INSTANCES OF EQUAL COMPUTED DIFFICULTY DIFFER IN LLM AGENT SUCCESS BY MORE THAN CHANCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Agent benchmarks usually set difficulty by how many models failed a task, building the axis from the systems measured: a fall in success along it is not itself evidence about a model. Does difficulty computed from the task's own structure predict how often an LLM agent wins? An instance is one setup in a small autobattler: the agent buys and places three units, one at a time, against a script. Its 3,240 complete plays can all be listed, and its difficulty d is the fraction that win: larger d means an easier instance. Every unit costs the same and the agent spends its whole budget, so choosing at random at each step draws uniformly from those plays: random play wins with probability exactly d. Two LLM agents, luna and glm, played 126 instances from the range's two ends, as did a third model reported separately, a random player and 21 scripted strategies. An analysis plan fixed four Holm-adjusted tests in advance; all four reject their null hypotheses. Success rises with computed d ($p = 0.0004$), and luna and glm each beat random play's 0.100 at $d = 0.1$, one test each. No instance sits at $d = 0.1$, fixed in advance between the ends, so both come off a fitted curve. The fourth rejects too: instances of equal computed difficulty differ widely in success, beyond what difficulty explains; structure is not all that makes an instance hard for a model. Instance for instance, luna and glm respond to difficulty about as steeply as random play (90% interval 0.92 to 1.30, within the pre-registered range 0.6 to 1.4). That steepness is a slope, 1 for random play by construction. The plan's model assumed one shared slope; the agents' separate fits differ. The slope does not tell you how well a strategy plays: four scripted strategies spanning the design's range of information and search have slopes within about 0.05 of one another and of random play's fitted 1.03. Computed difficulty does predict success, but is no substitute for measuring models: a difficulty-graded suite should report the spread at equal difficulty beside the slope. These results rest on two agents, one game and one prompt.

1 INTRODUCTION AND CONTRIBUTIONS

On agent benchmarks, difficulty is usually set by how many models already failed a task, which builds the difficulty axis out of the systems being measured. A fall in success along that axis is not by itself evidence about a model. Difficulty computed from a task's own structure, before any model runs it, would say where a model will struggle without paying to run it, let you build a test set at a difficulty you pick, and separate a near-impossible task from a model that is merely bad at it. Does such a measure predict how often an LLM agent wins?

An instance is one fixed setup in a small autobattler: the agent buys three units, one at a time, and places them against a script. Each instance has 3,240 ordered complete plays, and the fraction that win is its difficulty d; larger d means an easier instance. Every unit costs the same and the agent spends its whole budget, so choosing at random at each step draws uniformly from those plays: uniform random play wins with probability exactly d. An agent's success is therefore compared with random play's own rate on the same instance, not with other agents' scores. Exactness alone would not give that comparison: a difficulty can be exact without listing every play. An analysis plan was fixed before any of the data it analyses were collected. It declared four tests on 126 instances from the low and high ends of d, every p-value Holm-adjusted across all four: whether success rises with d; whether luna and glm, two LLM agents, each win at $d = 0.1$ at a rate different from random

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.

play's, one test for each; and whether instances of equal computed difficulty differ in success by more than chance.

Success is higher on instances with higher computed d (Holm-adjusted $p = 0.0004$), and at $d = 0.1$, where no instance sits, both agents beat random play's 0.100: on the headline fit, averaged over instances, 0.2912 for luna and 0.2575 for glm, both rates read from the fitted curve. Instances of the same computed difficulty still differ widely in how often an agent wins, beyond what difficulty explains: the fitted spread of instances around that curve, $\hat{s}$, is about 1.4 in log-odds. Instance for instance, luna and glm respond to difficulty about as steeply as random play. The plan's model gave both agents one shared slope; fitted separately, their slopes differ (Section 5). Structural difficulty is worth computing, but it is no substitute for measuring models: it does not say which instances a model will find hard, and a slope near random play's barely separates strategies spanning this design's whole range of information and search (Section 6).

- Each instance's difficulty is computed exactly from its structure alone, as in CalBench (Zou et al., 2026), and here equals uniform random play's win probability, so the axis carries its own baseline. This paper tests that axis rather than using it (Section 3).
- Instances of identical computed difficulty differ widely in agent success, beyond what difficulty explains, so the measure does not capture all of what makes an instance hard for a model (Section 5).
- Four scripted strategies that differ greatly in what they see and how far they search have nearly the same slope (Section 6).
- Handed the exact worth of every legal move, the agents take the best one essentially every time: where they fall short is in working out what a move is worth, not in choosing among them (Section 7).

2 Related work

One agent benchmark already computes difficulty from a task's structure, before any model runs. CalBench (Zou et al., 2026) scores each task in its 90-task scheduling suite by the exact fraction of all schedules that CP-SAT, a constraint solver, finds feasible. This paper builds d the same way, counting complete plays that win against a fixed opponent. Equal unit prices and the agent's fully spent budget make every complete play equally likely, so uniform random play wins with probability exactly d. The same arithmetic holds for CalBench: a schedule drawn at random is feasible with probability equal to its task's score. But a feasible schedule is not a win when several agents hold private information and must coordinate. CalBench names no random play and runs none as a comparison. CalBench uses that score descriptively. It sorts tasks into easy, medium and hard buckets, names those buckets among its contributions, and breaks results out by model, cost setting, speaker role and failure pattern. It fits no slope of success on that score and never tests it. This paper fits one. Unlike a bucket label, a slope can be checked against data and come out wrong. This paper did not fit that slope on CalBench's 90 public tasks, whose exact difficulty makes a random schedule feasible with that same probability.

Others have measured an agent's shortfall against a computed reference, then split it in two. Deb & Krishnan (2026) splits agent failure against a reference policy into a perception term and an action term: misreading the world, and acting badly on it. Liu (2026) is a precursor rather than a competitor: it prices five scaffold components with exact Shapley values over all 32 subsets. This study prices none of its own. The nearest neighbour, Grams et al. (2025), supplies the algebra: the split runs against the best available play. Its comparison policy runs on the agent's own memory, so it knows what the agent knows. Here the comparison play knows more: it runs against the opponent's finished board, which the agent has not seen when it places its units (Section 8). Three nearby studies each cover one part of agent quality: world-model accuracy beside play quality (Lehrach et al., 2025), decision optimality named but never computed (Cipolina-Kun et al., 2025), and search depth read off reasoning traces (Chen et al., 2026b). None combines the parts into one agent's shortfall. Other papers own both names this split invites. "Planning error" belongs to TravelPlanner (Xie et al., 2024), which titles its Section 5.2 "Planning Error Analysis". World-Time Compute (Schwoebel et al., 2026) uses "model error" for a learned simulator gone wrong. The second experiment (Section 7) names its own.

Another line of work sets difficulty deliberately. Shojaee et al. (2025) and Mirzadeh et al. (2025) turn up one setting and report success falling as it rises. Difficulty here comes from each instance's own structure, which also fixes a baseline: how often uniform random play wins. Of the work compared in Appendix F, Malek et al. (2025) comes closest to separating simulating from planning on one environment. It asks a model for the two separately; here both vary inside a single task that is scored one way throughout. That appendix compares one tower-defence benchmark, four further world-model and planning studies, and Yuan et al. (2025), which scores language models on strategic games by metrics beyond win rate. This paper takes CalBench's construction into a different environment, adds the random baseline it already implies, and tests whether computed difficulty predicts how often an agent wins, rather than assuming it does.

3 THE GAME, AND HOW AN INSTANCE'S DIFFICULTY IS COMPUTED

An instance is one fixed setup in a small two-sided autobattler: each side buys units and places them on a board, then the battle runs itself with no further choices. The agent holds 3 gold and a shop of three unit types at 1 gold each, and spends it all on three placements, each filling one of the six squares on its own side. The opponent is a fixed script, dealt 5 gold and a shop of those three types plus a fourth. The script places four units in one of nine compositions, on squares fixed before the agent moves. Those units on their squares are its opponent board, the one thing that differs between instances; that list in placement order is the opponent's queue. The two sides place in turn, the agent first, and once both have placed the battle is deterministic: the same placements always give the same result. A complete play is the agent's three placements against one opponent board. The agent either wins that battle or does not.

Each instance has 3,240 ordered complete plays, few enough to list them all, a count Appendix A derives; that list is the enumeration behind d. An instance's difficulty d is the fraction of those plays that win, so an instance where one complete play in ten wins has $d = 0.1$, and larger d means an easier instance. Nothing about d is sampled or estimated: it is computed from the instance's own structure before any agent runs.

Every unit the agent can buy costs the same, so no earlier choice makes a later one unaffordable: at every step the options are those same three types and whatever squares are still free. Spending the whole budget fixes every complete play at three placements. At each step every play faces the same number of options, whatever came before, 18 then 15 then 12. So choosing uniformly at each step draws uniformly from the 3,240 ordered complete plays, and random play wins with probability exactly d.

The agent sees each opponent unit only as that unit is placed, so it commits its first unit having seen none of them. The enumeration runs every play against the finished board, so d rests on information the agent never has (Section 8).

The enumeration supports one claim about d: it is the win probability of uniform random play on that instance. d measures nothing else. It describes the instance's structure, not how hard the instance is for a model. The registered tests ask whether a model wins more often on instances with higher d, and how steeply its success rises with d. That steepness is the slope the analysis estimates.

4 DESIGN

The suite is 126 instances at the two ends of computed difficulty d: 0.0164 to 0.0265 at the low end, 0.1735 to 0.3762 at the high end, and nothing in between. The design therefore cannot test the shape of the curve between the ends, and any claim about that shape is exploratory rather than one of the confirmatory analyses the plan fixed in advance. No two of the suite's opponent boards mirror each other, which would repeat a test rather than add one (Westfall et al., 2014).

An arm is one player the suite was run against, and Table 1 gives the four: luna and glm, the two headline agents the main results are about; a third model, gemma, reported apart from them; and a random player. Random play is the null arm and a check on the harness, the software that ran the suite. Each arm played every instance ten times, and one attempt is one complete play of three placements: $126 \times 10 \times 3 = 3{,}780$ placements per arm, with all $4 \times 126 = 504$ arm-and-instance cells played in full. The three model arms ran one prompt configuration, `I1_E0_R0`: rules, no

examples, no reflection. Calls to luna and glm set the provider's reasoning effort to medium and calls to gemma set none, so part of what separates the agents is reasoning against non-reasoning, which the plan marks as an inference rather than a measurement. No call set a sampling parameter or a request seed, so each model arm sampled at its provider's default, which the run records nowhere.

When a model's reply could not be used as a placement, the harness placed for it; Table 1 counts those fallbacks. The random arm had no API key, so every one of its placements was random and counts as a fallback by construction, not as a harness failure. The fallback seed does not depend on which model is playing, so a model's fallback on an attempt draws on the same random numbers the random arm used for that attempt. Since the draws are not independent, no confirmatory quantity compares a model's fallbacks with the random arm.

The registered model is a generalised linear mixed model, binomial on the wins each of luna and glm takes from its ten attempts on an instance, fitted at $Q = 15$ adaptive Gauss-Hermite quadrature points. It estimates a separate starting point $a[m]$ for luna and glm; γ, how steeply an agent's odds of winning rise along $\mathrm{logit}(d)$, the log-odds of computed difficulty; and s, the spread in an agent's win rate among instances of the same computed difficulty. In the plan's notation, where *puzzle* means an instance and *GLM* is the agent this paper calls glm,

$$\begin{aligned} \mathrm{logit}\, P(\text{win} \mid \text{model } m,\ \text{puzzle } p) &= a[m] + \gamma \cdot \mathrm{logit}(d[p]) + u[p], \\ u[p] &\sim N(0, s^2), \quad \text{one } u[p] \text{ shared by luna and GLM} \end{aligned} \tag{1}$$

where the instance effect $u[p]$ is how far an instance sits above or below the fitted curve; the median instance is the one whose $u[p]$ is zero. One slope and one instance effect cover both agents rather than one each, and the design fixed these two as assumptions. That pooled fit is the headline fit; Section 5 reports what each agent's own fit gives instead.

The registered family is four tests, with every p-value Holm-adjusted across all four at $\alpha = 0.05$ (Holm, 1979). Q1 asks whether agents win more often on instances with higher computed d. An agent's level is its win rate at $d_0 = 0.1$, read off the fitted curve at a difficulty fixed in advance in the gap between the ends. The level test asks whether that rate differs from random play's level there, which is 0.100 because random play wins with probability exactly d. It runs once for each headline agent (Q3, luna and Q3, glm), so it is two of the four. Q4 asks whether instances of the same computed difficulty differ in win rate by more than chance alone would produce. That excess is misfit, and s measures it.

A level has two readings: the marginal rate is the agent's average win rate across instances at that difficulty, and the conditional rate is its win rate on the median instance (Heagerty & Zeger, 2000; Hubbard et al., 2010; Hedeker et al., 2018). The conditional reading depends more than the marginal one does on the assumption about how instances vary. The plan fixes the marginal one; Appendix B quotes its wording. An interval is reported beside every registered test (Cumming, 2014; Madaan et al., 2024), errors are grouped by instance, and differences between arms are paired within an instance (Miller, 2024). Intervals on registered quantities are 90% intervals, as the plan fixed.

"Converged" is the plan's own two-part verdict, not a solver's exit code: the fit had to settle, and it had to land in the same place when restarted from elsewhere. Had the headline fit failed either part, the slope, its interval and the levels could not be reported as confirmatory. Appendix B gives the plan's definition in full, with the fitter's tolerances and test-run record, and reports that the headline fit passed both.

The plan was committed and hashed before any of the data it analyses were collected, and Section 7 names the earlier run the scoping is written around. It fixed each outcome's interpretation and its reporting sentence, and it is registered under an embargo that records the hash rather than the text. The plan's file sits in a public git repository, but the commit holding it was never pushed there. The registration's sealed summary says the reverse: that the repository has not been published. That summary cannot be edited, so the correction is made here. No reader can check the hash or read the registration during review, so the pre-commitment rests on disclosure. A registered rule could have added 72 more instances. It ran blind on 2026-09-17, before any confirmatory quantity was computed, returned `stop`, and the suite stayed at 126 (Appendix G).

Arm	Role	Model stamp	Host pin	Quant	Effort	Fallback rate
luna	headline	openai/gpt-5.6-luna	openai	none	medium	194/3,780=0.0513
glm	headline	z-ai/glm-5.3-flash	parasail	fp8	medium	242/3,780=0.0640
gemma	reported separately	google/gemma-4-31b-it	parasail	fp8	none	4/3,780=0.0011
random	null arm and harness check	none	keyless			3,780/3,780=1.0000

Table 1: The four arms, all at prompt configuration I1_E0_R0. A fallback is an agent prep placement whose chooser is not model, counted by fallback_reason (Appendix G lists the five).

5 CONFIRMATORY RESULTS

Four arms played the 126 instances, ten attempts each (Table 1). Difficulty d is the exact fraction of an instance's plays that win, so larger d means an easier instance. All four registered tests reject their nulls (Section 4): difficulty, both agents' levels, and misfit.

5.1 INSTANCES OF EQUAL COMPUTED DIFFICULTY DIFFER IN SUCCESS BEYOND WHAT DIFFICULTY EXPLAINS

In the plan's wording, where *puzzle* means instance and φ a dispersion statistic: "Puzzles of equal computed difficulty differ in model success beyond what difficulty explains ($\varphi = 3.2342$; $\hat{s} = 1.4059$, 90% CI [1.1990, 1.6484]; Wald on log s; coverage not simulated)." Two readings give its size. Spearman ρ says how tightly an arm's success follows computed difficulty, 1 being perfect: the random arm, whose win probability is d by construction, reaches 0.7620, and every model tracks d less closely, glm 0.6801, luna 0.5743 and gemma 0.5123, each $p = 0.0001$ against zero, not against another arm. And which instances are hard beyond d is model-specific: on the same instances, luna's spread around its own fitted curve is more than twice glm's (point estimates; the plan registers no interval for them). So computing difficulty exactly from an instance's structure does not capture all of what makes that instance hard for a model.

5.2 INSTANCE FOR INSTANCE, LUNA AND GLM RESPOND TO DIFFICULTY ABOUT AS STEEPLY AS RANDOM PLAY

A slope of 1 means an agent's odds of winning rise with computed difficulty exactly as fast as uniform random play's do. Random play sits at 1 by construction; fitted from ten attempts an instance, the scripted random strategy in Section 6 returns 1.03. Printing that header needed what the plan calls a licence, a condition fixed in advance: "Puzzle for puzzle, luna and glm respond to difficulty about as steeply as random play (90% CI [0.9248, 1.2988], within the pre-registered range 0.6–1.4)." An edge test at each edge of the range asks whether the slope sits there, and both edges reject it (Table 2). The interval must also fall inside the range, and it does: it excludes 0, sits above 0.6 and contains both 1 and random play's fitted slope, the values the plan named. Eleven of the fifteen scripted strategies with a fitted slope, of 21 in all, fall inside 0.6 to 1.4 (Figure 1). γ is one slope shared by luna and glm by registered construction, pooled at 1.11. On their own fits they do not share it: luna's is 1.4429, a point estimate above the range's upper end of 1.4, with a 90% CI [1.1056, 1.7802] that excludes 1, and glm's is 1.0079, a difference of 0.4350 registered as descriptive with no test, whose interval excludes zero. The registered common-slope assumption is contradicted by the data it was fitted to: 1.11 is fitted to both agents at once, not averaged from 1.44 and 1.01.

Success does rise with computed difficulty: "Model success is higher on puzzles with higher computed d (permutation test, Holm-adjusted $p = 0.0004$)." Both agents also beat random play's 0.100 at difficulty 0.1, on the headline fit. For luna: "At difficulty 0.1, luna wins more often than random play on average over puzzles (Holm-adjusted $p = 0.0040$; marginal rate 0.2912, 90% CI [0.2525, 0.3332]). On the median puzzle, its conditional rate is 0.2298 (90% CI [0.1861, 0.2802])." For glm: "At difficulty 0.1, glm wins more often than random play on average over puzzles (Holm-adjusted $p = 0.0040$; marginal rate 0.2575, 90% CI [0.2210, 0.2977]). On the median puzzle, its conditional rate is 0.1923 (90% CI [0.1539, 0.2377])." The two rates differ because the marginal rate averages

	Estimate	90% CI	p	Holm p
Q1, slope above zero			0.0001	0.0004
Q3, level, luna			0.0020	0.0040
Q3, level, glm			0.0020	0.0040
Q4, misfit			0.0010	0.0030
Licence test, lower edge			0.0005	
Licence test, upper edge			0.0065	
$\hat{\gamma}$, pooled (headline)	1.1118	[0.9248, 1.2988]		
$\hat{\gamma}$, luna, own fit	1.4429	[1.1056, 1.7802]		
$\hat{\gamma}$, glm, own fit	1.0079	[0.8293, 1.1864]		
$\hat{\gamma}$, gemma, own fit (reported separately)	1.6834	[1.2005, 2.1663]		
$\hat{\gamma}$, luna minus glm (descriptive, no test)	0.4350	[0.0562, 0.8138]		
Level at $d_0 = 0.1$	Marginal, 90% CI		Conditional, 90% CI	
luna, headline fit	0.2912	[0.2525, 0.3332]	0.2298	[0.1861, 0.2802]
glm, headline fit	0.2575	[0.2210, 0.2977]	0.1923	[0.1539, 0.2377]
gemma, own fit (reported separately)	0.1910	[0.1493, 0.2409]	0.0479	[0.0253, 0.0891]

Table 2: All four registered tests reject their nulls, and every estimate row names the fit it came from. A blank cell means the column does not apply to that row, not a result that is missing: the test rows carry p values and no estimate, the slope and level rows carry a point estimate with its interval and no p, and the licence edges are unadjusted and so carry no Holm p. Slope intervals are Wald t. The luna minus glm row is descriptive with no test; its interval is a normal approximation, covariance ignored. Level intervals carry the pre-registered label "delta method; simulated coverage .86–.90". Steepness, which the plan numbers Q2, is registered as an estimate rather than a test, so it is here as the pooled slope and its interval, with no p value and in no family. Each unadjusted p value comes from resampling, as $(b+1)/(B+1)$ for b replicates at least as extreme as the observed statistic out of B resamples; one at its own floor, such as 0.0001 on 9,999, means no replicate was that extreme.

over the spread between instances and the conditional rate reads the median one. Every level interval here and in Table 2 carries the pre-registered label "delta method; simulated coverage .86–.90". Both level tests are read where no instance sits: $d = 0.1$ was fixed in advance between the suite's two ends, and the curve between the two ends is an assumption rather than an estimate, so any look at its shape is exploratory. A registered check that fits no curve agrees with the level tests on direction without replacing them; Appendix B gives each arm's value with its interval.

