra, Sol and Luna through Codex CLI 0.153.4, Gemini 3.8 Flash (Google DeepMind, 2026) through Antigravity CLI 1.2.3 (Google, 2026) and Muse Spark 1.3 (Meta AI, 2026) through OpenCode 1.18.29 (OpenCode, 2026). Each system plays 45 games under a standard budget and two tighter arms (Table 8). One arm halves the per-level action budget and the other halves the decision budget of the whole episode. Gemini ran partly through a subscription login and partly through a metered API key, with the same model and settings.

Astra gives 43 exact answers, all of them supported. Sol gives 39 exact answers, of which 28 are supported, and Luna gives 11, of which 1 is supported. Gemini and Muse both reach 23 supported identifications. Their exact answers differ by 3, and the difference consists entirely of underdetermined answers. On this grammar, exact-answer scoring therefore separates two systems that the evidence does not separate.

The base grammar is a poor difficulty scale. The number of candidate rules peaks below the largest tier (Table 8, right), so adding readings does not always add hypotheses. This result led us to the temporal grammars, whose difficulty follows from pattern length and catalog size (Appendix H).

arm	level actions	model decisions
standard	72	240
local-tight	36	240
episode-tight	72	120

tier k (readings in the rule)	candidate rules
1	13
2	65
3	158
4	209
5	160

Table 8: Budget panel on the base grammar. Left: budget arms. Right: candidate rules $|\mathcal{R}_k(G)|$ per tier k.

I.2 Temporal Pilot

The pilot tested whether the temporal grammar makes the game harder by changing one setting. It uses the smaller catalog, whose 18 readings give 408 candidate rules at $k = 3$. On a given seed the length-four and length-six grammars use corresponding patterns and the same maps and role bindings, so the first two rows of Table 9 differ only in pattern length. Worlds are small and fully open, with no guards. Besides the length-six game at the standard budget, we ran it with twice the decision budget and with a direct interface. The direct interface removes navigation and collection and charges the cost of building an offering in virtual actions. Astra and Gemini play the same 3 seeds on each arm. A scripted control that reads only public observations and picks experiments greedily from a fixed library checks whether an arm can be solved within its budget.

pattern length	interface	decisions	scripted control	Astra	Gemini
4	situated	240	2/3	3/3	2/3
6	situated	240	1/3	1/3	0/3
6	situated	480	2/3	3/3	1/3
6	direct	240	3/3	3/3	0/3
all arms			8/12	10/12	3/12

Table 9: Supported identifications per arm of the temporal pilot, out of 3 development seeds each.

Astra identifies the rule in 10 of 12 games and Gemini in 3. No exact answer in the pilot was underdetermined. Lengthening the patterns from four to six lowers both systems. At the standard budget with length six, the scripted control also fails on most seeds, so this arm may be too tight for any policy. The two relaxed arms give the control 5 of 6 identifications and Astra 6. Gemini identifies 1 there, and on the direct interface it reached the budget limit without committing in all 3 games. The pilot has 3 seeds per arm, so we do not rank systems on it. It motivated running both pattern lengths on fresh seeds in the full world of the main cohort.