SEPARATING STRUCTURAL SEMANTICS FROM INFERENCE ALLOCATION IN LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM-agent comparisons often change both what the model is asked to do and how much inference it receives, then treat realized resource consumption as a control. We separate structural semantics S, ex-ante allocation A and realized downstream cost C, and explain why adjusting for C does not in general estimate the effect at fixed allocation. Our intervention requests $m \in \{1, 3, 5\}$ separately seeded candidates under byte-identical prompts; in sequential control a fixed plurality rule selects the action. We audit prompt invariance using full-message hashes in program synthesis and system-prompt hashes plus prompt-construction tests in control. In a preregistered factorial over four models, four cooperative-cooking layouts and five seeds, allocation contrasts range from -0.563 to -0.338 soups per episode. All four percentile-bootstrap intervals lie below zero, but each admits a drop smaller than our preregistered practical margin, so these data do not establish practical harm. Structural effects remain imprecisely estimated. In a separate matrix of 2,624 HumanEval candidate pools, both the frozen and corrected original-test scorers place the structural effect at $m = 5$ within the fixed equivalence margin. This oracle-success metric has no aggregator and is not a replication of the control result. Observed changes run opposite to both preregistered mechanism predictions; the mechanism remains unresolved. The results concern the tested sampling-and-plurality policy, whose candidate count and selection rule are not separately identified.

1 INTRODUCTION

Suppose a role prompt, a critic, or a self-contribution prompt improves an LLM agent's score. Such a comparison may change the requested behaviour, the inference allocation, or both. When a scaffold also changes the ex-ante allocation policy, the resulting comparison does not by itself isolate the fixed-allocation effect of the prompt, and reading it as a fixed-allocation architectural effect changes the estimand. A rise in realized cost alone does not show that both factors moved: holding the allocation fixed and changing the prompt also can change cost, and that comparison is well posed. Recent work makes the same observation: apparent advantages of multi-agent systems are "better explained by unaccounted computation and context effects rather than inherent architectural benefits" (Tran & Kiela, 2026).

Our position is that the remedy is not a better scaffold but a cleaner set of quantities: *LLM-agent evaluations should distinguish changes to the semantic decision rule from changes to ex-ante inference allocation and from realized downstream cost.* We write S for the semantics of the decision rule, what the model is asked to do; A for the ex-ante allocation, how many times it is asked and the fixed rule that turns the answers into one action; and C for the realized cost (Figure 1). C is produced by the treatment, so adjusting for it does not in general estimate the effect at fixed allocation, and a common spending limit has to be imposed in advance rather than recovered by keeping the runs that spent alike (§2).

The separation is usable only if A can be moved without moving S. We do that with repeated separately seeded sampling under a byte-identical prompt followed by a fixed plurality rule over the parsed actions, and we define the requirement so that it is checkable: the request a given structural level and context produces must not depend on m. We audit that from the request logs and from pre-

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

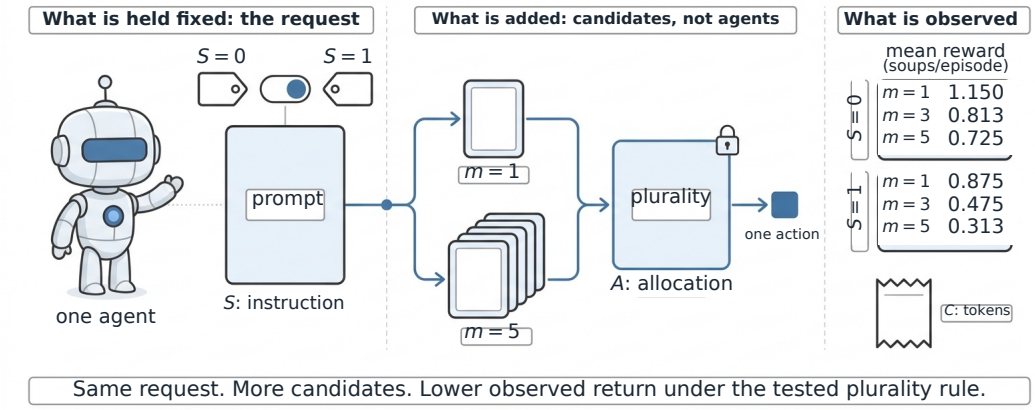


Figure 1: The separation this paper asks for. At a fixed structural level and decision context, raising m repeats the same request and hands more candidates to one fixed plurality rule: it does not add agents, and the request itself does not change (§3.1). The six Domain-1 cell means are printed as reported in Table 1, which also gives the contrasts and their intervals; token means rise across the same cells and are reported in Appendix F. The reading is an observation about the tested sampling-and-plurality policy at this panel of models and layouts, not an identified mechanism and not a claim about inference scaling in general. The drawing is a schematic; every character on it is typeset by the figure script from the audited values.

run tests, with full-message hashes in one domain and system-prompt hashes plus prompt-construction tests in the other (§3.1).

What we find. Two matrices were collected under one preregistered plan; subsequent scoring corrections and robustness checks are reported separately. In cooperative sequential control (four models $\times$ four layouts $\times$ five seeds $\times$ two structural levels $\times$ three allocation levels; 115,200 calls), increasing candidate allocation under the preregistered plurality policy *reduced* pooled reward at both structural levels, with contrasts from -0.563 to -0.338 soups per episode with all four intervals excluding zero, negative in 15 of 16 stratum-level allocation contrasts, all layouts, and all leave-one-model-out holdouts. No tested stratum reverses the direction and all four preregistered percentile-bootstrap intervals lie below zero, but each also admits a drop smaller than our preregistered practical margin, so these data do not establish practical harm. We find no statistically resolved benefit from the structural prompt at any tested allocation, and its point estimates are negative at all three. In program synthesis (2,624 nested candidate pools over the 164 HumanEval problems), a second structural intervention of the same form is practically equivalent to zero at $m = 5$ inside the preregistered margin; that domain is not a replication of the first, because pass@m has no aggregator and a voting-induced degradation has nowhere to appear in it. Two mechanisms were preregistered, and on the frozen data the observed changes run opposite to both predictions, so we report the mechanism as unresolved rather than replacing them with an explanation found afterwards.

What is not new here. The direction of the sequential-control result is known: Chen et al. (2024) show majority-vote accuracy that "can first increase but then decrease as a function of the number of LM calls", Lee et al. (2026) report a web-agent case where success drops as budget grows, and Wang et al. (2024) already vote over sampled actions per step. The repeated-sampling protocol is Brown et al. (2024)'s, and Proposition 3 restates Chen et al. (2024)'s hard-query branch for a single context. We claim none of those. Three things are ours. A separability criterion stated so that a script can check it, with the scope of the check reported per domain (§3.1). A preregistered factorial estimating, within each domain, its prompt intervention, the allocation ladder and their interaction on the same cells (§3). And an adjudication of every contrast against margins fixed before collection, including the verdicts that resolve nothing (§4–§6). A prior version of this design is disclosed as confounded in Appendix P.

2 Structural Semantics, Ex-Ante Allocation, and Realized Cost

Three quantities. At each decision point the agent is in a context x: the state, the partner's state, and whatever history the harness exposes. S**, the semantics of the decision rule**, is a structural level s fixing a distribution $P_s(\cdot \mid x)$ over completions, induced by a model, a decoder configuration, and a prompt that is a fixed function of (s, x). A**, the ex-ante inference allocation**, is a level m fixing how many times $P_s(\cdot \mid x)$ is sampled and the rule reducing the results to one action, chosen before the decision rather than in response to it. C**, the realized cost**, is what the run consumed, produced by the treatment and available only after (s, m) are set. A parse map ϕ returns either a legal action from the action menu $\mathcal{A}(x)$, which is not the allocation A, or $\perp$ when no legal action was named. Write q for the probability of a legal parse and π_s for the distribution of parsed actions given a successful parse, a distribution on $\mathcal{A}(x)$. Level m draws $Z_1, \ldots, Z_m$ i.i.d. from $P_s(\cdot \mid x)$, counts the valid parses $V = |\{j : \phi(Z_j) \neq \perp\}|$, and applies a fixed aggregator g_m, here plurality among the valid parses with ties broken by menu emission order and a no-op when $V = 0$.

Separability, defined so that it can be checked. The separation is usable only if A can be moved without moving S. Stating that as an assumption alone would leave it unchecked, so we state it as a property of request construction with explicit audit conditions.

Definition 1 (Allocation separability)**.** *A_m is separable from S if, for every context x and structural level s, the prompt string presented to the provider and the decoder configuration are the same functions of (s, x) at every m.*

Written this way it is auditable: write $Q(s, x, m)$ for the request produced at context x and level s under allocation m, hash it, and check that the hash does not depend on m. The requirement is $Q(s, x, m) = Q(s, x)$, not that distinct contexts produce distinct requests, since a renderer may map two states to one message. What that delivers is bounded: visible-request invariance implements Definition 1 only under the further assumption of a stationary provider kernel, and only then does m index how many times $P_s(\cdot \mid x)$ is sampled rather than what $P_s(\cdot \mid x)$ is; it certifies neither conditional independence of the draws nor backend stationarity. §3.1 reports what the audit covers in each domain, which is not the same in both. We claim no novelty for the intervention itself (Brown et al., 2024; Wang et al., 2024; Lee et al., 2026); what we require of it is only Definition 1.