Four statements ride with the steepness sentence above. Design sensitivity: "At a puzzle-effect SD of 2 and a true slope of exactly 1, this pre-registered rule licensed the phrase in 53% of 300 simulated studies, with a false-licence rate of at most .063 at the upper edge of 1.4; at SDs of 1.5 and 2.5 only a stricter fixed-126-puzzle stand-in was simulated (84% and 20%). This study's estimated SD is $\hat{s} = 1.4059$ (90% CI [1.1990, 1.6484]; Wald on log s; coverage not simulated)." At that SD of 2, the rule granted the phrase in about half of 300 simulated studies in which the slope really was 1. Fallbacks: "On pure attempts (selected subset: attempts in which every agent placement was the model's; 0 puzzles dropped for a headline model with no pure attempt), the headline refit gives $\gamma = 1.1439$ (90% CI [0.9391, 1.3487], row S2 read mechanically without a licence test). Attempts with at least one fallback: luna/`I1_E0_R0` 15.3175% (193 of 1260); glm/`I1_E0_R0` 17.6984% (223 of 1260)." A floor, a win rate no agent drops below however hard the instance: "A flat win-rate floor near $d = 0.028$ would bias γ by about -0.12, toward shallower than random play, and this design cannot check it"; on 126 instances the plan runs no floor refit, and none was run. The marginal scale, flatter than the conditional one by a square-root factor (Pavlou et al., 2015): "Averaged over puzzles, a model's curve is flatter than its conditional slope: at $\gamma = 1$ the marginal slope is 0.79 / 0.68 / 0.59 at $s = 1.5$ / 2 / 2.5."

gemma is reported separately and is not in the headline fit. Its own fit puts its two level readings at $d = 0.1$ on opposite sides of random play's 0.100: marginal 0.1910 above, conditional on the median instance 0.0479 below. Its own slope is 1.6834, a point estimate above the range, with a 90% CI that excludes 1.

The fallback screen names any model arm the harness placed for too often, and names none: no model arm's fallback rate reaches the plan's 0.106 cut at the bottom of its 95% interval. The harness

check asks whether the random arm won as often as $\mathrm{Bin}(10, d)$ says it should, and finds no departure ($G = 138.6470$, the departure statistic; $p = 0.0730$ against a flag threshold of 0.01), so the random arm behaved as the construction says it must.

6 FOUR VERY DIFFERENT SCRIPTED STRATEGIES HAVE NEARLY THE SAME SLOPE

The anchor ladder, 21 scripted strategies the plan fixed in advance, shows what slopes non-random play produces. No language model plays them: each is a script, so their win probabilities on the same 126 instances are computed exactly. Fitting a slope needs attempts, so their play is simulated too, 40 runs of ten attempts on every instance, as the agents had. Appendix C defines all 21 and prints the table.

Four of them differ about as much as any four here in information (what each one sees of the opponent) and in search (how far ahead it looks). `front_row` at 1.09 uses nothing about the opponent and does not search. `counter_visible` at 1.07 aims its units at the opponent units on the board; `counter_queue` at 1.12 aims at the whole queue, including those not yet placed. Neither searches. `lookahead_visible` at 1.12 searches over all its own placements, but scores each as if the opponent placed nothing more.

In Figure 1 each point is the median of one strategy's converged fits, and its bar covers nearly all of them, not a confidence interval. All four bars cover random play's own fitted 1.03. Those four slopes span about 0.05. The 90% interval on the pooled headline slope is 0.37 wide, roughly seven times that span, and each of the four bars is nearly that wide. The comparison is arithmetic, not a test: at this design's precision the slope does not separate these four strategies.

The softmax strategies play more randomly at higher T, and their slopes cross 1 between $T = 0.25$ and $T = 0.5$, so even the pre-registered range 0.6 to 1.4 holds play far from random. Six of the 21 have no fitted slope, because a fit needs outcomes that vary: `optimal`, `softmax_T0.1` and `rollout_greedy` win essentially every instance, and three always play the same placements.

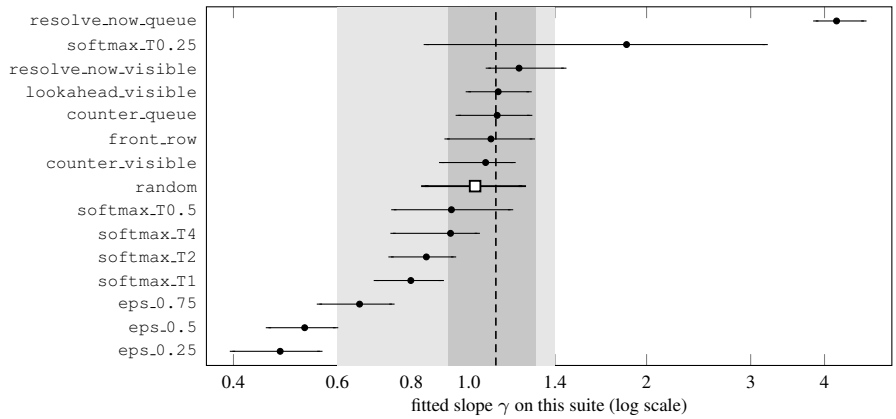
`eps_0.25`, `eps_0.5` and `eps_0.75` win far more often than `random`, yet their slopes are lower than its 1.03: 0.48, 0.53 and 0.65. `softmax_T0.5` wins more often than any of the three, a mean win probability over the suite of 0.88 against `eps_0.25`'s 0.74, and its slope is higher, 0.94. So a slope does not tell you how well a strategy plays.

Which strategies count as sensible non-random play, and how each is built, are judgements the slopes depend on. The plan pins these results by a hash of their file, which fixes the numbers but not their date: the file was written after the plays it records.

A slope γ near 1 describes how steeply success tracks difficulty. It does not say that play is random. Computed difficulty does predict success (Section 5), but it is no substitute for measuring models. Instances of identical computed difficulty differ widely in how often an agent wins, beyond what difficulty explains. The slope barely separates strategies that differ greatly in what they see and how far they search, and does not tell you how well a strategy plays. So anyone building a difficulty-graded test suite should read its slope as steepness rather than capability, and report that spread beside the slope.

7 WHERE THE REGRET IS: VALUING MOVES, NOT CHOOSING THEM

A second, smaller experiment asks where in a play an agent loses ground. Regret is the measure: at each of a play's three placements, the best final margin still reachable minus the margin the move taken leaves, added over the play. Margin is the agent's score minus the opponent's. Handed the exact margin of every legal move, luna and glm take the best one essentially every time: selection regret is 0.0000 and 0.0437 per play. Without those margins their regret is 2.87 and 3.04; without the battle rules as well it is 3.97 and 3.98, about 97 percent of uniform random play's own regret. About three quarters of that goes on working out what a move is worth, and all or almost all the rest on the battle's dynamics.



`random` is the open square; light grey 0.6 to 1.4, dark grey the 90% CI on dashed $\hat{\gamma}$. Not fitted, over their draws not converged, so not plotted (Appendix C): `optimal`, `softmax_T0.1`, `rollout_greedy`, `fixed_front_AAA`, `fixed_back_ABC`, `fixed_front_CAB`.

Figure 1: The scripted anchors on this 126-instance suite: the 15 strategies with a defined fitted slope, each at the median of its converged draws among 40 binomial draws at $k = 10$, bars to the 2nd and the 2nd from last of those, not a confidence interval; counts per strategy in Appendix C.

It played 20 of the same 126 instances, on cells never pooled with the main experiment's; no result above rests on it. Appendix D holds the apparatus, from the five prompts, called rungs, to a placebo contrast that checks the agents read the printed margins, and Appendix D.4 every estimate with its interval.

None of this shows that the agents' search is sound and only their model of the battle is broken. No rung was run that would separate the two, so the experiment can only bound each part, and those bounds assume each agent searches and selects the same way at every rung. The registered check on that assumption tripped for both agents, and this experiment's own plan fixed in advance what a set flag does: the bounds stay two-sided intervals, labelled A1-conditional, and hold only if the assumption does. Appendix D.5 gives the check and the one-sided statements that survive without it.

The experiment's frozen checking tool returned one verdict:

```
DECISION: undecidable:  harness defect or served-provider mismatch
```

Two of its checks look for results at rungs this experiment's own plan excludes, so the rule behind them ran over rungs with no data. Whether that is a harness defect or a served-provider mismatch, meaning the wrong provider served the calls, is not settled here. No re-run was performed; Appendix D.7 prints the verdict in full. These data were collected on the same suite about 17 hours before the plan was frozen and hashed, which is why Section 4 scopes its pre-commitment to the data that plan analyses; the outcomes stayed sealed until 2026-09-17T10:33Z, after the confirmatory analysis had run.

8 LIMITATIONS

The agent places first and blind; the enumeration behind d runs against the finished board, with full information the agent never has. This design cannot separate what the missing information costs from what reasoning costs. d is a full-information property of the instance, not a bound on achievable play, and no distance from it should be read as the room a first-moving player had.

What the agent is given bounds how much is missing: the battle rules, the opponent's shop and gold, and each opponent unit as it is placed. The opponent is a fixed script that never adapts, and its board takes one of nine compositions (Appendix A). So the types still to come are heavily constrained. Where they go is not. Across the 126 instances composition explains almost none of the difference in difficulty ($\eta^2 = 0.00006$). Grams et al. (2025) match their comparison policy to what the agent knows; nothing matched that way can be built here. It would need a prior over the boards still consistent with what the agent has seen, and this environment supplies none. With deterministic

battles the best play for a given board simply wins, so that ceiling is 1 on every instance here and separates nothing.

The misfit result does not say what makes equally difficult instances differ; which board features drive that difference was planned as exploratory and is not analysed here. So $\hat{s} = 1.4059$ should not be read as a second difficulty axis: it says one is missing, not what it is. A flat win-rate floor near $d = 0.028$ cannot be ruled out here, and the plan says it would bias the slope by about -0.12; correcting for a bias that size would move the licensed interval's upper end past the top of the pre-registered 0.6 to 1.4 range.

The straight line the fit draws along $\mathrm{logit}(d)$ between the suite's two ends is assumed, not shown, and any look at it is exploratory. If that line is wrong, the fitted slope is not the slope at any particular difficulty. Two of the four registered tests are read off that line, at $d = 0.1$, where no instance sits.

A provider could swap the model behind an unchanged model name and host, and nothing here would catch it. The provider-constancy check in Appendix G compares each arm's logged provider, and a swap of that kind leaves that provider unchanged, so passing the check rules nothing out. The role prompt given to the agent was trimmed, and the effect of the trimmed text is unmeasured.

All of these results rest on two headline agents at one reasoning-effort setting, one environment and one prompt configuration. The per-step action-space size (the number of choices open at each step) and play length are the same across all 126 instances. Nothing here says whether the results depend on that per-step size or on play length.

9 Deviations from the pre-registration

There are three. None changes a registered estimate; the third changes only what is reported. On 2026-09-17 the analysis printed the misfit sentence with an added label, "Wald on log s; coverage not simulated". The plan pre-wrote that sentence without the label, but the plan's Section 7.2 attaches it to the $\hat{s}$ estimate the analysis emits. The label stays.

On 2026-09-18 the authors built the comparison of the steepness interval against the scripted anchors' slopes, from the anchor set the plan fixes (Appendix C). The analysis produces no such comparison, so it is the authors' work.

Also on 2026-09-18 the authors read the plan's Section 7.1, which requires per-model slopes and levels with intervals, against the plan's Section 8: "Per-model [licence-test], permutation and level-test rows that the spec produces are not reported and enter no family". The spec is the analysis specification; the bracketed name replaces the plan's. This paper follows the plan's Section 7.1: the per-model levels are reported (Table 2; Appendix B), and no per-model licence-test or permutation row appears anywhere in this paper. Read narrowly, the plan's Section 8 excludes level-test rows, not level estimates, and the two sections agree; read broadly, it excludes them too, and the two sections conflict. Which reading the plan intended, and so whether this is a departure, is left open.

10 Conclusion

Success rises with computed d; larger d means an easier instance. Instance for instance, luna and glm respond to difficulty about as steeply as random play. The match is in steepness; it does not mean they play at random. Instances of equal computed difficulty differ in how often an agent wins, beyond what difficulty explains: structure is not all that makes an instance hard for a model. The agents lose most of their ground working out what moves are worth, not choosing among them (Section 7). None of these results says how well an agent moving first could play; all rest on two agents in one environment (Section 8). Whether success also follows the per-step action-space size is untested; varying it at constant play length is deferred, not ruled out. Anyone building a difficulty-graded test suite should report the spread at equal difficulty beside the slope.

AI use statement

A large language model assistant was used throughout this project, in categories the venue requires authors to disclose and in categories it recommends disclosing.

The assistant designed the methodology: the four arms, the number of instances and the number of attempts per cell, the confirmatory estimand of a slope on logit(d) common to the two headline models, with per-model slopes registered alongside it as descriptive secondaries, and the analysis plan that was committed and hashed before the data it analyses were collected. It gave research-methodology feedback on that design, including adversarial reviews of it against comparable published studies, and it refined the hypotheses, turning an informal question into the four registered tests, the Holm family they sit in, and the outcome sentence templates that fix each result's wording in advance. Working out the algebra the design rests on was part of that work: the additivity identity the decomposition arm uses, and the fact that uniform random play's win probability on an instance is that instance's computed difficulty, which is what puts the null arm's limiting slope at 1. It wrote the implementation: the environment, the enumerator that computes each instance's exact difficulty, the agent harness and the analysis code. The instance suite itself was produced by that code deterministically from a fixed seed, not sampled from a model. It also ran the study, launching and monitoring both runs and resuming the decomposition arm after an earlier invocation stopped. It did the data cleaning: the completeness and replay checks on every cell, the fallback screen, the pure-attempt refit, and the reconciliation of logged per-call costs against the account ledger. The only free text this study collects is the models' replies. The assistant fixed in the plan the reasons a reply could not be played, implemented that classification in the harness, and wrote the reading of what those categories show. And it interpreted the results, reading each estimate off the analysis output files and writing what those estimates license and what they do not. One interpretive call was handed to the assistant outright: it read the decomposition arm's harness-defect verdict as a false alarm, on the account Section 7 gives of why the gate rule behind that verdict fired over rungs with no cells, and it decided that no re-run would be performed. That section states the limit on that reading. No data set here was sampled from a model. The plays the study analyses are model output, but they are the measurement rather than material generated to stand in for it. What holds for the instance suite holds for the scripted anchor ladder's draws and the bootstrap replicates: code the assistant wrote produced them, from seeds fixed in the plan. Translation does not arise in this work at all, since the environment, the role prompt, the analysis and every paper read for the related work are in English.

In the categories the venue recommends disclosing, the assistant edited the paper, drafting and revising the prose of every section. It analysed the literature, running the searches behind the related work and behind the near-neighbour comparisons of Appendix F, and identifying the gap in that literature this study speaks to. It proposed the paper's structure, its title and its keywords. It built the figure and the tables, and it assembled the reference list, checking its entries against the arXiv and Crossref records.

The authors set the research direction, decided what would be spent and on what, chose how the headline result is framed, reviewed the work the assistant did, and are responsible for every claim, every line of text and every artifact in this paper.

Ethics statement

This study involved no human subjects and no personal data. Every datum analysed here was produced inside a synthetic game environment that runs locally: the agents see a board, return a placement, and have no tools, no network access of their own and nothing to act on but the game. The environment affects nothing outside itself, and the study touched no external system other than the model endpoints described below.

The models were reached through a commercial routing service, under its terms of use, at the model stamps and host pins listed in Table 1.

The two runs this paper reports cost $9.1544 and $8.2645 in model calls. The confirmatory run's $9.1544 sits against the stage-1 ceiling of $75.00 fixed in the pre-registered run manifest, and the decomposition arm's $8.2645 against the $49.04 budget fixed in its own plan. Appendix G breaks the first figure down by arm and gives the call counts behind it. The feasibility runs described in Appendix E also used paid model calls, and those are not part of either figure. They cannot be totalled: the earliest of them, the gate runs of 2026-09-10, logged no token usage, so no price can be derived from any of their arms, and per-call usage logging landed only after them. The one

feasibility spend this paper reports is the Option 1 pilot of 2026-09-13, whose 32 arm-by-instance cells cost $0.9726 against the $10 ceiling that run was given.

The submission is double-blind, so no funding source and no institution is named here.

REPRODUCIBILITY STATEMENT

The environment, the enumerator that computes each instance's difficulty, the agent harness, the analysis code and the instance suite are not released with this submission. They are held in a repository that is public under the authors' identity, so releasing them now would undo the anonymity this review requires, and an anonymised copy is not a substitute: rewriting a file changes its bytes, and a rewritten copy of any artefact this paper pins would no longer match the digest printed for it. The run logs, which that code produced and which are not themselves kept under version control, are withheld too. All of it accompanies the camera-ready. What stands in their place here is the specification rather than a pointer to code: Appendix A gives the board, the battle rules, the opponent, the enumeration and the construction of the 126 instances, Appendix B gives the fitting, the quadrature, the intervals and each registered test, Appendix D gives the decomposition arm's rungs, statistic and gates, and Appendix G gives the confirmatory run as it happened, including its wall clocks, its spend and call counts, and the fallback screen in full.

Each artefact identified below carries a content hash, so a reader can check that a file they hold is the file the study used without needing any location or account. This statement prints the pre-registration's and the appendices carry the rest.

`e587b3e18e21c229a2e0c3e8a2195d770b2c5907aafda50bd3d871c55a223c80`

The design section states what was committed and when, in the scoping that is licensed, and discloses where the sealed registration summary is inaccurate; neither claim is repeated here. Appendix B prints the analysis spec and run identity hashes beside the procedures they configure, Appendix A prints the suite hash beside the construction of the 126 instances and the role prompt's own hash, and Appendix C pins the scripted anchor ladder's results file by its SHA-256. The decomposition arm is pinned in the same way and separately: Appendix D carries further digests, for that arm's own plan, its gates file, its instance selection file and the two prompt parts it adds to this study's.

