WHEN DOES MODEL-SPECIFIC TOOL-CALL FORMATTING PAY OFF?

Anonymous authors
Paper under double-blind review

ABSTRACT

Different LLMs often prefer different tool-call formats. Should each model then get its own format? Our framework, FormatGain, separates the model–format interaction, the oracle gain of per-model formats over the best shared format, and the selection error of a small-budget rule. Decision margins link the first two: a model gains from another format only when its interaction advantage there exceeds that format's average deficit. So one interaction can come with a large gain or none, and a gain can still be lost in selection. Eight designs, five on our suite and three external, show three regimes (post hoc): little to gain, a gain lost, a gain kept. On WorkBench, the corrected interaction share is 0.38 but the oracle gain is only 0.001, and fixed `json` and a rule pooling the other models' history both had lower observed regret than every probe. On three related designs of our suite, a ten-task probe gains 0.030 to 0.105 over the hindsight best shared format. The gains transfer only in part: with new models they held on the strict-parser and 4k-token designs but not on three others, they held on new task templates, and on new BFCL task families the probe showed no detected advantage over the pooled-history rule.

1 INTRODUCTION

An LLM agent calls a tool by writing text in a fixed *tool-call format*, such as a JSON object or a Python-style function call. The same format need not work equally well for every model: format changes can move accuracy by tens of points, and model rankings can change across harnesses (Sclar et al., 2024; Zhang et al., 2026; Yao et al., 2026). This matters when a team replaces an agent's base model. Should it keep one shared format, or choose a new format for the incoming model?

Evidence that formats affect models differently does not answer this question. Choosing a format per model helps only when its choices beat a strong shared format and a team can find them from a few runs. We separate three quantities in a framework we call FormatGain. The *interaction* says whether the score gaps between models change with the format. The *oracle gain* is how much choosing each model's best format adds over the best shared format. The *selection error* is how much of that gain a rule loses when it picks from a small budget. The three need not move together.

Figure 1 shows how far apart the first two can be. Call the part of the variation in model gaps that moves with the format the *interaction share*. On our suite with a strict parser (panel a), each of the four formats is best for some model, and choosing the format per model adds 0.143 over the best shared format. On WorkBench (panel b) the corrected interaction share is larger (0.38 against 0.32 in panel a), yet `json` is best for five of the six models, and choosing the format per model adds only 0.001. Why can a system show large model-specific sensitivity and offer almost nothing to gain?

Proposition 1 answers it. Moving a model from the best shared format to another gains its own advantage there, minus that format's average deficit. The best such gain is the model's *decision margin*, and the oracle gain is the mean margin. The interaction share sees the advantages but not the deficits. On WorkBench they lie in formats that are worse on average, so they do not pay.

We make three contributions. First, we separate the interaction from the value of choosing a format per model: decision margins, not the size of the interaction, set the oracle gain over the best shared format (§3). Second, we measure how much of this opportunity a rule that selects from a small budget turns into gain on held-out tasks (§5). We see three regimes: a strong shared default leaves

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.

(a) our suite, strict parser

$\widetilde{\kappa} = 0.32$, $\widehat{G} = 0.143$

	json	react	xml	pythonic
gpt-oss-120b	**0.85**	0.58	0.67	0.63
gpt-oss-20b	0.12	0.29	**0.31**	0.12
llama-3.1-8b	**0.31**	0.00	0.02	0.29
llama-3.3-70b	0.50	**0.62**	0.44	0.54
mistral-small-24b	0.40	0.75	0.37	**0.92**
phi-4	0.33	**0.77**	0.19	0.42
qwen3-14b	**0.79**	0.48	0.62	0.63
average	*0.47*	*0.50*	*0.37*	*0.51*

(b) WorkBench

$\widetilde{\kappa} = 0.38$, $\widehat{G} = 0.001$

	json	react	xml	pythonic
gpt-4o-mini	0.52	0.51	0.38	**0.53**
claude-haiku-4.5	**0.77**	0.63	0.40	0.67
gpt-5.6-luna	**0.72**	0.47	0.35	0.26
qwen3-32b	**0.59**	0.50	0.28	0.58
llama-3.3-70b	**0.46**	0.33	0.24	0.35
deepseek-chat	**0.62**	0.61	0.34	0.61
average	*0.61*	*0.51*	*0.33*	*0.50*

Figure 1: **A larger interaction share, a smaller gain.** Success rate of each model under each call format; a box marks each model's best format, shading the best shared format. *(a)* Our suite with a strict parser, 7 models $\times$ 4 formats, 52 tasks: every format is best for some model. *(b)* WorkBench, 6 models $\times$ 4 formats on its 568 tasks that need an action: `json` is best for five of six models, so choosing the format per model adds only 0.001 over it, against 0.143 in (a) (App. K).

little to gain, selection error takes the gain that exists, or a ten-task probe keeps it, as on three variants of our suite. Third, we test what transfers when the models, the task templates or the task families change (§6). Frozen probe choices kept part of their gain with new models and on new task templates of our suite, but on new BFCL task families they showed no detected advantage over a rule that pools the other models' history. A new design's *old label* is the regime its configuration had with earlier models. On our suite and on BFCL, reading 20 tasks of the new design did no better than copying the old label.

2 Related Work

Harness studies. Kapoor et al. (2024) argue that agent evaluation mixes the model with the harness. Meaning-preserving format changes move accuracy by tens of points and correlate only weakly across models (Sclar et al., 2024), and the agent–computer interface shapes performance (Yang et al., 2024). Recent work reports capability per (model, harness) configuration (Yao et al., 2026), harness-induced variance that can exceed model-induced variance (Zhang et al., 2026), harness sensitivity that is not monotone in capability (Cho, 2026), and controlled and factorial designs (Ben Sghaier et al., 2026; Liu, 2026; Haase et al., 2026). Ge et al. (2026) split ability into model and harness parts that add on the logit scale, which gives an interaction on the success scale ($\kappa > 0$) but no model-specific best harness ($G = 0$). These studies report harness effects; none separates the interaction from the value of adapting to it (App. AD compares the five closest designs).

Algorithm selection. Choosing a call format per model is an instance of algorithm selection (Rice, 1976), with models as the instances and formats as the algorithms. That literature judges a selector against two references: the *single best solver*, the one algorithm that is best on average, and the *virtual best solver*, an oracle that picks the best algorithm for each instance (Bischl et al., 2016). The gap between the two is the most that per-instance selection can gain, and a selector is scored by how much of that gap it closes (Kerschke et al., 2019). Our oracle gain is this gap with models as instances, and our selection error is a rule's shortfall from the virtual best. We do not claim a new principle. What we add for agent call formats is the link from the model–format interaction to this gap, through decision margins, and measurements of when strong defaults and model-specific probes succeed or fail. The interaction share adapts functional ANOVA (Hutter et al., 2014) and variance-component analysis of evaluation noise (Riezler & Hagmann, 2023).

Model swaps. Deployed models change over time (Chen et al., 2023); a swap under a fixed harness resembles prediction churn (Milani Fard et al., 2016) and backward-compatible updates (Bansal et al., 2019; Yan et al., 2021; Cacioli, 2026). We use WorkBench (Styles et al., 2024), Harness-Bench (Yao et al., 2026) and BFCL (Patil et al., 2025) (survey: Yehudai et al., 2025).

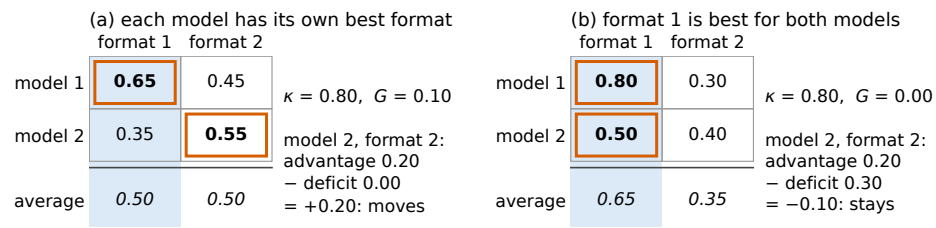

Figure 2: **The same interaction, two different gains.** Two 2×2 tables of success rates. A box marks each model's best format; shading marks the best shared format (in (a) both formats average 0.50, and ties go to format 1). Table (b) is table (a) with 0.15 added to every score on format 1 and taken from every score on format 2, so $\kappa = 0.80$ in both. Read model 2's row. Its interaction advantage on format 2 is 0.20 in both tables. In (a) format 2 has no deficit, so model 2 moves to it and $G = 0.10$. In (b) format 2 averages 0.30 less than format 1; the advantage does not cover this deficit, model 2 stays, and the gain falls to 0.

3 Interaction, decision margins and selection error

We model an LLM agent as a *base model* run under a *harness*: a system prompt, a tool interface, a call format and a step budget. Within each design, only the call format changes. For example, the same lookup on our suite is written {`"tool": "db", "args": "Zorin"`} in `json`, `Action: db(Zorin)` in `react`, `<act tool="db">Zorin</act>` in `xml` and `db("Zorin")` in `pythonic`. Harness-Bench adds *schema-guided* `json`, with the schema requested but not enforced (`strict` off). A *crossed design* runs a models under b formats on the same n tasks. Let p_{mh} be the expected evaluation score of model m under format h: task success on our suite and on WorkBench, turn accuracy on BFCL, and the graded score on Harness-Bench. P is the $a \times b$ table of these scores, and $\widehat{P}$ the table of cell means.

3.1 The interaction share κ

Split each cell mean into a model effect, a format effect and what is left over: $p_{mh} = \mu + \alpha_m + \beta_h + \gamma_{mh}$, with every effect summing to zero. The interaction cell γ_{mh} is the part of the cell that neither the model nor the format explains. Then

$$V_M = \tfrac{1}{ab}\textstyle\sum_{m,h} \alpha_m^2, \qquad V_{M\mathcal{H}} = \tfrac{1}{ab}\textstyle\sum_{m,h} \gamma_{mh}^2, \qquad \kappa = \tfrac{V_{M\mathcal{H}}}{V_M + V_{M\mathcal{H}}}, \tag{1}$$

where $\mathcal{H}$ is the harness, here the format. $\kappa = 0$ means that every pair of models keeps the same score gap under every format. κ grows when the interaction cells are large next to the model effects.

3.2 The oracle gain and decision margins

Figure 2 shows two small tables with the same κ and different gains. In table (a) each model has its own best format, and choosing the format per model adds $G = 0.10$ over the best shared format. Table (b) moves every model up on format 1 and down on format 2 by the same amount. This changes only the format effects β_h, so the model effects, the interaction cells and κ stay the same: both tables have $V_M = 0.0025$, $V_{M\mathcal{H}} = 0.01$ and $\kappa = 0.80$. But now format 1 is best for both models, and the gain is 0. The oracle gain is the average gain of the best per-model format over the best shared format $h_0 = \arg\max_h \frac{1}{a}\sum_m p_{mh}$:

$$G(P) \;=\; \tfrac{1}{a}\textstyle\sum_m \max_h p_{mh} \;-\; \max_h \tfrac{1}{a}\sum_m p_{mh}. \tag{2}$$

It is the most a per-model choice can add when the table is known and measuring costs nothing. Write $d_h = \beta_{h_0} - \beta_h \geq 0$ for format h's *deficit*: how much worse it is than h_0 on average.

Proposition 1 (κ is a pairwise share, and it does not determine G)**.** *Let $\Delta_{ij,h} = p_{jh} - p_{ih}$ be the gap between models i and j under format h. (i) Whenever the denominator is positive,*

$$\kappa \;=\; \frac{\mathbb{E}_{i<j}\big[\,\mathrm{Var}_h(\Delta_{ij,h})\big]}{\mathbb{E}_{i<j}\big[\mathbb{E}_h(\Delta_{ij,h}^2)\big]}, \tag{3}$$

with the expectations uniform over the $\binom{a}{2}$ model pairs and the b formats. (ii) The oracle gain is the mean decision margin*, and it is capped by the interaction:*

$$G \;=\; \frac{1}{a}\sum_{m}\max_{h}\left[\gamma_{mh}-\gamma_{mh_0}-d_h\right] \;\le\; \frac{1}{a}\sum_{m}\max_{h}\gamma_{mh}. \tag{4}$$

(iii) Adding a common offset c_h to every model's score under format h leaves α, γ and κ unchanged and changes d, so it can change G: κ does not determine G. On an unrestricted real table, the offset $c_h = -\beta_h$ sets every deficit to zero and reaches the bound of part (ii). When the entries are success probabilities, only offsets that keep every $p_{mh} + c_h$ in $[0, 1]$ are allowed.

The proof is in App. B.1, where both identities are also checked to 10^{-9} on the nine designs we analyze. Part (i) gives the one definition we use: κ **is the share of the variation in model-pair gaps that moves with the format, within one design.** It says nothing about a format outside the design. Part (ii) in plain words: moving model m from h_0 to format h gains its interaction advantage there, $\gamma_{mh} - \gamma_{mh_0}$, minus that format's deficit d_h. The decision margin is the largest such gain, never below zero. κ sees the advantages but not the deficits (Fig. 2b).

3.3 Selection error

A team does not see the incoming model's row of P before it runs that model. It picks the format with a rule π that uses a finite budget: other models' history, and perhaps a small *probe*, a run of the incoming model on a few tasks per format. We score the rule on held-out tasks. On one held-out table, let h_0 be that table's best single shared format and $\widehat{G}$ its oracle gain. Let $U(\pi)$ be the mean held-out score of π's choices. The *regret* of a choice is the model's best held-out score minus the chosen format's score, and the *selection error* $R(\pi)$ is π's mean regret. A rule's gains over the best single shared format and over the pooled-history rule π_{history} (§4) are, exactly,

$$\begin{aligned} A_{\text{shared}}(\pi) &= U(\pi) - U(h_0) = \widehat{G} - R(\pi), \\ A_{\text{history}}(\pi) &= U(\pi) - U(\pi_{\text{history}}) = R(\pi_{\text{history}}) - R(\pi). \end{aligned} \tag{5}$$

The second needs no common format: π_{history} may give each model a different format. For example, on our strict-parser design, averaged over $2{,}000$ random splits of its tasks, the held-out $\widehat{G}$ is 0.144 (0.143 on the full table), and a probe of 10 tasks per format has $R = 0.039$, so it gains 0.105 over the best shared format. A team can fail to gain in two ways. The first is *limited adaptation opportunity*: $\widehat{G}$ is small, so a per-model choice adds little. The second is *opportunity offset by selection error*: $\widehat{G}$ is not small, but $R(\pi)$ takes it, or leaves a gain whose sign depends on how ties are broken. If $\widehat{G} - R(\pi)$ stays above zero under every tie-breaking order we evaluate, we call it a *realized adaptation gain*. We keep three gains apart: the population gain $G(P)$; the *empirical* oracle gain $\widehat{G} = G(\widehat{P})$, which describes one measured table and is not a bound on $G(P)$; and a rule's achieved gain A_{shared}.

What each number is. Raw κ (Eq. (1) on cell means) is above zero even with no interaction. The corrected $\widetilde{\kappa}$ removes task noise and is clipped at zero; we use it only to describe sensitivity (App. S).

4 Designs, comparators and checks

Designs. We analyze nine crossed designs (Table 2). Six come from our tool-use suite: 52 multi-hop arithmetic questions over *fictional* entities, so answers cannot be recalled and need a database tool and a calculator (App. D). Its *permissive-parser design* runs seven models under four formats, and its parser takes the last call in a reply. Five variants use a strict parser (the first call wins), a 4k-token budget, typed arguments, reasoning models or strong models. Three designs come from external benchmarks with tasks we did not write. WorkBench (Styles et al., 2024) has 690 workplace tasks from 69 templates, run by six models in $16{,}560$ episodes; κ and $\widehat{G}$ use the 568 tasks that need an action. The other two are BFCL v3 multi-turn (Patil et al., 2025) and Harness-Bench (Yao et al., 2026). Prospective studies followed committed protocols (App. AW); post hoc analyses are labelled as such.

Table 1: **Comparators and estimands.** (a) What beating each comparator shows (*when chosen on history). (b) What each headline estimates; paired gain: frozen choices minus comparator, held out.