Estimands, and one identity. Let $Y(s, m)$ be the episode outcome and $\mu_{s,m} = \mathbb{E}[Y(s, m)]$. We report exactly three contrasts, all differences of means over the same design cells:

$$\Delta_S(m) = \mu_{1,m} - \mu_{0,m}, \quad \Delta_A(s; m, m_0) = \mu_{s,m} - \mu_{s,m_0}, \quad I(m_1, m_0) = \Delta_S(m_1) - \Delta_S(m_0). \tag{1}$$

Proposition 1 (Path dependence)**.** $I(m_1, m_0) = \Delta_A(1; m_1, m_0) - \Delta_A(0; m_1, m_0)$.

This is bookkeeping: both sides expand to $\mu_{1,m_1} - \mu_{0,m_1} - \mu_{1,m_0} + \mu_{0,m_0}$. It earns a number because it makes the two readings of the interaction one question with one answer, computed two ways that check each other.

Fixed allocation and fixed realized cost are different questions.

Proposition 2 (Fixed allocation and fixed realized cost are different estimands)**.** *Fix m and let $C(s, m)$ be realized cost, a random variable determined by the treatment together with run-level randomness. Where both conditional means are defined, write $D_{\mathrm{cond}}(m, c) = \mathbb{E}[Y(1, m) \mid C(1, m) = c] - \mathbb{E}[Y(0, m) \mid C(0, m) = c]$. Then $D_{\mathrm{cond}}(m, c) \neq \Delta_S(m)$ in general, because conditioning selects, within each arm, the runs that happened to spend c, and that subset is defined by a variable the treatment caused.*

This is a standard result rather than a new one: adjusting for a concomitant variable affected by the treatment can bias the estimate or change the estimand (Rosenbaum, 1984; Montgomery et al., 2018). Three comparisons should be kept apart. Holding m fixed and varying the prompt estimates the effect of structure at a common allocation. Reporting reward alongside realized cost describes what was spent and obtained. Asking which system wins under a common spending limit is a third question, needing a budget-constrained policy specified in advance with its own stopping rule; it is *not* answered by keeping the runs that happened to spend alike, since that subset is selected by a variable the treatment caused. What we add is to name the variable that plays this role in an agent evaluation, and to show in §7 that the distinction is live on our data.

More sampling carries no monotonicity guarantee. A negative allocation effect needs no exotic mechanism, because the aggregator promises no positive one: more draws mean a better estimate of the *mode* of π_s, and the modal action need not be the best action.

Proposition 3 (No monotone return guarantee, at perfect parsing)**.** *Fix a context with a finite menu, suppose every draw parses ($q = 1$), and suppose π_s has a unique mode $a^\star$. Then $\Pr[g_m(Z_{1:m}) = a^\star] \to 1$ as $m \to \infty$, so for any bounded per-step reward r, $\mathbb{E}[r(g_m)] \to r(a^\star)$. Since $q = 1$ gives $\mathbb{E}[r(g_1)] = \mathbb{E}_{\pi_s}[r]$, the hypothesis $r(a^\star) < \mathbb{E}_{\pi_s}[r]$ places the limit strictly below the value at $m = 1$.*

A two-action instance suffices: $\pi_s(a_1) = 0.6$, $\pi_s(a_2) = 0.4$, $r(a_1) = 0$, $r(a_2) = 1$ gives $\mathbb{E}[r(g_1)] = 0.4$ and $\mathbb{E}[r(g_m)] \to 0$ (Appendix D). The $q = 1$ hypothesis is deliberate: with parse failures the $m = 1$ value is instead $q\,\mathbb{E}_{\pi_s}[r] + (1 - q)\,r(a_{\text{noop}})$, and abstention is handled by the conditional bound of Appendix E.

Remark 1 (Attribution, and what this does not say). Proposition 3 is not ours. It is the single-context restatement of the hard-query branch of Chen et al. (2024), whose Theorem 2 gives the conditions under which mixing hard and easy queries produces a non-monotone aggregate. **It is a possibility result and not an explanation of the degradation we observe.** We never show that the modal action is the bad action in our environment, and §6 reports two preregistered mechanism attempts whose observed directions run the other way. It is also local. Which states an episode visits, and with what weight, is a function of the policy and therefore of m, so this is not an episode-level decomposition or a mechanism identification (Appendix N).

The supplementary Lean files cover additional algebra: a Shapley reading, a Lipschitz bound, a concentration bound for plurality under imperfect parsing, the pass@m identity, and the aggregator's invariance properties. Those and Proposition 1 are formalised in Lean 4 with Mathlib; Propositions 2 and 3 are not, and Appendix E says what the formalisation covers.

3 An Allocation-Separable Design, and Its Mechanical Verification

Estimands, strata, the bootstrap and its seed, the practical-equivalence margins and the stopping rules were fixed in a preregistration before any outcome was computed, and the design constants were frozen by hash before launch. Both matrices were collected once. The seeded analysis was re-run after repairing a resampler defect (Appendix L); later scorer corrections and robustness checks are explicitly post-hoc (Appendix Q).

Domain 1: cooperative sequential control. Two LLM-driven agents share an Overcooked-style kitchen (Carroll et al., 2019) and must assemble and deliver onion soups over a 200-tick episode. Every five ticks each agent is shown the kitchen state, both agents' holdings, and a menu of legal high-level actions, and returns one: 40 decisions per agent, 80 calls per episode at $m = 1$, 480 episodes and 115,200 calls in all. The outcome is soups per episode, an integer from 0 to 5 here. The four layouts are a within-family sweep of station contention fixed before collection by an a-priori geometric quantile rule, so layout selection is outcome-independent (Appendix C). At $S = 0$ the system prompt states the rules, the kitchen contents, the goal and the response format. At $S = 1$ it appends a block headed *value decomposition strategy*, instructing the agent to estimate its own contribution to the team score, choose the action maximising it, and state the estimate first. This is a prompt package, not learned value factorisation or joint training. The chosen goal is executed for five ticks by a scripted low-level controller. Same-executor competence controls are reported in Appendix Q.

Domain 2: program synthesis. The second matrix uses the 164 problems of HumanEval (Chen et al., 2021), unmodified, scored by the benchmark's public evaluation tests, withheld from the generation prompt. Each cell is an ordered pool of five candidate programs and m indexes a *prefix*, so $m = 1 \subset m = 3 \subset m = 5$ by construction and no candidate is regenerated when m changes; four models $\times$ 164 problems $\times$ two structural levels $\times$ two replicates gives 2,624 pools and 13,120 candidates. The outcome is pass@m over the frozen prefix, used throughout as an *oracle sampling-success* measure: it asks whether the pool contained a passing program, not whether a deployable system could have selected one. That matters more than usual here, because Domain 2 has no aggregator. Its $S = 1$ is *not* the Domain-1 prompt; it appends "before producing the final program,

check likely edge cases internally" to an otherwise identical coder instruction. The two share only their form, so they are two instances of a structural intervention rather than one manipulation applied twice, and nothing here treats Domain 2 as a replication of Domain 1's structural contrast.

The allocation intervention. In Domain 1, at level m the harness draws m completions from the same decision rule and executes the plurality action among the valid parses, ties broken by menu emission order and a designated no-op when none parses (Figure 1b). Decoding is temperature 0.6 with an output cap of 256 tokens in Domain 1 and 800 in Domain 2, one value per domain at every S and m. Each draw carries a distinct seed $\mathrm{sha256}(\text{salt} \mid \text{context} \mid \text{draw index})$. Distinct seeds and separate requests implement the intended sampling protocol; conditional independence of the draws and stationarity of the provider's backend remain assumptions that request hashes cannot certify. Domain-1 episodes were collected in fixed layout, structural-level, allocation-level and seed order within each model, not in a randomised treatment order. Treatment and collection time can therefore be associated; fixed request configurations do not rule out temporal provider drift.

3.1 Verifying allocation separability, and the manipulation checks

We are precise about what the check covers. In Domain 2 the recorded hash is over the full message list, taking 328 values, which is 164 problems times two structural levels, constant across m, model, replicate and candidate index. In Domain 1 it is over the system prompt alone, taking 8 values; the state-conditioned user message is not hashed per request. Whole-message invariance there rests instead on a pre-run byte-identity test over 32 harvested states and a signature test that m is not a parameter of any prompt, menu or decoder builder. One decoder hash covers all 128,320 calls, but it hashes the harness configuration object rather than the effective request parameters, so it is hygiene rather than a finding. Appendix C gives the remaining counts and the seven canonical run hashes.