Eight seeds fix the random draws in the confirmatory run and in its analysis spec. One is `fallback_seed` 20260913, a field of the run identity, from which the game reseeds for each attempt. The other seven are declared in the analysis spec: `arm_seed` is 20260914 at its top level, and one seed sits on each estimand block, 20260921 for the Q1 permutation, 20260922 for the licence test, 20260923 for Q3, 20260924 for Q4, 20260925 for the raw Spearman correlations and 20260926 for the harness check on the null arm. Appendix B gives the permutation, bootstrap and licence-test procedures these seeds drive, with the resample count each uses. The scripted anchor ladder is seeded separately, from forty further seeds fixed in the plan and recorded in the anchors file with its other settings, which Appendix C reports. The decomposition arm is seeded separately again, under its own plan rather than this one: its 20 instances were selected under seed 1347107531, every per-attempt seed it used derives from run seed 715440741, and every resampling figure it reports comes from report seed 1010. Appendix D gives all three, with the rule the selection seed comes from and the derivation the run seed drives.

REFERENCES

Jixuan Chen, Jianzhi Shen, Haoqiang Kang, Zhi Hong, Qingyi Jiang, Soham Bose, Yiming Zhang, Leon Leng, Amit Vyas, Lingjun Mao, Siru Ouyang, Kun Zhou, and Lianhui Qin. AgentSpec: Understanding embodied agent scaffolds through controlled composition. *arXiv preprint arXiv:2606.14674*, 2026a.

Shiqi Chen, Tongyao Zhu, Zian Wang, Jinghan Zhang, Kangrui Wang, Ruochen Zhou, Siyang Gao, Teng Xiao, Yee Whye Teh, Junxian He, and Manling Li. Why do LLM agents fail in exploring new environments? a world-modeling perspective. *arXiv preprint arXiv:2510.15047*, 2025. Findings of EMNLP 2026.

Sixing Chen, Ji-An Li, Saner Cakir, Sinan Akcali, Kayla Lee, and Marcelo G. Mattar. Extracting search trees from LLM reasoning traces reveals myopic planning. *arXiv preprint arXiv:2605.06840*, 2026b.

Lucia Cipolina-Kun, Marianna Nezhurina, and Jenia Jitsev. Game Reasoning Arena: A framework and benchmark for assessing reasoning capabilities of large language models via game play. *arXiv preprint arXiv:2508.03368*, 2025.

Geoff Cumming. The new statistics: Why and how. *Psychological Science*, 25(1):7–29, 2014. doi: 10.1177/0956797613504966.

Sagar Deb and Ashwanth Krishnan. STOCKTAKE: Measuring the gap between perception and action in LLM agents with a fair oracle. *arXiv preprint arXiv:2607.13618*, 2026. 15 July 2026.

Tim Grams, Patrick Betz, Sascha Marton, Stefan Lüdtke, and Christian Bartelt. Disentangling exploration of large language models by optimal exploitation. *arXiv preprint arXiv:2501.08925*, 2025. Version v3.

Patrick J. Heagerty and Scott L. Zeger. Marginalized multilevel models and likelihood inference. *Statistical Science*, 15(1), 2000. doi: 10.1214/ss/1009212671. With comments and a rejoinder by the authors.

Donald Hedeker, Stephen H. C. du Toit, Hakan Demirtas, and Robert D. Gibbons. A note on marginalization of regression parameters from mixed models of binary outcomes. *Biometrics*, 74 (1):354–361, 2018. doi: 10.1111/biom.12707.

Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70, 1979.

Sukai Huang, Chenyuan Zhang, Fucai Ke, Zhixi Cai, Naim Rastgoo, Gholamreza Haffari, and Hamid Rezatofighi. What we talk about when we talk about LLM planning: Evidence for two distinct planning abilities. *arXiv preprint arXiv:2607.11197*, 2026.

Alan E. Hubbard, Jennifer Ahern, Nancy L. Fleischer, Mark Van der Laan, Sheri A. Lippman, Nicholas Jewell, Tim Bruckner, and William A. Satariano. To GEE or not to GEE. *Epidemiology*, 21(4):467–474, 2010. doi: 10.1097/ede.0b013e3181caeb90.

John K. Kruschke. Rejecting or accepting parameter values in Bayesian estimation. *Advances in Methods and Practices in Psychological Science*, 1(2):270–280, 2018. doi: 10.1177/2515245918771304.

Wolfgang Lehrach, Daniel Hennes, Miguel Lázaro-Gredilla, Xinghua Lou, Carter Wendelken, Zun Li, Antoine Dedieu, Jordi Grau-Moya, Marc Lanctot, Atil Iscen, John Schultz, Marcus Chiam, Ian Gemp, Piotr Zielinski, Satinder Singh, and Kevin P. Murphy. Code World Models for general game playing. *arXiv preprint arXiv:2510.04542*, 2025.

Ming Liu. More is not always better: Cross-component interference in LLM agent scaffolding. *arXiv preprint arXiv:2605.05716*, 2026. arXiv:2605.05716v1 [cs.AI], 7 May 2026.

Lovish Madaan, Aaditya K. Singh, Rylan Schaeffer, Andrew Poulton, Sanmi Koyejo, Pontus Stenetorp, Sharan Narang, and Dieuwke Hupkes. Quantifying variance in evaluation benchmarks. *arXiv preprint arXiv:2406.10229*, 2024.

Idriss Malek, Omar Choukrani, Daniil Orel, Anh Duy Le Dinh, Zhuohan Xie, Zangir Iklassov, Martin Takáč, and Salem Lahlou. LLM-BabyBench: Can language models plan in worlds they can simulate? *arXiv preprint arXiv:2505.12135*, 2025.

Evan Miller. Adding error bars to evals: A statistical approach to language model evaluations. *arXiv preprint arXiv:2411.00640*, 2024.

Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. GSM-Symbolic: Understanding the limitations of mathematical reasoning in large language models. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2410.05229, October 2024.

Davide Paglieri, Bartłomiej Cupiał, Samuel Coward, Ulyana Piterbarg, Maciej Wołczyk, Akbir Khan, Eduardo Pignatelli, Łukasz Kuciński, Lerrel Pinto, Rob Fergus, Jakob Nicolaus Foerster, Jack Parker-Holder, and Tim Rocktäschel. BALROG: Benchmarking agentic LLM and VLM reasoning on games. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2411.13543.

Chanwoo Park, Xiangyu Liu, Asuman Ozdaglar, and Kaiqing Zhang. Do LLM agents have regret? a case study in online learning and games. *arXiv preprint arXiv:2403.16843*, 2024.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology (UIST)*, 2023. arXiv:2304.03442.

Menelaos Pavlou, Gareth Ambler, Shaun Seaman, and Rumana Z. Omar. A note on obtaining correct marginal predictions from a random intercepts model for binary outcomes. *BMC Medical Research Methodology*, 15, 2015. doi: 10.1186/s12874-015-0046-6.

Devin Pereira and Willem Zuidema. Transformers struggle to use their emergent world models: Revisiting the Tower of Hanoi, and the illusion of thinking. *arXiv preprint arXiv:2608.07077*, 2026.

James Schwoebel, Ingrida Semenec, Jenia Rousseva, Marcos Ortiz, Collin Overbay, Christopher Klaus, Anderson Edmond, Manish Bhatt, Rome Thorstenson, Jessica Tsai, and Martin G. Frasch. World-Time Compute with verified Code World Models. *arXiv preprint arXiv:2609.09163*, 2026.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. arXiv:2303.11366.

Parshin Shojaee, Iman Mirzadeh, Keivan Alizadeh, Maxwell Horton, Samy Bengio, and Mehrdad Farajtabar. The illusion of thinking: Understanding the strengths and limitations of reasoning models via the lens of problem complexity. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. arXiv:2506.06941.

Pratyush Singh. Scaffolding the strategist: Architecture-dependent reasoning interventions in Hotelling spatial markets. *arXiv preprint arXiv:2607.09743*, 2026. ICLR 2026 Workshop on LLM Reasoning.

Dawei Wang, Chengming Zhou, Di Zhao, Xinyuan Liu, Marci Chi Ma, Gary Ushaw, and Richard Davison. TowerMind: A tower defence game learning environment and benchmark for LLM as agents. *arXiv preprint arXiv:2601.05899*, 2026.

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

Jacob Westfall, David A. Kenny, and Charles M. Judd. Statistical power and optimal design in experiments in which samples of participants respond to samples of stimuli. *Journal of Experimental Psychology: General*, 143(5):2020–2045, 2014. doi: 10.1037/xge0000014.

Jian Xie, Kai Zhang, Jiangjie Chen, Tinghui Zhu, Renze Lou, Yuandong Tian, Yanghua Xiao, and Yu Su. TravelPlanner: A benchmark for real-world planning with language agents. In *International Conference on Machine Learning (ICML)*, 2024. Spotlight; arXiv:2402.01622v4.

Xiaopeng Yuan, Xingjian Zhang, Ke Xu, Yifan Xu, Lijun Yu, Jindong Wang, Yushun Dong, and Haohan Wang. Tracing LLM reasoning processes with strategic games: A framework for planning, revision, and resource-constrained decision making. *arXiv preprint arXiv:2506.12012*, 2025.

Luan Zhang, Ruochen Zhou, Dandan Song, Zhengyu Chen, Yuhang Tian, Jun Yang, Huipeng Ma, Chenhao Li, Guangyuan Feng, Xudong Li, Yizhou Jin, and Yan Xu. HarnessCompass: Guiding automatic harness evolution toward generalizable and effective agent harnesses. *arXiv preprint arXiv:2608.01918*, 2026.

Table 3: The four unit types. Costs are `engine.UNIT_COSTS` and the behaviours are the four action functions of the battle engine. The rules the agent is shown under prompt configuration `I1_E0_R0` state the same behaviours in the agent's own terms, and they state less than this table does. Section A.6 prints those rules in full beside the gaps it names, so a reader can check that comparison rather than take it. "Ahead" means one row toward the opponent's edge, which is downward for the agent and upward for the opponent. The column directions of `B` and `C` are board-absolute and are the same for both sides.

Unit	Cost	Behaviour when it acts
`A`	1 gold	Removes an enemy in the cell directly ahead. With no enemy there it advances into that cell.
`B`	1 gold	Removes an enemy in the cell one row ahead and one column lower, toward column 0. On column 0 that cell is off the board, so it can only advance. Otherwise, with no enemy there, it advances straight ahead.
`C`	1 gold	The mirror of `B`: one row ahead and one column higher, toward column 2, and it can only advance on column 2.
`D`	2 gold	Ranged. Removes the closest enemy anywhere on the board by Manhattan distance, ties broken by the smaller column and then the smaller row. It never moves and so never escapes.

Chelsea Zou, Yiheng Yao, Selena She, Noah D. Goodman, and Robert D. Hawkins. CalBench: Evaluating coordination-privacy trade-offs in multi-agent LLMs. *arXiv preprint arXiv:2605.09823*, 2026. Version v3, 5 June 2026; cite v3.

A THE ENVIRONMENT AND INSTANCE CONSTRUCTION

This appendix gives the construction behind the environment in full: the board and its units, the battle rules, the derivation of the agent's 3,240 ordered complete plays, the opponent space and its nine compositions, how the 126 instances were selected, the three prompt files the agent was sent, printed in full, and the content hashes that pin the suite file and the prompts.

A.1 THE BOARD, THE UNITS AND THE TWO BUDGETS

The board is four rows of three columns. The agent owns rows 0 and 1 and the opponent owns rows 2 and 3, so each side has exactly six squares and neither can take a square from the other. The agent's units face down the board, toward increasing row, and the opponent's face up.

Four unit types exist, labelled `A`, `B`, `C` and `D`. Table 3 gives each one's price and what it does when it acts. In every instance of the confirmatory suite the agent is dealt 3 gold and a shop of `A`, `B` and `C`, and the opponent is dealt 5 gold and a shop of all four types. The rest of this appendix follows from both budgets and from one stipulation beyond them: the agent's budget is what makes its action space uniform, and the opponent's budget fixes the compositions of Section A.4 only once the opponent's board is stipulated to hold four units, and then only up to that section's exclusion of `AAAD`. Five gold on its own also buys a board of three units, or one of five.

A.2 THE BATTLE RULES

Once both sides have finished placing, the battle runs with no further input from either player and is fully deterministic.

- The two sides alternate steps and the agent takes the first step.
- On its step, a side acts with its highest-priority unit that can act. Priority is by type, `A` before `B` before `C` before `D`, and within a type by placement order, earlier first. If the highest-priority unit cannot act the scan continues down the order, and if no unit of that side can act the step is skipped.
- A unit can act if it can remove an enemy as Table 3 describes, or if the cell ahead of it is empty or off the board. A `D` can act whenever any enemy is on the board.

- A unit that advances off the far edge escapes and is worth 1 point to its owner for the rest of the battle.
- The battle ends when neither side can act, or when exactly one side's units are all gone. A step counter of 200 is carried as a divergence guard rather than as a rule: every step of a real battle removes, moves or escapes a unit, so a real battle ends on its own.
- A side's score is its units still on the board plus its units escaped, and the higher score wins. A tie is not a win: the win flag the oracle counts is the game's own, the agent's score strictly greater than the opponent's.

Because placement order sets priority within a type, the same three units placed in a different order are a different battle and can give a different result. That is why a play is an ordered sequence and not a set, and it is what the next section counts.

A.3 The 3,240 ordered complete plays

A play is one complete legal run of preparation by the agent: an ordered sequence of placements, each naming a unit type and a square, running to the point where the game stops handing the agent a turn. The game stops handing it one when it can afford nothing in its shop or its own rows are full, and the agent cannot stop early, so every legal play spends the budget down.

The opponent never enters this count. It places only on rows 2 and 3, so it can neither take a square the agent might want nor change what the agent can afford, and the agent's options are the same whatever the script did on its turns. The agent's action space is therefore settled by its shop and its gold alone.

With 3 gold and three types at 1 gold each, every complete play is exactly three placements, and before each one the menu is the same three types and whichever of the six squares are still free:

$$3^3 \times (6 \times 5 \times 4) \;=\; 27 \times 120 \;=\; 3{,}240. \tag{2}$$

Equal prices are what keeps that menu constant, since no earlier purchase can make a later one unaffordable, and full spending is what fixes the length at three. The suite's counting code does not use Equation 2: it walks the enumeration itself and counts what it yields, so that the count cannot disagree with the plays actually produced. The walk returns 3,240, and every one of the 126 instances records `play_count` 3,240 in the suite file.

A.4 The opponent, and its nine compositions

The opponent is a fixed non-adaptive script. Its units and their squares are fixed in the instance file before any agent runs, and it places them in a fixed queue order, so nothing it does responds to what the agent places.

Its board always holds four units and always spends its 5 gold in full. The four units are a stipulation of the design and not a consequence of the budget. With one type priced at 2 gold and three at 1 gold, 5 gold can be spent in full in three ways, and all of them fit inside the opponent's six squares: five 1-gold units, one `D` plus three 1-gold units, and two `D` plus one 1-gold unit. Only the second puts four units on the board.

Once the board is stipulated to hold four, the arithmetic leaves nothing to choose. Four units of which k are `D` cost $k + 4$ gold, so four units with no `D` cost 4 and leave a gold unspent, four units with two `D` cost 6 and cannot be paid for, and only $k = 1$ comes to 5. Every opponent board is therefore one `D` plus three units drawn, with repetition, from `A`, `B` and `C`. Drawing three from three types with repetition gives ten multisets. The design excludes `AAAD`, which leaves the nine compositions of Table 4: `AABD`, `AACD`, `ABBD`, `ABCD`, `ACCD`, `BBBD`, `BBCD`, `BCCD` and `CCCD`. The exclusion of `AAAD` was fixed in the registered plan and applied by the generator, and no reason for it is recorded in the plan, in the generator or anywhere else in this work; it is disclosed here as a design choice that carries no recorded justification. These nine are the ones the limitations section refers to, and each label names the whole board: the single `D` and the three units beside it.

Within a composition, the four units are dealt in order into four of the opponent's six squares, which gives $6 \times 5 \times 4 \times 3 = 360$ boards per composition and $9 \times 360 = 3{,}240$ opponent boards in the

Table 4: The nine opponent compositions, in the order the generator emits them, with the number of confirmatory instances each supplies. Low end and High end are the instances a composition supplies at each difficulty end, and Instances is their sum. Every board is one `D` plus the three units named, four units for 5 gold. `AAAD` is the tenth multiset and is excluded by the design. The 360 boards per composition are the orderings of four units over the opponent's six squares, $6 \times 5 \times 4 \times 3$.

Composition	Units beside `D`	Boards in pool	Low end	High end	Instances
`AABD`	`A`, `A`, `B`	360	7	7	14
`AACD`	`A`, `A`, `C`	360	7	7	14
`ABBD`	`A`, `B`, `B`	360	7	7	14
`ABCD`	`A`, `B`, `C`	360	7	7	14
`ACCD`	`A`, `C`, `C`	360	7	7	14
`BBBD`	`B`, `B`, `B`	360	7	7	14
`BBCD`	`B`, `B`, `C`	360	7	7	14
`BCCD`	`B`, `C`, `C`	360	7	7	14
`CCCD`	`C`, `C`, `C`	360	7	7	14
Total		3,240	63	63	126

pool, which is the whole opponent space and was enumerated rather than sampled. The order is part of what a board is, but only within a type. By the priority rule of Section A.2, which sorts a side's units by type first and by placement order only inside a type, two boards differing in which of two same-type units was dealt first are different battles and so are different boards, while dealing order across different types changes no battle at all. A board is recorded as each type's squares in placement order. Its *picture* drops the order among squares holding the same type, and its *mirror class* unites the picture with its column reflection, column c to $2-c$, with `B` and `C` exchanged because they are reflections of each other. A mirror class is a deduplication unit for the stimulus population and not a claim that its members are equally hard: the rules are not symmetric under reflection, and one measured pair of twins wins 23 and 311 respectively of the 3,240 plays of Section A.3.

The queue the file records puts the `D` at position `emission_index` modulo 4, so across the 126 instances the `D` is placed first, second, third and fourth on 32, 32, 31 and 31 boards respectively. The other three units are ordered by a digest of a fixed hash seed, `20260913`, and the board's own text, and squares holding the same type keep the order the board records, so the emitted queue is the same board as its key: only the order across types moves, and by the paragraph above that changes no battle.

A.5 Selecting the 126 instances, and the two ends

All 3,240 opponent boards were measured over all 3,240 agent plays, so every board in the pool carries an exact d before anything is selected. The suite is chosen from that measured pool, and nothing about which boards were chosen depends on a draw.

A board is eligible if it is non-degenerate, meaning that its d is neither 0 nor 1, and if $0.01 \leq d \leq 0.40$. A degenerate board is decided before the agent chooses anything, so it would measure nothing about an agent. Of the 3,240 boards, 3,037 are eligible.

Selection is rank matched across the two ends, seven ranks per end per composition. It runs the low end first, since that order decides which end a contested mirror class goes to. At each rank and each end, every composition takes the free board whose $\mathrm{logit}(d)$ is nearest a common target, the target being the least extreme of the compositions' most extreme free boards, with ties broken by the board's text. At most one board per mirror class is taken across the whole suite. Seven per end per composition over nine compositions gives the 126 instances of the confirmatory suite, 63 at each end, and the right-hand columns of Table 4 show that the composition balance is exact within each end by construction.

Measured on the frozen file, the low end runs from $d = 0.0164$ to 0.0265 and the high end from 0.1735 to 0.3762, with the pre-registered $d_0 = 0.1$ fixed in advance in the gap between them and no instance anywhere near it. The standard deviation of $\mathrm{logit}(d)$ over the 126 instances is 1.376. Composition and difficulty are close to orthogonal across the suite because the rank matching pulls

them apart: every composition takes, at each rank and each end, its own free board nearest a common target on $\mathrm{logit}(d)$, so the nine compositions are laid along one difficulty ladder rather than falling where they may. The η^2 of $\mathrm{logit}(d)$ on composition is 0.00006, which is a check that the construction did what it was built to do and not a property of the environment.