(a)

comparator	chosen on	formats	reads incoming model?	deployable?	beating it shows
hindsight best shared, h_0	held-out tasks	one shared	yes, held-out	no	a gain over every shared format
common format	fixed, or all models' history	one shared	its history runs*	yes	a gain over a common default
pooled-history rule	other models' history	per model	no	yes	a gain over one historical rule

(b)

headline	target	unit	assumptions
split averages (Table 3)	mean gain over random splits	split	splits share tasks; percentiles are spread, not intervals
one frozen split (§5.2)	paired gain on similar tasks	task (32)	models, choices fixed; t interval, coverage measured
new models (§6.1)	gain on one held-out table	task	no interval; tie-order range only
new templates (§6.1)	paired gain over templates	template (24)	sign-flip: under no effect, template gains symmetric, independent
new families (§6.2)	paired gain, held-out tasks	task (64)	models, choices fixed; t interval, coverage checked
tie-order ranges	the result under each tie order	tie order	not sampling uncertainty

Table 2: **The nine crossed designs (post hoc).** The corrected share $\widetilde{\kappa}$, the typical interaction cell $\sqrt{V_{M\mathcal{H}}}$ in each benchmark's own units, the empirical oracle gain $\widehat{G}$ (Eq. (2), full table), the share of models whose best format is the shared best, and that format. Similar $\widetilde{\kappa}$ can come with very different $\widehat{G}$. †Fails the parse check ($0.159 \geq 0.15$). Strong models use the strict parser.

design	models×formats×tasks	$\widetilde{\kappa}$	$\sqrt{V_{M\mathcal{H}}}$	$\widehat{G}$	best = shared	shared best format
BFCL v3 multi-turn (App. J)	7×4×40	0.00	0.022	0.008	0.43	`react`
Harness-Bench, six formats (App. Y.5)	13×6×29	0.35	0.082	0.023	0.46	`xml`
our suite, 4k-token budget (App. G)	7×4×52	0.21	0.128	0.096	0.57	`pythonic`
our suite, permissive parser†	7×4×52	0.49	0.171	0.132	0.14	`json`
our suite, typed arguments (App. H)	7×4×52	0.15	0.093	0.038	0.57	`json`
our suite, reasoning models (App. G)	7×4×52	0.50	0.102	0.074	0.29	`react`
our suite, strict parser (App. F)	7×4×52	0.32	0.139	0.143	0.14	`pythonic`
our suite, strong models (App. I)	4×2×52	0.39	0.120	0.000	1.00	`json`
WorkBench (App. K)	6×4×568	0.38	0.067	0.001	0.83	`json`

The parse check. A design passes this check (the appendix calls it the validity gate) if fewer than 15% of its episodes end with no parsed tool call and its pooled score is between 0.10 and 0.90, thresholds fixed in advance (App. V). Eight designs pass. The permissive-parser design fails, at **0.159**, so we report it as a separate sensitivity row, never pooled with the eight (App. E).

Comparators and estimands. Each comparator answers a different question (Table 1a). The *pooled-history rule* π_{history} (the appendix's pooled default) gives each incoming model the best format on the other models' history, so models can get different formats. A *common format* chosen on the history of all models sees more: it also reads the incoming model's own history runs. The regimes and new models use the hindsight best shared format h_0; on new templates and families the primary comparator is the pooled-history rule. Table 1b says what each headline estimates. An interval that contains zero means "no detected advantage", never equivalence.

5 Results on the eight designs

5.1 Interaction does not determine opportunity

Sensitivity changes with the design. Across the eight designs that pass the parse check (Table 2), $\widetilde{\kappa}$ runs from 0.00 on BFCL to 0.50 with reasoning models, and 0.38 on WorkBench. A trend seen on our suite, lower $\widetilde{\kappa}$ when a schema fixed more of the call format, did not replicate on Harness-Bench with new models (App. V). So κ describes a design, not a format.

Similar interaction, very different opportunity. The two designs with $\widetilde{\kappa}$ of 0.39 and 0.38 have $\widehat{G}$ of 0.000 (strong models, all on `json`) and 0.001 (WorkBench, five of six on `json`). The strict parser has a smaller share, $\widetilde{\kappa}$=0.32, but the largest gain, $\widehat{G} = 0.143$. $\widetilde{\kappa}$ does not order the gain: over the eight designs that pass the parse check, their rank correlation is -0.21, a loose estimate.

Decision margins explain the difference. Proposition 1 says where to look: at each model's decision margin (Table 21 in App. L lists them for every design). On WorkBench 1 model of 6 has a positive margin, 0.005 (`gpt-4o-mini` on `pythonic`). The largest advantage in a format worse than `json` is 0.140 (`gpt-4o-mini` on `xml`), below that format's deficit of 0.281. On the strict parser, by contrast, 6 of 7 models have a positive margin, the largest 0.346 (`phi-4` on `react`).

5.2 Selection error can erase the available gain

Rules and splits (post hoc). How much of $\widehat{G}$ does a finite-budget rule keep? On the eight designs that pass the parse check we run the same rules: (1) *inherit* the outgoing model's format; (2) the pooled-history rule; (3) a *probe*: run the incoming model on 3, 5 or 10 tasks per format and keep the best; (4) on WorkBench only, fixed `json` and a probe on all its probe tasks. Each design is split 2,000 times into 10 history, 10 probe and the remaining held-out tasks (on WorkBench, templates). Ties go to `json`, and we rerun every rule under every tie order.

Opportunity, selection error and achieved gain. Table 3 gives three numbers per design: the held-out oracle gain $\widehat{G}$, the probe's selection error R and its achieved gain $\widehat{G} - R$. We name three patterns in them with the regimes of §3.3. The names are post hoc labels for these designs at 10 probe tasks per format; they depend on the probe, the tie order and the cut on $\widehat{G}$, 0.02 in each benchmark's own units (Apps. L.1, L.5). A new design's *old label* is the regime its configuration had with earlier models (§6.1). On our suite and on BFCL, reading 20 tasks of the new design did no better than copying the old label.

Little to gain (limited adaptation opportunity). On strong models, WorkBench and BFCL the held-out $\widehat{G}$ is 0.001 to 0.013. On WorkBench every other rule scores below fixed `json`.

A gain lost in selection (opportunity offset by selection error). On Harness-Bench and on typed arguments $\widehat{G}$ is 0.044 (0.023 and 0.038 on the full tables), but the 10-task probe scores -0.015 on Harness-Bench. On typed arguments it keeps 0.007 with ties to `json`, but from -0.011 to 0.011 over the 24 tie orders.

A gain kept (realized adaptation gain). On reasoning models, a 4k-token budget and the strict parser, $\widehat{G}$ is 0.072 to 0.144. In 3 of the 8 designs, the 10-task probe's mean held-out gain over the best shared format is positive under every tie order. The means are 0.030, 0.044 and 0.105. Against the deployable pooled-history rule, which here reads 10 history tasks, the gain A_{history} is 0.078, 0.138 and 0.197 (Table 3). Only the strict parser has a positive 2.5th split percentile (0.040). More probe tasks cut the selection error: on the strict parser it falls from 0.083 to 0.062 to 0.039 at 3, 5 and 10 tasks per format, and it falls this way on 7 of the 8 designs, not on strong models (Table 24). On one declared split of the strict parser, the probe's frozen choices scored 0.161 above the pooled-history rule on the 32 held-out tasks, with a nominal 95% interval of $[0.089, 0.232]$ and measured coverage 0.955–0.957 (post hoc; an interval for that split's tasks only, App. L.2).

Cost on WorkBench. On WorkBench we also measured each rule's regret with an interval, its harm and its cost (App. K, Table 16, Fig. 4). Both deployable defaults had lower observed regret than every probe. In pairs (Table 12), the pooled-history rule and the 10-task probe had lower observed regret than inheriting, by 0.034 and 0.023. The probe had no detected advantage over the pooled-history rule: the rule's regret minus the probe's is -0.011, interval $[-0.019, 0.004]$. Its coverage was only 0.58–0.74, so these are descriptive. The 10-task probe costs 40 episodes per swap and repays that cost against the pooled-history rule for one of six incoming models, after 9,710 episodes of use (point estimates).

5.3 When does historical information help selection?

The pooled-history rule reads only the other models, and the probe only the incoming model. Two post hoc selectors read both (App. L.3): a *history tie-break*, and a *history-shrunk probe* that pulls each format's probe score toward its history score. History helped where the other models were a good guide: shrinking cut the probe's selection error by at least 0.005 only on Harness-Bench and on WorkBench, where the pooled-history rule already had lower R than the plain probe (on WorkBench from 0.016 to 0.009, still above both defaults). History hurt where the new model departs from the

Table 3: **Oracle gain, selection error and achieved gain: the same rules on every design (post hoc).** Held-out: tasks per split, of all (t.: templates). Probe episodes: 10 tasks × formats. $\widehat{G}$ and $R(\pi)$: means over 2,000 held-out splits. Last two columns: the 10-task probe's gain over the best shared format, $\widehat{G} - R(\pi)$, with ties to `json`, and its tie-order range, not an interval. Its gain over the pooled-history rule is the difference of the two R columns (Eq. (5)). †Fails the parse check.

	held-out	probe		selection error $R(\pi)$				10-task probe's achieved gain	
design	of all tasks	episodes per model	$\widehat{G}$	inherit	pooled history	fixed `json`	10-task probe	$\widehat{G} - R$	range over tie orders
limited adaptation opportunity									
our suite, strong models	32/52	20	0.001	0.025	0.011	–	0.010	−0.009	[−0.028, −0.009]
WorkBench	49/69 t.	40	0.003	0.040	0.005	0.003	0.016	−0.014	[−0.041, −0.012]
BFCL (turn accuracy)	20/40	40	0.013	0.025	0.023	–	0.022	−0.009	[−0.009, −0.007]
opportunity offset by selection error									
Harness-Bench (graded)	9/29	60	0.044	0.093	0.051	–	0.060	−0.015	[−0.018, −0.014]
our suite, typed arguments	32/52	40	0.044	0.073	0.102	–	0.037	0.007	[−0.011, 0.011]
realized adaptation gain									
our suite, reasoning models	32/52	40	0.072	0.123	0.120	–	0.042	0.030	[0.022, 0.035]
our suite, 4k-token budget	32/52	40	0.101	0.154	0.195	–	0.057	0.044	[0.038, 0.060]
our suite, strict parser	32/52	40	0.144	0.202	0.236	–	0.039	0.105	[0.091, 0.105]
fails the parse check (a sensitivity row, not pooled)									
our suite, permissive parser†	32/52	40	0.133	0.191	0.246	–	0.050	0.083	[0.075, 0.089]

Table 4: **The frozen 10-task probe when one thing changes.** Gain in points of task success (turn accuracy on BFCL), mean over the same 7 new models; choices frozen before any held-out run.

what changed	held-out tasks	over pooled history	over hindsight best	test or interval of the first gain
models (§6.1)	32 (strict parser)	+7.6	+7.1	none: one table
task templates (§6.1)	96 (24 templates)	+3.9	+3.9	sign-flip $p = 0.0128$, no interval (tie orders 3.4–6.5)
BFCL families (§6.2)	64 (6 families)	+0.06	−1.80	95% interval [−0.65, 0.78]

others: on the strict parser, where every format is best for some model, shrinking raised the error from 0.039 to 0.056, and the pooled-history rule's own R is 0.236. One common format, chosen on the history runs of all models, the new one included, had lower regret than the pooled-history rule on 6 of the 8 designs and never higher (App. L.6). Against it the probe still won on the three designs with a realized gain, but by less: 0.126 instead of 0.197 on the strict parser.

Margin and top-two gap. Two things decide whether a probe gains. A useful alternative must beat the shared default by enough, its decision margin, and the probe tasks must tell a good choice from costly ones, which is easier when a model's best format is far ahead of its second best (a large *top-two gap*). Missing the exact best format and suffering a large regret are different outcomes: when a model's top two formats are within 0.042, the probe often misses the best one but pays little, and such close rows make 65% of the misses but 48% of the selection error (App. L.4).

6 Transfer depends on what changes

6.1 New models, then new task templates

We first changed only the models: five new designs, each a copy of one design above, its *source design* (App. M, Table 31). The four suite designs reuse two of the paper's task suites and four of its configurations with new models, and the BFCL design reuses BFCL's 40 tasks, which we did not write, with the same new chat models; this tests new models on task families the paper already reports, not new tasks or new benchmarks. The models, the rule and each design's 20 calibration tasks (10 history, 10 probe) were fixed before any new model ran a task, and the probe's choices before any held-out task ran. On the held-out tasks, the frozen probe choices gained 0.071 and 0.076 over the best shared format on the strict-parser and 4k-token copies, positive under every tie order; the two copies share their models and held-out tasks, so the gains are not independent. On P-bfcl, the BFCL copy, the gain was −0.009.

Predicting whether adaptation helps. A new design's *old label* is the regime of its source design. On the four new designs of our suite, the regime read from 20 calibration tasks matched the regime read from 32 held-out tasks collected afterwards. On the new BFCL design, the regime read from 20 calibration tasks did not match the regime read from 20 held-out tasks collected afterwards. Copying the source configuration's regime matched on 4 of 4 suite designs and on the BFCL design. On our suite, reading 20 tasks of the new design did no better than copying the old label. On BFCL, reading 20 tasks of the new design did no better than copying the old label. The 20-task reading differed from the source configuration's regime on P-bfcl; there, the old label matched the held-out regime. The held-out regime is read from one table of 32 / 20 tasks; on the source designs this one-table reading differed from the design's averaged regime on up to 22% of partitions. On the source designs, whose regimes were already known, the same rule matched the regime read from the held-out tasks of the same partition on 63% to 92% of random partitions on our suite and on 55% on BFCL; copying the known regime matched on 78% to 99% and on 81%.

New task templates, same tools. We then changed only the questions of the strict-parser copy: new arithmetic over the same fictional database and tools (App. N). On 96 tasks from 24 new templates, the frozen 10-task probe beat the pooled-history selector by 3.9 points (exact one-sided sign-flip test over the 24 template means, $p = 0.0128$). The advantage seen on the suite's templates carried over to these 24 new templates. We give no interval for this gain: in planning, no interval method reached its coverage target on every simulated population (appendix). The test assumes template gains are symmetric about zero; in data simulated from ours with no gain but a few heavy losses, a p this small occurred in up to 6.9% of datasets, so the evidence is weaker than p suggests (App. Q). The gain is about half the 7.6 points of the strict-parser copy over the pooled-history rule (Table 4), and it rests on two models: qwen3-30b-a3b gained 19.8 points with xml over json (56.3 there) and gpt-4.1-nano 7.3 with react (-3.1). Because the gain comes mainly from qwen3-30b-a3b's json error, this result shows that a model-level formatting error persists on new tasks; it does not show that the probe learns task-dependent preferences. Our mechanism prediction missed: the xml gain was not larger on tasks that need a calculator call (App. N).

Does the parser make the gain? Our `react` and `pythonic` parsers stop a tool argument at its first ")": `calc((a+b)*c)` reaches the calculator as `(a+b`. All realized gains above used them; the 4k-token and reasoning designs were not audited. On the new templates a corrected parser finds the cut in 235 `react` and 212 `pythonic` cells of 672; replaying the replies with it keeps the strict-parser copy's held-out gain over the pooled-history rule between 7.1 and 7.6 points (App. O). We reran the frozen choices of qwen3-30b-a3b and gpt-4.1-nano; the probe was not rerun (App. P). With a parser that reads nested parentheses and quotes, and one fixed provider per model, the frozen choices still gain 7.7 points over json on the 96 new-template tasks (first run: 3.9). Half the rise came in formats the cut never touched; qwen3-30b-a3b's `xml`, for one, rose by 10.4 points.

6.2 A FROZEN TEST ON NEW TASK FAMILIES

Last, we froze six selectors and changed the task families: BFCL families that no design had used (Table 5, App. R; turn accuracy in points). The primary comparator is the pooled-history rule fitted on the calibration families, called the *shared default* below; it gave `json` to five models and `react` to two. On the 64 held-out tasks, the 10-task probe's gain over the shared default chosen on the calibration families was 0.06 points (95% interval $[-0.65, 0.78]$). The interval contains zero, so this study does not show that the probe beats the default; gains above 0.78 points are outside the interval. The gain is averaged over the 7 models; 3 of 7 models chose a different format under the probe than under the default. The interval covers the sampling of tasks from these 6 families; it treats the 7 models and their frozen choices as fixed. Against the same rule fitted on BFCL's 40 older tasks, the probe's gain was -1.61 points (95% interval $[-2.56, -0.66]$). Five models' formats were served by more than one provider; leaving any one of them out gave gains from -0.46 to 0.33 points (descriptive). The held-out families are new API combinations, but both main APIs appear on both sides, and 83% of the held-out gold calls use a function seen in calibration. Each turn is scored with the gold history of the earlier turns, so it measures next-call accuracy, not end-to-end task success.

Table 5: **Frozen selection on held-out BFCL task families.** Regret in turn-accuracy points, averaged over 7 models on 64 held-out tasks; lower is better. Pooled-history rules read only their declared history; probes also run 10 tasks per format. Details: App. R.

selector	choice (7 models)	new episodes per model	regret (points)
(a) fixed `json`	`json` ×7	0	2.60
(b) pooled history, calibration families (primary comparator)	`json` ×5, `react` ×2	0	2.38
(c) pooled history, BFCL's 40 older tasks	`react` ×6, `pythonic` ×1	0	0.71
(d) 10-task probe	`json` ×4, `react` ×3	40	2.31
(e) probe, history tie-break	`json` ×3, `react` ×4	40	2.10
(f) history-shrunk probe, $\lambda = 20$	`json` ×3, `react` ×4	40	2.10
best shared format in hindsight (not deployable)	`pythonic` ×7	—	0.51

This test uses one benchmark, BFCL v3 multi-turn, and 99 tasks built on two APIs (`TravelAPI`, `VehicleControlAPI`) that no earlier design in this paper used, with 7 chat models and 4 formats. A task family is a task's set of APIs. Formats were chosen on 35 tasks from 4 families and scored on 64 tasks from the 6 other families. Both sides use both main APIs, and 83% of the held-out gold calls use a function that also occurs in the calibration tasks. Each model was treated in turn as the incoming model; all seven had run BFCL's 40 earlier tasks, and λ was chosen on those cells, including the incoming model's own. This is one case, not a sample of benchmarks, families or models. Gains use unrounded regret. The probe costs each incoming model 40 episodes (4 formats × 10 tasks, 160 calls, USD 0.0230 on average); the shared default costs no new episode and reads 600 episodes of the other six models. Over the 24 tie orders the gain ranged from 0.06 to 0.54 points. Under the collector's own scorer the gain was 0.06 points.