Three manipulation checks were computed before any treatment contrast, as separate entry points in the analysis program. Parse validity is $q = 0.9989$ against a preregistered floor of 0.90, and the observed fraction of decisions receiving no valid draw is at most 0.0022 and is zero in the observed records at every $m > 1$, so the aggregator's no-op branch was never reached in an observed $m > 1$ decision; at $m = 1$ it is reached in at most 0.0022 of them. The allocation arm is live: the draws disagree in 0.425 of decisions at $m = 5$ and plurality differs from the first draw in 0.137, about one decision in seven. Ties are rare (0.040) on a menu averaging 4.03 actions, so the sparse-vote failure that makes exact-match voting unusable in large action spaces (Wang et al., 2024) is not our regime.

3.2 Inference, margins, and the limits of this panel

All intervals are 95% percentile intervals from a crossed cluster bootstrap resampling models and layouts independently, 10,000 draws, with a seed fixed in the preregistration; in Domain 2 the second factor is the problem. Clustering on models treats the panel as a sample of models rather than as replicates, which is the assumption we want and the reason the structural intervals are wide. Two properties of the intervals we obtained belong here. Every Domain-1 interval is wider than the margin it is judged against, half-widths 0.250 to 0.638 soups against ± 0.20, and an interval wider than the margin cannot be contained in it, so no equivalence verdict is reachable for these estimates. That bears on equivalence only: the reliably-worse verdict asks whether the upper endpoint falls below -0.20, which a wide interval can satisfy and ours do not. Separately, we do not calibrate this bootstrap's coverage at four clusters, so its nominal 95% level is not guaranteed here, and every "interval excludes zero" statement below should be weighed against the stratum-wise evidence, which does not depend on the resampling procedure. Every contrast is reported per model, per layout and under leave-one-model-out (Appendix I).

Practical margins, and the panel. The preregistered margins are 0.20 soups per episode in Domain 1 and 0.02 in pass@m for Domain 2, inherited unchanged from the preceding study's preregistration; 0.20 soups is about 17% of the $S = 0$, $m = 1$ mean. An estimate is practically equivalent only if its interval is contained in $[-\delta^*, +\delta^*]$, reliably worse only if it lies entirely below $-\delta^*$, and unresolved otherwise. The panel is `glm-5`, two Qwen3 models (235B mixture-of-experts, 32B dense), and one entry identified only by the provider alias **DeepSeek API** (`deepseek-chat`), because the provider served a revision the alias does not name (Appendix H).

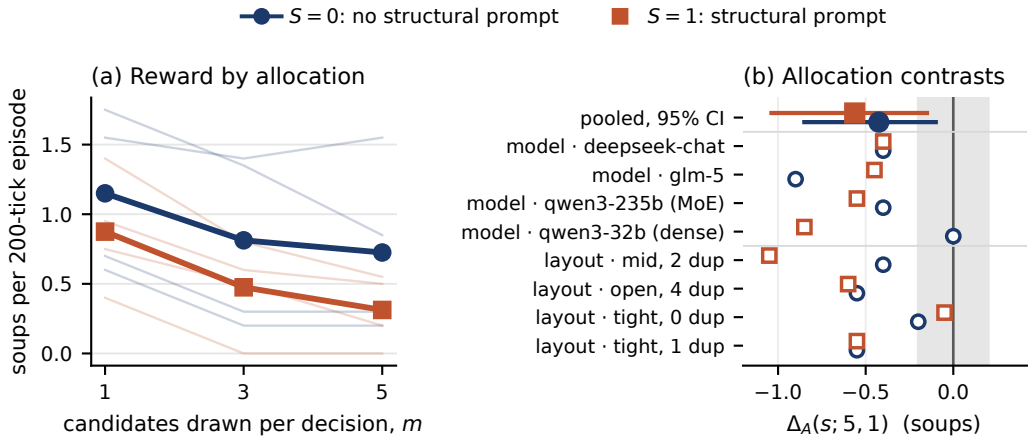


Figure 2: Domain 1. **(a)** Reward falls in the number of candidates drawn per decision at both structural levels; bold lines pooled, faint lines individual models. **(b)** $\Delta_A(s; 5, 1)$ by stratum, $S = 0$ below each row's tick and $S = 1$ above; filled markers with bars are pooled estimates with 95% crossed cluster-bootstrap intervals, open markers individual strata, and the shaded band is the preregistered ± 0.20 region of practical equivalence. Every pooled interval excludes zero and also overlaps the band.

Table 1: Domain 1, soups per 200-tick episode. Each column entry pools 80 episodes, the four models × four layouts × five seeds at that (S, m). Intervals are 95% crossed model × layout cluster bootstrap, 10,000 draws.

	$m = 1$	$m = 3$	$m = 5$
cell means $\mu(S, m)$			
$S = 0$ (no structural prompt)	1.150	0.813	0.725
$S = 1$ (structural prompt)	0.875	0.475	0.313
allocation at fixed structure, $\Delta_A(s; m, 1)$			
$S = 0$ estimate	—	−0.338	−0.425
95% CI	—	[−0.575, −0.075]	[−0.850, −0.100]
$S = 1$ estimate	—	−0.400	−0.563
95% CI	—	[−0.725, −0.100]	[−1.038, −0.150]
structure at fixed allocation, $\Delta_S(m)$			
$\Delta_S(m)$ estimate	−0.275	−0.338	−0.413
95% CI	[−0.750, +0.138]	[−0.900, +0.063]	[−1.075, +0.000]

4 Domain 1: The Allocation Effect in Sequential Control

4.1 The allocation effect

Drawing more candidates per decision and executing the plurality choice lowers reward. The effect is −0.338 and −0.425 soups per episode without the structural prompt, −0.400 and −0.563 with it, and all four intervals exclude zero (Table 1, Figure 2). Reward is non-increasing in m in the pooled means at both structural levels and in seven of the eight model-by-level curves; the exception, visible in Figure 2a, is `qwen3-32b` at $S = 0$, which falls from 1.55 soups at $m = 1$ to 1.40 at $m = 3$ and returns to 1.55 at $m = 5$. The manipulation checks of §3.1 were computed before these contrasts, so the extra draws did change the executed action; that rules out a manipulation that never fired, not why reward fell.

No stratum reverses the direction. Of the sixteen stratum-level allocation contrasts (four models × two arms × the ladder steps $m = 3$ and $m = 5$), 15 of 16 are negative and the sixteenth is exactly zero; all four layouts are negative in both arms, and all four leave-one-model-out holdouts are

negative, from -0.617 to -0.267 (Appendix J). These are not sixteen independent confirmations: the two ladder steps share an $m = 1$ baseline and the holdout rows restate the per-model rows almost arithmetically, so they support only the weak claim that no single model, layout or holdout carries the sign. One stratum is truncated rather than freely estimated: the `deepseek-chat` entry delivered zero soups in all 20 episodes of both $S = 1$, $m > 1$ cells (cell means at $m = 1, 3, 5$ are $0.60/0.20/0.20$ at $S = 0$ and $0.40/0.00/0.00$ at $S = 1$).

The pooled sign is negative; the practical threshold is not met. Three things should be kept apart. No tested stratum reverses the sign. All four preregistered percentile-bootstrap intervals lie below zero, though their coverage is uncalibrated at four model clusters (§3.2). And every one of them also overlaps the ± 0.20 band (Figure 2b), so under the decision rule fixed before collection, which calls an estimate reliably worse only when its interval lies entirely below -0.20, the practical-harm threshold is not met. So sampling more and voting was worse in direction, and better in no stratum, by an amount these estimates cannot bound away from practically negligible.

A_m is repeated sampling *and* plurality aggregation, and we varied m without varying the aggregator, so this design cannot separate the two readings (§9). The claim is about this sampling-and-voting policy, in this task, on this panel.

4.2 The structural effect at fixed allocation

The comparison this study was built to make cleanly holds m fixed and varies the prompt, and it is the weaker of the two results. The lower block of Table 1 gives the contrast: Δ_S is -0.275, -0.338 and -0.413 soups at $m = 1$, 3 and 5, growing more negative as allocation rises, with crossed intervals that include zero at every level. Leave-one-model-out keeps the sign negative under every holdout.

We do not call this a null. The three point estimates are all negative, all lie outside the preregistered ± 0.20 band, and at $m = 5$ the interval's upper endpoint is exactly $+0.000$. Under the decision rule of §3.2 this is *unresolved*, which is a different statement from equivalence: we find no resolved benefit from the structural prompt and cannot exclude a moderate harm. Reporting it as "no effect" would claim a precision this panel does not have.

The between-model spread is what widens the interval. Per-model $\Delta_S(5)$ runs from -0.100 to -1.000 soups, every per-model value at $m = 5$ is negative, and the most favourable stratum anywhere is $+0.200$ for one model at $m = 3$ ($+0.050$ at $m = 1$). This heterogeneity is not noise to be averaged away, and it is why a bootstrap treating models as a random factor returns intervals this wide (Appendix J).