The exact d is the field `measured_difficulty`, which is `win_count` divided by `play_count`. The field named `difficulty`, easy to mistake for it, is only a rank by win fraction within one file, 1 being the easiest, so it repeats across files. Every pooled quantity in this paper keys on `id` and on `measured_difficulty`, and never on `difficulty`.

A.6 The prompts the agent was sent, and the content hashes

The confirmatory suite file is `study_stage1.json`, holding the 126 instances `study-001` to `study-126`, with SHA-256

`f707dabb370ba10ddb0fbc64ec1d4c01e0c334b563b6ee877d5c21ba86615436`.

That digest is recorded in the pre-registration and in the run identity, every logged attempt states it, and a resume refuses a file that does not match it.

Every arm ran at prompt configuration `I1_E0_R0`: rules, no worked examples, no reflection. Under that configuration the system prompt is three files joined by a blank line separator, in this order: the role prompt `llm_role.txt`, the rules `llm_rules.txt` and the response format `llm_response_format.txt`. Each file already ends in a newline of its own, so two blank lines fall between one file and the next in the assembled prompt. Those files are the whole of it, and all are printed below.

Each digest quoted for a file is over the file itself and not over the block printed here, and one rendering rule covers all three blocks. The three files' longest lines run to 115, 130 and 115 characters, too wide to set in this column, so a line that does not fit is broken at a space and the continuation is indented by four spaces. No line of any of the three files begins with four spaces, so the wrapping reverses without ambiguity: replacing every newline followed by four spaces with a single space recovers the file. Further differences are worth stating exactly. The first is confined to the role prompt, which carries a single trailing space at the end of its `2. BATTLE` line that is not shown, and that is the only trailing space in any of the three files. The second is a matter of glyph rather than of content: the typewriter font used for these blocks sets an apostrophe as a right single quotation mark, and every apostrophe printed in the three blocks is the plain ASCII apostrophe, byte 0x27, in the file. All three files are plain ASCII and hold no tab character. Nothing else is altered, so a reader who unwraps the continuations, restores that one space and reads every printed apostrophe as byte 0x27 recovers the bytes the three digests are over.

The role prompt is the trimmed `llm_role.txt`, with SHA-256

`c24a6ac5538a867f102e3ba74dfffd14fae8717f80b9b6e6dcefe1ddae372e68`.

```
You are playing a 4x3 autochess game. The board has 4 rows and 3
    columns.
You control the top 2 rows (rows 0-1). Your opponent controls the
    bottom 2 rows (rows 2-3).
Your units are indicated with an "L." The opponent's units are
    indicated with an "H."

GAME PHASES:
1. PREPARATION: You and the opponent alternate placing units. You
    place first each round.
2. BATTLE: Units fight automatically (you don't control this phase).
    Units attack one-by-one in alternating turns.

UNIT TYPES AND COSTS:
- A (1 gold)
- B (1 gold)
- C (1 gold)
- D (2 gold)
```

```
You can see BOTH shops: your shop/gold and the opponent's shop/gold.
Your objective is to outscore the opponent at game end.

STRATEGY TIPS:
- Consider what the opponent might place and position your units to
    counter.
- Use the opponent's visible shop options and gold count to anticipate
    likely purchases and counter-placements.
```

The rules file is `llm_rules.txt`, with SHA-256

`da3c78b853aba7941d885de6d5cdcf44a6500a668c3fa891d5c72e961c531580`.

It carries the battle mechanics, and it is what rung `L1` of the decomposition ladder adds to `L0` in Appendix D. The role prompt above already tells the agent that the battle runs itself and that units act in alternating turns, so this block is not the agent's only source on the battle.

```
BATTLE MECHANICS:
- Your units face DOWN (toward higher rows). Opponent units face UP
    (toward lower rows).
- Unit behavior details:
  - A (1 gold): Attacks the cell directly ahead. If an enemy is there,
    removes it. Otherwise advances forward one cell.
  - B (1 gold): Attacks diagonally left-ahead. If on leftmost column
    (col 0), cannot attack. Otherwise advances forward.
  - C (1 gold): Attacks diagonally right-ahead. If on rightmost column
    (col 2), cannot attack. Otherwise advances forward.
  - D (2 gold): Ranged unit. Removes the closest enemy by Manhattan
    distance (ties broken left-to-right, then top-to-bottom).
- Turn order scan: A units are checked first, then B, then C, then D.
    Within same type, earlier-placed units are checked first.
- If the highest-priority unit is blocked and cannot act, the scan
    continues to the next unit in priority order until one can act.
- If no unit on that side can act, that side's step is skipped.
- Units that advance past the opponent's edge ESCAPE and earn 1 point.
- Battle alternates: you go first, then opponent, repeat until neither
    side can take any action.
- Score is: units still on board + escaped units. Higher score wins.
    Equal scores are a tie.
```

Set against Table 3 and the battle rules of Section A.2, that file states the same behaviours in the agent's own terms, and in at least two places it states less. It does not say that a `D` never moves, and so never says that a `D` cannot escape. And it gives one of the two ways a battle ends, that neither side can act, where the engine also ends a battle the moment exactly one side's units are all gone. What the file does carry, in its own wording, includes the four costs, the facing rule, the column that blocks a `B` and the column that blocks a `C`, both named as board columns and so the same for either side, the scan order and its within-type tie-break, the escape point, and that equal scores are a tie.

The response format is `llm_response_format.txt`, with SHA-256

`7f80e0529877df513674eff925cabddaefc44f898d8adfc7d1b46753a8044a62`.

It is the whole of the answer-extraction contract. The `PLACE` line it specifies is the line the harness reads a placement out of, and this file is where the agent is given the exact form a placement must take, down to the row and column values it may write. A reply the harness cannot read a placement out of, and a reply naming a placement the game will not accept, are the two fallback causes this contract governs, and Appendix G counts both.

```
RESPONSE FORMAT:
Think through your reasoning, then include one final concise paragraph
    immediately before the PLACE line.
That final paragraph must state your core rationale in 1 sentence (why
```

Table 5: The seven prompt files covered by the prompt digest, with their SHA-256 digests. Each digest is printed as two halves of 32 hexadecimal characters, which is a presentation break and not part of the value. Combining the seven whole digests in name order by the rule in the text reproduces the run identity's `prompts_sha256`. The first three are the files `I1_E0_R0` sends.

File	SHA-256 (two halves of 32 characters)
`llm_role.txt`	`c24a6ac5538a867f102e3ba74dfffd14` `fae8717f80b9b6e6dcefe1ddae372e68`
`llm_rules.txt`	`da3c78b853aba7941d885de6d5cdcf44` `a6500a668c3fa891d5c72e961c531580`
`llm_response_format.txt`	`7f80e0529877df513674eff925cabdda` `efc44f898d8adfc7d1b46753a8044a62`
`llm_examples.txt`	`38bdf5320487225a748347a72a89d2be` `a3dbc7c1733536acae98f6702e36a1c0`
`llm_reflection.txt`	`4ebef520f76959e5e63dedfbea2c794e` `ff1eab246e0b370beded3060316e035b`
`reflection_system.txt`	`ac14d89fa6caba8377270aed422a796c` `78ea527150585af2c7390d98e0c16f0b`
`reflection_user_footer.txt`	`0dbc1a501d01310f12910d95e2f075fc` `55de5d0342aac6f6c07d8bbb96cd14b3`

```
    this unit and this cell are best right now).
Avoid generic closers like "This seems solid"; include specific
    board/shop factors.
Then end your response with exactly this line:
PLACE: <type> (row, col)

Where <type> is A, B, C, or D, and row is 0 or 1 (your rows), col is
    0, 1, or 2.
The concise rationale paragraph must be the last paragraph before
    PLACE.
The PLACE line must be the last non-empty line of your response.
```

The header the agent sees on each of its turns is `=== PREP TURN n ===` with n equal to 0, 2 and 4, a placement index rather than a round count. That header, the board and the two shops are the user message, and no part of them is in the system prompt above. The effect of trimming the role prompt is disclosed as unmeasured; the limitations section carries that disclosure.

Every logged attempt also carries a single prompt digest, `prompts_sha256`, over the whole prompt directory rather than over the role prompt alone, and the run identity records it as

`eace89fd5823afb281470ebab558f7dbf1fdf3dca8cff3e52d50488edb8636cc`.

It is built by sorting the prompt files by name, writing one line per file as the file name, a tab, the file's SHA-256 and a newline, and taking the SHA-256 of the concatenation. Table 5 gives the seven files and their digests, so the combination can be checked without the files themselves. The other four files, the ones not printed above, belong to the example and reflection modes, which no arm in this study used; they are in the digest because the digest covers the directory.

B Statistical apparatus

B.1 Fitting and quadrature

Every generalised linear mixed model the plan calls for, the headline fit and the three per-model fits alike, is fitted at $Q = 15$ adaptive Gauss-Hermite quadrature points, fixed before any of the data analysed here were collected. The plan records the check behind that number: against $Q = 60$, over 30 data sets per cell, $Q = 15$ moves $\hat{\gamma}$ by at most .0024 at $s \leq 2.5$ (.0136 at $s = 3$, about .08 SE) and $\hat{s}$ by at most .010 at $s \leq 2.5$ (.094 at $s = 3$). There is no quadrature switch and no rescue refit,

so a fit either meets the verdict below at $Q = 15$ or is reported as not meeting it. The headline fit and the three per-model fits each record 15 quadrature points and the verdict `converged`.

So does the pure-attempt refit, whose slope and 90% interval main text gives as the plan's fallback sensitivity: it too is fitted at $Q = 15$, returns `converged`, and has no reason recorded against that verdict. It is fitted on all 126 instances with none dropped, so the interval main text quotes for it is built by the rule below on the same 124 degrees of freedom as the headline fit's. Its instance-effect SD came back away from zero, $\hat{s} = 1.5172$, so it is not the boundary case the SD interval rule below sets aside.

B.2 Convergence is a verdict

"Converged" is a two-part check inside the fitter rather than a solver's exit code, and the plan defines it in full: "Convergence is a verdict (`response.py`): an undamped Newton step within tolerances (parameters 1e-7, log-likelihood 1e-10); and a refit from a start perturbed by 0.5 that agrees within 1e-4 in γ and s and 1e-6 in log-likelihood. Separation (a coefficient at ± 25, s at 50, or an arm with 0 or all wins) is named, not fitted through. The committed fitter converged on 1,040/1,040 test fits and 59,700/59,700 bootstrap refits." Both counts come from the fitter's test runs before the study. What that definition does not carry, and what belongs here beside it, is the branch it drives.

The verdict is load-bearing because a registered branch hangs on it. A headline fit that is not `converged`, or that is `separated`, selects the plan's row S0, under which no slope, interval, licence or level is reported as confirmatory and both level tests enter the Holm family at $p = 1$; the permutation test, the misfit test, the per-model fits and the rank correlations would still be reported, and no substitute model would be confirmatory. The headline fit returned `converged` with s not at zero, so that branch was not taken and the reported row was reachable.

B.3 The level estimand and its second quadrature

The registered level is a marginal quantity: the fitted curve for a model at $d_0 = 0.1$, averaged over the estimated instance-effect distribution. The plan's own words for it are "Per headline model, from the headline fit: the marginal win rate at $d_0 = 0.1$, $m = \int \operatorname{expit}(a[m] + \gamma \cdot \operatorname{logit} 0.1 + u)\, \varphi(u; 0, \hat{s}^2)\, du$ (40-point quadrature, `response.level`), against 0.100." The 40 points belong to that integral and not to the fit's $Q = 15$ points; the two are different steps, an average over instances and an approximation inside the likelihood. The secondary reading is the conditional level, the median instance's curve at $u = 0$, reported beside the marginal one with its own delta-method interval. For either reading the precision this paper reports is the 90% interval, and no standard error on the level scale is quoted in its place.

B.4 How the intervals are built

The slope interval is $[L, U] = \hat{\gamma} \pm t(0.95,\ n-2) \cdot \mathrm{SE}$, with the standard error taken from the converged headline fit's observed information and n the number of instances, so 124 degrees of freedom on this suite. The plan records two grounds for choosing it over a test-inversion bootstrap and ranks neither: the test-inversion interval would have cost 17 to 23 CPU-hours per analysis, and simulated coverage is comparable. Through the staged rule at $s = 2$, coverage was .910, .913 and .877 at a true γ of 0.6, 1 and 1.4, against .900, .927 and .883 for the bootstrap-t; neither reaches the nominal .900 at the upper edge, which is the edge the licence turns on. The staged rule is the plan's two-stage design, in which a blind interim fit on the first 126 instances could have extended the suite to 198 before any confirmatory quantity was computed. It did not fire here, so that coverage belongs to a procedure carrying an interim look and a possible sample-size increase, while the interval reported here comes from a fixed 126-instance analysis. Appendix G records the trigger and what it returned. The gap is small rather than unquantified: in the same simulation the extension triggered in .087 to .120 of replicates at $s = 2$, so about nine replicates in ten were fixed-126 anyway.

The plan also records what the interval costs when it is read against 1.4 without the edge test. Read that way it is liberal: the simulated false-licence rate at the upper edge is .083 against a nominal .05, and the plan's instruction is that any row read from the interval alone carries that rate, which includes the pure-attempt refit main text reports as row S2 read mechanically. The plan's own answer sits beside it: the bootstrap-t's comparable rate was .077, so no cheap interval did better.

Quantity	Statistic	Reference distribution	Draws	Seed
Q1, slope above zero	$T = \sum_p \mathrm{logit}(d_p)\, W_p$, with W_p luna's plus glm's wins on instance p	instance-label permutation, one-sided greater, $p = (b+1)/10{,}000$	9,999	20260921
Q3, level at $d_0 = 0.1$	t on $\mathrm{logit}(m) - \mathrm{logit}(0.1)$, two-sided	parametric bootstrap from the fit constrained to the null, profiling over $\log s$	999	20260923
Q4, misfit	Pearson $\varphi = X^2/(\text{cells} - \text{parameters})$, Firth penalty, on a binomial GLM with model intercepts, a common slope and no instance effect	parametric bootstrap from that GLM, one-sided	999	20260924
Licence test, each edge	one-sided Wald t at that edge, $\delta = 0.4$	parametric bootstrap from the fit constrained to that edge	1,999	20260922
Rank correlation, per arm	Spearman against d on midranks, one-sided	resampling	9,999	20260925
Harness check, random arm	G against $\mathrm{Bin}(10, d)$	resampling	999	20260926

Table 6: The registered procedures as run. A replicate that does not converge counts as at least as extreme as the observed value, in the level test and at each licence-test edge alike; no replicate failed on any row this paper reports. Q1's observed statistic on this suite is $T = -1077.1455$ on $n = 126$ instances.

Level intervals are the delta method with a Student t quantile on the same instances minus two degrees of freedom. Every one of them, marginal or conditional, wherever it appears in the body or the appendices, carries the pre-registered label "delta method; simulated coverage .86–.90".

The instance-effect SD is interval-estimated on the log scale and back-transformed, $\exp(\log \hat{s} \pm t(0.95,\, n-2) \cdot \mathrm{SE}(\log s))$, with the standard error read from the fit's observed-information covariance, and it carries its own registered label, "Wald on log s; coverage not simulated". A converged fit with s at zero has no log-s variance and would be reported without an interval; this fit is not one.

B.5 The registered tests, the licence test, and the Holm family

Table 6 gives each procedure as it was run.

Q4 is fitted on a model of its own rather than on the headline fit: the plan puts it on a binomial GLM with model intercepts, a common slope on $\mathrm{logit}(d)$ and no instance effect, which on this suite is 252 cells and three parameters, so φ is read on 249 degrees of freedom. Its bootstrap draws from that GLM. So Q4 is not a test of whether the headline GLMM fits, and that is also why it survives the S0 branch above, which turns on the headline fit alone.

The licence test is a pair of one-sided edge tests, one at 0.6 and one at 1.4, each referring the Wald statistic at that edge to a parametric bootstrap from the fit constrained to it. The registered phrase is released only when both edges reject and the 90% interval lies inside the band. On this fit the lower edge gave $t = 4.5356$ against a critical value of 1.6433 and the upper edge $t = -2.5537$ against -1.7418, with no failed replicate at either edge. Had a constrained fit not converged, the licence would have been withheld and the interval still reported; had the test and the interval disagreed, the phrase would not have been used and the disagreement reported instead.

The band itself, 0.6 to 1.4, is a region of practical equivalence in the sense of Kruschke (2018): an interval of slope values fixed in advance and treated, for the one sentence it governs, as close enough to random play's slope of 1 to license that sentence and nothing wider. Its upper edge is not independent of the ladder it is read against: the plan fixes 1.4 as a round edge below the skilled softmax $T = 0.25$ anchor, whose GLMM γ was 1.68 on the earlier ends design. An edge drawn below a strong anchor makes the licence harder to earn rather than easier, and the plan records that the band does not move when the anchors are recomputed here. The lower edge has a different source, keeping 60 percent of random play's sensitivity, and is not affected. This appendix is the only place in the paper where that vocabulary is used. The procedure run against the band is the

Seed	Declared as	What it fixes
20260913	`fallback_seed`, run identity	the fallback stream; the game reseeds per attempt from the SHA-256 of `[fallback_seed, puzzle_id, config, attempt]`
20260914	`arm_seed`, analysis specification	a top-level field of the specification, carried inside its hash
20260921	`slope.permutation_gamma_0`	Q1's instance-label permutation
20260922	`slope.tost_gamma_1`	the licence test's two edge bootstraps
20260923	`level.level_vs_random`	the level test's null bootstrap
20260924	`misfit`	Q4's parametric bootstrap
20260925	`raw_rank`	the Spearman resampling
20260926	`harness`	the random arm's G-test resampling

Table 7: The eight seeds the registered analysis and the run fix. The last six are `estimands` entries of the analysis specification and are named here by their path within it.

pair of edge tests above, and neither edge, nor the pair, is called an equivalence test anywhere in this paper; the plan's own terms, kept throughout, are licence test, edge tests, and both edges reject.

The multiplicity family is exactly four tests at $\alpha = 0.05$: Q1, the level test for luna, the level test for glm, and Q4. Nothing else enters it. The slope interval and the licence are estimation rules rather than family members, and the error rates that govern them are simulated rather than adjusted: the interval's coverage and its upper-edge reading rate are given above, and the licence rule's own false-licence rate at the upper edge, at most .063, is in Section 5 inside the design-sensitivity sentence. Everything in the plan's secondary and sensitivity section, together with the conditional levels, the suite-mean paired difference printed below and the level intervals themselves, is unadjusted and labelled as such. A family test that could not be computed would have entered at $p = 1$ with its reason; none did.

B.6 The eight seeds

Table 7 lists the eight seeds the registered analysis and the run fix, seven in the analysis specification and one in the run's own identity record. They are not every seed in the study: the pre-registered anchor ladder alone draws on forty further seeds, which Appendix C records with the anchors file's own settings.

B.7 Identity of the analysis

The analysis specification is pinned by content, and every table in this paper that comes from the analysis was produced under it; the value is carried in a `spec_sha256` column on every output file. The run the analysis read is pinned the same way, by a run identity hash recorded beside a state of `complete` and 504 of 504 cells.

Analysis specification
`b93efda48348e7d21b7e075552ea7fa3486e6e4e3fc4879ef68279c8b20c3185`
Run identity
`83a899cc31af2a4fb439f270c9ad941df9c79c915e68ae37b24cd8eacb163d4c`

The suite and role-prompt hashes are in Appendix A, and the anchor file's hash is in Appendix C.