7 Discussion and limitations

Sensitivity is not opportunity (Proposition 1), and opportunity is not realized gain (Eq. (5)). Our limits: (1) *Reach.* Every design varies only the call format, not prompts, tools or budgets, and our claims are about agents that write their own calls as text. (2) *Related designs.* The three designs with a realized gain are variants of one suite, share the parser cut, and are grouped post hoc. The two new-model gains share their models and held-out tasks, and the new-template gain reuses the suite's tools and rests on two models. (3) *Inference.* Table 1b gives each headline's unit. A tie-order range is not sampling uncertainty, and the WorkBench template-bootstrap intervals covered the true value only $\mathbf{0.58}$ to $\mathbf{0.94}$ of the time, so they are descriptive (App. K.12). (4) *When to probe.* No signal we tried, κ included, was shown to say when to probe (App. AB.3). (5) *Collections.* Comparisons between collections are bounded by run-to-run noise (Apps. E, Y.8); corrections are in App. AX.

8 Conclusion

Interaction measurements describe sensitivity to the call format, not whether a per-model format is worth it. Decision margins show where an opportunity exists, and selection error decides how much of it a finite-budget rule keeps. We saw all three regimes on eight designs (post hoc), with realized gains only on variants of our suite. There the gains transferred in part, to new models and to new task templates with the same tools; on six new BFCL task families, a frozen probe showed no detected advantage over the pooled-history rule. On our suite and on BFCL, reading 20 tasks of the new design did no better than copying the old label. So interaction alone should not justify adaptation; held-out comparisons against strong defaults should decide.

Reproducibility Statement

The supplement holds the code (one script per experiment plus 106 unit tests), the data and the scripts behind every table, figure and pinned number. Section 4 and Apps. C–K specify each design: models, formats, tasks, parsers, budgets and seeds. Each main result has one producer script, named in the first line of the file it writes: Table 2 (`experiments/review_20260919/`), the decision margins (`experiments/review_20260923_hd/`), Table 3 (`experiments/review_20260924_hd/`), the selectors, close calls and common format (`experiments/review_20260925c_hd/`, `review_20260925d_hd/`), and the three transfer tests (`experiments/127_prospective_regime/`, `135_new_templates/`, `129_heldout_families/`), and the parser audit, its rerun and the sign-flip check (`experiments/review_20260926_hd/`, `136_parser_rerun/`). Every live study ships its scored cells, so each analysis reruns offline. `verify.sh` recomputes 1000 pinned quantities and audits 231 more against the released data, then rebuilds the paper; it does not check numbers that no pin covers. `release_check.sh` runs the same checks in a fresh extraction of the supplement. Protocols with locked predictions were committed before their data (App. AW). Live re-collection is only approximate: hosted models at temperature 0 are not bit-reproducible, so routes and collection dates are recorded with each run.

AI Use Statement

AI tools were used for: synthetic data generation (the controlled simulator of App. AH and its task generators); developing the conceptual framework (the separation of interaction, oracle gain and selection error, the model×harness factorization, and the earlier five-component decomposition); formulating mathematical claims and drafting proofs (Propositions 1–4 and their proof sketches) and dataset cleaning and reformatting (normalizing heterogeneous public logs into our action vocabulary). We additionally used AI tools for writing and editing code, drafting and copy-editing portions of this paper, literature search and brainstorming.

Ethics Statement

This work analyzes publicly released benchmark logs and our own agent runs on synthetic, fictional-entity tasks. No human subjects, personal data, or sensitive content are involved. Improved attribution of agent behavior change supports safer deployment practices.

One thing a reader should know rather than discover: this repository is itself the object of measurement in a concurrent submission, which injects defects into it to score its verification machinery. That paper's results are not used here, and nothing in this paper depends on them. We say it because the two supplements share a repository, and a reviewer who reads both should not have to work that out.

References

Gagan Bansal, Besmira Nushi, Ece Kamar, Daniel S. Weld, Walter S. Lasecki, and Eric Horvitz. Updates in human-AI teams: Understanding and addressing the performance/compatibility tradeoff. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019.

Oussama Ben Sghaier, Hao Li, Bram Adams, and Ahmed E. Hassan. Don't blame the large language model: How agent harness evolution shapes coding agent quality. *arXiv preprint arXiv:2607.03691*, 2026. URL `https://arxiv.org/abs/2607.03691`.

Bernd Bischl, Pascal Kerschke, Lars Kotthoff, Marius Lindauer, Yuri Malitsky, Alexandre Fréchette, Holger Hoos, Frank Hutter, Kevin Leyton-Brown, Kevin Tierney, and Joaquin Vanschoren. ASlib: A benchmark library for algorithm selection. *Artificial Intelligence*, 237:41–58, 2016. doi: 10.1016/j.artint.2016.04.003.

Jon-Paul Cacioli. Beyond the mean: Within-model reliable change detection for llm evaluation. *arXiv preprint arXiv:2604.27405*, 2026. URL `https://arxiv.org/abs/2604.27405`.

Lingjiao Chen, Matei Zaharia, and James Zou. How is chatgpt's behavior changing over time? *arXiv preprint arXiv:2307.09009*, 2023. URL `https://arxiv.org/abs/2307.09009`.

Yong-eun Cho. It's not the capability: Harness sensitivity is non-monotone across llm agent tiers. *arXiv preprint arXiv:2605.26731*, 2026. URL https://arxiv.org/abs/2605.26731.

Chris Ge, Daria Kryvosheieva, Daniel Fried, Uzay Girit, and Kaivalya Hariharan. Agent psychometrics: Task-level performance prediction in agentic coding benchmarks. *arXiv preprint arXiv:2604.00594*, 2026. URL https://arxiv.org/abs/2604.00594.

Atticus Geiger, Hanson Lu, Thomas Icard, and Christopher Potts. Causal abstractions of neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. URL https://arxiv.org/abs/2106.02997.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, Yuanzhuo Wang, and Jian Guo. A survey on llm-as-a-judge. *arXiv preprint arXiv:2411.15594*, 2024. URL https://arxiv.org/abs/2411.15594.

Jennifer Haase, Jana Gonnermann-Müller, Paul H. P. Hanel, Nicolas Leins, Thomas Kosch, Jan Mendling, and Sebastian Pokutta. Within-model vs between-prompt variability in large language models for creative tasks. *arXiv preprint arXiv:2601.21339*, 2026. URL https://arxiv.org/abs/2601.21339.

Frank Hutter, Holger Hoos, and Kevin Leyton-Brown. An efficient approach for assessing hyperparameter importance. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, volume 32 of *Proceedings of Machine Learning Research*, pp. 754–762, 2014. URL https://proceedings.mlr.press/v32/hutter14.html.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *International Conference on Learning Representations (ICLR)*, 2024. URL https://arxiv.org/abs/2310.06770.

Sayash Kapoor, Benedikt Stroebl, Zachary S. Siegel, Nitya Nadgir, and Arvind Narayanan. Ai agents that matter. *arXiv preprint arXiv:2407.01502*, 2024. URL https://arxiv.org/abs/2407.01502.

Pascal Kerschke, Holger H. Hoos, Frank Neumann, and Heike Trautmann. Automated algorithm selection: Survey and perspectives. *Evolutionary Computation*, 27(1):3–45, 2019. doi: 10.1162/evco_a_00242.

Ming Liu. More is not always better: Cross-component interference in llm agent scaffolding. *arXiv preprint arXiv:2605.05716*, 2026. URL https://arxiv.org/abs/2605.05716.

Xing Han Lù, Amirhossein Kazemnejad, Nicholas Meade, Arkil Patel, Dongchan Shin, Alejandra Zambrano, Karolina Stańczak, Peter Shaw, Christopher J. Pal, and Siva Reddy. Agentrewardbench: Evaluating automatic evaluations of web agent trajectories. *arXiv preprint arXiv:2504.08942*, 2025. URL https://arxiv.org/abs/2504.08942.

Mahdi Milani Fard, Quentin Cormier, Kevin Canini, and Maya Gupta. Launch and iterate: Reducing prediction churn. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

Shishir G. Patil, Huanzhi Mao, Charlie Cheng-Jie Ji, Fanjia Yan, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, 2025. URL https://proceedings.mlr.press/v267/patil25a.html.

John R. Rice. The algorithm selection problem. In *Advances in Computers*, volume 15, pp. 65–118. Elsevier, 1976. doi: 10.1016/S0065-2458(08)60520-3.

Stefan Riezler and Michael Hagmann. Towards inferential reproducibility of machine learning research. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://arxiv.org/abs/2302.04054.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. Quantifying language models' sensitivity to spurious features in prompt design or: How i learned to start worrying about prompt formatting. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://arxiv.org/abs/2310.11324.

Olly Styles, Sam Miller, Patricio Cerda-Mardini, Tanaya Guha, Victor Sanchez, and Bertie Vidgen. WorkBench: a benchmark dataset for agents in a realistic workplace setting. *arXiv preprint arXiv:2405.00823*, 2024. URL https://arxiv.org/abs/2405.00823.

Sijie Yan, Yuanjun Xiong, Kaustav Kundu, Shuo Yang, Siqi Deng, Meng Wang, Wei Xia, and Stefano Soatto. Positive-congruent training: Towards regression-free model updates. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL https://arxiv.org/abs/2405.15793.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. 2024. URL https://arxiv.org/abs/2406.12045.

Yilun Yao, Xinyu Tan, Chao-Hsuan Liu, Yaoming Li, Zhengyang Wang, Wenhan Yu, Zhewen Tan, Yuxuan Tian, Guangxiang Zhao, Lin Sun, Xiangzheng Zhang, and Tong Yang. Harness-bench: Measuring harness effects across models in realistic agent workflows. *arXiv preprint arXiv:2605.27922*, 2026. URL https://arxiv.org/abs/2605.27922.

Asaf Yehudai, Lilach Eden, Alan Li, Guy Uziel, Yilun Zhao, Roy Bar-Haim, Arman Cohan, and Michal Shmueli-Scheuer. Survey on evaluation of llm-based agents. *arXiv preprint arXiv:2503.16416*, 2025. URL https://arxiv.org/abs/2503.16416.

Yunbei Zhang, Janet Wang, Yingqiang Ge, Weijie Xu, Jihun Hamm, and Chandan K. Reddy. Stop comparing llm agents without disclosing the harness. *arXiv preprint arXiv:2605.23950*, 2026. URL https://arxiv.org/abs/2605.23950.

A How to read this appendix

This section is a map. Nothing here is new evidence. The appendix follows the order of the argument: sensitivity (the interaction), opportunity (the oracle gain) and realized gain (selection error). A reader who stops before the archival record has read every current result. Names: a *single shared format* gives one format to every model; the *pooled-history rule* of the main text, which the appendix often calls the *pooled default* and E129's protocol the *shared default*, gives each incoming model the best format on the other models' history, so different models can get different formats.

1. **Formal results** (App. B). The proof of Proposition 1: the pairwise identity for κ, the oracle gain as the mean decision margin, and a check of both identities on the nine designs. Then the corrected share $\widetilde{\kappa}$ and its null floor.
2. **Experimental specifications** (Apps. C–K). Table 6 maps each design's main-text name to its appendix name, its data and its endpoint. The main text calls the gate-failing seven-model grid of our suite the *permissive-parser design*; older appendices call it the *headline grid*.
3. **Unified policy results** (App. L). The decision margins of every design; the same rules on every design that passes the parse check (App. L.1), with the tie sensitivity and the rule behind the three regimes; the frozen-choice comparisons on WorkBench and on the strict parser (App. L.2); and, post hoc, two stronger selectors, where the probe's mistakes happen, how the regimes move with their cut, and one common format against the pooled default (Apps. L.3, L.4, L.5 and L.6).
4. **The prospective study, the new-template test, the parser checks and the held-out-family test** (Apps. M–Q and R). Five new designs with new models (E127). Then the strict-parser copy's frozen choices on 96 tasks from 24 new task templates over the same database and tools (E135). Then a frozen selection test on six BFCL task families no design had used (E129). E135

and E129 use the pooled-history rule as the primary comparator. Then an audit of how often our `react` and `pythonic` parsers cut a valid call (App. O), a rerun of E135's two carrying models with a corrected parser and one fixed provider each (E136, App. P), and a check of the sign-flip test under skewed nulls (App. Q).

5. **Statistical validation** (Apps. S–U). Table 41 names, for every statistic, its response, estimator, null, resampling unit and the one conclusion it licenses, including $\widehat{G}$, $R(\pi)$, the achieved gain and the E127, E135 and E129 results; Table 1b in the main text is its short form. Then the task-blocked test and the re-run floor.
6. **Sensitivity analyses** (Apps. V–AD). The frozen replication that refuted the specification trend outside our suite (App. V); the specification ladder on our suite and on Harness-Bench; how far κ moves with the design; how often the small probe picks the wrong format; the earlier selection studies, including where κ showed no value for deciding when to probe (App. AB.3); a non-result on frontier models; and a comparison with the closest 2026 designs on interaction measurement, oracle opportunity, finite-budget selection, strong defaults and transfer (App. AD).
7. **Archival record** (App. AE to the end). Earlier framings and the history of our instruments, with the two estimands (App. AF), the pre-registration ledger and the corrections. These sections do not define the current contribution.

Part I — Formal results

B FORMAL RESULTS

This appendix holds the proofs behind Section 3. Proposition 1 comes first: κ as a share of model-pair gaps, the oracle gain as the mean decision margin, and the common offset that changes the gain but not κ. Proposition 2 then says why the raw share has a positive floor with no interaction and why the corrected share $\widetilde{\kappa}$, which Tables 2 and 3 print, removes most of it. The accounting identity of Eq. (5) needs no proof beyond its definitions: $U(\pi) - U(h_0)$ and $\widehat{G} - R(\pi)$ are the same sum written two ways, and the scripts check it on every split to 10^{-9}.

B.1 PROPOSITION 1: THE PAIRWISE SHARE, THE DECISION MARGINS AND THE OFFSET

Two questions a reader asks of κ are answered here by algebra, then checked numerically on every crossed design the paper reads (`experiments/review_20260919/kappa_identity.py`, `data/review_20260919/kappa_identity.json`). First: what does a share of *variance* have to do with a model *swap*? Second: if the interaction is large, is a per-model harness choice worth having? The first has a clean answer. The second does not, and that is the point.

Proposition 1 is stated in §3.2. We restate its setting. Let p_{mh} be the task-averaged response of model $m \in \{1, \dots, a\}$ under form $h \in \{1, \dots, b\}$, with equal weights, and write $p_{mh} = \mu + \alpha_m + \beta_h + \gamma_{mh}$ with $\sum_m \alpha_m = \sum_h \beta_h = \sum_m \gamma_{mh} = \sum_h \gamma_{mh} = 0$, so that $V_M = \frac{1}{a}\sum_m \alpha_m^2$, $V_{M\mathcal{H}} = \frac{1}{ab}\sum_{m,h} \gamma_{mh}^2$ and $\kappa = V_{M\mathcal{H}}/(V_M + V_{M\mathcal{H}})$. Part (i) is Eq. (3), part (ii) is Eq. (4) with $d_h = \beta_{h_0} - \beta_h \geq 0$, and part (iii) is the statement about a common protocol offset c_h.

Proof. $\Delta_{ij,h} = (\alpha_j - \alpha_i) + (\gamma_{jh} - \gamma_{ih})$, and the second term averages to zero over h, so $\mathbb{E}_h \Delta_{ij,h} = \alpha_j - \alpha_i$ and $\mathrm{Var}_h(\Delta_{ij,h}) = \mathbb{E}_h(\gamma_{jh} - \gamma_{ih})^2$, while $\mathbb{E}_h(\Delta_{ij,h}^2) = (\alpha_j - \alpha_i)^2 + \mathbb{E}_h(\gamma_{jh} - \gamma_{ih})^2$ (the cross term vanishes for the same reason). For any centred vector $x \in \mathbb{R}^a$, $\sum_{i<j}(x_j - x_i)^2 = a\sum_i x_i^2$, so $\mathbb{E}_{i<j}(\alpha_j - \alpha_i)^2 = \frac{2a}{a-1} V_M$ and $\mathbb{E}_{i<j}\mathbb{E}_h(\gamma_{jh} - \gamma_{ih})^2 = \frac{2a}{a-1} V_{M\mathcal{H}}$; the common factor cancels in the ratio. For the second part, $p_{mh} - p_{mh_0} = \gamma_{mh} - \gamma_{mh_0} - d_h$. So the bracket in Eq. (4) is what model m gains by leaving h_0 for h, its maximum over h is model m's decision margin, and G is the mean margin because h_0 maximizes $\frac{1}{a}\sum_m p_{mh}$. Since $\sum_m \gamma_{mh_0} = 0$ and $d_h \geq 0$, $G = \frac{1}{a}\sum_m \max_h(\gamma_{mh} - d_h) \leq \frac{1}{a}\sum_m \max_h \gamma_{mh}$, with equality when every d_h is zero. The offset c_h enters β_h only, so it moves d and leaves α and γ alone. The two 2×2 tables $\begin{pmatrix} 0.65 & 0.45 \\ 0.35 & 0.55 \end{pmatrix}$ and

$\begin{pmatrix} 0.80 & 0.30 \\ 0.50 & 0.40 \end{pmatrix}$ differ by $c = (+0.15, -0.15)$; both have $V_M = 0.0025$, $V_{M\mathcal{H}} = 0.01$ and $\kappa = 0.80$, and $G = 0.10$ for the first, 0 for the second. On the first table, a column shift with $\delta = c_1 - c_2$ gives $G = \max\{0,\ 0.1 - |\delta|/2\}$, so the largest G any shift attains is 0.1, the unshifted value and the bound of Eq. (4). So a shift cannot move G up to the largest per-model range, 0.2 on this table. □

What this licenses. Eq. (3) is the sentence the main text uses to define κ: the share of a model-pair mean contrast's second moment that moves with the form, within one design. It is a statement about *mean* contrasts; it says nothing about how many task outcomes a swap flips, which is the paired quantity of App. AF. The identity holds for the raw share; the EMS-corrected $\widetilde{\kappa}$ subtracts the same noise estimate from both terms and has no equally short form. The script checks Eq. (3) to 10^{-9} on the seven gate-passing designs, the permissive-parser design and the WorkBench cube, and prints the empirical oracle gap $\widehat{G} = G(\widehat{P})$ of each measured table, $\sqrt{V_{M\mathcal{H}}}$ and the share of models whose best form is the shared best (Table 2 in the main text). On the two designs with $\widetilde{\kappa}$ nearest 0.39, $\widehat{G}$ is 0.000 (four models, two forms, all best on `json`) and 0.001 (WorkBench, five of six best on `json`); the permissive-parser design has $\widehat{G} = 0.132$ at $\widetilde{\kappa} = 0.487$, with one of seven models best on the shared form. Across the nine designs the rank correlation between $\widetilde{\kappa}$ and $\widehat{G}$ is 0.033; with nine points that number is loosely estimated and is printed only so a reader can see that κ does not order $\widehat{G}$. Eq. (4) says why. A model gains from leaving h_0 only when some form's interaction contrast $\gamma_{mh} - \gamma_{mh_0}$ exceeds that form's main-effect deficit d_h, and κ sees the contrasts but not the deficits. The post hoc script `experiments/review_20260923_hd/decision_margins.py` prints every model's margin on the nine designs (`data/review_20260923_hd/decision_margins.json`).