No detectable interaction, on intervals too wide to exclude a small one. $I(3, 1)$ is -0.063 with interval $[-0.425, +0.275]$ and $I(5, 1)$ is -0.138 with interval $[-0.825, +0.450]$. Both contain zero and both are wide, so we do not detect an interaction and these intervals are too coarse to exclude a modest one. We report it because Proposition 1 makes it one quantity reachable along either path, and the two computations agree, which checks the estimator rather than the world.

5 Domain 2: The Separation on a Task Without an Aggregator

A distinct test of the design. Domain 2 asks whether the S/A separation is implementable in program synthesis. It does not replicate the degradation in sequential control: the structural prompt and outcome differ, and oracle pass@m has no deployed selector whose errors could produce a voting-induced degradation.

Corrected scoring. Table 2 reports the corrected original-test scores. The frozen scorer omitted the task prompt and its helper definitions; adding them rescues 151 candidates under the same tests, without generating new programs. This correction was identified after the preregistered analysis. The full paired rescoring and cell means are in Appendix Q; the original scores are retained as an explicitly historical record in Appendix K.

Using the originally fixed ± 0.02 reference margin, the corrected intervals at $m = 3$ and $m = 5$ are contained in the band, whereas $m = 1$ remains unresolved. At $m = 5$ both structural levels have oracle success 0.9573, giving a zero point contrast. The frozen scorer also placed the $m = 5$ interval inside the band, but left $m = 3$ unresolved. These interval-based judgements retain the small-model-panel caveat of §3.2; they do not establish absence of a benefit on harder tasks. The separate HumanEval+ evaluation also retains the $m = 5$ margin judgement (Appendix Q).

Table 2: Corrected original-test structural contrasts. Intervals use crossed model/task resampling.

m	Δ_S	95% interval
1	-0.0038	[-0.0229, 0.0152]
3	0.0023	[-0.0076, 0.0130]
5	0.0000	[-0.0069, 0.0084]

Oracle coverage. Success over nested prefixes is non-decreasing in m by construction. The metric guarantees the sign of that change, not its size, and does not tell us whether a deployed selector would choose a passing candidate. After the original-test correction, 955 of 13,120 candidates fail, for a per-candidate pass rate of 92.7%. The limited headroom makes this domain a check of the evaluation design, not strong evidence against structural improvements elsewhere.

6 Preregistered Mechanism Diagnostics

Two explanations for a plurality-induced degradation were written down before the outcomes were computed, each with the observable it would move, and tested on the frozen data. The observed rates move against both predictions (Appendix O).

H1: the modal action is the passive one. Prediction: the executed share of the wait action rises with m. It falls, from 0.138 at $m = 1$ to 0.107 and 0.109. Restricting to decisions where the plurality action overrode the first draw, waiting is swapped *out* on net, by -0.118 at $m = 3$ and -0.117 at $m = 5$.

H2: both agents mode-seek onto the same action. Prediction: the share of ticks on which the two agents execute an identical action rises with m. It falls too, from 0.137 at $m = 1$ to 0.115 and 0.120, and still falls when ticks where both agents waited are excluded (0.111, 0.093, 0.094). Menu size is flat in m (4.03 actions on average).

The observed changes run opposite to both predictions. That phrasing is deliberately weaker than a test: this section reports pooled executed shares, not an inferential comparison, so what it establishes is the direction of the observed change. H2's observable is literal action-string identity, and two agents can duplicate effort without emitting the same string, so the falling share speaks to the version we preregistered and not to every version of the idea. The mechanism remains unresolved; the post-hoc scripted controls in Appendix Q do not identify it. The obvious untested candidate is the one Proposition 3 names, that the modal action is systematically the worse action here; testing it would require controlled action-value evaluation beyond these logs. A hypothesis chosen after seeing the effect would be exploratory on these data, not a replacement for the preregistered predictions.

7 Realized Cost, and Why It Is Not a Control

Held at the same m, the structural prompt consumes about 25% more tokens per episode than its absence (24.5% to 25.1% across the three allocation levels; Appendix F gives the cells). Allocation cost, by contrast, is almost exactly mechanical: total tokens scale by $3.00\times$ from $m = 1$ to $m = 3$ in both arms, and by $5.02\times$ and $5.00\times$ to $m = 5$ at $S = 0$ and $S = 1$. So two arms matched on m are not matched on realized cost, and the converse fails as well: equalising tokens between the arms at $m = 1$ would require the non-structural arm at a fractional allocation that is not on the design grid. A study that reports one of these comparisons and calls it budget-matched has silently chosen an estimand, and by Proposition 2 the two answer different questions. We do not claim the token gap explains the reward differences and we make no such test.

Candidate disagreement is also lower under the structural condition along its observed trajectories (0.311 to 0.120 at $m = 5$ for one model, 0.655 to 0.508 for another). That compares two policies'

trajectories, so it does not separate a change in the conditional decision rule from a change in which states are visited, and we do not read it as the prompt sharpening the distribution at a fixed context. These are descriptive differences between policy trajectories; per-model disagreement at $m = 5$ spans 0.12 to 0.66 (Appendix J).

8 Related Work

Sampling, voting, and non-monotone returns. Self-consistency samples reasoning paths and returns a majority answer (Wang et al., 2023); repeated sampling under a fixed prompt is a scaling axis (Brown et al., 2024; Snell et al., 2024), including per-step agent decisions (Wang et al., 2024; Lee et al., 2026). Chen et al. (2024) show majority-vote accuracy rising and then falling with calls, explaining the aggregate through easy and hard queries. Agent evaluations also contain negative budget contrasts (Lee et al., 2026; Li et al., 2026b), and inverse scaling can occur within a single reasoning trace (Gema et al., 2025). Our contribution is the combination of an auditable request-invariance criterion, a preregistered prompt-by-allocation factorial, and contrasts adjudicated against fixed practical margins. Neither repeated sampling nor the possibility of a negative return is new. The source-level comparison is in Appendices M and N.

Budget matching and post-treatment conditioning. Tran & Kiela (2026) compare architectures under common thinking-token budgets; Li et al. (2026a) examine coordination overhead in multi-agent systems. These motivate distinguishing allocated compute from consumed compute. Our cost argument applies the established warning against adjusting for variables affected by treatment (Rosenbaum, 1984; Montgomery et al., 2018). Reporting resource use alongside quality also has an established history (Dodge et al., 2019; Schwartz et al., 2020; Liang et al., 2023). Here we report fixed-allocation effects and their interaction while treating realized cost as an outcome, not a substitute control.

9 Limitations, Scope, and What Survives

One aggregator, which is the most important limitation here. A_m is repeated sampling *plus* plurality with a fixed tie-break. We varied m without independently varying the aggregator in the frozen factorial, so this study cannot separate "more samples hurt" from "plurality hurts"; Proposition 3 makes the second reading concrete. This is a substantive limitation: a concurrent comparison with a different fixed aggregator is needed to distinguish candidate count from aggregation. A post-hoc trajectory audit also found incomplete reconstruction from logged goals; original outcomes are retained, but exact state-based mechanism analyses are not supported for every episode (Appendix Q).

Panel size, and what the intervals can support. four models from three providers, two of them same-family siblings; two domains; $m \in \{1, 3, 5\}$; one structural prompt per domain; four layouts. We characterise interval width and not coverage (§3.2), and nothing licenses extrapolation beyond this panel, horizon, menu and scaffold.

Prompt package and executor. The cooking prompt changes both contribution semantics and the requested response format; their effects are not separately identified. The low-level controller is also part of the tested system. Post-hoc scripted controls demonstrate deliveries using that same executor (Appendix Q), but do not replace a comparison with trained cooperative-agent methods or isolate an LLM mechanism.

Magnitude, ceiling, mechanism, provenance. The practical size of the allocation effect is not established, and the mechanism is open (§6). Domain 2's corrected per-candidate pass rate of 92.7% leaves little headroom for a positive structural effect, so that domain is not evidence against one. One panel entry is identified only by a provider alias (§3.2); every cell used the same requested alias and configuration hashes, so we observe no change in request configuration, which is weaker than an absence of backend change. The supported conclusion is to report allocation as an experimental factor with uncertainty and realized cost, with claims scoped to the tested system.

REFERENCES

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

Micah Carroll, Rohin Shah, Mark K. Ho, Tom Griffiths, Sanjit Seshia, Pieter Abbeel, and Anca Dragan. On the utility of learning about humans for human-AI coordination. In *NeurIPS*, 2019.

Lingjiao Chen, Jared Davis, Boris Hanin, Peter Bailis, Ion Stoica, Matei Zaharia, and James Zou. Are more LLM calls all you need? towards the scaling properties of compound AI systems. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. arXiv:2403.02419.

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

Jesse Dodge, Suchin Gururangan, Dallas Card, Roy Schwartz, and Noah A. Smith. Show your work: Improved reporting of experimental results. In *Proceedings of EMNLP-IJCNLP*, pp. 2185–2194, 2019. doi: 10.18653/v1/D19-1224.