B.8 Per-model fits and their levels

The plan fits luna, glm and gemma each alone, with the same form as Equation 1 and its own s, at $Q = 15$, and requires each model's slope and each model's levels at $d_0 = 0.1$ to be reported whatever the fallback screen shows. The three slopes are in Table 2, as are gemma's two levels. luna's and glm's levels are the two rows below, and appear nowhere else in this paper.

The plan also requires a line above those rows, naming every keyed arm whose fallback-rate Wilson 95% lower bound, on the data analysed there, exceeds the 0.106 reference level, and saying which data that line read. *It names none.* The data it read is the stage-1 fallback screen, computed once on the complete stage-1 data, which Appendix G prints in full. No arm's lower bound reaches 0.106, so no row below is labelled and nothing about how any of them was fitted or is read changes.

Arm, own fit	Conditional	90% CI	Marginal	90% CI
luna	0.1786	[0.1228, 0.2524]	0.2998	[0.2524, 0.3519]
glm	0.2158	[0.1792, 0.2575]	0.2594	[0.2258, 0.2960]

Those are levels at $d_0 = 0.1$ from each model's own fit, against random play's rate of exactly 0.100, and all four intervals carry the pre-registered label "delta method; simulated coverage .86–.90". They are not the headline-fit levels of Table 2, and that distinction is the reason this subsection exists rather than a matter of presentation. Table 2's luna and glm level rows come from the pooled headline fit, in which one slope and one instance effect are shared by the two models, and those rows are what the registered tests were run on. The rows above come from fits in which each model carries its own slope and its own instance-effect SD, and they are unadjusted. The two sets differ: luna's marginal level is 0.2912 on the headline fit and 0.2998 on its own, and glm's is 0.2575 against 0.2594. No claim in the main text reads off the rows above. They are printed because the plan requires them by name.

Those fits differ from the headline fit, and from each other, in the instance-effect SD they estimate as well as in the slope: luna 2.3307, glm 1.0889 and gemma 2.9282, against the 1.4059 that Equation 1 makes luna and glm share. Those are point estimates and are printed without intervals, because the plan registers the interval for s on the headline row alone and the analysis emits one there alone; manufacturing three now would be an analysis this study did not register. The factor of 2.1 between luna's and glm's own SDs is the variance half of the same registered constraint whose slope half Section 5 reports as contradicted by the data it was fitted to, and the headline interval [1.1990, 1.6484] contains neither of them, glm's falling below it and luna's above. What that does to the licence rule's error rates is not established here: the shared estimate sits between the two, the direction is not determined by that alone, and nothing in the plan or in this paper simulates it.

The plan registers one further quantity on the level question, and it is model-free: the suite-mean paired difference per model, the mean over instances of wins$/k$ minus d, with a 90% t interval over instances, which the plan describes as "model-free and clustered by puzzle". It is a registered robustness check, it is unadjusted, and its interval is a t interval on 125 degrees of freedom. On this suite it is 0.0057 [−0.0115, 0.0230] for the random arm, 0.1605 [0.1180, 0.2029] for luna, 0.1327 [0.1020, 0.1634] for glm and 0.0676 [0.0285, 0.1068] for gemma. The plan also registers a sign-disagreement check of this quantity against the primary level result, and it fires for neither luna nor glm. What it corroborates is the direction of the level tests and nothing further: it is a mean over the instances the suite actually holds, which sit at the two ends, so it is not a reading at $d = 0.1$ and it does not estimate what the level tests estimate. Section 5 reports the direction and points here for the values.

One further point of procedure belongs here rather than in the result. The plan's Section 7.1 requires these per-model levels, while its Section 8 says that per-model rows the specification produces are not reported and enter no family. What is reported is the levels alone: no per-model licence-test row and no per-model permutation row appears anywhere in this paper, which is the wording the deviations section uses for the same exclusion. Whether that reading is itself a departure is left open in the paper's deviations section, and is not settled here.

C THE SCRIPTED ANCHOR LADDER

The plan fixes 21 scripted strategies, says what each one is by pinning the code that implements them, and commits the file that records their exact results by its SHA-256:

```
9a4a441225961da1f89b98ae786d62092a3be2213e86223897805d95738ab98c
```

That string is carried twice inside the frozen plan, once in the timing bullet that commits the anchor files and once in the plan's table of committed hashes, so the pin is external to this paper and

to its authors. The file records the settings it was computed under: `attempts` 10, `draws` 40, `seeds [20260930, 20260969]`, which the plan defines as $20260930 + i$ for $i = 0$ to 39, and `quadrature_points` 15. It also records the suite it was computed on, `study_stage1.json`, 126 instances, which is the confirmatory suite, at SHA-256

```
f707dabb370ba10ddb0fbc64ec1d4c01e0c334b563b6ee877d5c21ba86615436
```

Every probability in the ladder is exact and none of it involves a language model. The limit γ is the ordinary least squares slope of $\mathrm{logit}\, P(\mathrm{win})$ on $\mathrm{logit}(d)$ over those exact probabilities, and is undefined when any instance's P is 0 or 1. The GLMM γ is fitted on 40 binomial draws at $k = 10$, each at $Q = 15$, and is the median of the n draws that converged; a strategy is *separated* when more than half of its draws fail to converge, and then no median is reported. The low and high are the values at 1-based positions $\lceil 0.05n \rceil$ and $n + 1 - \lceil 0.05n \rceil$ of the n sorted converged draws, which comes to the 2nd and the second highest for every strategy reported here: positions 2 and 39 for the twelve whose 40 draws all converged, 2 and 38 for `softmax_T1` and `eps_0.75` at $n = 39$, and 2 and 35 for `softmax_T0.25` at $n = 36$. The n column of Table 8 carries the count for every row, so each row's positions can be read off it. Those draws describe attempt noise with the instances held fixed, so the low and high columns are not a confidence interval over instances, and the bars in Figure 1 are those same two values. The information column records what the strategy is allowed to use: `none` is nothing about the opponent, `visible` is only the opponent units already on the board at that decision, which is what the agent's prompt shows, and `queue` is the whole fixed queue, which is the instance as the oracle knows it. Ties are split uniformly in every strategy, which is why strategies that carry no noise term can still have fractional win probabilities.

An anonymised submission cannot cite the pinned code, so what it implements is set out here. Every strategy is a stepwise policy: at each decision it puts a weight on every legal placement, and its win probability on an instance is the total probability of its winning complete plays. Where a strategy uses V^*, that value is the best final score margin still reachable after a placement, taking the agent's own best continuation against the fixed opponent script; computing it needs the whole queue.

- `random` puts equal weight on every legal placement, which makes every complete play equally likely because the agent's three unit types all cost 1 gold and the whole budget is spent. It is the softmax family at infinite temperature, and the design makes its exact win probability the instance's difficulty d.
- `optimal` reads the whole queue and takes the placement with the largest V^*.
- `softmax_T0.1`, `softmax_T0.25`, `softmax_T0.5`, `softmax_T1`, `softmax_T2` and `softmax_T4` sample from the same V^* rather than maximising it: each legal placement gets weight proportional to $\exp(V^*/T)$, so T divides the margin scale. A lower T approaches `optimal` and a higher one approaches `random`, which is the family's limit as T grows without bound.
- `eps_0.25`, `eps_0.5` and `eps_0.75` mix those two ends: at every decision the strategy places uniformly at random with probability ϵ and plays `optimal` otherwise. ϵ is the random fraction, so `eps_0.75` is the most random of the three and wins least often of them.
- `rollout_greedy` reads the whole queue, searches one step and then continues at random: it takes the placement from which the largest fraction of complete plays wins.
- `lookahead_visible` searches fully over its own remaining placements, but scores each continuation against only the opponent units already on the board when the decision was taken, as if the opponent placed nothing more. It is the ladder's only full search restricted to what the agent is shown.
- `resolve_now_visible` and `resolve_now_queue` are myopic: each takes the placement that maximises the margin of a battle resolved at once on the board built so far, the first counting only the opponent units visible at that decision and the second the whole queue.
- `counter_visible` and `counter_queue` run no battle at all. Each takes the placement that maximises how many opponent units stand in a column its own units attack, an `A` attacking its own column, a `B` the column one lower and a `C` the column one higher (Table 3); the first counts the visible units and the second the whole queue. The count ignores

Strategy	Info	Search	Mean P	Limit γ	GLMM γ	n	Low	High
`random`	`none`	none	0.1316	1.0000	1.0254	40	0.8430	1.2324
`optimal`	`queue`	full	1.0000	*undefined*	*separated*	0		
`softmax_T0.1`	`queue`	full, noisy	1.0000	3.4335	*separated*	0		
`softmax_T0.25`	`queue`	full, noisy	0.9906	1.4380	1.8500	36	0.8479	3.1743
`softmax_T0.5`	`queue`	full, noisy	0.8797	0.9113	0.9350	40	0.7472	1.1773
`softmax_T1`	`queue`	full, noisy	0.5785	0.8015	0.7983	39	0.6980	0.8987
`softmax_T2`	`queue`	full, noisy	0.3291	0.8313	0.8480	40	0.7391	0.9426
`softmax_T4`	`queue`	full, noisy	0.2156	0.8950	0.9318	40	0.7448	1.0341
`eps_0.25`	`queue`	full, noisy	0.7378	0.4766	0.4798	40	0.3985	0.5600
`eps_0.5`	`queue`	full, noisy	0.4982	0.5233	0.5279	40	0.4584	0.5955
`eps_0.75`	`queue`	full, noisy	0.2923	0.6457	0.6538	39	0.5591	0.7416
`rollout_greedy`	`queue`	one step + random continuation	1.0000	*undefined*	*separated*	0		
`lookahead_visible`	`visible`	full over own placements	0.2551	1.1344	1.1219	40	0.9987	1.2653
`resolve_now_visible`	`visible`	one step, myopic	0.2692	*undefined*	1.2169	40	1.0798	1.4487
`resolve_now_queue`	`queue`	one step, myopic	0.6568	*undefined*	4.1922	40	3.8669	4.6631
`counter_visible`	`visible`	none, heuristic	0.2089	1.1233	1.0683	40	0.9002	1.1886
`counter_queue`	`queue`	none, heuristic	0.2397	*undefined*	1.1175	40	0.9604	1.2684
`front_row`	`none`	none, positional	0.1581	*undefined*	1.0907	40	0.9193	1.2814
`fixed_front_AAA`	`none`	none, fixed play	0.3095	*undefined*	*separated*	0		
`fixed_back_ABC`	`none`	none, fixed play	0.0238	*undefined*	*separated*	0		
`fixed_front_CAB`	`none`	none, fixed play	0.0635	*undefined*	*separated*	0		

Table 8: The full scripted anchor ladder on this 126-instance suite, in the order the pinned file lists it. Mean P is the strategy's mean exact win probability over the suite. n is how many of the 40 binomial draws converged, and Low and High are the 2nd and the second highest of those n converged draws, not a confidence interval.

rows, and `counter_visible` is exactly uniform at the agent's first decision, when no opponent unit is on the board yet.

- `front_row` uses nothing about the opponent: it is uniform over the legal placements in row 1, the agent's row nearer the opponent.
- `fixed_front_AAA`, `fixed_back_ABC` and `fixed_front_CAB` read nothing at all and each plays one fixed complete play on every instance: three `A` units across row 1; `A`, `B` and `C` left to right across row 0; and `C`, `A` and `B` left to right across row 1. Because the play is fixed, its win probability on an instance is that one play's 0 or 1 win flag. They are an illustration of a front line, a back line and a mixed line rather than tuned choices.

Twelve of the 21 strategies have a defined limit γ and 15 have a defined GLMM γ. Eleven of those 15 fall strictly inside the pre-registered range 0.6 to 1.4; the four that are defined and outside it are `softmax_T0.25` at 1.8500, `eps_0.25` at 0.4798, `eps_0.5` at 0.5279 and `resolve_now_queue` at 4.1922.

The `random` row is the ladder's check on itself. The plan requires `random`'s exact win probability to equal the computed difficulty on every instance, so its limit γ has to come out at exactly 1 with an intercept of exactly 0. It does: the pinned file records 1.0000 and 0.0000. A ladder whose null strategy missed that would not be measuring what the design says it measures.

The separated strategies, which carry no GLMM median, are `optimal`, `softmax_T0.1`, `rollout_greedy`, `fixed_front_AAA`, `fixed_back_ABC` and `fixed_front_CAB`, and none of the 40 draws of any of them converged, which is the 0 in their n column. `optimal` and `rollout_greedy` win with certainty on all 126 instances, so 10 attempts on every instance is a clean sweep and the fit has no variation to work on. `softmax_T0.1` is within a thousandth of certainty on every instance: its mean rounds to 1.0000 while no single instance's P is exactly 1, which is why its limit γ is defined at all, and its draws are separated all the same. The last three play one fixed complete play on every instance, so each instance's probability is that single play's 0 or 1 win flag.

Separation is a property of the fit rather than a measure of strength, and the two slope columns above are what show it. `softmax_T0.1` is separated in the GLMM and still has a defined limit γ of 3.4335. `softmax_T0.25`, which has the highest mean P of any strategy outside the three that print 1.0000, has a defined GLMM γ of 1.8500, the median of the 36 of its 40 draws that converged, and it is outside the pre-registered range, on a bracket running from 0.8479 to 3.1743 that is the widest of any row carrying a median. So a high win rate on this ladder neither removes a strategy's slope nor pins it near 1.

The properties Section 6 reads off the ladder are in the table above. The softmax family's GLMM γ crosses 1 between $T = 0.25$ and $T = 0.5$, at 1.8500 and 0.9350, and its limit γ crosses between the same two rows, at 1.4380 and 0.9113, with no draw noise in it at all. And γ is not monotone in skill: it falls from `random`'s 1.0254 at mean P 0.1316 across the three epsilon-greedy rows to `eps_0.25`'s 0.4798 at 0.7378, then rises again at `softmax_T0.5`, which wins more often still, at mean P 0.8797, and has a slope of 0.9350.

D THE DECOMPOSITION ARM'S APPARATUS

Section 7 states what this arm found and the disclosures that travel with it. This appendix holds the apparatus behind those sentences: the run frame, the ladder, the statistic and the condition its additivity rests on, the estimates with their intervals, the gate sheet as the arm's own tool evaluated it, the tool's verdict in full, and the per-instance terms.

D.1 THE RUN FRAME

The arm ran as one confirmatory invocation over five rungs, two models and 20 instances of the 126-instance confirmatory suite, at 24 attempts per rung, model and instance. That is 200 cells, and the finished run records 200 of 200 complete, with no row left uncertified by the run log's replay check. The ledger closed at $8.2645 against a budget of $49.04. An earlier invocation had recorded one API-failure stop, on glm at rung `L1` on instance `study-021`, where a call ended in an HTTP error and was not resent, so that attempt stopped; a later invocation resumed past it, and the cell is among the 200 complete.

The 20 instances were fixed before the run by a rule that leaves nobody a choice. The selection seed is the SHA-256 of a fixed ASCII prefix followed by the suite's own digest, read as a big-endian integer modulo 2^{31}, which is 1347107531; the selection file it produced is pinned by its own SHA-256 and its `arm` list is what the runner played. That rule took 15 instances from the suite's low end, at d 0.0164 to 0.0259, and 5 from its high end, at 0.1765 to 0.2512, against a suite built 63 and 63; no end quota was imposed on it. Every term below is therefore an average over an uneven mixture of the two ends, and 20 instances split 15 to 5 will not support an end-by-end split of the terms, so none is reported. Every per-attempt seed derives from the run seed 715440741 through a SHA-256 over the model, the rung, the instance, the slot, the replacement index and the purpose. Both models were sent at reasoning effort `medium` with a 16,384-token output cap and no temperature, one API call per placement, with no game history in any user message.

Every resampling figure in this appendix comes from one bootstrap: 2,000 resamples at report seed 1010. Table 9 pins the files the run was frozen against. The suite digest and the role prompt digest are in Appendix A.6 and are not repeated here.

D.2 THE FIVE RUNGS

Table 10 gives each rung and what it hands the agent. The system prompt is assembled from named parts joined by a blank line: `L0` is the role text plus the response format, `L1` is the role text plus the rules file plus the format, `L1P` is the role text plus the placebo block plus the format, and `L2` and `L2P` reuse `L1`'s system prompt and add the oracle block to the user message. The oracle block is its own paragraph immediately before the final instruction line, and its lines are ordered by a uniform permutation drawn from that decision's seed.

The placebo block's token match was measured before the run rather than assumed: the whole-prompt input-token ratio of `L1P` to `L1` is 1.0023 for luna, 878 tokens against 876, and 1.0114 for glm, 887 against 877, both inside the band of 0.95 to 1.05 the gates file reads. `L2P` is matched to `L2`

Table 9: Content hashes the decomposition run was pinned to, beyond those Appendix A.6 already prints. Each digest is shown as two halves of 32 hexadecimal characters, which is a presentation break and not part of the value. The placebo block and the oracle header are the two prompt parts this arm adds to the headline study's.

Pinned file or prompt part	SHA-256 (two halves of 32 characters)
This arm's pre-registration	`a81ce02fde2913b35532bc6aefdece6c` `9f3b235c6cc3d151bbbf7ea948d57812`
Its gates file	`f100c341b18298ae02a4f09a1b52706d` `9f29d0c89af0c407f0f586401b35547a`
The instance selection file	`c93a8299f6933871aa7f0324770c17b0` `968a4919dd6e8cb1ab7180b065854444`
The `L1P` placebo block	`ef531661ba899ab446f6c063c42f8713` `87e35a8cd25127b76c28e5bfdaa2b440`
The `L2` oracle header	`830d696f43425f6636d1854bfdc6a13b` `ec132ae7c70961be2ab7c9e17a953334`

Table 10: The five rungs of the reduced ladder and what each one hands the agent. The identification rungs `L3` and `L3N` of the arm's design were never bought and this arm's own plan excludes them.

Rung	What the agent gets	What it isolates
`L0`	The trimmed role text and the response format. The battle mechanics are withheld	The bottom of the ladder, where the agent must infer the dynamics
`L1`	`L0` plus the rules file verbatim	The true dynamics stated in words
`L1P`	`L0` plus a dynamics-free block, token-matched and register-matched, restating only what the interface already shows	Whether stating the rules helped or whether more text helped. It is deliberately not a false rule set, because misinformation is not a placebo
`L2`	`L1` plus one line per legal candidate giving that candidate's exact final margin under best continuation	The dynamics work is removed entirely, so whatever regret survives is selection
`L2P`	`L2`'s bytes with the same multiset of margins reattached to candidates under a seeded derangement, and a forced single-candidate row printed truthfully	Whether the agent reads the printed margins at all

on byte length and on the printed multiset, so the two differ only in which candidate carries which value. No rung prints `L2`'s value lines with the values removed, so for those lines this arm cannot separate the information from the extra text; `L1P` does that for the rules text alone.

There is no fallback anywhere in this arm. A content failure is one of four things the plan fixes in advance: empty content, a finish reason of `length` with no command line, no command the reader accepts, or a placement the agent cannot make. The identical call is retried twice after one, and a third content failure at one decision censors the whole attempt, with the censoring recorded. A censored attempt leaves no hole: it is replaced in its slot under the next replacement index with seeds of its own, because the run log refuses a gap in a cell's attempt indices. A cell may make at most two replacements, and a cell that would need a third is incomplete and enters no contrast. Censoring the attempt rather than the decision is what keeps the per-attempt summation of decision regret exactly equal to whole-play regret; dropping one decision and keeping its attempt would break that silently.