B.2 The corrected share: identifiability and the null floor (Proposition 2)

The over-estimate of the raw share at $\kappa \approx 0$ (App. S, E15) is not folded-normal in origin—it is the *eta-squared* bias of a variance share built from raw sums of squares, which we now characterize and correct.

Proposition 2 (Identifiability, the null floor, and a consistent debiased κ)**.** *In a balanced two-way design with $a \geq 2$ models, $b \geq 2$ harnesses, $r \geq 2$ task-replicates per cell, additive cell means, and i.i.d. residual variance σ^2 (our own designs reuse the* same *tasks across every cell, so task difficulty induces correlation and this hypothesis holds only approximately; App. T refits the live grid with task as a third factor and reports what the approximation costs: with the same corrected-sum-of-squares normalization the value moves from* 0.487 *to* 0.491*, and the variance-component index, which weights the interaction by $b/(b-1)$, reads* 0.563 *(*0.481 *on a repeat* 21 *days later, App. Y.9) — "conservative" applies to the error term only — and we rest significance on the permutation and mean-matched-null tests rather than on the correction): (i) the components $(V_M, V_{\mathcal{H}}, V_{M\mathcal{H}}, \sigma^2)$ are identifiable from the expected mean squares, so κ is identified; (ii) since $\mathbb{E}[\mathrm{SS}_{\mathrm{eff}}] = \nu_{\mathrm{eff}}\,\sigma^2 +$ (true effect SS) with ν the degrees of freedom, the raw $\widehat{\kappa} = \mathrm{SS}_{M\mathcal{H}}/(\mathrm{SS}_M + \mathrm{SS}_{M\mathcal{H}})$ has a strictly positive floor at $V_{M\mathcal{H}}=0$, where $\mathbb{E}[\mathrm{SS}_{M\mathcal{H}}] = (a-1)(b-1)\sigma^2$ while the true share is zero, so it is biased upward at and near the null; (iii) the EMS-corrected estimator $\widetilde{\kappa} = \mathrm{SS}^{\star}_{M\mathcal{H}}/(\mathrm{SS}^{\star}_M + \mathrm{SS}^{\star}_{M\mathcal{H}})$, $\mathrm{SS}^{\star}_{\mathrm{eff}} = \max(0, \mathrm{SS}_{\mathrm{eff}} - \nu_{\mathrm{eff}}\,\mathrm{MS}_{\mathrm{resid}})$, removes the leading $O(\sigma^2)$ inflation, and $\widehat{\kappa}, \widetilde{\kappa} \to \kappa$ a.s. as $r \to \infty$.*

Proof sketch. The balanced-layout EMS give $\mathbb{E}[\mathrm{MS}_{\mathrm{resid}}] = \sigma^2$ and $\mathbb{E}[\mathrm{MS}_{M\mathcal{H}}] = \sigma^2 + r\,\theta_{M\mathcal{H}}$ (analogously for the main effects), a triangular system that identifies the components, giving (i). Multiplying by the d.f. gives the inflation of each sum of squares and the positive floor at the null (ii). That is all (ii) claims: both sums of squares are inflated, and a ratio of two inflated quantities has no fixed sign of bias away from the null, so "upward biased" holds only near the null. At a pure interaction the sign reverses: the statistic cannot exceed one, and noise in the model main effect only pulls it down. On the $5\times2\times31$ binary design, 1,000 draws, the raw share averages 0.296 at a true 0 and 0.708 at a true 1 (`data/review_20260915/ratio_bias_direction.json`). Subtracting $\nu_{\mathrm{eff}}\mathrm{MS}_{\mathrm{resid}}$ (an unbiased estimate of $\nu_{\mathrm{eff}}\sigma^2$) centers each corrected SS on the true effect SS; consistency follows from $\mathrm{MS} \to \mathbb{E}[\mathrm{MS}]$ (SLLN) and continuity of the ratio where the denominator is eventually positive. □

Part II — Experimental specifications

C EXPERIMENTAL SPECIFICATIONS: THE NINE DESIGNS

This part specifies every design in Table 2. In each design only the call format changes; the models, the tasks, the prompts, the tools, the step budget and the endpoint stay fixed. Table 6 maps the names the main text uses to the names the later appendices and the released code use, and says where each design is specified. The appendices that follow in this part give the suite (App. D), the permissive-parser design (App. E), the variants of our suite, BFCL and WorkBench; the Harness-Bench design is specified in App. Y.5, beside the external ladder it came from.

Table 6: **Names.** The main text's name for each design and check, the name older appendices use, the data directory, the endpoint, and where the design is specified.

main text	appendices	data	endpoint	specified in
permissive-parser design (our suite)	the headline grid	`live_grid4`	task success	Apps. D, E
our suite, strict parser	the strict grid	`live_strict`	task success	App. F
our suite, 4k-token budget	chat models at a 4k budget	`live_chat4k`	task success	App. G
our suite, typed arguments	heterogeneous arguments	`live_hetero`	task success	App. H
our suite, reasoning models	reasoning models	`live_reason`	task success	App. G
our suite, strong models	strong models, strict parser	`live_strict_strong`	task success	App. I
WorkBench	E120	`e120_workbench`	task success; harm	App. K
BFCL v3 multi-turn	E51	`bfcl_grid`	turn accuracy	App. J
Harness-Bench, six formats	the repaired external grid	`hbench_fixed`	graded score in $[0, 1]$	App. Y.5
the parse check	the validity gate	—	—	App. V

D LIVE MULTI-MODEL STUDY (E5): FULL DETAILS

We apply the trajectory decomposition to real LLM agents under genuine model substitution. The fixed tool-use harness (§4) is a ReAct-style agent with a `db` lookup over a *fictional* knowledge base and a `calc` evaluator, run under two interface variants—text REACT and JSON tool calls—with disk-cached, resumable calls. It is run across three open/free models—`gpt-oss-20b`, `gpt-oss-120b` (a within-family scale upgrade), and `gemini-2.5-flash` (a cross-family swap)—under two interface variants (ReAct, JSON) on the verifiable task suite. Free-tier rate limits cap the scale: the within-family contrast uses all 24 shared tasks per harness; the cross-family contrasts are preliminary (4–5 shared tasks, as `gemini`'s free quota throttled collection). All calls are cached and resumable; App. AL scales this harness to eight models and 52 tasks, and App. AI supplies benchmark-scale outcomes.

Outcomes already reveal a harness interaction. Table 7 shows success rates. `gemini-2.5-flash` solves every task under *both* harnesses (1.00/1.00); `gpt-oss-20b` solves none (0.00/0.00); and `gpt-oss-120b` is sharply harness-sensitive—0.04 under ReAct but 0.21 under JSON, a $\sim 5\times$ swing from changing only the interface. The same interface change leaves `gemini` unmoved. This is precisely a model×harness interaction. We resist reading it as a capability effect: a pre-registered strong-model replication does not support that account (App. I), and here `gemini`'s invariance is confounded with its ceiling (1.0 under both interfaces).

The trajectory decomposition separates brittle from robust substitutions. Applying the variance decomposition to each ordered substitution (response: per-task tool-call count over the {old,new} × {ReAct,JSON} design) yields interaction ratios that separate the two regimes, with the caveat that the two cross-family rows rest on 4 shared tasks each and `gemini` is at its ceiling in both:

Substitution	S_M	$S_{M\mathcal{H}}$	$\kappa = V_{M\mathcal{H}}/(V_M+V_{M\mathcal{H}})$
`gpt-oss-20b` → `gpt-oss-120b` (within-family)	0.29	0.14	**0.33**
`gpt-oss-20b` → `gemini-2.5-flash` (n=4)	0.90	0.02	≈ 0.0
`gpt-oss-20b` → `gemini-2.5-flash` (n=4)	0.96	0.00	≈ 0.0

Table 7: E5 live success rates (fraction solved; n tasks). `gpt-oss-120b` is strongly harness-dependent; `gemini` is harness-invariant.

Harness	`gpt-oss-20b`	`gpt-oss-120b`	`gemini-2.5-flash`
ReAct	0.00 (n=24)	0.04 (n=24)	1.00 (n=5)
JSON	0.00 (n=24)	0.21 (n=24)	1.00 (n=4)

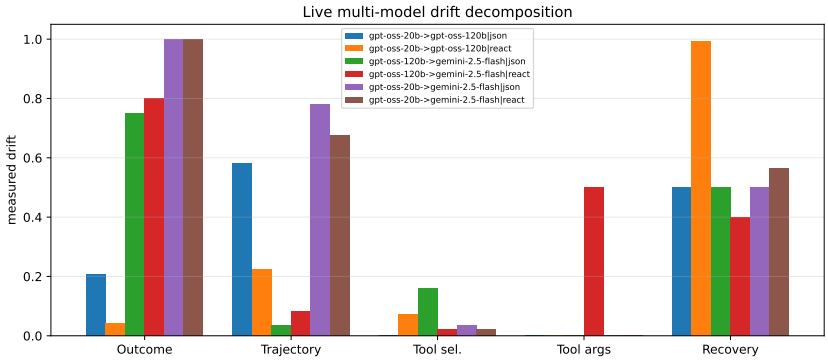

Figure 3: E5: component drift of real model substitutions under each harness. The within-family upgrade's drift is dominated by trajectory and recovery; magnitudes differ across harnesses (interaction). Cross-family contrasts are preliminary (small n).

The within-family "upgrade" is *harness-brittle* (κ=0.33 on tool-call count): much of its behavioral change comes from the interface rather than the model, so it must be re-validated per harness. Adopting `gemini` is *harness-robust* ($\kappa \approx 0$): its effect is dominated by the model ($S_M \geq 0.90$) and generalizes across interfaces. κ is facet-dependent (for the within-family pair, 0.11 on step count to 0.33 on tool-call count/success) and, at n=24 tasks, its point estimate is noisy; the robust signal is the order-of-magnitude *separation* between the harness-brittle within-family swap and the harness-invariant cross-family swaps ($\kappa \approx 0$), which echoes the $\sim 5\times$ success swing above. The multi-model study of App. AL provides the large-sample counterpart with CIs and a significance test. For the within-family pair, the component decomposition (Fig. 3) attributes the drift mainly to trajectory (0.58 under JSON) and recovery (0.50) with modest outcome change, and the phase profile concentrates the drift early in the trajectory ($0.93, 0.76, 0.47$). A practitioner watching only success would see "the upgrade helps a little"; the trajectory decomposition additionally reveals that the help is interface-contingent and the process is substantially rewired—actionable signal that benchmark scores omit.

E Re-collecting the headline grid under one provenance (E55)

An external reviewer objected that the headline κ was assembled from directories that could silently overwrite one another. The objection was correct, and following it up changed the headline number and produced a result we had not anticipated.

What was wrong. The four-form grid was stitched from three collections made between 2026-05-30 and 2026-08-05, and XML appeared in two of them. The loader resolved collisions by last-write-wins: 180 cells appeared in more than one directory and 47 disagreed on success, with the winner decided by directory order rather than by anything principled.

What we did. We re-collected the entire design in a single run at a single setting: 7 models $\times$ 4 protocol forms $\times$ 52 tasks $=$ 1456 cells, 400-token budget, temperature 0, every model complete on all 52 tasks, **zero** cross-directory conflicts by construction. All κ below are from that grid alone. In the terms of §3.1, the grid reads κ=0.501 (task-bootstrap percentile range $[0.409, 0.603]$, descrip-

tive), a typical interaction cell, $\sqrt{V_{M\mathcal{H}}}$, is 0.171 in success rate, and a second collection of the grid reads κ=0.421 (App. Y.9).

	task success by protocol form				
model	json	react	xml	pythonic	best
`gpt-oss-120b`	0.442	0.250	0.462	0.327	`xml`
`gpt-oss-20b`	0.212	0.308	0.173	0.135	`react`
`llama-3.1-8b`	0.192	0.000	0.000	0.269	`pythonic`
`llama-3.3-70b`	0.481	0.635	0.462	0.500	`react`
`mistral-small-24b`	0.404	0.712	0.250	0.923	`pythonic`
`phi-4`	0.788	0.846	0.038	0.269	`react`
`qwen3-14b`	0.750	0.327	0.596	0.654	`json`

The headline moves, and the pre-registered pair moves the most.

design	κ	bias-corrected	95% CI	stitched grid
b=2 JSON vs. ReAct (pre-registered)	0.2187	0.2038	$[0.13, 0.31]$	0.466
b=3 + XML	0.4658	0.4546	$[0.37, 0.54]$	0.377
b=4 all four	**0.5009**	**0.4874**	$[0.39, 0.59]$	0.394

All three clear a mean-matched no-interaction null at $p < 10^{-3}$, so the interaction itself was never in doubt; its magnitude and which design carries it were. Two things invert. The four-form estimate *rises* to 0.487, and the two-form pair we pre-registered *falls* to 0.204 — from the largest of the six two-form sub-designs on the stitched grid to the **smallest** on the clean one. Our locked choice was therefore the most conservative of the six, not the most flattering, which we note because the opposite would have been a fair criticism. The practical claim is untouched: **all four** forms remain somebody's optimum.

Three things this grid is asked to survive, which we had not reported. *(i) Our own parse gate.* E52 (App. AC) was declared a non-result partly on a locked no-parsed-call rate of 0.15, which it missed at 0.167. We had never reported that rate for the headline grid. It is 0.1587 — also above 0.15 — with the worst model (`gpt-oss-20b`) at 0.399, about the rate of the `qwen3-32b` whose grid we set aside. That 0.1587 is the gate runner's, counting episodes (231 of $1{,}456$); E55's own summary file averages the seven per-model rates after rounding them and reads 0.1586, and the gate verdict is the same either way. Applying our own standard symmetrically is the only defensible move, so we did: dropping the single worst-parsing model brings the grid *under* the gate at 0.1186 and κ *rises* to 0.5463 ($p < 10^{-3}$), with all four forms still somebody's optimum. Dropping every model above the gate leaves three, where the design has no power to estimate an interaction at all. The interaction is therefore not an artifact of unparsed cells — but the gate should have been reported here first, and E52's non-result should be read as a statement about *that* grid's two badly-parsing models rather than as a standard we applied only where it was convenient.

(ii) Resampling the qualitative claim. "All four forms are somebody's optimum" is true of the point estimate, and we checked how often it survives resampling rather than asserting it: over 4000 task bootstraps, at least *three* distinct forms win in 99.85% of draws but all four in only 50.5%. The stable claim is that no single interface is a safe default; "exactly four" is a coin flip, and the abstract no longer says it.

(iii) Which axis the interval covers. The published CI is a *task* bootstrap, and tasks are the axis κ moves least along (no task shifts it by more than 0.018). Over the axis the claim generalises to — models — the leave-one-out jackknife standard error is $\mathbf{0.22}$ on the corrected share and 0.21 on the finite-design share, comparable to the estimate itself. κ should be read as well determined for *these seven models* and only loosely determined for models in general.

A reviewer asked us to pick one inferential target and make the interval language match it, which is fair because the two paragraphs above sit on either side of the line. We pick **fixed levels**. The seven models and four protocol forms are chosen, not sampled: there is no population of models we drew from, and no sampling model that would license a confidence interval over one. So every κ in this paper is a descriptive share of the design that produced it, the task bootstrap is a statement

about *these* models under *other* tasks, and the leave-one-model-out jackknife is a **sensitivity**, not a confidence interval — it answers "how much does this number depend on which seven models we happened to pick?" and the answer, 0.22 against an estimate of 0.49, is "a great deal". We report it because that is the honest thing to hand a reader, not because it generalises anything.