Aryo Pradipta Gema, Alexander Hägele, Runjin Chen, Andy Arditi, Jacob Goldman-Wetzler, Kit Fraser-Taliente, Henry Sleight, Linda Petrini, Julian Michael, Beatrice Alex, Pasquale Minervini, Yanda Chen, Joe Benton, and Ethan Perez. Inverse scaling in test-time compute. *Transactions on Machine Learning Research*, 2025. arXiv:2507.14417.

Nicholas Lee, Lutfi Eren Erdogan, Chris Joseph John, Surya Krishnapillai, Michael W. Mahoney, Kurt Keutzer, and Amir Gholami. Agentic test-time scaling for WebAgents. *arXiv preprint arXiv:2602.12276*, 2026.

Jialing Li, Zhouhong Gu, Yin Cai, and Hongwei Feng. Scaling behavior of single LLM-driven multi-agent systems. *arXiv preprint arXiv:2606.00655*, 2026a.

Xiaochuan Li, Ryan Ming, Pranav Setlur, Abhijay Paladugu, Andy Tang, Hao Kang, Shuai Shao, Rong Jin, and Chenyan Xiong. Benchmark test-time scaling of general LLM agents. *arXiv preprint arXiv:2602.18998*, 2026b.

Percy Liang, Rishi Bommasani, Tony Lee, et al. Holistic evaluation of language models. *Transactions on Machine Learning Research*, 2023. arXiv:2211.09110.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by ChatGPT really correct? rigorous evaluation of large language models for code generation. *arXiv preprint arXiv:2305.01210*, 2023.

Jacob M. Montgomery, Brendan Nyhan, and Michelle Torres. How conditioning on posttreatment variables can ruin your experiment and what to do about it. *American Journal of Political Science*, 62(3):760–775, 2018. doi: 10.1111/ajps.12357.

Paul R. Rosenbaum. The consequences of adjustment for a concomitant variable that has been affected by the treatment. *Journal of the Royal Statistical Society, Series A*, 147(5):656–666, 1984. doi: 10.2307/2981697.

Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. Green AI. *Communications of the ACM*, 63(12):54–63, 2020. doi: 10.1145/3381831. arXiv:1907.10597.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Dat Tran and Douwe Kiela. Single-agent LLMs outperform multi-agent systems on multi-hop reasoning under equal thinking token budgets. *arXiv preprint arXiv:2604.02460*, 2026.

Han Wang, Archiki Prasad, Elias Stengel-Eskin, and Mohit Bansal. Soft self-consistency improves language model agents. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024. arXiv:2402.13212.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed H. Chi, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations (ICLR)*, 2023. arXiv:2203.11171.

CONTENTS OF THE APPENDIX

A AI use statement 11
B Reproducibility and the claim audit 12
C Design details, in full 12
D Proofs 13
E Supporting results, and what their machine-checking covers 14
F Realized token cost, by cell 14
G Collection integrity and the floor stratum 15
H Provenance of the model panel 15
I Inference procedure, in full 15
J Full stratum tables 16
K Domain 2 supplementary numbers 16
L An analysis defect, found and fixed before interpretation 17
M Prior-art audit 17
N Related work, extended 17
O Mechanism diagnostics, in full 18
P A superseded design, and why it was not a clean allocation intervention 19
Q Post-hoc Robustness Checks 19

A AI USE STATEMENT

Large language models were used in preparing this work in two distinct roles, which we separate because they carry different risks. First, as the *object of study*: every model output analysed here was produced by provider API calls to the panel models, and every outcome was scored by the environment or by the benchmark's own tests. Those records are the phenomenon under investigation, not synthetic data standing in for observations we could not otherwise make, which is why the first required category below is answered no. Second, as an *assistant*, itemised here against the venue's categories rather than summarised.

Required disclosures. We used generative AI tools to help develop the theoretical framing; to formulate the mathematical claims, to supply critical ingredients for their proofs, and to write those proofs, including the Lean formalisation whose coverage is stated in Appendix E; to refine the preregistered hypotheses and the margins they are adjudicated against; to design and to give feedback on the experimental methodology; to implement the methods, meaning the collection, scoring and analysis code; to reformat and re-score the candidate records when the original-test scorer was corrected (§5, Appendix Q); and to interpret results, in the sense of drafting readings that the authors then adjudicated against the preregistered rules. We did not use them to generate synthetic data sets, and we did not use them for translation. Support for qualitative and thematic data analysis does not apply here: no analysis in this paper is qualitative.

Recommended disclosures. We also used them to create and edit software code; to create and modify the figures, including drawing, from a written specification, the text-free schematic panel of Figure 1, over which every label and number is typeset by the figure script from the audited values; to draft and to edit parts of this paper; to summarise existing literature and to construct the prior-art audit of Appendix M; to identify references and check them against their sources; and to review the manuscript for statements stronger than their evidence.

What the authors are responsible for. We have reviewed all AI-assisted work. The authors take responsibility for the experimental design, the preregistration, the adjudication rules and every scientific claim. Two things bound what that assistance could have hidden, and we state them because a disclosure without them is only a confession. Every quantitative claim in the paper is generated from a frozen artifact by a script, and the analysis and the claim audit were re-run end to end against the frozen records and the separately versioned post-hoc artifacts for the submitted version. Of the proofs, the results named in Appendix E are machine-checked in Lean 4 against Mathlib with no result depending on `sorryAx`, so that much of what an assistant helped write is certified by a checker rather than by the party that wrote it; Propositions 2 and 3 are not among them, and the checking covers the algebra alone, not the estimator, the collection code or the scoring harness. The resampler defect and the later scoring correction are documented in Appendices L and Q.

B Reproducibility and the claim audit

The anonymous supplementary material contains the preregistration and charter as submitted before collection; the frozen record files for both domains with the SHA-256 digests quoted in them; the collection, scoring and analysis programs; the Lean sources and their check script; and the audit program described here. Its entry point is `verify.sh`, which prints the artifact digests and then re-runs the audit and the separate post-hoc checks. After installing the listed dependencies, verification requires no network access or provider calls. The analysis is one command per stage and is deterministic given the fixed bootstrap seed, so every interval in this paper reproduces bit for bit.

Frozen-study numbers in the body are defined in a macro file, and each macro is transcribed from a named line of a frozen artifact. An audit program recomputes each of them from the frozen records, from the stage outputs, or from the preregistration text; compares at the precision the paper prints rather than against a tolerance, so that a value rounded the wrong way fails instead of passing; verifies that the generated stratum tables below are not stale; and fails if any number in the body was typed by hand rather than drawn from the macro file. At submission it reports every macro checked, none unchecked, and no mismatches. Corrected-score macros and the post-hoc tables are generated and checked separately from the saved paired evaluations by `iclr3_verify_completion.py`, also invoked by `verify.sh`. What these checks cannot establish is which qualifying sentence a correct number sits in; the scope corrections we made late were found by reading, not by running it.

Two consequences of that discipline are worth recording, because both are cases where the audit caught us. The panel-wide structural token overhead is 25%; an earlier draft reported a figure computed from a single model's row. And every model in the panel has a negative $\Delta_S(5)$, where an earlier summary said three of four; the corresponding statement for Δ_A is not true, and Table 3 shows the one non-negative cell.

C Design details, in full

This appendix carries the parts of §3 that a reader checking the design will want and a reader following the argument does not need in line.

The four layouts. The four Domain-1 layouts are a within-family sweep of station contention, chosen by an a-priori geometric quantile rule over the family and fixed before collection. They differ in how many station types are duplicated, which is what determines whether the two agents can work in parallel or must queue for the same object.

- *open, 4 duplicated*: two onion dispensers, two cookers, two plate dispensers, two delivery areas. Almost nothing must be shared.

- *mid, 2 duplicated*: two onion dispensers and two cookers; one plate dispenser and one delivery area.
- *tight, 1 duplicated*: two onion dispensers; everything else single.
- *tight, 0 duplicated, close*: one of each station, placed close together, so every step of the recipe is a potential collision.

The separability audit, itemised. §3.1 states what the recorded hashes cover; the counts and the pre-run tests are as follows. Seven canonical hashes are unchanged across the entire multi-day run: benchmark source, decoder configuration, design constants, parser contract, the probe states, and the two system prompts. Output-cap truncation affected 82 (0.071%) of Domain-1 calls and 21 (0.160%) of Domain-2 calls, under a cap identical at every S and m within a domain. All observed Domain-1 cap hits occur at $S = 0$; equal caps do not imply equal truncation rates. This asymmetry is compatible with the prompt-format difference, but does not identify its cause. All 13,120 Domain-2 draw seeds are distinct with no seed reused. The pre-run invariance test builds the whole message list from 32 harvested benchmark states at every m and both structural levels and checks byte identity, checks that the action menu is identical across m and across S, and checks by signature that m is not a parameter of the message builders, the menu builder or the decoder configuration, so a later edit cannot introduce a dependence without failing it. What none of this certifies is the provider: identical visible configuration is consistent with a changed backend, and Appendix H records where that matters.