D.3 The statistic, and the condition its additivity rests on

Regret is summed over a play, each decision contributing that decision's best value minus the value the agent took. $\bar{R}(\text{rung})$ is instance-stratified: the mean over instances of each instance's mean whole-play regret over its kept attempts. Table 11 gives it for all five rungs, beside the pre-registered scale, which is the same quantity for a uniform random placer and for the worst legal play.

The terms are defined from those rungs: $D_{\text{infer}} = \bar{R}(\texttt{L0}) - \bar{R}(\texttt{L1})$, $D_{\text{sim}} = \bar{R}(\texttt{L1}) - \bar{R}(\texttt{L2})$ and $S_{\text{sel}} = \bar{R}(\texttt{L2})$. Terms are signed and are never clamped, and each is computed on the intersection

Table 11: Instance-stratified mean whole-play regret at each rung, on the 20 instances, with each rung's regret as a share of the uniform random placer's and of the worst legal play's. The pre-registered scale for reading a term's size is the share of the uniform placer's regret.

	luna			glm		
Rung	$\bar{R}$	of uniform	of worst	$\bar{R}$	of uniform	of worst
L0	3.9688	0.9664	0.6250	3.9833	0.9699	0.6273
L1	2.8708	0.6990	0.4521	3.0396	0.7401	0.4787
L1P	4.1229	1.0039	0.6493	3.8000	0.9253	0.5984
L2	0.0000	0.0000	0.0000	0.0437	0.0107	0.0069
L2P	4.0917	0.9963	0.6444	4.2708	1.0399	0.6726

of the instances its own pair of rungs completed. D_{sim} carries the label A1-conditional, which the plan fixed on it in advance and Section D.5 explains; the label changes and the number does not.

Additivity rests on one condition and no more: $D_{\text{infer}}+D_{\text{sim}}+S_{\text{sel}} = \bar{R}(\texttt{L0})$ holds exactly when L0, L1 and L2 share one instance set. Here they do, all three at 20 of 20, so the identity holds exactly on the unrounded values. The four-decimal figures printed below sum to 3.9687 against luna's $\bar{R}(\texttt{L0})$ of 3.9688 and to 3.9832 against glm's 3.9833, and that gap is rounding and nothing else. The tool's own telescoping line reads the four-term form only and reports it as not computable, as it does the four-term intervals, because D_{exec} and D_{search} have no instances at all: they are differences against L3, and L3 was never bought. D_{sim} is their sum and cannot be split without that rung, so this arm yields no number that separates a wrong world model from a poor search.

S_{sel} pooled is $\bar{R}(\texttt{L2})$ by definition, which is why glm's pooled S_{sel} and its L2 row in Table 11 are the same 0.0437. The restricted quantity is a different statistic with a different denominator: for each instance the L2 regret is summed over every decision with a unique optimum across every kept attempt and divided by the number of those decisions, and that per-decision rate is averaged over the instances that have at least one. Every one of the 20 has at least one, and the counts are 642 decisions for luna and 650 for glm. The two quantities are never interchangeable and neither denominator belongs to the other row.

Inference is fixed by the plan. Attempts are resampled within rung and instance, 2,000 resamples with nearest-rank 95 percent endpoints, each cell's draws coming from a stream named by model, rung and instance under the report seed 1010. The test is an exact paired sign-flip permutation over instances, all 2^{20} assignments, with the sum of the per-instance differences as the statistic and the two-sided p read off it. Multiplicity is Holm within each model at family-wise $\alpha = 0.05$ over a family of five fixed in advance: D_{infer}, D_{exec}, D_{search}, L1P against L0, and L2P against L2. The plan settled beforehand that a rung not run never shrinks the family, so D_{exec} and D_{search} each enter at $p = 1$ and still count, and the three contrasts that can reject are adjusted as five. D_{sim} and S_{sel} sit outside the family and carry intervals only.

D.4 The terms and the two placebo contrasts

Table 12 gives the terms. Both registered manipulation contrasts are the placebo rung minus its comparator, which is why the L2P row is large and positive: it is $\bar{R}(\texttt{L2P}) - \bar{R}(\texttt{L2})$, so $4.0917 - 0.0000$ for luna and $4.2708 - 0.0437$ for glm. The other is $\bar{R}(\texttt{L1P}) - \bar{R}(\texttt{L0})$, so $4.1229 - 3.9688$ for luna and $3.8000 - 3.9833$ for glm. Every difference in this subsection is taken on the unrounded means and rounded once at the end, the rows Table 12 labels as subtractions no less than the operands displayed here, so subtracting the four-place values in Table 11 can disagree in the last digit. Luna's two operands above subtract to 0.1541 against the 0.1542 the table prints, and that gap is rounding and nothing else. The L2P contrast rejects at Holm-adjusted $p < 0.0001$ for both models; the L1P contrast is Holm-null for both, at 0.2346 and 0.2317, and the two models take opposite signs, so the reading Section 7 takes from the placebos rests on the L2P contrast alone. L1 against L1P is not a registered contrast, is in no family, and is estimated nowhere in this arm.

Restricted to decisions with a unique optimum, the bootstrap censored no resample for either model, and no instance had to be dropped for want of such a decision.

Table 12: The decomposition arm's terms and registered contrasts, on 20 instances and two models. Intervals are 95 percent bootstrap, 2,000 resamples at seed 1010, and p values are two-sided sign-flip, Holm-adjusted within each model over the family of five; both differ from the 90 percent Wald and delta-method intervals carried everywhere else in this paper. The two S_{sel} rows are different quantities: the pooled row is a per-play mean over 20 instances, and the restricted row is a per-decision rate whose denominators are 642 decisions for luna and 650 for glm. D_{sim} and S_{sel} are outside the test family and carry no p. The D_{sim} row is A1-conditional, which is the label the plan fixed on that term in advance, and Section D.5 gives the one-sided statements that stand without that assumption. A Holm-adjusted p is printed as < 0.0001 where it falls below that: the exact sign-flip test over all 2^{20} assignments cannot return a two-sided value below $2/2^{20}$, about 0.000002, and Holm over the family of five floors the adjusted value near 0.00001. Where the statistic is zero in every kept attempt the percentile bootstrap is degenerate and returns a zero-width interval, so a selection row reading [0.0000, 0.0000] records that that row's statistic was zero in every kept attempt, and is not a width.

	luna			glm		
Quantity	Estimate	95% CI	Holm p	Estimate	95% CI	Holm p
D_{sim}, `L1` minus `L2`	2.8708	[2.7750, 2.9625]		2.9958	[2.8896, 3.1042]	
D_{infer}, `L0` minus `L1`	1.0979	[0.9604, 1.2375]	<0.0001	0.9437	[0.7958, 1.0937]	<0.0001
S_{sel}, pooled, per play	0.0000	[0.0000, 0.0000]		0.0437	[0.0167, 0.0750]	
S_{sel}, unique-optimum decisions	0.0000	[0.0000, 0.0000]		0.0000	[0.0000, 0.0000]	
`L2P` minus `L2`	4.0917	[4.0000, 4.1813]	<0.0001	4.2271	[4.1229, 4.3354]	<0.0001
`L1P` minus `L0`	0.1542	[0.0104, 0.2979]	0.2346	−0.1833	[−0.3354, −0.0354]	0.2317

D.5 THE IDENTIFICATION INTERVALS

$R_{\text{model}}(\texttt{L0})$ and $R_{\text{search}}(\texttt{L0})$ are not separate measurements. They are groupings of the primary terms, defined so that $R_{\text{model}}(\texttt{L0}) + R_{\text{search}}(\texttt{L0}) = \bar{R}(\texttt{L0})$: the plan puts D_{infer} on the model side and S_{sel} on the search side, and D_{sim} is divided between the two sides in a proportion this run cannot recover, which is the whole of the width. S_{sel}'s charge is imposed rather than derived, and the plan says so and calls it contestable; the plan argues D_{infer}'s side nowhere, and D_{infer} sits in both endpoints of the plan's own model-side interval. An agent that knows the rules and cannot run them forward is put on the model side, which is where D_{exec} would have sat had `L3` been bought; an agent handed every candidate's value and still not taking the largest is put on the search side, which is S_{sel}. A reader who puts either on the other side reads the intervals differently, and the terms in Table 12 are what that reader should use.

With `L1` as the middle rung, the arm's identification intervals are $R_{\text{search}}(\texttt{L0}) \in [S_{\text{sel}}, \bar{R}(\texttt{L1})]$ and $R_{\text{model}}(\texttt{L0}) \in [\bar{R}(\texttt{L0}) - \bar{R}(\texttt{L1}),\ \bar{R}(\texttt{L0}) - S_{\text{sel}}]$, and both have width $\bar{R}(\texttt{L1}) - S_{\text{sel}}$, which is D_{sim} on that instance set. On the 20 instances that is $R_{\text{search}} \in [0.0000, 2.8708]$ and $R_{\text{model}} \in [1.0979, 3.9688]$ for luna at width 2.8708, and $R_{\text{search}} \in [0.0437, 3.0396]$ and $R_{\text{model}} \in [0.9437, 3.9396]$ for glm at width 2.9958. As with the terms above, every figure here is printed from the unrounded means and rounded once at the end, so an endpoint difference taken on the printed digits can disagree with the printed width in the last place: $3.9688 - 1.0979$ gives 2.8709 against luna's width of 2.8708. The bootstrap is applied to each endpoint and not to the width: luna's lower endpoint of R_{search} has interval [0.0000, 0.0000] and its upper [2.7750, 2.9625], and glm's are [0.0167, 0.0750] and [2.9396, 3.1417].

Two assumptions are imposed rather than derived, and the plan states both as assumptions: that the search and selection procedure is the same function across rungs with only the dynamics channel moving, and that correct dynamics weakly improve the agent's subjective model and never degrade search. The first carries a pre-registered check on reasoning tokens, comparing the median over every answered call of every kept attempt at `L0` with the same median at `L2`, with a shift beyond a factor of two in either direction setting a flag. The check was evaluable for both models and the flag is set for both: luna's medians are 417 at `L0` against 101.5 at `L2`, and glm's are 50 against 12, which are factors of 4.1 and 4.2 against a trigger of 2. What a set flag changes was fixed in advance by the plan: the two-sided intervals and their width are still printed and are labelled A1-conditional, the plan's name for a figure that holds only if that first assumption does, which is how the endpoints

Table 13: The seven gates as the arm's frozen tool evaluated them on this run, with G5's outcome withheld for the reason given below. The gates file voids this run's gate outcomes, so nothing in this paper is argued from this table.

Gate	What it reads	luna	glm
G1	Parse rate at least 95 percent and censoring under 2 percent in every rung, and a censoring differential under 2 points between any pair of rungs, at the 16,384-token cap. A differential failure marks the contrasts that pair enters as invalid rather than dropping them, and the tool recorded no such contrast	PASS	PASS
G2	`L2P`'s regret materially worse than `L2`'s and near the precomputed trusting-agent value	PASS	PASS
G3	S_{sel} above zero on at least one instance, over unique-optimum decisions	FAIL	FAIL
G4	`L1P`'s regret no better than `L0`'s	PASS	FAIL
G5	The registered monotonicity chain on the rung point estimates, which runs through `L3`	outcome not reported	
G6	Tool usage rate, round trips per placement and cap hits at the identification rung	not evaluable	not evaluable
G7	The identification rung's regret against its null-tool control's	not evaluable	not evaluable

above should be read, and the one-sided statements that survive without it are $R_{\text{search}}(\texttt{L0}) \leq \bar{R}(\texttt{L1})$ and $R_{\text{model}}(\texttt{L0}) \geq D_{\text{infer}}$.

What the check cannot do is settle the assumption. `L2` prints every candidate's exact final margin, so selection there is reading off the largest printed value, and a large fall in reasoning tokens from `L0` to `L2` is what the design predicts whether or not search and selection is the same function across the rungs. The check therefore has little power against the violation it is written for: its firing neither confirms nor refutes the assumption, and what stays unchecked is the assumption itself. The arm's 2026-09-15 pilot set the same flag for both models, with drops of about a factor of 3 for luna and 4 for glm, and the arm's own plan recorded before this run that recurrence was expected, marking that expectation as an inference rather than a finding.

D.6 The gate sheet, and the weight it does not carry

The arm's frozen gates file sorts a run into one of four classes and attaches an action to each. This run fell into the harness-defect class, and that class's action says in its own terms that the run's data are reported and never enter a gate. Table 13 is therefore a record of what the tool computed rather than a result, and no claim in this paper rests on any row of it.

G3 records `FAIL` for both models with an empty list of instances above the floor, which is the same fact the restricted S_{sel} row states: over the unique-optimum decisions there is no instance where selection loses anything. Its pre-written consequence, that the lower endpoint of R_{search} is then trivially zero, does not follow from the gate by itself. G3 reads the restricted per-decision rate, while that endpoint is pooled S_{sel}, a per-play mean over instances, and neither quantity implies the other. luna's lower endpoint is 0.0000 because its pooled S_{sel} is 0.0000; glm's is 0.0437 and glm fails G3 all the same. G4 records `FAIL` for glm because glm's `L1P` regret of 3.8000 is lower than its `L0` regret of 3.9833, and `PASS` for luna at 4.1229 against 3.9688. G2 read `L2P` at 4.0917 and 4.2708 against the precomputed trusting-agent value of 4.3117 for both models. G6 and G7 returned the codes `no_l3_decisions` and `no_rungs`. G5 is listed without an outcome because its registered chain runs through `L3`, a rung this arm never bought, and this paper reports no outcome for that gate.

Table 14 gives the rates G1 reads. Their denominators are different and both are fixed by the plan: the parse rate is answered calls whose answer read as a command over all answered calls, retries included, and the censoring rate is censored attempts over all attempts, replaced attempts included. Of the four content failures, an `illegal` placement read as a command and so counts as parsed, while `empty`, `length` and `unparseable` do not. That is what reconciles luna's `L1` row, which shows a parse rate of exactly 1.0000 beside a censoring rate of 0.0062: every content failure at that rung was a placement the game could not make, and its three censored attempts each took three of them at one decision. Six answered glm calls hit the 16,384-token output cap, with none for

Table 14: Parse and censoring rates per rung and model, with Wilson 95 percent intervals, which are the quantities G1 reads.

	luna				glm			
Rung	Parse	95% CI	Cens.	95% CI	Parse	95% CI	Cens.	95% CI
L0	0.9955	[0.991, 0.998]	0.0103	[0.004, 0.024]	0.9994	[0.996, 1.000]	0.0000	[0.000, 0.008]
L1	1.0000	[0.998, 1.000]	0.0062	[0.002, 0.018]	0.9955	[0.991, 0.998]	0.0021	[0.000, 0.012]
L1P	0.9911	[0.985, 0.995]	0.0000	[0.000, 0.008]	0.9987	[0.995, 1.000]	0.0000	[0.000, 0.008]
L2	1.0000	[0.997, 1.000]	0.0000	[0.000, 0.008]	1.0000	[0.997, 1.000]	0.0000	[0.000, 0.008]
L2P	1.0000	[0.997, 1.000]	0.0000	[0.000, 0.008]	0.9986	[0.995, 1.000]	0.0000	[0.000, 0.008]

luna, and all sit at rung L1; their effect on what was parsed is unmeasured, and these rates are not a reading on it.

D.7 THE CLASSIFIER'S VERDICT, IN FULL

Section 7 quotes the tool's decision line. Its full text is that line and the paragraph printed under it:

> DECISION: undecidable: harness defect or served-provider mismatch
>
> estimate.decide is not run (gates file section 3: it runs only when every not-evaluable gate is model behaviour and no over-reservation stop is held); fix without changing any stimulus, threshold or the gates file (for an over-reservation stop the fix is re-pricing; that run stays final and is not resumed), then re-run the whole pilot once in a fresh run directory; this run's data never enter a gate, and a re-run again in this class stops the arm at the pilot

The readings classified are G6 and G7 for each of the two models, and each names one rule and no other. For G6 on luna the tool prints that the gate is not evaluable with code no_l3_decisions while every cell of its rungs is complete, citing *gates file section 3 rule d: a gate not evaluable while every cell of its rungs is complete is a harness defect*. The other three readings are the same sentence with G7, with no_rungs, and with the other model. No other rule in that section is named for any of the four, and the section's other harness-defect triggers, which include a missing trusting-agent value, a provider-mismatch stop and an over-reservation stop, are not cited.

What makes rule d resolve this way is that this arm's own pre-registration excludes an L3 phase. Both gates read decisions at rungs that have no cells in this run, so the rule's requirement that every cell of a gate's rungs be complete is satisfied over an empty set. No re-run was performed. That account is a reading of the rule rather than a measured property of the run, and this paper asserts neither that a defect exists nor that none does.

D.8 PER-INSTANCE TERMS

Table 15 gives every term and registered contrast on each of the 20 instances, for both models, in ascending instance identifier. Each column's mean over the 20 is the pooled estimate in Table 12; glm's S_{sel} column averages to 0.0438 rather than 0.0437 only because the per-instance figures are rounded to four places before averaging. glm's selection regret is non-zero on seven of the 20 instances and zero on the other thirteen, and luna's is zero on all 20. Those seven are the whole of glm's pooled 0.0437, and none of them survives the restriction to decisions with a unique optimum.

Table 15: Per-instance terms and registered contrasts, luna then glm. Columns are D_{infer}, S_{sel} pooled, D_{sim}, `L1P` minus `L0` and `L2P` minus `L2`. Instance-stratified means of these columns are the estimates in Table 12.

	luna					glm				
Instance	D_{inf}	S_{sel}	D_{sim}	L1P	L2P	D_{inf}	S_{sel}	D_{sim}	L1P	L2P
study-010	0.7083	0.0000	2.7500	0.7500	3.9583	1.8750	0.0000	2.8750	−1.0833	4.0833
study-021	2.5417	0.0000	1.4583	0.4167	4.5417	1.9167	0.0833	2.0417	−0.4167	3.5833
study-024	1.5000	0.0000	1.9583	−0.2083	4.3750	1.1250	0.0417	2.8750	−0.5833	4.0000
study-032	0.7500	0.0000	2.9167	0.1667	4.2917	1.2500	0.1250	2.5417	−0.1250	3.9167
study-062	−0.2917	0.0000	4.0833	0.6667	3.4167	−0.3750	0.1667	4.0833	−0.3333	4.4583
study-071	1.0000	0.0000	2.0417	−0.2083	3.1667	0.4583	0.0000	2.0833	0.2083	3.0833
study-075	0.3750	0.0000	3.0000	0.7083	3.5833	0.5833	0.1667	3.0417	0.4583	4.0417
study-081	1.3333	0.0000	2.2083	0.4583	3.9167	0.9167	0.0000	2.4583	0.2083	4.2500
study-082	1.0000	0.0000	1.6250	−0.0833	2.5417	0.6667	0.0000	1.6667	−0.4167	2.8333
study-083	1.2500	0.0000	2.1667	0.0417	3.3750	0.5417	0.1250	2.7500	−0.3333	3.3333
study-084	1.7917	0.0000	0.8333	0.2917	2.9167	1.7083	0.0000	1.4167	−0.2083	3.6250
study-088	0.7917	0.0000	3.4167	−0.0833	3.6250	0.1250	0.0000	3.5000	0.3333	4.1250
study-095	1.4167	0.0000	4.4167	0.2500	5.4167	1.7500	0.0000	4.1250	−1.0833	5.4583
study-105	1.0833	0.0000	3.4167	−0.1667	4.1667	1.7083	0.0000	3.0417	−0.5000	5.0417
study-109	1.0000	0.0000	3.1667	−0.2083	4.7500	0.8750	0.0000	3.3333	−0.3750	4.7500
study-113	0.0000	0.0000	4.2917	0.2083	4.5833	0.8750	0.0000	3.4583	−0.0417	4.5417
study-115	1.4167	0.0000	3.4583	−0.4583	4.2083	1.2917	0.0000	3.2500	0.1250	4.4167
study-120	0.6667	0.0000	4.1667	0.6667	4.9583	0.5417	0.0000	4.6667	0.0417	5.2500
study-122	0.7500	0.0000	4.0000	−0.2083	4.7500	0.0833	0.1667	3.7500	0.2500	4.6667
study-125	2.8750	0.0000	2.0417	0.0833	5.2917	0.9583	0.0000	2.9583	0.2083	5.0833

E FEASIBILITY RUNS (NOT PART OF THE CONFIRMATORY ANALYSIS)

Nothing confirmatory rests on anything in this appendix. None of the data described here enters any fit reported in this paper, and that holds of every run below. Further statements about boards are on record, and they cover part of the set rather than all of it. The plan says of the two 2026-09-15 probes, in its own words, that no board is shared with either manifest. And of the crossed suite, which the 2026-09-12 run, the 2026-09-13 probe and the Option 1 pilot all played: every instance in the confirmatory suite has 3,240 ordered complete plays, six of the eight crossed instances have larger action spaces than that and so cannot coincide with any of them, and the two that do have 3,240 are recorded as different boards from any instance in either manifest. For the 2026-09-10 gate runs no such statement is on record and none is made here; what is claimed of them is that their data enter no fit and that they are exploratory history. What this appendix is for is the other direction, the audit trail from the feasibility work to the frozen design, so that a reader holding the plan can see which run produced each choice in it.