Why it moved: one model changed under an unchanged name. Comparing the two collections cell by cell, 205 of 728 overlapping JSON/ReAct cells (28.2%) disagree — and six of the seven models account for almost none of it (deltas ≤ 0.067, two exactly 0.000). Essentially all of it is `phi-4`, whose task success went $0.240 \rightarrow 0.817$:

`phi-4`, JSON/ReAct	2026-05/06	2026-08
no-parsed-call rate	0.404	0.010
mean tool calls per run	1.90	3.38
mean errors per run	1.85	0.77

In the earlier collection `phi-4` answered without calling a tool on 40% of cells; in the later one it almost always calls one.

A third collection narrows when it moved. The comparison above uses two collections and there is a third between them, five days before the later one, which an earlier version of this appendix did not use. On the JSON column `phi-4` scores 0.077 in May/June, 0.000 on 19–22 August, and 0.788 on 24 August (E99, `data/e99_ladder_provenance.json`); every other model in this design moves by at most 0.154 across all three, on either column. So most of the JSON-column change did not accumulate over ten weeks: it happened in the two days between the last two collections, and in the middle collection `phi-4`'s JSON score was lower than at the start, not higher. Its ReAct column is a different story and we should not blur them — there it rose steadily, $0.404 \rightarrow 0.692 \rightarrow 0.846$, so 0.288 of that move predates the last two days. The sharp claim holds for the JSON column, which is the one the ladder's first rung uses. Our prompts, parser and scorer are unchanged between the two. We state the limit of what we can conclude: we cannot fully separate "the hosted model behind this alias changed" from "the earlier collection used an invocation we can no longer reconstruct", because that run's exact settings are not recoverable. Both readings lead to the same place, and it is the thesis of this paper turned on itself — **the provider's serving stack is part of the harness, so the same model alias at two dates is not the same agent**. Evaluations that quote a model name without a date are quoting an underdetermined system. We now ship, for every one of the $1{,}456$ cells, the raw model replies together with the model alias, the resolved route, the seed and the token budget, so that a reader can re-derive our parsing and scoring rather than taking them on trust. We are deliberately narrow about what that provenance does *not* contain. The replies are served from a request cache, so their *text* is the original model output but their recorded latency is a cache lookup (0.000 s for all of them) and the file timestamp is when the grid was last written, not when the model was queried; original wall-clock time and original latency were never captured and cannot be reconstructed. What can be checked is the comparison above — the same alias, the same prompts, two collections, different behaviour — and that is the claim we make. Prospective per-call provenance is now recorded going forward.

F Is the interaction an artifact of a permissive parser? (E38)

Every one of our harness prompts demands one action per step ("output exactly one line", "ONE JSON object per step and nothing else", and so on). Our parsers did not enforce it: they scanned the whole response, took the *last* match, and accepted a trailing answer even when action lines preceded it. A model that ran ahead of the environment therefore ended its episode on an unsupported number before any tool executed. A production ReAct scaffold stops generation at `Observation:` for exactly this reason. Since the single largest cell in our corpus was produced by that artifact (App. I), the objection "your interaction is a parser artifact" is the strongest available attack on our central result, and we measure it rather than concede it.

The intervention. A *strict* mode enforcing the stated contract—the *first* action wins, and an answer counts only if no action precedes it—which is what a stop sequence achieves. Nothing else

changes: same models, same 52 tasks, same prompts, same tools, temperature 0. On the exact response that carries App. I, permissive parsing returns the fabricated `Answer: 490500` while strict parsing returns the first tool call. Pre-registered with four locked predictions and with the rule for which value we would report fixed *in advance*. Table 8 compares the permissive and strict grids.

Table 8: E38: the interaction under permissive and strict parsing, on the same 52 tasks and the same seven models. The interaction survives faithful parsing at every harness-factor size, and becomes *more* stable across b; its magnitude falls.

b	interfaces	κ permissive	κ **strict**	strict CI	strict p
2	JSON, ReAct	0.4662	**0.3536**	$[0.212, 0.501]$	5×10^{-4}
3	+ XML	0.3772	0.3332	—	—
4	+ `pythonic`	0.3940	0.3401	$[0.243, 0.466]$	5×10^{-4}
range over b		0.089	**0.021**		

The artifact was real, is now quantified, and does not carry the result (P1, P2). Enforcing one action per step collapses the rate of trajectories that never emit a parsed tool call, and—beyond what we predicted—it does so on *every* interface, not just ReAct: JSON $0.154 \to 0.052$, ReAct $0.192 \to 0.052$, XML $0.181 \to 0.041$, `pythonic` $0.239 \to 0.071$. Premature termination was the dominant failure mode across the board rather than a ReAct quirk. The interaction nonetheless stands: $p{=}5 \times 10^{-4}$ at both $b{=}2$ and $b{=}4$ under the cell-level permutation, all four protocol forms remain optimal for at least one model (P4 confirmed; `llama-3.1-8b`'s optimum moves from `pythonic` to JSON, exploratory), and on the strong pool `hermes-3-70b` still swings 0.827 under JSON against 0.173 under ReAct. **What does change is the magnitude:** the permissive parser inflated $\kappa(b{=}2)$ by 0.113, a 24% reduction when corrected.

A refuted prediction (P3). We locked that `hermes-3-70b`'s ReAct success would rise above 0.20 under strict parsing. It rose from 0.019 to 0.173—a nine-fold increase, with its no-tool-call count going from $38/52$ to $0/52$, so the predicted *mechanism* is confirmed exactly—but the threshold we committed to is not met. We report it as refuted rather than rounding 0.173 up. The practical reading is that roughly 19% of that model's 0.808 swing was our parser and 81% is a real harness effect: it genuinely cannot work under ReAct, just not *quite* as badly as first measured.

Which number we report, and a note on our own pre-registration. The pre-registered rule permitted keeping the permissive value as the headline whenever the strict value landed inside its confidence interval, which it did. But that rule's stated rationale was cross-experiment *consistency*—every other experiment used the permissive parser—and not *fidelity*, and on fidelity the strict parse is plainly the better implementation of the protocols our own prompts specify. A 24% movement in the headline is not an appendix detail. We therefore keep the permissive value as the reference for comparability across our experiments *and* state the strict value beside it in the main text (§5). We record here that our own rule permitted burying it and that we declined, because fixing the rule in advance was meant to stop the flattering number being chosen after the fact.

Exploratory. Under strict parsing κ varies across b by only 0.021 ($0.354/0.333/0.340$) against 0.089 permissively. The permissive parser was adding b-dependent noise, which makes sense: how often a model gets away with running ahead of the environment depends on which protocol it is running ahead in.

G Reasoning models: the interaction does not go away, and κ depends on the token budget (E49)

Every other live study in this paper uses chat models. Cho (2026) report that harness sensitivity depends on model type, so reasoning models are the one missing *class* rather than a missing size. This appendix adds them, and reports two results we did not predict.

The trap we had to remove first. A reasoning model emits its chain of thought as billed output tokens. Our harness capped output at 400, and at that cap `o4-mini` spends all 384 tokens reasoning and returns *empty content*; one candidate burns 4,369 reasoning tokens and still returns nothing. Run that way a reasoning model scores 0 everywhere for a *harness* reason, and that artifact would have been the entire result. Because the token budget is itself part of the harness, raising it only for reasoning models would trade one confound for another, so we **re-ran the chat models at the identical raised budget** (4,000 tokens). Both grids then share budget, protocol forms, tasks and strict parsing, and model class is the only difference.

All seven reasoning models across four labs completed the grid (Table 9). `deepseek-r1` needed roughly 10 minutes per cell and 30 hours in total, which is worth recording as a practical cost of evaluating reasoning models in an agent loop, but it did finish. Two further candidates were excluded before collection (one never terminates its reasoning, one emits special-token artifacts instead of JSON). `gpt-oss-20b` fails the pre-registered parse gate in the chat grid and is excluded from it, leaving six models on each side.

Table 9: E49: the same design at matched token budget. Reasoning models are stronger and the interaction is *larger*, not smaller, though the intervals overlap. $S_{\mathcal{H}}$ and $S_{\mathcal{M}}$ swap importance between the classes.

class	models	pooled success	κ	95% CI	$S_{\mathcal{M}}$ / $S_{\mathcal{H}}$
reasoning	7	**0.705**	0.541	[0.383, 0.730]	0.042 / 0.048
chat	6	0.505	0.306	[0.200, 0.436]	0.168 / 0.018

P2 refuted, and we do not claim the reverse either. We predicted $\kappa_{\text{reason}} < \kappa_{\text{chat}}$ with non-overlapping intervals, reasoning that a model which thinks before acting can repair an awkward protocol by itself. That is **refuted**: the point estimate goes the other way (0.541 against 0.306, both raw, as in Table 9). But the intervals overlap, so the honest statement is the weaker one. There is *no evidence that reasoning models are less harness-sensitive*, and we cannot claim they are more sensitive either. What we can say is that the interaction does not disappear for this class: against a mean-matched *no-interaction* null, κ_{reason} — bias-corrected to 0.501 from the raw 0.541 in Table 9 — sits at z=8.7 (0/4000 null draws reach it), so protocol form still matters when the model reasons first. We test it this way rather than by asking whether the bootstrap interval excludes 0: the raw η^2 ratio that interval resamples is a ratio of non-negative sums of squares and cannot reach 0, so "the CI excludes zero" would be true here whatever the data said (App. J shows a case where exactly that test manufactured a result).

What did change is the structure, not just the size. The two variance shares swap importance. For chat models the *model* term clearly dominates ($S_{\mathcal{M}}$=0.168 against $S_{\mathcal{H}}$=0.018). For reasoning models the two are close (0.042 against 0.048). The honest reading is the weaker one: model identity stops being the dominant term, rather than the harness becoming dominant. Reasoning models are more alike in raw capability, so which model you pick matters less.

XML is nobody's best (P3, confirmed). Three of the four protocol forms are the best choice for at least one reasoning model (JSON for `o4-mini` and `kimi-k2-thinking`, ReAct for `o3-mini` and `qwen3-next-80b-thinking`, `pythonic` for the two remaining Qwen models). There is still no interface to default to: no single form is best for more than three of the seven.

Why reasoning models are harness-sensitive: a mechanism we already had. An earlier experiment in this project (E30) measured protocol *conformance* for three self-hosted reasoning workers of the previous generation, and the result is stark. Conformance was 0.000 for `r1-qwen-32b`, 0.042 for `r1-llama-8b` and 0.312 for `qwq-32b`, and **every single non-conformant episode had the same cause**: the chain of thought leaked into the protocol channel (48/48, 46/46 and 33/33 think-contaminated). A reasoning model that cannot keep its thinking out of the response channel cannot follow a response protocol at all.

Read against E49 this gives the arc. In the current API reasoning models the leak is all but fixed: no-parsed-call rates are 0.000–0.010, so conformance has gone from near-zero to near-perfect in

one model generation. But harness sensitivity did *not* go away with it. It moved. The problem is no longer whether a reasoning model can emit the protocol, it is which protocol you hand it, and that is exactly what the variance shares say: the harness term now outweighs the model term for this class. The symptom was fixed; the disease was not. This is also why we report κ rather than a conformance rate, since a conformance rate would now read as solved.

κ depends on the token budget (fallback (d), triggered). The chat re-run gives κ=0.306 where the same models, tasks and protocol forms at a 400-token budget give 0.466. The budget is part of the harness, so this is a real harness effect rather than an inconsistency, but it qualifies the headline: κ=0.47 **belongs to a 400-token harness**, and a more generous budget lets models recover from an awkward protocol, shrinking the interaction to 0.31. Both numbers are reported here and each is labelled with the harness it belongs to. A practitioner reading this should take the operational point: raising the output budget is itself a partial remedy for harness sensitivity, and a cheaper one than switching interfaces.

H HETEROGENEOUS ARGUMENTS, AND TWO CORRECTIONS TO OUR OWN ARGUMENT METRIC (E48)

What this design is for now. This appendix specifies the typed-argument design of the main text (*our suite, typed arguments*): the tool surface, the tasks and the validity gate below. In the current argument it is a case of opportunity offset by selection error at the declared plain probe (Table 3); with history tie-breaking the probe keeps a small gain under every tie order (App. L.3). The paper's claims about this design rest on its interaction share, oracle gain and selection error. The profile predictions (Table 10) and the two corrections below record how an earlier framing, the five-component drift profile, fared on this design; that profile carries no current claim (App. AE).

The earlier framing. E39 originally found the five-component profile interface-invariant (cosine 0.957) with `tool_args` dominant under all four protocol forms, refuting all of its predictions. That reading did not survive the two argument-alignment corrections recorded later in this appendix: re-run, E39 gives cosine 0.870 and *confirms* interface-dependence. The narrative below is kept in the order it happened, and every verdict is restated at the end from the corrected numbers. We diagnosed a cause in the task suite: every tool in the original harness takes exactly *one string argument*, so the argument component is one-dimensional by construction and is dominated by free-form arithmetic strings. E48 removes that constraint and re-runs the identical estimator. **Our diagnosis was wrong.**

Design. A new tool surface with four tools taking multiple *named, typed* fields—`lookup(table,key,field)`, `filter(table,field,op,value,limit?)`, `aggregate(table,field,op,keys)`, `compute(op,a,b,round_to?)`—spanning enum, string, number, list and optional-integer types, with no free-form arithmetic string. 52 deterministic tasks needing 1–5 calls, the same seven models, the same four protocol forms, $7 \times 4 \times 52$ complete with no missing cells. The estimator is untouched: `tool_arg_drift` already compares (name, value) sequences, so recording each field as its own pair makes the component multi-dimensional with *zero* change to the metric. The manipulation is the tool surface and nothing else, and the profile analysis is run by E39's own script via `--dirs`, so the code that produced the refuted result is the code that evaluates the new one.

Three zero-cost gates ran before any API call: an oracle executed all 52 intended call plans through the real tools and required the declared answer (52/52); a round-trip rendered every planned call in each interface's documented syntax and required exact recovery (536/536); and a fuzz suite threw 448 adversarial calls at the tools requiring that none raise. The suite is solvable and parses: pooled success 0.698 (pre-registered band $[0.15, 0.85]$) and pooled no-parsed-call rate 0.107 (threshold 0.15), so the validity gate **passed** and the refutations below are results rather than artifacts. Table 10 puts E48 next to E39 on the identical analysis.

What did move (P5, confirmed). The headline interaction replicates on this third task family: κ=0.182 raw, bias-corrected **0.148**, task-bootstrap 95% CI $[0.085, 0.269]$, with $S_{\mathcal{M}}$=0.186 and $S_{\mathcal{H}}$=0.012. Significance rests on the mean-matched no-interaction null (p<0.001 against a null mean of 0.007), *not* on the interval excluding zero: as App. J shows, the raw ratio has a non-zero

Table 10: E48 versus E39 on the identical analysis, after the two argument-alignment corrections described below. The verdicts are mixed and suite-dependent, not uniformly refuted as an earlier version of this caption claimed: E39's P1 and P2 confirm (post hoc), E48's P1 still refutes, and E48's P2 and P3 confirm. E39's *third* locked prediction is the decisive one and is reported separately below — it refutes, which makes the interface-dependence outcome-driven.

locked prediction	threshold	E39	E48
P1 profiles differ by interface	median cosine <0.90	0.870 conf.	$\mathbf{0.940}$ refuted
P2 dominant component varies	≥ 2 distinct	2 conf.	$\mathbf{2}$ conf.
P3 `tool_args` not universal	dominant in $\leq 2/4$	$3/4$ refuted	$\mathbf{0/4}$ conf.

null floor, so "the CI excludes 0" is unfalsifiable for it, and an earlier version of this paragraph rested on exactly that void criterion with a one-generation-stale interval. It is smaller than the arithmetic family's 0.47, which is expected: pooled success is 0.70 here and three models sit at 0.88–0.96 in every interface, so there is little room left for an interaction. The optimum is still split—JSON for four models, ReAct for three—so the practical claim survives on a third family.

Correction: the saturation was our own alignment bug, not a property of the metric. The account above was wrong, and the error was ours. The canonical `tool_args` estimator concatenated the tool calls of *every* pair into two lists and compared them *by index*. As soon as the two sides emit different numbers of calls — the phenomenon the metric exists to measure — it compared call i of one trajectory against call i of a *different* trajectory and silently dropped the tail past $\min(\text{len})$. A correctly aligned implementation (Needleman–Wunsch on tool names, per pair) already existed in the same module and was used by three other experiments, which had stored both: they differ by a mean of 0.481 and up to 0.810, the flat version reading 0.90–0.98 where the aligned one reads 0.16–0.65.

With the aligned estimator the axis is nowhere near its ceiling. On the heterogeneous suite the per-interface mean `tool_args` runs 0.08–0.13 (E48); on the WebArena pairs the aligned+semantic estimator runs 0.26–0.45 across the 6 model pairs where the naive flat-zip estimator reads 0.91–1.00 on the very same trajectories (Table 53); on our own live suite the per-pair values run 0.00–0.50 with a mean of 0.08. So the sentence we previously wrote — that normalized edit distance "saturates as soon as two models diverge at all" and that the decomposition needs a gracefully-degrading argument metric — was a mis-diagnosis of our own defect. The metric was adequate; our alignment was not.

Second correction: the alignment we replaced it with was also wrong. The paragraph above credits "Needleman–Wunsch on tool names" as the correct implementation. It is not. Matching on the tool name alone leaves many pairings tied whenever a name repeats, and the traceback then chose among them by its own direction. Two trajectories issuing exactly the same two `db` calls in a different order scored argument drift 1.0 — maximal — and the statistic was asymmetric: `[zorin, acme]` against `[zorin]` scored 1.0 while the same against `[acme]` scored 0.0. Making the substitution cost the argument distance itself fixed the asymmetry but not the reordering, because a sequence alignment is order-preserving and cannot represent the crossing. What we now use pairs calls by minimum-cost assignment *within* each tool name, which is the honest consequence of a choice already made one level down for argument keys: emission order is not argument drift, and is measured by tool-selection and action-sequence divergence instead. Six cases are pinned as regression tests (reordering, both subsequence directions, symmetry, a changed value, a field-semantics swap).