Pool construction. Domain-2 pools are exactly five candidates, in generation order, with timestamps monotone in candidate index.

Parse validity by stratum. Pooled parse validity is 0.9989 against the preregistered floor of 0.90; the per-model, per-m range is 0.997 to 1.000. Wang et al. (2024) document the failure mode that makes exact-match voting unusable when the action space is large, reporting that self-consistency "fails to produce a single majority action 86% of the time" for one of their tasks. The tie rates in §3.1 show that is not the regime here, which rules out one explanation of a negative result before it is offered.

Scale of the two matrices. Domain 1 is 480 episodes, 115,200 calls and 44,597,180 tokens; Domain 2 is 2,624 pools, 13,120 calls and 4,953,242 tokens.

D PROOFS

Proof of Proposition 1. Write $\mu_{s,m} = \mathbb{E}[Y(s,m)]$. Then $\Delta_S(m_1) - \Delta_S(m_0) = (\mu_{1,m_1} - \mu_{0,m_1}) - (\mu_{1,m_0} - \mu_{0,m_0})$, and $\Delta_A(1; m_1, m_0) - \Delta_A(0; m_1, m_0) = (\mu_{1,m_1} - \mu_{1,m_0}) - (\mu_{0,m_1} - \mu_{0,m_0})$. Both expand to $\mu_{1,m_1} - \mu_{0,m_1} - \mu_{1,m_0} + \mu_{0,m_0}$. □

Proof of Proposition 2. One instance suffices. Fix m, let $U \sim \mathrm{Bernoulli}(1/2)$ be run-level randomness, and set $Y(s,m) = U$ and $C(s,m) = 1 + s + U$. Then $\Delta_S(m) = \mathbb{E}[U] - \mathbb{E}[U] = 0$: the structural intervention has no effect on the outcome at this allocation. But $C = 2$ arises at $s = 0$ only when $U = 1$, and at $s = 1$ only when $U = 0$, so $D_{\mathrm{cond}}(m, 2) = 0 - 1 = -1$. Conditioning on realized cost returns a contrast of -1 where the fixed-allocation effect is 0, and the two estimands therefore differ. The construction is deliberately minimal: no confounding, no measurement error, and no dependence on the size of the cost gap. It is enough that cost responds to both the treatment and the run-level randomness that also drives the outcome. □

Proof of Proposition 3. Let N_a count the draws whose parse is a, and let $p_a = \pi_s(a)$ with $p_{a^\star} > p_b$ for all $b \neq a^\star$. By the strong law $N_a/m \to p_a$ almost surely for each of the finitely many menu entries, so almost surely $N_{a^\star} > N_b$ for all $b \neq a^\star$ for all large m, and the tie-break is never reached. Hence $\Pr[g_m = a^\star] \to 1$, and for bounded r dominated convergence gives $\mathbb{E}[r(g_m)] \to r(a^\star)$. Under the hypothesis $q = 1$ every draw parses, so g_1 executes the single draw and $\mathbb{E}[r(g_1)] = \mathbb{E}_{\pi_s}[r]$; the hypothesis $r(a^\star) < \mathbb{E}_{\pi_s}[r]$ then places the limit strictly below the value at $m = 1$. The two-action instance in §2 satisfies it. Without $q = 1$ the $m = 1$ value is instead $q\,\mathbb{E}_{\pi_s}[r] + (1-q)\,r(a_{\mathrm{noop}})$,

a term this proposition does not model; the abstention case is treated by the conditional bound of Appendix E. □

As Remark 1 states, Proposition 3 is the single-context restatement of the hard-query branch of Chen et al. (2024) and is included because that is the version a sequential agent occupies, not as a new result.

E SUPPORTING RESULTS, AND WHAT THEIR MACHINE-CHECKING COVERS

The following justify design choices rather than carry claims. They are formalised in Lean 4 with Mathlib, together with Proposition 1 and the attribution identities around it; the check script type-checks every file and verifies that no result depends on `sorryAx`. The audited count is 33 results across three modules (attribution, plurality, aggregator). Propositions 2 and 3 are *not* among them: the first states which estimand a conditioning choice targets, the second is a limit statement about a mode-seeking rule. Both are proved in Appendix D, the first by counterexample, and the Lean count should not be read as covering them.

Aggregator properties. The plurality aggregator with menu-order tie-breaking is deterministic, invariant to permutations of the candidate list, unchanged by candidates that parse off-menu, and cannot be swung by abstentions: in the case the preregistration names explicitly, three abstentions and two valid votes for a menu action execute that action. A constant candidate list executes that constant at every multiplicity, which is the duplicate-sample negative control, verified simultaneously at $m \in \{1, 3, 5\}$.

Plurality concentration. A Hoeffding-plus-union-bound argument gives an *upper bound* on the plurality rule's conditional failure probability that decreases in the number of valid votes, with a marginal form obtained by integrating against the binomial distribution of the valid-draw count V. The bound decreases; the failure probability itself need not. With two legal actions at probabilities 0.6 and 0.4 and a tie-break favouring the second, one valid vote errs with probability 0.4 while two valid votes err with probability 0.64, because a tie counts as an error. Since abstention can leave V even, that case is reachable here, and only the bound is monotone. This motivates the manipulation checks in §3.1; it supports no reported effect. At $q = 0.9989$ invalid parses are rare in aggregate. This is not a uniform bound over reachable contexts and does not bound an episode-level reward effect; validity-conditioned distributions and state visitation may still depend on the policy.

Coverage. The property §5 relies on is set inclusion: a longer index prefix cannot contain fewer passing programs, so pass@m is non-decreasing in m by construction. The familiar closed form $\text{pass@}m = 1 - (1 - p)^m$ is also formalised. It holds *per model-problem unit* under the conditional i.i.d. assumption, and the pooled single-sample rate cannot be substituted into it: the correct aggregate is $\mathbb{E}_u[1 - (1 - p_u)^m]$, not $1 - (1 - \mathbb{E}_u[p_u])^m$. Substituting the pooled per-candidate rate under the historical scorer, 91.6%, would predict pass@5 near one, against an observed 0.95. Heterogeneity across model-problem units is by itself sufficient to produce that gap; the aggregate discrepancy does not establish dependence between draws within a unit, and we do not claim it does.

What formalisation does not buy. It verifies the algebra we print, which was never the risky part of this study. The defects that actually occurred were in the resampler (Appendix L) and in the original-test scoring harness (Appendix Q), and no proof about the aggregator would have caught either. Formalising the mathematics does not certify the estimator, the collection code, or the scoring harness.

F REALIZED TOKEN COST, BY CELL

Tokens per episode, pooled over the four models, at each cell of the Domain-1 factorial: 27,504 against 34,366 at $m = 1$, 82,530 against 103,212 at $m = 3$, and 138,002 against 171,851 at $m = 5$, for $S = 0$ against $S = 1$ respectively. These are the numbers behind the 25% figure in §7. Whole-run totals are 44,597,180 tokens in Domain 1 and 4,953,242 in Domain 2.

Draw disagreement, and what the comparison confounds. The rate at which the m draws disagree at $m = 5$ is lower under the structural condition for the two models §7 names. The comparison is between trajectories generated by two different policies, so both the conditional decision rule and the distribution of visited states may differ between the arms, and the observed difference does not separate them. Establishing that the structural prompt sharpens the action distribution at a fixed context would need a paired comparison on common states, which this study does not report.

G COLLECTION INTEGRITY AND THE FLOOR STRATUM

Domain 1: 480 episodes, which is four models $\times$ four layouts $\times$ two structural levels $\times$ three allocation levels, each run at five seeds; 115,200 provider calls, 44,597,180 tokens, zero transport errors, 82 (0.071%) output-cap truncations. Domain 2: 2,624 pools, 13,120 candidates, 4,953,242 tokens, zero transport errors, 21 (0.160%) truncations. No duplicate, missing or extra cells in either matrix. The recorded prompt hash—the system prompt in Domain 1, the full message list in Domain 2—takes exactly 8 distinct values in Domain 1 (four layouts $\times$ 2 structural levels) and 328 in Domain 2 (164 problems $\times$ 2). The seven canonical hashes are benchmark source, decoder configuration, design constants, parser contract, probe states, and the two system prompts.

The floor stratum discussed in §4: the `deepseek-chat` entry has Domain-1 cell means $0.60/0.20/0.20$ soups at $S = 0$ and $0.40/0.00/0.00$ at $S = 1$ for $m = 1, 3, 5$, and delivered zero soups in all 20 episodes of each $S = 1$, $m > 1$ cell.