The plan also fixes how these runs may be reported: estimates with intervals and no p-values. That is a constraint rather than a preference, so no p-value, no adjusted p-value and no rejection appears anywhere below, and where a source recorded only a p-value for a quantity, the quantity is left out rather than restated without it.

E.1 THE RUNS

The gate runs of 2026-09-10. Three keyed arms, `openai/gpt-5.6-luna`, `openai/gpt-5.6-terra` and `openai/gpt-5.6-sol`, reached directly rather than through a router, plus the keyless random arm, over two generated five-instance suites, `pilot_suite.json` at d 0.0741, 0.3386, 0.5151, 0.7627 and 0.9900 and `lowtail_suite.json` at d 0.0060, 0.0108, 0.0182, 0.0241 and 0.0480; the three keyed arms also ran over the three hand-made instances that serve as the oracle's fixture. Configuration `I1_E0_R0`, 10 attempts per instance, every attempt played. These runs logged no token usage, so no price can be derived from any of their arms, though the keyed ones were paid calls; per-call usage logging landed afterwards and the 2026-09-12 run is the first to price itself. What does survive of their per-call record is partial: completed call counts for the three keyed arms over the first suite and again over the fixture instances, and fallback counts for those three over the first suite. They also predate independent attempts, which is the reason given below for pooling them with nothing. The crossed suite itself was first driven through the game keyless on 2026-09-11 at no cost, which established only that the suite loads, plays and logs.

The 2026-09-12 run. The first paid run on this suite, and the first to price itself from its own logs: the same three keyed arms and the random arm over `crossed_suite.json`, eight instances crossing four action-space sizes against two difficulty levels, 3,240, 29,160, 174,960 and 524,880 ordered complete plays with two instances at each. Configuration `I1_E0_R0`, 10 attempts, every attempt played, output cap 4,096, 360 agent placements per arm. Input was about 973 tokens per call and flat, with no cached input on any call. Wins out of 80 were random 16, luna 12, terra 18 and sol 20, and model ordering is not established on this suite. Like the 2026-09-10 runs it is disclosed as exploratory history, and it enters no fit.

The 2026-09-13 probe. Nine arms on the same eight instances, one of them stopped early by its host, at output cap 16,384 and at the router's standard service tier, run to compare candidate models, hosts and effort settings rather than to estimate anything. The suite takes 36 placements per attempt, so the 72-call arms are two attempts and the 36-call arms one; the arm that stopped early reached four instance-attempts in its 21 calls.

The Option 1 pilot, 2026-09-13. The run the design was sized on: `gemma`, `luna`, `glm` and the random arm over the same eight instances, configuration `I1_E0_R0`, 10 attempts, every attempt played, attempt history not carried, output cap 16,384, fallback seed 20260913, spend ceiling $10. All 32 arm-by-instance cells completed, at a logged cost of $0.9726. Wins out of 80 were random 15, gemma 8, luna 16 and glm 18.

The 2026-09-15 fallback and replacement probes. Twenty boards built at 3,240 ordered plays, 10 at each end and covering all nine opponent compositions, none of which shares a board key, a picture or a B to C mirror class with any of the 198 study boards. Configuration `I1_E0_R0`, 5

Arm	Host, quantization	Effort	Calls	Fallbacks	Cap hits	Rate (Wilson 95%)
2026-09-10, `pilot_suite.json`, five instances, 10 attempts						
`luna`	OpenAI direct		156	0		0.000 [0.000, 0.024]
`terra`	OpenAI direct		151	4		0.026 [0.010, 0.066]
`sol`	OpenAI direct		160	2		0.013 [0.003, 0.044]
`random keyless`						
2026-09-10, the three fixture instances, 10 attempts						
`luna`	OpenAI direct		71			
`terra`	OpenAI direct		72			
`sol`	OpenAI direct		76			
2026-09-10, `lowtail_suite.json`, five instances, 10 attempts						
`luna`	OpenAI direct					
`terra`	OpenAI direct					
`sol`	OpenAI direct					
`random keyless`						
2026-09-12, `crossed_suite.json`, eight instances, 10 attempts, cap 4,096						
`luna`			360	22	0	0.061 [0.041, 0.091]
`terra`			360	38	0	0.106 [0.078, 0.142]
`sol`			360	11	10	0.031 [0.017, 0.054]
`random keyless`			360	360		1.000

Table 16: Arms of the 2026-09-10 gate runs and of the 2026-09-12 run. A blank cell is a setting or a count the surviving record does not carry, not a zero. The 2026-09-10 arms logged no token usage and no output cap is recorded for them; their call counts are completed API calls counted from their own logs, roughly one per placement, and no cap-hit count survives for them. No reasoning effort was recorded for any arm of either run, and host pinning was introduced only after the 2026-09-12 run, so no arm here carries a pin and the 2026-09-12 arms' route is not recorded either. All ten of sol's cap hits were followed by a fallback. The random arm's fallbacks are by construction, one per placement, since it holds no key, so no interval is given for it.

attempts, every attempt played, output cap 16,384, seed 20260913, 300 placement calls per model in the fallback probe. The builder matched these boards to the average of all 198 rather than to the confirmatory 126, so their ends sit at d 0.1142 to 0.2256 and 0.0241 to 0.0540 and six of the twenty fall outside the confirmatory suite's difficulty range. The replacement probe reran the same boards and settings on two candidate models; neither of its runs was allowed to finish, and why is given with its arms below. Each probe was preceded by a keyless check on one board, and the checks establish different things. The replacement probe's used the real model blocks, changing only the key, so it confirmed that the intended pins, effort and hashes reached the harness. The fallback probe's used keyless stubs carrying no model id, host, quantization or effort, so it established only that the suite loads and logs; that probe's settings are confirmed from its paid run's own logs instead. Neither check cost anything.

E.2 One row per arm

Tables 16, 17 and 18 give every arm of every run with its settings as sent. Calls are agent placements: for a keyed arm each placement is one API call, and the random arm makes none. Rates are the proportion of placements that fell back. Every Wilson 95% interval in these three tables is computed here from the two counts printed in its own row, so no row carries a rate without one, except the keyless random arm, whose rate is fixed by construction rather than estimated.

E.3 The pilot's instance-effect SD, as a range

The quantity the design was sized against is s, the SD of the instance effect, and the pilot fixes it only loosely. Fitted on all eight instances with all three keyed arms, its 95% profile interval runs from about 1.25 to about 4.5; on the two arms that became the headline pair, from about 1.25 to about 5.5; and on the four instances with 29,160 ordered plays or fewer, from about 0.5 to about 5.25. Those are ranges, and the pilot supports nothing narrower: that is why the design was sized at s of 1.5, 2 and 2.5 rather than at one value, and why a rule able to add 72 instances was registered in advance. The pilot's own common slope was 1.99 with a standard error of 0.95 on the three

Arm	Host, quantization	Effort	Calls	Fallbacks	Cap hits	Rate (Wilson 95%)
2026-09-13 probe, eight instances, cap 16,384						
gpt-oss-20b	deepinfra bf16	not sent	72	30	12	0.417 [0.310, 0.532]
gemma	parasail fp8	not sent	72	0	0	0.000 [0.000, 0.051]
glm	baseten fp8	not sent	21	8	8	0.381 [0.208, 0.591]
glm	parasail fp8	not sent	72	72	72	1.000 [0.949, 1.000]
glm	parasail fp8	low	36	5	1	0.139 [0.061, 0.287]
glm	parasail fp8	medium	36	4	0	0.111 [0.044, 0.253]
glm	together	low	36	4	0	0.111 [0.044, 0.253]
luna	openai	low	72	4	0	0.056 [0.022, 0.134]
luna	openai	high	72	4	0	0.056 [0.022, 0.134]
Option 1 pilot, 2026-09-13, eight instances, 10 attempts, cap 16,384						
gemma	parasail fp8	not sent	360	0	0	0.000 [0.000, 0.011]
luna	openai	medium	360	25	0	0.069 [0.047, 0.101]
glm	parasail fp8	medium	360	41	3	0.114 [0.085, 0.151]
random	keyless	none	360	360		1.000

Table 17: Arms of the 2026-09-13 probe and of the Option 1 pilot, both on crossed_suite.json. The probe's arms ran one or two attempts rather than ten and are not a matched set: they vary host, quantization and effort deliberately. The baseten arm ended at 21 calls on five consecutive rate-limit responses from that host's shared upstream pool. In the 72-fallback arm every call spent the whole 16,384 on reasoning and returned empty text, so its 5 wins of 16 came from random placements. Of the pilot's fallbacks, luna's 25 were all illegal placements and glm's 41 were 38 illegal and 3 truncated, the 3 being exactly its cap hits.

Arm	Host, quantization	Effort	Calls	Fallbacks	Cap hits	Rate (Wilson 95%)
2026-09-15 fallback probe, twenty boards, 5 attempts, cap 16,384						
luna	openai	medium	300	12	0	0.040 [0.023, 0.069]
glm	parasail fp8	medium	300	24	1	0.080 [0.054, 0.116]
gemma	parasail fp8	not sent	300	0	0	0.000 [0.000, 0.013]
2026-09-15 replacement probe, the same boards and settings, both arms stopped early						
deepseek-v4.1-flash	deepinfra fp8	medium	23	12	12	0.52 [0.33, 0.71]
qwen3.8-flash	alibaba	not sent	9	9	9	1.00 [0.70, 1.00]

Table 18: Arms of the two 2026-09-15 probes. Calls are resolved placement calls. In the fallback probe luna's 12 fallbacks were all illegal placements and glm's 24 were 23 illegal and one truncated. In the replacement probe both candidates truncated at the cap rather than returning a placement; the deepseek rate is a floor, because 11 further requests timed out at the game's 600 second limit and were probably capped calls, and that host also returned 15 rate-limit responses. The qwen arm ran at its host's default reasoning: the harness sends only the effort parameter that host does not accept, which breaks the fixed-effort condition rather than measuring the model under it.

keyed arms, 2.07 with a standard error of 1.09 on the two that became the headline pair, and 2.73 with a standard error of 1.24 on the four instances with 29,160 ordered plays or fewer. The pilot's own analysis computed no interval for that slope and sized the design from the standard error; the normal-approximation 95% intervals implied by those three standard errors, computed here rather than taken from that analysis, are 0.13 to 3.85, -0.07 to 4.21 and 0.30 to 5.16. The calibration's limits were recorded at the time: s and the intercepts were fitted across instances of very different action-space size while the design simulated only the smallest, and on the two smallest instances alone the model is not identifiable at all.

E.4 Each design choice, with the run that triggered it

Tables 19 and 20 are the point of this appendix. Each row names a choice fixed in the registered design, the feasibility run that produced it, and what that run showed.

Design choice	Triggered by	What the run showed
Output cap 16,384, held fixed across arms and effort levels	the 2026-09-12 run	sol hit the old 4,096 cap on 10 of its 360 calls and every one of those calls was followed by a fallback, so the cap was changing what was being measured rather than bounding it. Raising it also made that run non-comparable with everything after it.
Explicit effort `medium` on the two reasoning arms and none on gemma	the 2026-09-13 probe	at the provider default glm returned 72 truncated replies in 72 calls, each spending the whole 16,384 on reasoning and returning empty text; at `medium` on the same host it had no cap hits in 36 calls. The plan carries the consequence as an inference rather than a measurement: part of the capability spread is then reasoning against non-reasoning.
The keyed set kept as luna, glm and gemma	the 2026-09-13 probe	the alternative set turned on `openai/gpt-oss-20b`, which returned 30 fallbacks in 72 calls, 0.417 [0.310, 0.532], so the switch recorded as conditional on this probe did not happen.
The host pins and quantizations of Table 1	the Option 1 pilot	the plan takes them from this run, as the only pins that had carried a run to completion.
Host pinning kept, with its cost disclosed rather than removed	the 2026-09-13 probe and the 2026-09-15 replacement probe	pinning an arm to one host exposes it to that host's shared upstream pool. One probe arm ended at 21 calls on five consecutive rate-limit responses, and the replacement probe's pinned host returned 15 of them at two different concurrency settings, so backoff, resume and the choice of host are part of the design rather than operational detail.

Table 19: Each design choice with the feasibility run that produced it, first of two parts. Model names are the router's identifiers as sent.

E.5 What the 2026-09-10 runs are and are not

They are exploratory history, and they are kept here because the design answers to them. First, neither of their two generated suites separates what it varies: on the low-band suite the number of legal plays and the mean length of a play are rank-collinear at 1.000 across its five instances, so two structural accounts of the same numbers fit equally well and neither is recorded as the explanation; and on the eight-instance curve formed by pooling the fixture instances with the first suite, the agent's material advantage over the scripted opponent is rank-collinear with d at 1.000, so that curve cannot tell the enumerated measure apart from one subtraction on the instance file. Second, two of the three keyed arms won every attempt of four of the five instances of the first suite, so that suite could not have ordered the arms whatever they did.

None of those numbers is withdrawn and none of them is pooled with anything in this paper. They were measured under the shared-replay regime described above, which the 2026-09-11 change ended, so a later run is not a replication of them and they are not a baseline for one.

The decomposition arm is a separate study with its own plan, and Appendix D carries its apparatus. Its pilot data enter no fit in this paper either, which is the same rule applied to a run that is not a feasibility run.

Design choice	Triggered by	What the run showed
One prompt configuration, `I1_E0_R0`	every run above	every run above used it, the only run-side ground for the choice. The rest are not measurements of these runs: reflection engages only from attempt 3, Liu (2026) resolves components at a finer grain, and Appendix D's arm asks what a second configuration would.
Independent attempts, with attempt history not carried	the 2026-09-10 gate runs	one replay window was shared by the placement prompt and the reflection channel and was cleared once per instance, so attempts 2 to 10 saw the earlier attempts' battle replays and the ten attempts were not ten independent draws. The window was split on 2026-09-11, independent attempts became the default, and which regime a run used is stamped on every log entry.
The trimmed role prompt	the 2026-09-10 gate runs and the fix that followed them	the shipped role prompt instructs the agent to use past game tracethroughs as experience, and under independent attempts there are none, so that paragraph was removed for this study and for the decomposition arm's baseline rung.
The fallback screen's reference level, 0.106	the 2026-09-12 run	it is terra's fallback rate on that run, 38 of 360, carried forward as the level a keyed arm's Wilson lower bound is compared against.
The fallback sensitivity refit is reported and gates nothing	the Option 1 pilot	attempts containing a fallback won 2 of 23 for luna and 3 of 36 for glm, against 14 of 57 and 15 of 44 for attempts without. The instance confounds both comparisons and the subset is selected by the outcome's own mechanism, so the refit is a disclosure rather than a condition on any registered sentence.
glm kept as the second headline arm, at risk and disclosed	the 2026-09-15 fallback and replacement probes	glm's probe rate was 0.080 [0.054, 0.116] and its fallbacks clustered by board, which is what put it at risk on a suite that repeats each instance thirty times: clustering widens the spread of the study-scale count and so raises the modelled chance that glm's fallback-rate Wilson lower bound over 3,780 stage-1 placements would exceed the 0.106 reference level. The at-risk verdict reads that chance, at its largest across the ways the probe modelled it, against bands set after the probe had run: a chance below 0.05 safe, one above 0.50 failing, at risk between. No threshold behind it is registered; the rule those bands fed is registered, with its date and its hash, and it was to replace an at-risk arm before the study rather than drop it from inside. Neither shortlisted replacement beat glm, one truncating on 12 of 23 resolved calls and the other on 9 of 9, and the second cannot be sent a fixed effort at all, so no candidate was found to replace it with.
126 instances at the two ends, with a registered rule able to add 72	the Option 1 pilot	the pilot pins s only to the range 1.25 to 4.5, so no single instance count follows from it and the design was sized across that range instead.

Table 20: Design choices and their feasibility runs, second of two parts. Model names are the router's identifiers as sent.

F NEAR NEIGHBOURS IN DETAIL

The related work section names most of the close neighbours below once and says in a clause how each differs; it does not name Paglieri et al. (2025) at all. This appendix gives the comparison in both directions for every one of them, what each work reaches that this study does not and what this study reaches that it does not, and then covers two groups and five further studies, four of which main text only points at. Nothing in the results depends on anything written here. The single near-neighbour comparison a claim in main text rests on is Grams et al. (2025), and it is made where it is used, in the related work and limitations sections, not here.

Where a claim about another paper is an absence, the corpus that absence is measured over is named, because an absence with no stated bound is not checkable. Where a reading is ours rather than a paper's own statement, it is marked as ours.

F.1 THE NEAR NEIGHBOURS, BOTH DIRECTIONS

Optimum-anchored decomposition. Grams et al. (2025) is the closest work and the source of the algebra used here. Its intermediate is anchored at optimal play given what the agent has actually discovered, and its comparison policy is recomputed after each episode, in its own words, "to measure exploration progress based on the agent's current memory", which makes that intermediate information-matched to the agent rather than better informed than it. This design has no information-matched ceiling of that kind. Its environment is a maze or grid of rooms in which the agent's task is to find out which objects are worth reaching, so a related work search scoped to game benchmarks will not return it. Running the other way, an exact continuation value is available here for every prefix of a play in a deterministic, fully enumerable action space, so that per-decision regret sums exactly to regret over the whole play, and the exploitation side of the gap is split again into a term for simulating the dynamics and a term for selecting among them. One reading connecting the two is ours and not theirs: their exploration term prices what an agent has yet to discover across episodes, this environment gives an agent a single play per attempt and no reward structure to uncover across attempts, so the term has no counterpart here and it is their exploitation gap that the decomposition arm subdivides.

The difficulty construction. Zou et al. (2026) is the source of the difficulty measure and is credited for it in main text. Its task is multi-agent coordination under private information, over a real scheduling constraint system solved by CP-SAT, with difficulty grouped into ranges: "Tasks are then bucketed into easy, medium, and hard ranges based on this score." None of that setting has a counterpart here: no coordination under private information, no constraint solver, and no bucket label. Its own regret is reported as a bound rather than an exact quantity, in its words "The plotted cost uses an upper-bound regret proxy", and the bound belongs to that regret and not to its difficulty, which is an exact structural fraction computed before any agent runs. In the other direction, regret here is resolved per decision against an exact continuation value over prefixes rather than as one number per agent per game, and difficulty is the covariate a slope is estimated on, where the bucket label never appears in a result. The citation pins version 3 because the difficulty definition was rewritten across versions.