This mattered. On the live suite it moved exactly one of six model pairs, but it moved it from 1.0 to 0.0; on the WebArena pairs the aligned+semantic values fell from 0.45–0.65 to 0.26–0.45, and the cross-component maximum $|r|$ from 0.13 to 0.10 (App. AJ), because charging reordered repeated calls as maximal argument drift had been correlating that axis with trajectory drift for a purely mechanical reason. We report this second correction rather than folding it into the first because the two have the same shape — a defect in our own instrument, found only when we tested the instrument on cases we had not thought to construct — and one instance of that would understate the pattern.

What survives, re-derived. The pre-registered verdicts move, and we report the new ones. On the heterogeneous suite **P2 and P3 CONFIRM**: `tool_args` dominates 0 of 4 interfaces (it dominated all four before) and *two* distinct components become dominant. **P1 still refutes there**, median cross-interface cosine 0.940 against the locked 0.90, so E48 lands on its fallback (b) rather than (a). On the live suite, however, both of E39's predictions now **confirm** once the argument alignment is corrected: median cosine $\mathbf{0.870}$ (permutation p=0.001) and *two* distinct dominant components, where the pre-fix run reported 0.918 and a single dominant component. Those two confirmations are **post hoc** — the estimator changed after the data were seen — so the ledger counts them under their locked verdict of REFUTED and reports the flip separately (App. AW); we claim no pre-registration credit for them. The verdicts also differ *between suites*, and we report them that way rather than pooling.

E39's third locked prediction refutes, and its locked fallback decides the claim. E39's protocol locked a *third* prediction that neither this appendix nor App. AR previously reported: P3, that interface-dependence survives **dropping the outcome component** and renormalising over the four process components. We computed it. It **refutes**. With outcome removed the median cross-interface cosine returns to $\mathbf{0.919}$ — back above the locked 0.90 — and `tool_args` once again dominates all four interfaces (1 distinct argmax, not 2). So the interface-dependence that P1 and P2 pick up is carried by the outcome component, which is the success bit the headline κ already measures; the process axes are not doing the work.

E39's protocol locked what to do in exactly this case, and we follow it verbatim: *"(b) If P1 holds but P3 fails: we report the effect as outcome-driven and explicitly decline to claim the process components carry a claim."* We so decline. We also record that P3 had been excluded from the pre-registration ledger as a derived diagnostic rather than counted as the locked prediction it is; it is now a ledger row, reported as refuted (App. AW).

What we conclude, and what we now claim. The demotion **stands**, and we withdraw the stronger reading an earlier revision of this appendix reached on the strength of P1 and P2 alone. What changed is only the *reason*. It was never that a saturated axis drowns the profile — that was a bug, twice over, and we were wrong about the cause for one revision cycle. With a correctly aligned argument axis the full five-component profile does separate protocol forms on the live suite in degree ($0.870 < 0.90$) and in rank (2 distinct dominant components), but P3 shows that separation is outcome churn in disguise: strip the outcome component and both signals vanish (0.919, 1 distinct argmax). On the heterogeneous-argument suite the profile does not separate them in degree at all (0.940), though `tool_args` stops being universal.

So the claim we make is the narrow one the protocol's fallback licenses: the decomposition is a **descriptive** instrument, validated against known ground truth on the simulator, whose *process* axes we do not claim discriminate between protocol forms on either suite. No current claim of this paper rests on the five-component profile (Limitation T6); the current claims rest on the interaction share, the oracle gain and the selection error (Sections 3–6). We flag that we reported a negative result for the wrong reason for one revision cycle, then briefly reported a positive one for an incomplete reason, and that an external reviewer, not our own checks, is what surfaced it. The pre-registered prediction that settles it is the one we had not been reporting.

I DOES MODEL CAPABILITY CONFER HARNESS-ROBUSTNESS? (E34, AND A CORRECTION)

What this design is for now. This appendix specifies the strong-model design of the main text (*our suite, strong models*). In the current argument it is the clearest case of interaction without opportunity: every model is best on `json`, so $\widehat{G}$ is zero (Table 2), and it is a case of limited adaptation opportunity (Table 3). The rest of this appendix records why the design was run: to test a capability explanation of κ that the paper no longer makes.

Why it was run. Our earlier reading of κ—large on our interface-sensitive suite, near zero on SWE-bench coding scaffolds—implicitly attributed the difference to *model capability*. That inference was confounded: the two settings differ in model strength *and* in task/scaffold regime simul-

taneously, because one is our study and the other is public leaderboard data. We pre-registered the capability explanation with a locked prediction ($\kappa < 0.30$ for strong models) and two locked fallbacks, then tested it by holding the suite and both interfaces *fixed* and changing only model strength. We then subjected the result to the uncertainty analysis it should have had from the start, which changed what we conclude; both stages are reported here.

Design. The identical 52-task suite and the identical JSON/ReAct interfaces as E9, with four models at complete 52×2 coverage. Marginal API cost $\sim$\$0.03–0.09.

model	JSON	ReAct	contrast d	κ if dropped
`deepseek-chat` (V3.1)	0.942	0.808	+0.135	0.441
`hermes-3-70b`	0.827	**0.019**	**+0.808**	**0.097**
`mistral-large`	0.923	0.750	+0.173	0.421
`mistral-medium`	0.808	0.712	+0.096	0.363

Point estimate, then uncertainty. $\kappa = 0.402$ ($\widetilde{\kappa} = 0.399$) against 0.466 for the small/mid E9 pool—a fall of only 0.064, so the locked $\kappa < 0.30$ did not occur, and the saturation control passes (cells span 0.019–0.942). We initially read that as refuting the capability explanation. Three checks show that reading was too strong.

(i) The interaction is carried by one model. The rightmost column above is leave-one-model-out: dropping `hermes-3-70b` collapses κ from 0.402 to 0.097, while dropping any other model leaves it at 0.36–0.44. The other three strong models have harness contrasts of only +0.10 to +0.17.

(ii) Against the correct null the four-model effect is real; the three-model one is not. One test must first be discarded: the within-model permutation returns p=1.000, an artifact rather than a finding. At b=2, permuting harness labels within a model merely flips the sign of that model's harness contrast; all four contrasts share a sign (every strong model prefers JSON), so sign-flipping *increases* their spread and places the null above the observed value by construction. **This test is invalid for a two-level design with same-sign contrasts** and we do not report it. Against the mean-matched no-interaction null used for the headline (App. S), the four-model κ is far outside (0.034 ± 0.028 null, z=13.0, $p < 10^{-4}$); excluding `hermes-3-70b`, the remaining three give 0.097 against a null of 0.253 ± 0.217 (p=0.71)—indistinguishable from no interaction.

(iii) The interval spans the locked threshold. The task-bootstrap 95% CI on κ is $[0.275, 0.518]$, which contains 0.30. The point estimate did not fall as predicted, but the data cannot separate it from the prediction, and it does not reach 0.40 either. We also disclose that the model-inclusion rule (minimum per-harness task coverage) was a post-hoc bug fix rather than a pre-registered choice, and that it moves the verdict: at a threshold of 20 tasks the design admits five models on 18 shared tasks and gives κ=0.266, below the locked boundary. The 52-task design is the defensible one and we keep it, but the sensitivity is real.

The mechanism is protocol adherence, not tool-use skill. Inspecting the cell that carries it, `hermes-3-70b` under ReAct *never emitted a parsed tool call in* 38 *of* 52 *tasks* (under JSON, 0 of 52). The raw responses show it emitting an entire trajectory in a single turn—several `Action:` lines followed by an `Answer:`—so the episode terminates on an unsupported answer before any tool runs. Our ReAct level uses no stop sequence, whereas a production ReAct scaffold halts generation at `Observation:` for exactly this reason (Limitation T4). Fabricated answers do not inflate success, however: across all harnesses only 4 of 327 no-tool-call trajectories succeeded, and 1 of 38 here.

What we conclude. Capability does not confer harness-robustness. Three of four strong models are harness-robust, but one strong 70B model is the most harness-brittle model anywhere in this work, because it cannot hold to one protocol—so a stronger incoming model is no substitute for per-scaffold validation. What we do *not* claim is that κ is invariant to model strength: with four models the interval $[0.275, 0.518]$ spans both the small/mid 0.466 and the locked 0.30. The capability explanation of the SWE-bench contrast is therefore *unsupported* rather than refuted, and we assert no capability-based regime distinction anywhere in the paper.

Why this appendix records its own correction. E34 was first written up from a point estimate with no interval and no leave-one-out, and the conclusion drawn from it was stronger than the design supports. That is the second time in this work an effect was reported before those two checks were run—the first was the brittleness AUC of App. AM. A point estimate that trips a pre-registered fallback still needs uncertainty attached before the fallback is acted upon.

J THE BFCL DESIGN: AN EXTERNAL BENCHMARK WITH ALMOST NO INTERACTION (E51)

The most obvious objection to this paper is that the task suite is ours. So we pre-registered a design on an external, widely used benchmark: BFCL v3 multi-turn (Patil et al., 2025), run as 7 models $\times$ 4 protocol forms $\times$ 40 tasks $=$ 1,120 cells. The corrected interaction estimate is near zero there. In the current argument this design is a case of limited adaptation opportunity (Table 3).

Design. We use BFCL's `multi_turn_base` split with gold-history teacher forcing: at turn i the model sees the gold calls for turns $0..i-1$, so a derailment at one turn does not contaminate the rest and every cell is scored from the same states. BFCL's own harness supplies the full function documentation, so we do too, *identically for all four forms*; only the syntax of the reply differs. Scoring matches the emitted call set against gold on normalised (name, arguments).

Validity gate. The pre-registered primary endpoint, whole-task success, **fails** its gate: pooled success is 0.0179, far under the $[0.10, 0.90]$ band, because these tasks require every turn of a multi-step episode to be exactly right. The secondary endpoint, per-turn accuracy, passes at 0.3533 with a 0.0214 no-parse rate, and carries κ. We report both rather than quietly promoting the one that worked.

Result: refuted.

	per-turn accuracy by form					
model	json	react	xml	pythonic	best	margin
`gpt-oss-120b`	0.347	0.337	0.346	0.326	`json`	0.001
`gpt-oss-20b`	0.300	0.280	0.155	0.235	`json`	0.021
`llama-3.1-8b`	0.352	0.386	0.315	0.385	`react`	0.001
`llama-3.3-70b`	0.314	0.360	0.312	0.358	`react`	0.002
`mistral-small-24b`	0.408	0.407	0.376	0.396	`json`	0.001
`phi-4`	0.422	0.425	0.422	0.448	`pythonic`	0.023
`qwen3-14b`	0.353	0.385	0.373	0.368	`react`	0.012

The bias-corrected κ is $\mathbf{0.00}$, bootstrap CI $[0.00, 0.13]$, against a mean-matched no-interaction null of 0.0447: p=1.00, $z = -0.62$. The estimate sits *below* its own null floor. The variance shares say it plainly — $S_{\mathcal{M}}$=0.0491, $S_{\mathcal{H}}$=0.0037, $S_{\mathcal{MH}}$=0.0081 — model identity matters about twelve times more than protocol form, and the interaction is negligible.

P3 ("at least two distinct best forms") reads CONFIRMED with four distinct argmaxes, and we decline to count it. With $S_{\mathcal{MH}}$ at 0.0081 the per-model best form is decided by margins in the third decimal, and App. AA showed that this is precisely the regime where argmax is tie-noise. A prediction that confirms itself out of noise is not evidence.

What the near-zero estimate does and does not say. The corrected interaction estimate is near zero in this BFCL design. Because the specification trend did not replicate consistently across settings (App. V), this result does not isolate schema specification as its cause.

What it cost to believe this, and why the number is not the flattering one. Three grids were collected here and two were discarded, both for asymmetries we had built into our own prompt, and both of which produced a *better* result than the truth:

1. **Gold history in the wrong syntax.** The teacher-forced history was rendered as Python calls under all four forms. That matches `pythonic` exactly, resembles `react`, and is

foreign to `json` and `xml`. It gave κ=0.477 — an almost exact replication of our 0.466 — and it was an artifact of the prompt. History is now rendered in the same form the model must reply in.

2. **An XML exemplar that taught the wrong thing.** `SYNTAX["xml"]` was `<call name="fn" k="v"/>`: the only one of the four shapes in which a *literal* attribute (`name`) and a *placeholder* (`k`) share a slot, with nothing marking which is which. Models copied it. 257 of 1257 XML replies (20.4%) emitted a literal `k=` or `v=` attribute, e.g. `<call name="get_stock_info" k="Zeta Corp"/>`, which can never match gold — at a strongly model-dependent rate (`phi-4` 27%, `gpt-oss-120b` 0%), so it landed straight in the interaction term. All four exemplars now use identical `FUNCTION`/`PARAM`/`VALUE` placeholders — and that repair was also incomplete. We verified it by counting the *old* symptom (literal `k=`/`v=`: 0 of 1028) rather than the failure mode, and the failure mode survived. A placeholder is copyable however it is spelled. The exemplars are now a *concrete* call to a function in no BFCL task (`set_volume(level="7")`), the convention this paper's own live suite has always used (`db("Zorin")`; its headline grid copies at 0.469%). Measured over each grid's own replies:

exemplar style	json	react	xml	pythonic
v1 `fn(k="v")`	—	—	**20.4%**	—
v2 `FUNCTION(PARAM="VALUE")`	0.00%	0.76%	7.30%	3.88%
v3 `set_volume(level="7")`	0.00%	0.00%	2.25%	0.00%

We locked "below 1% in every form" before collecting v3 and **that prediction is refuted**: three of four forms reach exactly 0.00%, but XML sits at 2.25%, every case a model copying `level="7"` into its own tag. The pattern across three unrelated exemplar styles is that XML's *attribute slot* is intrinsically exemplar-prone in a way that nested-JSON and keyword-argument forms are not, and no spelling of the example removes it.

This does not change the verdict, and two things establish that. First, direction: a copying penalty falls on XML and varies by model, so it manufactures model×form variance and can only *inflate* κ, which is already 0.00 and below its own null. Second, replication: κ was measured at **0.00** under *both* the 7.30% instrument and the 2.25% one, so the null does not depend on the residual.

A third defect was in the analysis, not the data. The locked P2 test read "bootstrap CI excludes 0" on the *raw* η^2 ratio. Sums of squares are non-negative, so that statistic cannot reach 0: simulated at this exact design with *no interaction present* it has mean 0.446 and 2.5th percentile 0.206, and returns CONFIRMED on 12/12 pure-null draws. It is not a test. On this grid it reads κ=0.141, CI $[0.098, 0.290]$ — the discarded test would have reported a successful replication on a real benchmark. P2 now requires the bias-corrected κ to beat a mean-matched no-interaction null at $p < 0.05$, the standard the rest of this paper already uses (App. S); it returns CONFIRMED on 1/12 pure-null, 8/12 weak and 12/12 strong simulated interactions. The original test is recorded as VOID rather than deleted, and the amendment was committed before the third grid was collected.

We report this history because two of the three defects surfaced only under an adversarial audit of our own instrument, and because both produced favourable numbers. That is a bias in how we build instruments, not two coincidences, and a reader weighing our other κ estimates deserves to know it.

K Does measuring the incoming model beat a strong fixed default on tasks we did not write? (E120)

A reviewer asked two questions about significance. First: what realistic evaluation or deployment decision changes because of κ, beyond the direct success, no-call and reversal measurements? Second: does a prospectively specified probe policy improve a meaningful endpoint over a strong fixed default, at a fair measured cost, on an independent task setting? What the paper had: on our own 52 arithmetic tasks, probing the incoming model beat keeping the incumbent's format, but the comparator was never a strong fixed default; on the frozen external grid, probing showed no advantage (App. V); the seven-design synthesis correlates κ with what probing saves at 0.18 and the reversal rate at 0.61, both intervals covering zero (App. AB.3); and cost had been counted in episodes, never

in provider calls or dollars. This experiment runs the comparison the reviewer described, once, on a public workplace-agent benchmark we did not build, with every rule, split, threshold and comparator fixed in the protocol (`experiments/120_workbench_decision/protocol.md`, committed alone and first, sha256 `0aaca1d5...`) and every departure from its literal text recorded in a dated addendum written before any smoke episode or grid cell existed. A positive result was not required. The analyzer was committed before the first cell and run once.

The decision, in plain words. A team runs model A as a tool-using agent on one call format. They replace A with model B. B must run on one format. They can (a) keep A's format, (b) use a fixed default, (c) spend a probe, running B on a few tasks in every format and picking the best, or (d) let a statistic on data they already have decide whether to spend the probe. Every option is priced and every option is scored on the same held-out tasks.

K.1 The family, and why it is independent

WorkBench (Styles et al., 2024) is a public benchmark of workplace tasks over five sandbox databases (calendar, email, analytics, project management, CRM), MIT-licensed, pinned at commit `49c7dfd0`, ground truth v2. It has 690 tasks: 69 base templates $\times$ exactly 10 instances, the 10 instances of a template being the same instruction with different names and dates. We wrote no task and dropped none. 568 tasks have at least one gold write action (*action-required*); 122 are the false branch of a conditional template ("if X, do Y") with an empty gold action list (*no-op*), sitting in 31 templates that also hold action-required instances. The primary success endpoint is scored on the 568, because a form under which a model emits no parseable call scores every no-op task correct (a silent agent scores 0.177 on all 690 and 0 on the 568). The harm endpoint is scored on all 690, because acting on a no-op task is exactly the harm a team cares about.