H PROVENANCE OF THE MODEL PANEL

Three of the four panel entries echo their own identity in the provider response. The fourth was requested as the alias `deepseek-chat`, and the provider echoed `deepseek-v4-flash` in the pre-run smoke test, in all 3,280 recorded Domain-2 responses for that alias, and in a post-run probe. That figure is this entry's own share of the 13,120 Domain-2 candidates, not the four-model total. The Domain-1 collector did not record the provider-returned model identifier, so for Domain 1 we cannot verify the backend revision per request; that collection ran between two agreeing observations. Every design cell used the same requested alias and the same visible configuration hashes, so we observe no change in request configuration across cells. That is weaker than an absence of backend change: a provider may serve different revisions behind one alias without any of it appearing in what we record, and for Domain 1 we have no per-request evidence either way. We name the entry **DeepSeek API** (`deepseek-chat` alias) throughout and make no claim about which release served it. Collectors written after this study log the returned identifier on every call.

I INFERENCE PROCEDURE, IN FULL

§3.2 states the procedure and its two consequences. The qualifications behind them are these. With four model clusters the resampling support is coarse, no small-cluster correction is applied, and two of the four models are same-family siblings, which if anything understates within-cluster dependence. Interval *width* is what §3.2 discusses; interval *coverage* is a different property, and we do not calibrate it here. Percentile cluster bootstraps at small cluster counts are commonly reported to undercover, but we did not measure the coverage of this procedure at this cluster count and did not apply a small-cluster correction, so the nominal 95% level is not guaranteed for the intervals we report. A reader should therefore treat every "interval excludes zero" statement in this paper as the output of an uncorrected percentile cluster bootstrap rather than as a calibrated 95% statement, and weigh it against the evidence that does not depend on the procedure at all: the allocation estimates are non-positive across the reported model, layout, arm and holdout summaries, with one stratum exactly zero. A wild cluster bootstrap-t or an exact randomisation test over the two factors may provide additional sensitivity evidence, and neither is in this paper. Agreement from a second procedure would not by itself establish the coverage of the intervals reported here.

J FULL STRATUM TABLES

Table 3 gives the allocation contrast in every model, layout and leave-one-model-out stratum, and Table 4 does the same for the structural contrast. Both are emitted by a generator from the frozen episode records rather than transcribed, and the audit of Appendix B fails if the checked-in copy drifts from the data.

Table 3: Domain 1, $\Delta_A(s; m, 1)$ by stratum, soups per episode. Generated directly from the frozen episode records.

	$S = 0$		$S = 1$	
	$m = 3$	$m = 5$	$m = 3$	$m = 5$
by model				
`deepseek-chat` alias	-0.400	-0.400	-0.400	-0.400
glm-5	-0.400	-0.900	-0.350	-0.450
qwen3-235b-a22b (MoE)	-0.400	-0.400	-0.250	-0.550
qwen3-32b (dense)	-0.150	$+0.000$	-0.600	-0.850
by layout				
mid, 2 duplicated stations	-0.400	-0.400	-0.650	-1.050
open, 4 duplicated	-0.350	-0.550	-0.500	-0.600
tight, 0 duplicated, close	-0.100	-0.200	-0.050	-0.050
tight, 1 duplicated	-0.500	-0.550	-0.400	-0.550
leave-one-model-out				
without `deepseek-chat` alias	-0.317	-0.433	-0.400	-0.617
without glm-5	-0.317	-0.267	-0.417	-0.600
without qwen3-235b-a22b (MoE)	-0.317	-0.433	-0.450	-0.567
without qwen3-32b (dense)	-0.400	-0.567	-0.333	-0.467

Table 4: Domain 1, $\Delta_S(m)$ by stratum. The between-model spread is what widens the pooled intervals in Table 1. Generated directly from the frozen episode records.

	$m = 1$	$m = 3$	$m = 5$
by model			
`deepseek-chat` alias	-0.200	-0.200	-0.200
glm-5	-0.800	-0.750	-0.350
qwen3-235b-a22b (MoE)	$+0.050$	$+0.200$	-0.100
qwen3-32b (dense)	-0.150	-0.600	-1.000
by layout			
mid, 2 duplicated stations	-0.350	-0.600	-1.000
open, 4 duplicated	-0.150	-0.300	-0.200
tight, 0 duplicated, close	-0.350	-0.300	-0.200
tight, 1 duplicated	-0.250	-0.150	-0.250
leave-one-model-out			
without `deepseek-chat` alias	-0.300	-0.383	-0.483
without glm-5	-0.100	-0.200	-0.433
without qwen3-235b-a22b (MoE)	-0.383	-0.517	-0.517
without qwen3-32b (dense)	-0.317	-0.250	-0.217

K DOMAIN 2 SUPPLEMENTARY NUMBERS

Table 5 gives the Domain-2 cell means, the structural contrast and its intervals, and the interaction under the historical scorer; these are retained for auditability and superseded by the correction for interpreting original-test success. Pool diversity, the number of distinct programs among the five candidates, spans 2.58 to 3.54 by model and arm, with the structural arm slightly more diverse in every model.

Table 6 reports the Domain-2 allocation contrast that §5 declines to treat as a finding.

Table 5: Historical frozen-scorer Domain 2. pass@m per pool, an oracle sampling-success measure over frozen index prefixes; 2,624 pools. Intervals are 95% crossed model × problem cluster bootstrap. Corrected results are reported in Table 8.

	$m = 1$	$m = 3$	$m = 5$
$S = 0$	0.913	0.938	0.947
$S = 1$	0.911	0.945	0.951
$\Delta_S(m)$	-0.0023	$+0.0076$	$+0.0038$
95% CI	$[-0.0252, +0.0191]$	$[-0.0046, +0.0213]$	$[-0.0053, +0.0137]$
$I(m, 1)$, CI	—	$+0.0099$ $[-0.0061, +0.0282]$	$+0.0061$ $[-0.0137, +0.0274]$

Table 6: Historical frozen-scorer Domain 2, the allocation contrast. The nested metric guarantees the sign; the size of the gain is empirical. These are oracle coverage gains, not evidence of improved selected-output performance.

	$m = 3$	$m = 5$
$\Delta_A(0; m, 1)$	+0.0244	+0.0343
95% CI	$[+0.0107, +0.0419]$	$[+0.0168, +0.0556]$
$\Delta_A(1; m, 1)$	+0.0343	+0.0404
95% CI	$[+0.0183, +0.0534]$	$[+0.0221, +0.0625]$

Under the historical scorer, per-model $\Delta_S(5)$ ranges from -0.003 to $+0.012$; leave-one-model-out $\Delta_S(m)$ stays within ± 0.012 throughout. Failure classes among the 1,106 failing candidates: AssertionError 842, NameError 158, SyntaxError 53, IndexError 25, ValueError 19, TypeError 4, Timeout 4, other 1. No candidate extracted to empty code. Pool diversity by model and arm runs 2.58 to 3.54 distinct programs per five-candidate pool, with syntax validity at least 98.3% everywhere.

L An analysis defect, found and fixed before interpretation

The first execution of the crossed bootstrap returned zero-width intervals on Domain 1. The cause was our own resampler: a hand-rolled linear congruential generator modulo 2^{31}, whose low-order bits have period 2^{k+1}, so reducing its output modulo the four-model panel cycled through the models in a fixed order and every "bootstrap" draw rebuilt an identical multiset. Domain 2 masked the defect because 164 problems is not a power of two. We replaced the generator with a seeded Mersenne Twister, which is equally reproducible and does not have that structure, and added an assertion that raises when 10,000 draws collapse to a single distinct value. No interval in this paper comes from the broken resampler, and the zero-width intervals were never interpreted.

M Prior-art audit

Novelty wording was set by a predefined audit run after the data were frozen and before this manuscript was written, under two rules: a component counts as precedented only on a verbatim quotation from the source, and anything not read is recorded as "not stated" rather than as absent. The audit found that per-step voting over sampled actions, the repeated-sampling protocol, the fact that more calls can lower accuracy, the non-monotonicity result, and the practice of reporting compute alongside accuracy are each precedented; the citations in §8 are its output. One source relevant to the monotonicity question could not be read and is recorded as unread rather than as negative. We report the audit's outcome as a success: it cost wording, not results.

N Related work, extended

This appendix carries the parts of §8 that the page limit displaced. Nothing here is a claim the body does not make.

On the mechanism given by Chen et al. (2024). Their account of non-monotonicity is a mixture argument: a benchmark containing both easy and hard queries, where more calls raise accuracy on the easy ones and lower it on the hard ones, so the aggregate can rise and then fall. Their evaluation is multiple-choice question answering, and their unit of mixture is the query. In a sequential task the mixture is over the roughly 40 decisions inside one episode, so an episode is a mixture by construction. That is why we restate their hard-query branch per context (Proposition 3, Remark 1) rather than citing the aggregate form.

Why the sequential case is only an analogy. Chen et al. (2024) decompose a benchmark into easy and hard queries whose mixture is fixed while the number of calls varies. A sequential episode is not that object. Which states the agent reaches, and with what weight they enter the return, is itself determined by the policy and therefore by m, so raising m changes the mixture rather than voting more often on a fixed one. Proposition 3 is accordingly stated for a single context, and Remark 1 says it is a possibility result rather than an episode-level decomposition. We do not derive a sequential analogue and do not claim one.