Splitting failure against an oracle. Deb & Krishnan (2026) establishes that splitting agent failure against an oracle is an existing move. Its reference policy runs on the same observation stream the agent receives, with one asymmetry the paper discloses, that its filters use the true generative parameters while the agent sees only a qualitative description of the world. Its setting is a 26-week partially observed supply-chain task, it reads a perception axis off the agents' own stated rationales, and it scores against an oracle that is optimal in expectation rather than per run, which it discloses: "Individual scores above 1 occur (33 of 196 scoreable runs) and are legitimate, since the oracle is Bayes-optimal in expectation but not per-tape." This study takes none of those three steps. It works in a deterministic environment, so the comparison value is exact for the play it is computed on and belief is never inferred from what a model says about itself, and the split it takes is simulating the dynamics against selecting within them rather than state estimation against control. The distinction matters for what each design can separate: in that work "search" never appears as a quality axis, so an agent with a correct model that searches badly and an agent with a wrong model are scored the same way. The bound on that absence is that one paper's full text.

Component-level scaffolding. Liu (2026) runs a full factorial over an agent scaffold's components and reports interference between them. That is a resolution this study does not reach, and it is the reason main text calls the work a precursor rather than a competitor: this study runs no factorial of any kind. What this study holds instead is a difficulty measured for each instance before any model runs, and play compared against an exact optimum. No quantity reported in this paper is derived from it, and no sentence of it is quoted.

World models measured separately from play. Lehrach et al. (2025) synthesises an executable world model as code, measures how faithfully it reproduces transitions, plans on top of it with tree search, and also runs a language model directly as a move-selecting policy. On its own headline it reports that its method "outperforms or matches Gemini 2.5 Pro in 9 out of the 10 considered games", which is weaker than a claim of beating that model in nine of them. Combining model quality and decision quality into the realised loss of one agent is the step this study takes and that one does not. There, fidelity and play are measured on different objects and never combined.

An optimality metric named but not computed. Cipolina-Kun et al. (2025) builds an OpenSpiel-backed harness across many games, records the reasoning strings agents produce alongside their moves, and lists among its metrics "decision optimality, measured as the proportion of moves matching the equilibrium or optimal policy". Measured over the copy of that paper read here, "decision optimality", "equilibrium", "optimality" and "optimal policy" each occur exactly once, all inside that one sentence, so the metric is defined and no result is reported against it. Here that quantity is computed. The breadth runs the other way: that harness spans many games on a general backend, where this environment is a single deliberately small one.

The knowing-doing gap as a questionnaire. Paglieri et al. (2025) benchmarks long-horizon agentic play across games, and probes the same gap this study measures with a structured questionnaire on NetHack knowledge whose questions have stated correct answers. It reports that models "can answer questions about the game mechanics and optimal strategies" and yet "fail to apply this knowledge in practice". Its coverage is long-horizon games across several environments, which this study does not attempt. That their scale puts an exact optimum out of reach, and that this is why the gap is probed by questionnaire, is our reading and not the paper's. Here the same gap is read as a continuous loss against an exact optimum rather than as the difference between questionnaire correctness and play.

Search recovered from traces. Chen et al. (2026b) extracts search trees from reasoning traces in a four-in-a-row game and fits models to them, finding search shallower than human search and performance predicted by its breadth rather than its depth. It recovers the shape of a model's deliberation without computing anything about the game's optimum. No such shape is recovered here. That it computes nothing about the optimum is an absence we observe rather than one it states, and it is observed over its abstract and nothing wider. The optimum is held here instead, so a shallow search can be priced in lost win probability rather than described in tree shape.

The two occupied phrases. Main text names Xie et al. (2024) and Schwoebel et al. (2026) as the prior occupants of "planning error" and "model error". Neither is a method neighbour, and both are cited so that a reviewer does not have to point out the collision. The detail behind that decision is small and checkable. "Planning error" occurs twice in TravelPlanner, as the title of its Section 5.2 and in its table of contents, and that section is a pass-rate breakdown by constraint type, not a split of realised loss against an optimum, so the phrase is occupied without the result being scooped. "Model error" occurs twice in World-Time Compute, in the sense of a learned simulator being wrong. For the terms this paper does use, we checked rather than assumed: across the eight texts read in full for this comparison (Xie et al., 2024; Miller, 2024; Malek et al., 2025; Shojaee et al., 2025; Chen et al., 2025; Huang et al., 2026; Pereira & Zuidema, 2026; Schwoebel et al., 2026), "search error" does not occur and neither does "world model error". The bound on that statement is those eight texts and nothing wider. Those eight are a set of downloaded full texts rather than a survey, and they were fixed as a corpus before this comparison was written, so the bound is reported as it stands rather than widened after the fact. They do not include the near neighbours compared above. Six of those were checked separately, and neither phrase occurs in the full texts of Grams et al. (2025), Zou et al. (2026), Deb & Krishnan (2026), Lehrach et al. (2025), Cipolina-Kun et al. (2025) or Paglieri et al.

(2025). The other two are not covered by any check reported here: no full text of Liu (2026) was obtained, and only the abstract of Chen et al. (2026b) was. Neither phrase was searched in either, and no absence of either phrase is claimed for them.

F.2 Scaffolding and component-level prior art

This study varies no scaffold. Its agent runs under one fixed prompt configuration throughout the confirmatory arm, and the ladder in the decomposition arm varies what the agent is handed rather than what components it is built from. The scaffolding literature is therefore adjacent rather than competing, and it is cited here so that the boundary is explicit. Chen et al. (2026a) isolates scaffold components under controlled composition, which is the design this study does not run. Zhang et al. (2026) carries the interference framing into harness optimisation, and optimises a harness rather than measuring what each part of one buys. Singh (2026), a 2026 workshop preprint, reports scaffold effects that differ across difficulty, which is consistent with the question asked here and does not settle it. The components themselves have standard references: reflection over an agent's own attempts (Shinn et al., 2023), experience synthesised into higher-level feedback (Park et al., 2023), and skill acquisition with memory in an open-ended environment (Wang et al., 2023). All of these vary an agent's architecture and report what each part contributes. Here the architecture is fixed and the measurement is against an exactly computed optimum, on an axis defined before any model runs.

F.3 Domain adjacency

Existing benchmarks put language models into the same family of games as this environment, and a reviewer will reasonably ask about them. Yuan et al. (2025) evaluates planning, revision and resource-constrained decision making across strategic games, scored by metrics beyond win rate such as overcorrection risk rate, correction success rate, improvement slope and over-budget ratio. Wang et al. (2026) is a tower defence learning environment and benchmark. The environment here is narrow by design: three placements against a fixed non-adaptive script. That narrowness is what makes it small enough to enumerate every complete play, which is what supplies both an exact per-instance difficulty and an exact optimum to score a decision against. That is the trade this environment makes: the games here are small, and in exchange every quantity about them is computed rather than estimated.

F.4 Difficulty, world models, and what a planning score names

The related work section reaches the two controlled-complexity reference points. Success falls as instances get harder in this suite as well, so no result of theirs is contradicted; what this design changes is how such a curve should be read rather than whether it holds. Further studies bear on the design and are cited here.

Pereira & Zuidema (2026) trains small transformers on the Tower of Hanoi and shows that they develop a linearly decodable, geometrically faithful representation of the task's state space that is causally involved in solving it, then finds that two frontier reasoning models encode that same structure near-perfectly at the end of the prompt and still fail on most tasks at four rings or more, with the failure located in the representation decaying as generation proceeds. It is the clearest available case of a correct internal model that is not used in action. It reads the model's internals and intervenes on them, which this study does not do. The same gap is measured here from realised play against an exact optimum, which says nothing about internals but requires no access to them, and so applies to models served behind an API.

Chen et al. (2025) attributes agent failure in unfamiliar environments to the quality of the agent's grounding in states and dynamics rather than to its search, and shows that teaching state estimation and transition prediction before optimising for reward recovers much of the loss. It changes the agent by training it, which is how it establishes the attribution. The same two candidates are separated here without training anything, by changing what the agent is handed and measuring what the change buys, which is the only route available when the models under study cannot be fine-tuned.

Of the work compared in this appendix, Malek et al. (2025) is the closest prior separation of simulating from planning on one environment. It recasts a gridworld as a fully observable textual task

in which every source of failure except planning is removed by construction, asks separately for the state that follows an action sequence, for a plan, and for a decomposition into subgoals, and reports simulation far ahead of planning for every model tested, with the length of the required solution rather than the size of the grid governing where planning collapses. Its split comes from asking for different outputs from the same model on the same environment. The split here comes from handing the agent different information in a single task that is scored one way throughout, and the driver of failure is a property computed for each instance rather than a knob the environment was generated with.

Huang et al. (2026) fits a multidimensional item response model to planning benchmark results and finds two distinct competencies rather than one: an operational one, which improves with scale and with longer reasoning, and a structural one, which is comparatively insensitive to both. The consequence for this study is a caution rather than a contrast, and it is worth stating against our own design. The dependent variable here is a single binary outcome on one environment, which is an aggregate of exactly the kind that result warns about. What limits the damage is that no claim here is about a latent planning ability: the registered quantity is the slope of success on a covariate measured from each instance's own structure, and a slope on a measured covariate does not require the outcome to load on one trait. What would require it is a claim ranking one model's planning ability against another's, and the one cross-model quantity this paper reports is a registered descriptive secondary that carries no test.

Park et al. (2024) is the direct precursor for regret as a measure of language model agents' behaviour, in the online learning sense: regret accumulated against a sequence of adversarial rounds. That repeated-game setting has no counterpart here. Regret in this paper is defined per decision against an exact continuation value for the prefix reached, in a single deterministic play. The two senses should not be read as the same quantity. This paper's regret is confined to the decomposition arm, is not part of the confirmatory set, and carries no claim in the results.

G Run record

This appendix is the ledger behind the confirmatory results section. That section states the harness check, the screen result and the fallback sensitivity with their numbers; what follows is the record those numbers were read off, in the order the run produced it. Every figure here is read off the run's own artefacts: its status index and its log, the screen and trigger records the plan requires, the call records the provider check re-reads, and the spend, call-outcome and screen tables the analysis wrote beside the specification it ran. What comes from outside those files is the plan's own figures and the freeze record's, named as such where they appear: the plan's ceilings, cost expectation and reference levels, and the plan's hash time and the analysis wall clock, which the run's freeze record carries.

G.1 Wall clocks and the order of operations

The run measured 504 cells, four arms over 126 instances at one prompt configuration. It ran as a single invocation, from 2026-09-16T22:21:18Z to 2026-09-17T06:33:18Z, 8 hours and 12 minutes, with at most six router-backed cells and at most two local cells running at once. It finished with its state recorded as complete, 504 of 504 cells measured, and no stop reason.

The plan's content hash was computed at 2026-09-16T19:23:45Z, just under three hours before that first call. Those are the two timestamps behind the scoping the design section uses: the plan was fixed and hashed before any of the data it analyses were collected.

What followed ran in the order the plan fixes. The fallback screen was computed at 2026-09-17T07:27:27Z, once, on the complete stage-1 data. The blind extension trigger wrote its record 56 seconds later, at 2026-09-17T07:28:23Z. The confirmatory analysis ran last and exited 0 at 2026-09-17T10:33Z, which is the write time its own output carries. The run's freeze record puts that at 3 hours and 3 minutes of serial wall clock; no file timestamps the start.

G.2 Spend and call counts

The confirmatory run cost $9.1544 against a stage-1 ceiling of $75.00, over 11,514 calls.

Arm	Calls	Answered	Failed	Priced (calls)	Unbilled (calls)	Spend (USD)
luna	3,780	3,780	0	3,780	0	5.9639
glm	3,791	3,780	11	3,786	5	2.5202
gemma	3,943	3,780	163	3,781	162	0.6704
random	0	0	0	0	0	0.0000
Total	11,514	11,340	174	11,347	167	9.1544

Each keyed arm's answered calls come to exactly 3,780, one per agent prep placement, so no failure cost a placement. All 504 cells completed and no api_error fallback was logged, which is what a retry inside the failing call's own cell would look like; those retries were not audited call by call. The null arm made no calls at all, because it holds no key. The failures themselves were gemma's 161 HTTP errors, one provider error and one transport failure, and glm's four HTTP errors, six provider errors and one transport failure; luna had none. The Priced column counts calls the router priced in its own response, and that stated price is what was summed; the Unbilled column counts calls that failed without reporting usage, which are charged nothing. Both are counts of calls and neither is a sum of money; the only money in the table is its last column, whose Total is the sum of the unrounded per-arm costs, so it prints $9.1544 where the four rounded figures above it add to $9.1545. No call in this run was priced from an estimate or from the manifest's price table, so the total is billed cost rather than a reconstruction. The ceiling is read in the same raw logged dollars, and the run stops when the plain sum of per-call cost reaches it.

The plan's raw expectation for three models over 126 instances was $7.38. The run cost more than that and is reported at what it cost; the $75.00 ceiling is the number that binds, and it was never approached.

G.3 Fallbacks by cause

A fallback is an agent prep placement whose chooser is not the model. The plan names five reasons, and a keyed arm showing either of two further reasons, api_error or no_api_key, is flagged as a harness question rather than counted as ordinary model behaviour. All seven are below, with the placements the model itself made on the last row.

Cause	luna	glm	gemma	random
empty_content	0	0	0	0
truncated	0	19	0	0
unparseable	2	6	0	0
illegal_placement	192	217	4	0
unstated	0	0	0	0
api_error	0	0	0	0
no_api_key	0	0	0	3,780
All fallbacks	194	242	4	3,780
Placed by the model	3,586	3,538	3,776	0

Nearly every fallback in a keyed arm is an illegal placement, which is a reply the harness could read and could not play. The null arm's 3,780 are no_api_key by construction, since it is keyless and every one of its placements is drawn at random; that is the arm behaving as designed and is the one case where a fallback count of 100% is the intended reading. No keyed arm produced a single api_error or no_api_key fallback, so the harness question the plan attaches to those two reasons did not arise for any of them.

glm's 19 truncated placements are its 19 output-cap hits at this run's cap of 16,384 tokens; luna and gemma had none. They are counted as fallbacks and are inside the screen below. The capped calls reported in the decomposition section belong to that arm's separate run and not to this one.

G.4 The fallback screen in full

The plan's screen is computed once, on the complete stage-1 data, before the extension trigger's fit, and it excludes nothing. It reports, per keyed arm, the fallback rate over agent prep placements,

a Wilson 95% interval on that rate, the counts by reason already given above, output-cap hits, the share of attempts holding at least one fallback, and whether the interval's lower bound exceeds the 0.106 reference level. The interval uses a z of 1.96, and the screen read 504 complete cells.

	luna	glm	gemma
Agent prep placements	3,780	3,780	3,780
Fallbacks	194	242	4
Rate	0.0513	0.0640	0.0011
Wilson 95% lower bound	0.0447	0.0567	0.0004
Wilson 95% upper bound	0.0588	0.0723	0.0027
Lower bound above 0.106	no	no	no
Output-cap hits	0	19	0
Attempts holding a fallback	193	223	4
Share of the 1,260 attempts	0.1532	0.1770	0.0032
`api_error` or `no_api_key`	0	0	0
Harness question raised	no	no	no

No arm's lower bound reaches 0.106, so the screen names none. That is the finding, not an absence of one: the line the plan requires above the per-model tables in Appendix B is written and says none, and the same emptiness is what the confirmatory results section reports. The distance is large rather than marginal. At 3,780 placements a lower bound above 0.106 takes about 438 fallbacks, a rate near 11.6%, and the highest rate observed is glm's 242 at 0.0640. Crossing the level would have been reported and would have named that arm in the Appendix B line. It is a reference level for reporting and nothing more: whatever the screen shows, the plan drops no model from any fit, keeps it out of no trigger input and changes no registered sentence.

G.5 Provider constancy

Each keyed arm is pinned to one host, and the plan requires the provider actually logged beside each call to be checked for constancy within the arm. The check passed, with exit 0 and one provider per arm: luna was served by OpenAI on 3,780 of 3,780 calls, glm by Parasail on 3,786 of 3,791, and gemma by Parasail on 3,781 of 3,943, each across 126 complete tries. The calls in the two shortfalls are each arm's HTTP errors and transport failures, gemma's 161 HTTP errors with one transport failure and glm's four with one: the router never served them, so they carry no provider in their response metadata and name none. A provider error was served, names its provider and is counted above, which is why each shortfall falls short of the Failed column of the spend table by that arm's provider errors. Those unserved calls are also what the Unbilled column counts, and that is more than two counts agreeing: a call is unbilled only if it was not served and reported no usage, and since no call in this run was priced from an estimate, each arm's priced count equals its served calls, so the calls left over are exactly the unserved ones. No call in any arm named a second provider. The provider is read from the router's own response metadata rather than from the field the game copied out of it, and the two are held against each other on every call. What passing this check does not establish is set out in the limitations section.

One thing in the analysis's own outputs can be misread: a header-only output file reads like a failed run and is not one. Five of the 14 tables it wrote hold their header row and nothing else. Three belong to an estimand the specification does not declare, each keyed arm's edge over the null arm, and they hold that edge instance by instance, its mean over instances, and its tests; those tests are not the licence test's two edge tests at 0.6 and 1.4, which the confirmatory results section reports and which were computed. `wins_by_end.csv` describes per-model slope fits that did not converge, and all three converged; `floor_refit.csv` holds a refit the plan runs only on the 198-instance pooled analysis, which is why the confirmatory results section carries the floor as a stated limitation and not an estimate.

G.6 The blind extension trigger

The plan fixes a rule that could have added 72 instances to the suite, and fixes it so that the decision is taken before any result is visible. After stage 1 was complete and had passed the plan's pre-analysis conditions, a trigger fitted the registered model to the two headline arms' stage-1 data at $Q = 15$

and wrote nothing but the plan's four values: the fit's verdict with its reason list, $\hat{s}$, the decision and a timestamp. The registered blinding is that no slope, level or test is computed before the decision is recorded. Fitting the registered model estimates a slope, so what the implementation can enforce is narrower, and it enforces exactly that: no slope, level or test was written or read. A fit diagnostic that names the slope parameter is replaced in the reason list by a fixed string recording that it was withheld.

The record reads `converged` with no reasons, $\hat{s} = 1.4059$, decision `stop`, timestamp 2026-09-17T07:28:23Z. That $\hat{s}$ is the same number to every digit as the one the confirmatory results section reports, and it is the same quantity: the trigger fits the registered model, so what it writes is that model's instance-effect SD, and the registered rule is keyed on that SD and could not be evaluated any other way. It is the only number in the record the confirmatory analysis reports as well. It is not one of the four registered tests' statistics, Q4's being φ, and no test statistic and no p value existed when the record was written. The registered rule extends when the fit is not `converged`, when it is separated, or when $\hat{s}$ exceeds 2.25; none held. So the confirmatory set stayed at 126 instances, the 72 extension instances were never collected, the extension manifest was never run, its own $40.00 ceiling was never spent, and the pooled 198-instance specification was never used.

The checks the plan requires were made before that fit ran. Provider constancy passed with exit 0. No headline model had been lost, which would have stopped the study at 126 without a trigger. The attempt-history check was discharged by the pre-launch rehearsal, since it needs a stub's request record and there is none in a paid run. And the fallback screen above had been computed and reported. The screen is not an input to the fit: the specification declares no screen exclusion, so a trigger handed a screen file under it refuses to run, because no model could be excluded by one.