The oracle is WorkBench's own. `is_correct` executes the predicted and the gold action lists on fresh sandboxes and compares final database states; it returns False when the episode carries an error, including WorkBench's own stop message when the step cap is hit, so a stopped episode is incorrect by its authors' rule. `has_side_effects` is True when the state changed and the result is not correct; the cells store it in WorkBench's own `compute_metrics` construction (addendum item 7). The model's text is never scored. We wrote no scorer. Tools are WorkBench's 27, dispatched per domain by their own rule plus `company_directory`; one disclosed repair maps the domain name `crm`, which 50 multi-domain tasks use and their toolkit map does not know, to the CRM toolkit. Nothing the model writes is executed as code.

K.2 Six models, and one substitution

Selection rules, fixed before any WorkBench episode of ours existed: a paid OpenRouter route, a list price on the public model list of 15 September 2026, a context window of at least 100k tokens, and not a thinking-tier model. Within that pool, two models from each of three tiers a team in 2026 might swap between: frontier-lab small (`gpt-4o-mini`, `claude-haiku-4.5`, `gpt-5.6-luna`), open-weight dense (`qwen3-32b`, `llama-3.3-70b`) and open-weight MoE (`gpt-oss-120b`). Reserves, in order: `deepseek-chat`, `gemma-3-27b`, `nova-lite`. The locked reserve rule: a reserve replaces a primary model only when the outcome-blind smoke test shows an HTTP error or an empty reply on at least 4 of that model's 48 smoke episodes; a model is never replaced for parsing or success behaviour.

The rule fired once. `gpt-oss-120b` returned an empty reply on 5 of its 48 smoke episodes (the smoke cells, kept outside the repository, record finish reason `tool_calls`: the model attempted the provider's native tool call instead of the text format the prompt asked for). The locked wording counts an empty reply as unavailability, so the first reserve, `deepseek-chat` (`deepseek/deepseek-chat-v3.1`), replaced it; the substitution was written to `config.json` before the grid started and the smoke was re-run for the new model set with zero API errors. We note the consequence: the substitution removed the one headline-grid model besides `llama-3.3-70b`, and it removed it for a behaviour (reaching for native tool calling) that is itself a harness effect. The six models that ran are in Table 11. Prices were re-read from the same public list on the collection day and were unchanged from the 15 September list.

Table 11: The six models, their per-form success on the 568 action-required tasks, their held-out-best form, their pooled success and no-parsed-call rate, and their measured cost per episode. Every number is a mean over the model's 2,760 (form, task) cells or its 568×4 action-required cells. `json` is the best form for five of six; `gpt-4o-mini`'s best is `pythonic`, by 0.005.

	success by form					pooled	no	per episode		
model	`json`	`react`	`xml`	`pythonic`	best	success	call	calls	tokens	USD
`gpt-4o-mini`	0.521	0.512	0.380	0.526	`pythonic`	0.485	0.000	4.25	6,831	0.0007
`claude-haiku-4.5`	0.775	0.634	0.398	0.673	`json`	0.620	0.001	5.61	11,690	0.0130
`gpt-5.6-luna`	0.717	0.467	0.354	0.255	`json`	0.448	0.092	5.91	12,550	0.0031
`qwen3-32b`	0.590	0.502	0.284	0.585	`json`	0.490	0.000	4.61	11,085	0.0022
`llama-3.3-70b`	0.463	0.329	0.245	0.354	`json`	0.348	0.000	4.66	7,512	0.0010
`deepseek-chat`	0.618	0.609	0.340	0.606	`json`	0.543	0.000	5.33	9,094	0.0016
pooled	0.614	0.509	0.333	0.500		0.489	0.016			

K.3 Four forms and a form-neutral tool block

The harness factor is the four free-text forms of the headline grid, `json`, `react`, `xml`, `pythonic`, in that order (ties in any argmax go to the first). No schema-constrained form: the reviewer's question was about probing free-text formats, and a constrained rung adds a failure mode (provider schema support) this study is not about. The four prompts share, byte for byte, everything but the syntax block: WorkBench's date prefix verbatim; a rules block (one tool call per reply and nothing else, WorkBench's own act-without-confirmation sentences verbatim, reply with the single word `DONE` when finished, the step cap); and a **form-neutral tool block**, for each tool its signature line, the first line of its docstring and one line per parameter. WorkBench's own tool renderer is not used, because every one of the 27 docstrings ends in an Examples section of positional Python calls, which would show the `pythonic` form's syntax under all four forms. The compact block is 11,737 characters for all 27 tools (3,368 per task on average under the per-domain rule; the protocol quoted a design-stage count of 11,990 and 3,441, addendum item 5). The syntax block gives one worked example per form using `set_volume`, a tool that does not exist, plus the form's carrier for a multi-line value. Tool names are shown with underscores (`calendar_delete_event`), WorkBench's own substitution for native tool calling, and mapped back for execution and scoring.

The parser is the headline grid's `parse_call` plus one change: in the `xml` branch any child element `<k>v</k>` carries the argument `k`, because WorkBench's multi-line argument is `body`. Two episode-loop changes were made before any cell and are logged (addendum items 8 and 9): a parsed call named `DONE` with no arguments ends the episode under every form, exactly as the bare word does, since `DONE()` passed the finish regex under `pythonic` while `{"tool": "DONE"}` did not under `json`; and an unknown tool name is recorded with dots replaced by underscores so WorkBench's dispatcher refuses at scoring exactly the calls the episode refused. Fixed settings: step cap 10, `max_tokens` 4,000, temperature 0, seed 120 in every cache key, observation cap 6,000 characters, 8 workers. A cell is written only when every call in its episode succeeded and is never overwritten.

K.4 Gate X0, the smoke test and gate X1

X0, mechanical, before collection. (a) WorkBench's `is_correct(gold, gold)` is True for 690 of 690 gold lists. (b) A renderer turns each task's gold action list into each of the four forms with exactly the syntax the prompt shows; the port parses each rendered call; the list is converted with WorkBench's own recorder and scored against gold: 2,760 of 2,760 lists score correct and 3,788 of 3,788 calls parse back to the identical (tool, kwargs), including the 40 with a newline in an argument. The parser revision counter reads 1 with 0 failure-driven revisions: revision 1 removed two changes the protocol does not allow (an escape table, XML entity decoding) and X0(b) was 2,760 of 2,760 before and after (addendum item 4). (c) The smoke test ended with zero API errors on every (model, form) after the substitution and WorkBench's state executor raised on none of the 288 smoke episodes. **PASSED.**

The smoke test (12 tasks $\times$ 6 models $\times$ 4 forms, seed 999, cells outside the repository, no outcome field read) drives three locked rules. Rule 1, parser repair if any form's no-parsed-call share exceeds 0.15: the shares were 0.000, 0.000, 0.000 and 0.042 (`pythonic`), so no repair. Rule 2, raise the

token budget to 8,000 if any model hits `length` on more than 0.20 of its episodes: every model read 0.000, so no raise. Rule 3, the budget ladder: the projected grid cost was USD 32.97 and 12.0 hours at k=10 instances per template and 8 workers, under the USD 120 and 20-hour lines, so k stayed at 10; the re-run for the substituted model set projected USD 34.96. The measured total came to USD 60.01, above the projection: a grid episode cost more than a smoke episode of the same model (USD 0.0130 against 0.0069 for `claude-haiku-4.5`), and the halt line of USD 140 measured was never approached.

X1, the validity gate. Pooled `correct` over the 568 action-required tasks × 6 models × 4 forms (13,632 episodes) is 0.489, inside $[0.10, 0.90]$, and the no-parsed-call rate over the same episodes is 0.016, below 0.15; over all 690 tasks the two read 0.476 and 0.016. **PASSED.** Per model and per form the numbers are in Table 11; the one cell with a high no-call rate is `gpt-5.6-luna` under `pythonic`, 0.339, where the model also writes no side-effect call in 0.667 of episodes and hits the step cap in 0.349. That cell is the largest interaction in the cube, and it is a cell a fixed `json` default never visits.

Completeness, provenance, re-score. 16,560 of 16,560 cells exist (69 of 69 templates complete at k=10; label NORMAL; 0 dropped). One pass wrote every cell between 2026-09-17 20:22 and 2026-09-18 14:07 UTC with 0 API-error skips, 0 halts and 0 scoring errors. Every cell carries the same `config.json` sha256 and the same port digest, and the port's source is unchanged since collection. Provider-reported usage is present on 1.000 of cells, so the USD column is labelled MEASURED. The oracle re-score re-ran WorkBench's pinned `is_correct`, `has_side_effects` and `is_exact_match` on every cell's stored action strings: 0 mismatches on 16,560 cells in all four fields.

K.5 Splits over templates, the eleven rules, and their costs

Splits are drawn over templates, not tasks, because a task-level split would let the probe see near-copies of held-out tasks. Each of 2,000 draws permutes the 69 template ids: the first 10 are **history**, the next 10 **probe**, the remaining 49 **held-out**. History is every action-required task of the history templates (about 82) for every model on every form: what a team already has from models it has deployed, sunk cost. Probe is exactly 10 tasks, one action-required task drawn from each probe template, run by B on all four forms: 40 episodes, the paper's convention. Held-out is every action-required task of the 49 held-out templates (about 403). Every rule is scored on B's held-out cells and nothing else; **regret** is B's best held-out form mean minus B's mean on the chosen form, and zero is perfect. The uncertainty unit is the template: every interval is a template-cluster bootstrap, 500 resamples of the 69 template ids with replacement (a template drawn twice enters twice, relabelled), 40 partition draws inside each, 2.5th and 97.5th percentiles of the named paired difference. The model set is fixed at six.

The eleven rules and what each costs per swap, beyond the sunk history: `keep_incumbent_format`, A's best form on A's history cells (0); `fixed_json`, always `json`, the data-free default chosen from the seven gate-passing designs' pooled form means and WorkBench's own JSON-blob agent (0); `fixed_hist_pool`, **the strong fixed default**, the form with the best pooled history mean over the five models other than B (0); `best_fixed_mean`, the house rule, the pooled best form over all six models' probe cells (40 episodes of B, because it reads B's probe); `probe_argmax`, B's best form on the 10 probe tasks (40 episodes); `probe_large`, B's best form on every action-required task of the 10 probe templates (about 330 episodes); `kappa_gate`, probe iff the EMS-corrected $\widetilde{\kappa}$ of the five-model history cube is above its median over all 12,000 (draw, B) instances, else `fixed_hist_pool` (40 when it probes); `reversal_gate`, the same gated on the reversal rate of the same cube; `random_gate`, probe with probability equal to the κ gate's realised rate, a seeded coin; `kappa_gate_0.10`, the deployable rule, probe iff $\widetilde{\kappa} \geq 0.10$; and `oracle`, B's best held-out form, a floor and not a rule. Cost is counted from the cells in four units: episodes of B, provider calls (from the per-step records), tokens (from the provider's `usage` object) and USD (provider-reported). The one-off calibration the fixed default and the gates need, the five deployed models' action-required history cells, is 1,652 episodes, 8,417 calls and USD 6.03, counted once and not per swap. Break-even for a swap is $N_B^* = 40/(\text{regret}_{\text{fixed_hist_pool}}(B) - \text{regret}_{\text{probe}}(B))$ deployment episodes, infinite when the denominator is not positive, one extra success worth one episode.

K.6 THE RULE TABLE ON SUCCESS, AND ON HARM

Table 12 is the full version of Table 16: every rule with its cost in episodes, calls and USD, and the differences between rules with the coverage their intervals had in simulation.

Table 12: **WorkBench: the full version of Table 16.** Every rule, with the gates, the oracle floor and the differences between rules. Held-out regret of the incoming model, 30 swaps $\times$ 2,000 template splits, with a template-bootstrap percentile interval (descriptive; per-rule coverage not measured). Harm: the chosen format's rate of unwanted side effects on held-out tasks of all kinds. Episodes, calls and USD per swap. The pooled default and the κ rules also use the other models' history, 1,652 episodes and USD 6.03 paid once. Bottom: differences in regret, the interval and its coverage measured in simulation (§K.12); a grouped resampler moves two intervals by more than 0.005, to grouped $[0.005,\ 0.039]$ (inherit − probe) and grouped $[-0.028,\ 0.005]$ (pooled − probe).

rule	regret	interval	harm	ep.	calls	USD
inherit	0.040	[0.030, 0.062]	0.286	0	0	0
pooled default	0.005	[0.001, 0.023]	0.284	0	0	0
fixed `json`	0.003	[0.001, 0.013]	0.285	0	0	0
10-task probe	0.016	[0.012, 0.026]	0.284	40	205	0.146
probe, all probe tasks	0.010	[0.007, 0.015]	0.279	330	1,681	1.19
κ rule	0.013	[0.007, 0.029]	0.285	20	100	0.086
reversal rule	0.012	[0.007, 0.027]	0.285	20	102	0.088
random rule	0.011	[0.007, 0.024]	0.284	20	102	0.072
probe if $\widetilde{\kappa} \geq 0.10$	0.017	[0.012, 0.029]	0.284	36	185	0.135
oracle (floor)	0		0.275	0	0	0

difference		interval	its coverage
inherit − pooled	+0.034	[0.024, 0.047]	0.62–0.80
inherit − probe	+0.023	[0.012, 0.043]	0.64–0.94
pooled − probe	−0.011	[−0.019, 0.004]	0.58–0.74
κ − random	+0.002	[−0.003, 0.009]	0.88–1.00
κ − reversal	+0.000	[−0.003, 0.005]	not measured; descriptive

Table 13 is the full rule table on the primary endpoint. Three readings. First, the rules with no per-swap probe split: inheriting A's format costs 0.040, a fixed default pooled from the other five models' history costs 0.005, and always-`json` costs 0.003. Second, every probe of B sits between them: 0.022 at 3 tasks per form, 0.020 at 5, 0.016 at 10, 0.010 at every task of the probe templates (330 episodes, USD 1.19 per swap). Third, the three gates at a matched probe rate of one half show no detected advantage over one another at this precision, 0.013, 0.012 and 0.011 for the κ, reversal and random gate, and the deployable κ rule probes 91% of the time, so it costs 0.135 of the probe's 0.146 and lands at the probe's regret, 0.017. The chosen-form shares say why a default with no per-swap probe does well here: `fixed_hist_pool` picks `json` in 0.994 of instances, the oracle picks it in 0.788, and a 10-task probe picks it in 0.744 while spending the rest of its choices on the three forms that are best for no one but `gpt-4o-mini`.

Table 14 repeats every rule on the harm endpoint, `unwanted_side_effects` on all 690 tasks, scored on the same 2,000 draws (addendum item 11), where regret is B's lowest held-out harm rate minus the harm rate of the chosen form. The analyzer bootstraps the named paired differences, not the harm table's rows, so the harm rows carry split percentiles across draws and are labelled as split sensitivity. Two things to read. Harm is a different decision: the oracle on harm picks `xml` in 0.460 of instances, a form that is best on success for no model: acting less is safer, and it is also failing more. And the observed harm difference between the form the success probe chooses and the fixed default's is small: the held-out harm rate of `probe_argmax`'s success-chosen form minus `fixed_hist_pool`'s is −0.0002, template interval $[-0.007,\ 0.004]$. X8's locked verdict is confirmed, because that upper end is below the 0.02 the protocol set. The interval's coverage was not measured (§K.12), so we do not read its upper end as a bound, and no non-inferiority is established.