On the agent-side reports of the same direction. Lee et al. (2026) name per-step majority voting as a static baseline and report both the drop from 83.3% to 80.6% and gains that are "smaller and non-monotonic" on one of their benchmarks. Li et al. (2026b) report configurations in which "self-choice performance even degrades as K increases". Wang et al. (2024) report that self-consistency "fails to produce a single majority action 86% of the time" for one of their tasks, which is the sparse-vote regime our tie rates rule out. Each of these is a point observation rather than an estimated contrast: none is accompanied by a preregistration, a crossed design, an interval, a practical-equivalence margin, or a structural factor to interact with. That list is the whole of what we claim to add to the direction.

On budget matching. Tran & Kiela (2026) match on an upper cap on thinking tokens and note that realized usage often falls below it, which is the distinction Proposition 2 formalises: an allocated budget is ex-ante and a consumed budget is post-treatment. Li et al. (2026a) attribute multi-agent degradation to coordination overhead rather than to context length. Our structural results, negative but interval-spanning in one domain and equivalent to zero in the other, are consistent with both.

On compute reporting. Budget as an axis of the reported result goes back to Dodge et al. (2019), reporting the price of a result alongside its quality to Schwartz et al. (2020), and efficiency as a first-class evaluation metric to HELM (Liang et al., 2023).

O MECHANISM DIAGNOSTICS, IN FULL

§6 states both preregistered hypotheses, their observables and their verdicts. The reasoning behind each, and the $m = 3$ values, are here.

H1, stated as it was preregistered. A mode-seeking aggregator should favour whichever action the model emits most readily. In a cluttered kitchen the action a model emits most readily is plausibly the no-op, because it is legal in every state and never blocked. An agent that waits more will deliver fewer soups. The prediction is therefore that the executed share of the wait action rises with m. The observed shares are 0.138, 0.107 and 0.109 at $m = 1$, 3 and 5, and the swap-conditional net change is -0.118 at $m = 3$ and -0.117 at $m = 5$, both negative. The prediction is inverted, not merely unsupported.

H2, stated as it was preregistered. Two agents sampling from similar distributions, each taking its own mode, will converge on the same choice, so the pair stops covering complementary parts of the recipe. The observable is the share of ticks on which the two agents execute an identical action: 0.137, 0.115 and 0.120. Because both agents waiting counts as agreement under that definition, we also report the share excluding ticks where both waited: 0.111, 0.093 and 0.094. Both series fall.

Why the wording is about the operationalisation. The observable is literal action-string identity. Two agents can duplicate effort without emitting the same string: both walking toward the pot, both

fetching onions from different dispensers, both idling at different stations. The falling identical-action share refutes the version of H2 we preregistered and does not refute every version of the idea. Claiming that we have ruled out mode collapse would be a stronger statement than the observable supports.

P A SUPERSEDED DESIGN, AND WHY IT WAS NOT A CLEAN ALLOCATION INTERVENTION

The design that this study replaced varied inference by asking the model to reconsider its action in additional prompted passes: a second pass appended an instruction of the form "reconsider your next action", a third appended another. Read as a budget factor this looks reasonable, since more passes cost more calls. It is not an allocation intervention in the sense of Definition 1, and the reason is exactly the distinction this paper is about.

The reconsider instruction changes the prompt string, so it changes $P_s(\cdot \mid x)$. The second pass is not another draw from the first pass's decision rule; it is a draw from a different decision rule, one that has been told its previous answer was worth revisiting. The manipulation therefore moves S and A together, and any contrast built on it estimates their sum. No amount of care in the analysis recovers the separation, because the confounding is in the data-generating process rather than in the estimator. The failure is visible mechanically under the audit of §3.1: the prompt hash varies with the pass index, which is precisely the invariance Definition 1 requires.

We record this because that design survived a full collection and analysis before the defect was identified, and the check that would have caught it on day one is a hash comparison costing nothing. A budget arm that changes what the model is asked is a semantic intervention wearing a budget label.

Q POST-HOC ROBUSTNESS CHECKS

Post-hoc trajectory reconstruction. An additional audit replayed the stored executed goals through the control simulator. Three of the 480 trajectories did not reproduce all recorded menus, and two also differed in final reward. The complete episode-level report and replay script are included in the supplement. The original collector did not store primitive actions or executor RNG states, so those trajectories cannot currently support exact state-based mechanism analyses. Provider retry jitter and the executor both use global Python randomness, a plausible coupling that is not established as the historical cause. We retain all original outcomes and do not replace them with reconstructed rewards. This audit is separate from the preregistered analysis.

Stronger code tests. We rescored every original program with HumanEval+ (Liu et al., 2023), retaining the original extraction and fixed candidate prefixes. This is post-hoc test-suite robustness, not a new preregistered study. The pinned evaluator, dataset digest, sample mapping and per-program results are supplied separately from the frozen analysis. Table 7 reports both the evaluator's base tests and its base-plus-extra tests, since changing the scoring framework itself also changes outcomes. The comparison uses paired replicates and crossed model/task resampling with the same number of bootstrap draws as the main analysis; the small model-panel limitation remains.

Original-test harness correction. The frozen scorer concatenated the extracted program and tests without the task prompt. This omitted helper definitions supplied by the task: some tests, and some generated programs, refer to those helpers. We reran all 13,120 saved candidates under both concatenations, using identical original tests, a fixed test RNG seed, and isolated processes. The seeded legacy rerun reproduced every frozen pass/fail label. Prepending the prompt changed 151 failures to passes and no passes to failures: 89 at $S = 0$ and 62 at $S = 1$, across HumanEval tasks 10, 32, 38 and 50. This count is relative to the original tests, not EvalPlus correctness. The frozen records remain unchanged; Table 8 gives the versioned correction with the same fixed-prefix estimator and crossed resampling rule. Under the original reference margin, the corrected intervals at $m = 3$ and $m = 5$ lie inside the band; $m = 1$ remains unresolved. This post-hoc correction strengthens the saturation observation without repairing the benchmark's limited difficulty or the small model panel.

Table 7: Post-hoc test-suite sensitivity of fixed-prefix oracle success. Base and Plus denote the pinned EvalPlus evaluator; neither substitutes for the original scorer. Means pool the same model/task/replicate cells. Intervals are crossed percentile intervals. At $m = 5$, the Plus interval remains inside the original ± 0.02 reference margin; at $m = 1$ it does not. This does not exclude benchmark contamination or establish generality across tasks.

Tests	m	$S = 0$	$S = 1$	Δ_S	95% interval
Base	1	0.9192	0.9154	-0.0038	[-0.0206, 0.0114]
Base	3	0.9428	0.9466	0.0038	[-0.0069, 0.0168]
Base	5	0.9505	0.9505	0.0000	[-0.0076, 0.0091]
Plus	1	0.8826	0.8758	-0.0069	[-0.0244, 0.0091]
Plus	3	0.9154	0.9146	-0.0008	[-0.0122, 0.0114]
Plus	5	0.9253	0.9207	-0.0046	[-0.0152, 0.0038]

Table 8: Corrected original-test oracle success, with all original candidate prefixes retained. These scores isolate the prompt-prepend correction and must not be conflated with either EvalPlus row in Table 7.

m	$S = 0$	$S = 1$	Δ_S	95% interval
1	0.9253	0.9215	-0.0038	[-0.0229, 0.0152]
3	0.9489	0.9512	0.0023	[-0.0076, 0.0130]
5	0.9573	0.9573	0.0000	[-0.0069, 0.0084]

Same-executor scripted controls. To establish a competence reference for the actual low-level controller, we evaluated four transparent goal policies on the original movement code, canonical menus, five-tick decision period and 200-tick horizon, with the same five seed values on each layout. A greedy solo recipe chooser was compared with a rescue version: after two successive unchanged full-state decision observations, both players choose `move away.`; a matched control chooses `wait.` at the same trigger. A uniform legal-goal policy uses a separate chooser RNG. Table 9 reports the 80 episodes; every saved primitive-action trajectory was replay-verified. The rescue policy completes deliveries on all layouts with the original executor. These are post-hoc scripted baselines, not optimal planners or learned MARL competitors. The rescue ablation concerns this scripted policy; it neither identifies an LLM mediation effect nor establishes why plurality changes reward in the frozen study. Escape and wait share a trigger rule, although their subsequent state distributions differ.

Table 9: Same-executor controls: mean soups over five seeds. Layout order matches the frozen panel.

Policy	Open	Mid	Tight 1	Tight 0
Greedy	0.00	0.00	0.00	0.00
Greedy + wait	0.00	0.00	0.00	0.00
Greedy + escape	2.20	4.00	1.20	1.00
Uniform goal	0.20	0.60	1.20	0.00