Per incoming model, and break-even. Table 15 gives each rule's regret when the named model is the incoming B, averaged over its five predecessors and the draws, and the break-even count. For five of the six models the fixed default's regret is at or below the probe's, so the denominator is not positive and no number of deployment episodes repays the probe. For `gpt-5.6-luna`, the one model where the pooled default sometimes misses (`json` is its best form by 0.25 over `react`, and the other five models' pooled history picks `json` in most but not all draws), the probe saves 0.004

Table 13: E120, primary endpoint (`correct` on the 568 action-required tasks): held-out regret of each rule, averaged over the 30 ordered swaps and 2,000 template draws; template-bootstrap 95% percentile interval (500 resamples × 40 draws; planning coverage in §K.11); the held-out harm rate of the form each rule chose; marginal cost per swap in episodes of B, provider calls and provider-reported USD; the share of instances in which the rule chose `json`/`react`/`xml`/`pythonic`; and, for the gates, the realised probe rate. In every E120 table a cell is the result file's four decimals rounded half-up to the printed precision.

rule	regret	95% interval	harm	episodes	calls	USD	chosen form (j/r/x/p)	probes
keep A's format	0.040	[0.030, 0.062]	0.286	0	0	0.000	0.73/0.07/0.01/0.19	
fixed `json`	0.003	[0.001, 0.013]	0.285	0	0	0.000	1.00/0.00/0.00/0.00	
fixed default, pooled history	**0.005**	[0.001, 0.023]	0.284	0	0	0.000	0.99/0.00/0.00/0.01	
pooled best probe form	0.007	[0.001, 0.021]	0.286	40	205	0.146	0.97/0.02/0.00/0.01	
probe, 3 tasks per form	0.022	[0.015, 0.034]	0.285	12	62	0.044	0.81/0.10/0.03/0.06	
probe, 5 tasks per form	0.020	[0.014, 0.033]	0.285	20	103	0.073	0.77/0.12/0.03/0.09	
probe, 10 tasks per form	**0.016**	[0.012, 0.026]	0.284	40	205	0.146	0.74/0.12/0.01/0.12	
probe, all probe-template tasks	0.010	[0.007, 0.015]	0.279	330	1681	1.193	0.72/0.08/0.01/0.19	
κ gate, median	0.013	[0.007, 0.029]	0.285	20	100	0.086	0.87/0.06/0.01/0.07	0.50
reversal gate, median	0.012	[0.007, 0.027]	0.285	20	102	0.088	0.87/0.05/0.01/0.07	0.50
random gate, matched rate	0.011	[0.007, 0.024]	0.284	20	102	0.072	0.87/0.06/0.01/0.06	0.50
κ gate at 0.10	0.017	[0.012, 0.029]	0.284	36	185	0.135	0.75/0.12/0.01/0.12	0.91
oracle (floor)	0.000	[0.000, 0.000]	0.275	0	0	0.000	0.79/0.03/0.00/0.18	

Table 14: E120, secondary endpoint (harm: `unwanted_side_effects` on all 690 tasks), the same rules on the same draws. Regret is the lowest held-out harm rate minus the chosen form's; the bracket is the 2.5th and 97.5th percentile across the 2,000 draws (split sensitivity, not coverage); harm is the chosen form's held-out harm rate. The probe over all probe-template tasks runs 400 episodes here because no-op tasks are in the harm cube.

rule	regret	split pct.	harm	episodes	USD	chosen form (j/r/x/p)	probes
keep A's format	0.067	[0.049, 0.084]	0.296	0	0.000	0.29/0.06/0.34/0.32	
fixed `json`	0.056	[0.035, 0.080]	0.285	0	0.000	1.00/0.00/0.00/0.00	
fixed default, pooled history	0.066	[0.032, 0.098]	0.295	0	0.000	0.22/0.00/0.29/0.49	
pooled best probe form	0.056	[0.026, 0.100]	0.285	40	0.146	0.38/0.04/0.19/0.39	
probe, 3 tasks per form	0.052	[0.015, 0.090]	0.281	12	0.044	0.64/0.11/0.18/0.07	
probe, 5 tasks per form	0.048	[0.009, 0.089]	0.277	20	0.073	0.55/0.13/0.21/0.12	
probe, 10 tasks per form	0.040	[0.005, 0.082]	0.269	40	0.146	0.46/0.13/0.22/0.19	
probe, all probe-template tasks	0.022	[0.000, 0.061]	0.251	400	1.443	0.30/0.06/0.33/0.32	
κ gate, median	0.054	[0.010, 0.093]	0.283	20	0.063	0.32/0.07/0.25/0.36	0.50
reversal gate, median	0.054	[0.011, 0.095]	0.283	20	0.062	0.32/0.07/0.25/0.36	0.50
random gate, matched rate	0.053	[0.014, 0.095]	0.282	20	0.073	0.34/0.06/0.25/0.34	0.50
κ gate at 0.10	0.045	[0.006, 0.090]	0.274	34	0.115	0.41/0.11/0.23/0.25	0.84
oracle (floor)	0.000	[0.000, 0.000]	0.229	0	0.000	0.31/0.00/0.46/0.23	

and repays its 40 episodes after 9,710 deployment episodes. X4 asked for $N_B^* \leq 2{,}000$ on at least 4 of 6 models and was refuted on 0 of 6.

Table 15: Held-out regret on success by incoming model B, and the break-even count N_B^* (deployment episodes at which a 40-episode probe repays itself against the pooled-history default; ∞ when the default is at least as good).

incoming model B	keep A's	fixed `json`	pooled default	pooled probe	probe 10	probe all	N_B^*
`gpt-4o-mini`	0.008	0.009	0.009	0.009	0.020	0.022	∞
`claude-haiku-4.5`	0.038	0.000	0.000	0.005	0.017	0.006	∞
`gpt-5.6-luna`	0.130	0.000	0.016	0.011	0.012	0.000	9,710
`qwen3-32b`	0.015	0.004	0.004	0.006	0.020	0.010	∞
`llama-3.3-70b`	0.037	0.000	0.000	0.005	0.016	0.009	∞
`deepseek-chat`	0.009	0.003	0.003	0.004	0.011	0.015	∞

The regret-against-cost curve. Fig. 4 plots the rules at their USD cost per swap (two are not drawn: the reversal gate sits at the κ gate's point, and the pooled best probe form is a house rule the main text does not discuss). The figure is drawn by `experiments/review_20260924_hd/make_writer_outputs.py` from the same result file with the main text's rule names (inherit, pooled default, κ rule); `experiments/120_workbench_decision/fig_e120.py` draws the same points with the

Table 16: **WorkBench: regret and cost.** Held-out regret in success on the 568 tasks that need an action, 30 swaps × 2,000 template splits, with a template-bootstrap percentile interval (descriptive). Harm: the chosen format's rate of unwanted side effects (the 690-task endpoint). Episodes, calls and USD per swap; the pooled-history rule also reads the history once (1,652 episodes, USD 6.03). The 10-task probe repays its cost against that rule for one of six incoming models, after 9,710 episodes of use (point estimates).

rule	regret	interval	harm	ep.	calls	USD
inherit	0.040	[0.030, 0.062]	0.286	0	0	0
pooled history	0.005	[0.001, 0.023]	0.284	0	0	0
fixed `json`	0.003	[0.001, 0.013]	0.285	0	0	0
10-task probe	0.016	[0.012, 0.026]	0.284	40	205	0.146
probe, all probe tasks	0.010	[0.007, 0.015]	0.279	330	1,681	1.19

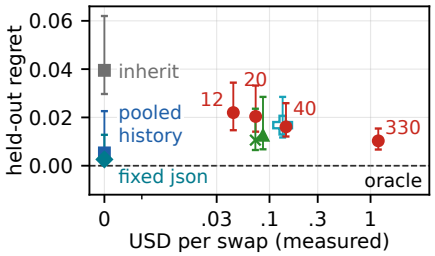

Figure 4: **Regret against cost** (USD per swap). Both defaults sit lowest, at no probing cost. Bars: percentile ranges. Circles: probes; others: rules that decide whether to probe (App. K); dashed: oracle.

protocol's names. Reading left to right: the rules at cost zero span 0.003 to 0.040 depending on what they know; the probe falls from 0.022 at 12 episodes (USD 0.044) through 0.020 at 20 (USD 0.073) and 0.016 at 40 (USD 0.146) to 0.010 at 330 (USD 1.19), and at no point on that curve does it reach the pooled default's 0.005. A 40-episode probe of four forms on ten tasks has a selection noise of its own, and on this family that noise costs more than the probe finds. The protocol's precision simulation gives this shape only under no interaction (-0.014); at the observed $\widetilde{\kappa}$ it predicted $+0.07$, and §K.11 says why the mapping failed here.

K.7 The gates

Each gate's threshold is the median of its statistic over the 12,000 (draw, B) instances, so each gate probes on one half of them and the random gate is a fair coin at the same rate; this is a within-design ranking test, separate from how often one probes. The thresholds came out at $\tau_\kappa = 0.343$ and $\tau_r = 0.50$. The tie rule (addendum item 1): the reversal rate of a five-model cube takes ten discrete values, so "at or above the median" would have fired the reversal gate on 0.650 of instances (0.211 of them tied at the median); instead every instance strictly above the median fires, none strictly below, and among the tied instances a seeded draw fires as many as bring the total to exactly one half. $\widetilde{\kappa}_{\text{hist}}$ had no ties at its median. Its distribution over the instances: 5th percentile 0.041, 25th 0.224, median 0.343, 75th 0.483, 95th 0.706; it is at or above 0.10 on 0.912 of instances, which is the deployable rule's probe rate. The reversal rate of the history cube is above zero on 0.999 of instances, which is why that gate is thresholded at its median and not at zero.

The three contrasts. κ gate minus random gate: $+0.002$, template interval $[-0.003, 0.009]$ (X5, refuted; the interval includes zero and its upper end, 0.0086, is above the 0.005 line that would have made the null informative, so the licensed sentence is the weaker one: no decision information from κ was detected at this precision). κ gate minus reversal gate: $+0.0001$, $[-0.003, 0.005]$, coverage not measured (X6, refuted; no advantage was detected for the κ gate over the reversal gate, and §K.15 says why the licence's wording is not used). The deployable rule: regret 0.017 at USD 0.135 per swap against the pooled default's 0.005 and the probe's 0.016 at USD 0.146; it neither beats the default by 0.01 while saving a quarter of the budget nor beats always-probing by 0.01 (X7, refuted). No threshold is added after the fact.

K.8 The descriptive $\widetilde{\kappa}$ and a null that could not be used

On the full $6 \times 4 \times 568$ cube the raw interaction share is 0.395 and the EMS-corrected $\widetilde{\kappa} = 0.384$, template-cluster interval $[0.292, 0.529]$ (500 resamples of the 69 templates); the reversal rate is 0.40: two of every five ordered model pairs change sign across some pair of forms. Pooled success by form is 0.614, 0.509, 0.333 and 0.500 for `json`, `react`, `xml` and `pythonic`. So the interaction here is larger than five of the seven synthesis designs' and below `live_reason`'s 0.501 and `live_strict_strong`'s 0.387, and it is concentrated: `gpt-5.6-luna` loses 0.46 from `json` to `pythonic` while `gpt-4o-mini` gains 0.005, and `json` is still the best form for five of the six models. A high $\widetilde{\kappa}$ and a fixed default that captures nearly all of the oracle are not in tension: κ measures how much the forms reorder the models, not whether the best form changes.

The protocol asked for the mean-matched Bernoulli null p-value (E55's construction, model and form main effects kept, independent Bernoulli cells, plus-one counting, seed 120, 4,000 draws) with a calibration check on 300 additive template-clustered grids of the same shape, and said the p-value is labelled unusable if the measured false-positive rate at $\alpha = 0.05$ is outside $[0.02, 0.10]$. The false-positive rate is 0.000: the construction rejected none of the 300 null grids (median null p-value 1.0). It is therefore labelled **UNUSABLE** and no p-value is quoted for κ on this family. The mechanism is visible in the calibration: on an additive grid the EMS-corrected $\widetilde{\kappa}$ floors at zero, every null draw is at least zero, and the plus-one p-value of a zero observation is 1.0, the median across the 300 grids, so the construction cannot reject a null grid of this shape and its behaviour on a grid with interaction is untested. $\widetilde{\kappa}$ on WorkBench is a descriptive share with its template interval, nothing more; that is the same status the paper gives every κ (§3.1).

K.9 E120 BESIDE THE SEVEN DESIGNS

The seven-design synthesis (App. AB.3) is not recomputed. E120's row is printed beside the seven committed rows in the synthesis's own format and by E106's own measure (seed 106): task-level splits of 113 history, 113 probe and 342 held-out tasks with the six models resampled, so its payoff column is not the template-blocked design above and is not compared with it.

Table 17: The seven gate-passing designs of the synthesis, and E120 as one out-of-sample point in the same format. Payoff is what probing saves over keeping the incumbent's format; the last column is what it saves over a fixed default. Nothing in the seven rows is recomputed.

design	models	forms	tasks	$\widetilde{\kappa}$	reversal	payoff	vs. fixed
`bfcl_grid`	7	4	40	0.000	0.143	+0.0037	−0.0015
`hbench_fixed`	13	6	29	0.346	0.577	+0.0346	−0.0171
`live_chat4k`	7	4	52	0.214	0.476	+0.0912	+0.0721
`live_hetero`	7	4	52	0.148	0.191	+0.0394	+0.0166
`live_reason`	7	4	52	0.501	0.762	+0.0870	+0.0602
`live_strict`	7	4	52	0.316	0.524	+0.1584	+0.1187
`live_strict_strong`	4	2	52	0.387	0.167	+0.0186	−0.0077
E120 WorkBench	6	4	568	0.384	0.400	+0.0269	−0.0029

E120's row has a positive payoff over inheriting ($+0.027$) and a slightly negative payoff against a fixed default (-0.003); three of the seven designs, `bfcl_grid`, `hbench_fixed` and `live_strict_strong`, show the same sign pattern. Within E120, over the six incoming models, the rank correlation of a model's mean $\widetilde{\kappa}_{\text{hist}}$ with what probing saved for it is $+0.09$, the same as the reversal rate's, and both are -0.60 against the saving over the fixed default; six points are a description, not a test.

K.10 SENSITIVITIES

Task-level split (the E70 convention, labelled sensitivity). With 10 history, 10 probe and 548 held-out *tasks* instead of templates, the history is 10 tasks rather than about 82 and the probe sees near-copies of held-out tasks; the probe's regret falls to 0.014 and the pooled default's rises to 0.020: the fixed default minus probe contrast flips to $+0.006$ (template-blocked: -0.011), and keep minus probe reads $+0.021$ ($+0.023$). This is the flattery the template split exists to remove, and it is the reason the headline is template-blocked. **Model resampling.** Resampling the six models with replacement inside each draw (1,960 usable draws, 4.03 distinct models on average, distinct models only) gives regrets of 0.040, 0.015 and 0.016 for keep, pooled default and probe, and a default-minus-probe contrast of -0.0003; this is not an interval on the model population. **Split percentiles across the 2,000 draws**, labelled split sensitivity: keep minus probe $[-0.041,\ 0.075]$, default minus probe $[-0.063,\ 0.068]$, κ gate minus random $[-0.034,\ 0.041]$, keep minus default $[-0.005,\ 0.079]$. **Argmax ties.** The 10-task probe had a tie in 0.351 of its argmaxes (ties go to `json`, which favours the probe on this family), the 3-task probe in 0.677, the pooled default in 0.001.

K.11 PRECISION, AS DECLARED AND AS COMMITTED

The protocol declared its precision from a seeded simulation of this exact procedure on $6 \times 4 \times 69$-template grids with WorkBench's action-required counts per template, additive template, task, model×template and form effects, Bernoulli draws, and a double-centred model×form interaction of amplitude 0, 0.05, 0.10 and 0.15. The protocol printed a table of ranges; the committed run (`data/e120_workbench/precision_sim.json`, seed 120, 14 datasets per cell) is the measurement, and Table 18 prints both. The committed run lands inside the declared ranges at amplitudes 0 and 0.10 and outside them at 0.05 (SD 0.009 against 0.013; the deployable rule fires on 0.044 against 0.07) and 0.15 (Δ $+0.071$ against 0.074 to 0.082; SD 0.015 against 0.023 to 0.030); a 30-dataset check under three other seeds gives SDs of 0.018 to 0.026 at the top amplitude, so both are draws from an SD estimate with a relative standard error near 0.2 (addendum item 6). The template bootstrap's planning coverage of the population Δ was 0.571, 1.000, 0.857 and 0.929 at the four amplitudes with 40 replicates, against the protocol's declared 0.71 to 0.86; the analyzer uses 500, and every interval in this appendix is a percentile interval with that planning coverage beside it. The point threshold 0.03 was chosen from this table and from break-even: 0.03 repays a 40-episode probe after $1{,}333$ deployment episodes.

Table 18: Precision of the design, declared before collection (the protocol's ranges) and as committed (the seed-120 run, 14 datasets per cell). Δ is regret(pooled default) minus regret(probe); "confirm" is the share of datasets with $\Delta \geq 0.03$ and lower end above 0; "fires" is the share of instances on which the deployable κ rule probes; coverage is the 40-replicate template bootstrap's coverage of the population Δ.

		protocol, declared				committed simulation				
amplitude	$\widetilde{\kappa}$	Δ	SD	confirm	fires	Δ	SD	confirm	fires	coverage
0.00	0.000	-0.017 to -0.014	0.011–0.014	0.00	0.00	-0.014	0.011	0.00	0.005	0.571
0.05	0.038	-0.002	0.013	0.00	0.07	-0.003	0.009	0.00	0.044	1.000
0.10	0.178	$+0.026$ to $+0.032$	0.012–0.020	0.50–0.64	0.45–0.70	$+0.031$	0.015	0.64	0.584	0.857
0.15	0.339	$+0.074$ to $+0.082$	0.023–0.030	0.87–1.00	0.88–0.99	$+0.071$	0.015	1.00	0.968	0.929

What the real cube did against this table is the one surprise of the study, and it is worth stating plainly. The observed Δ is -0.011, the value the simulation gives when the interaction is between 0 and 0.05 in amplitude, and the protocol's "what a null looks like" paragraph named Δ inside $(-0.02, 0.02)$ with an expected -0.014 under no interaction. Yet the cube's $\widetilde{\kappa}$ is 0.384, the top row, where the simulation expected $\Delta \approx +0.07$ and a confirmed X3 almost always. The simulator's interaction was a smooth ramp spread over every model and form; WorkBench's is concentrated in one model under one form that no default picks. So the mapping from $\widetilde{\kappa}$ to what a probe can buy, which the precision table assumed, does not hold on this family, and that is the same finding X5 and X6 make from the other side.

K.12 INFERENCE VALIDATION: WHAT THE PRODUCTION INTERVAL COVERS

The coverage in the table above is a *planning* number: 40 bootstrap replicates, 14 datasets per cell, computed before collection. The analyzer ships 500 replicates, and nothing had run it on data whose answer we knew. A reviewer asked for that, and asked a second question with it: the bootstrap relabels a template drawn twice as two clusters, so the partition drawn inside a replicate can put copies of one template in the probe set and the held-out set at once.

Design. `experiments/review_20260919/e120_inference_validation.py` (post hoc, 19 September 2026, seed 20260919) imports E120's own production procedure and runs it unchanged: `draw_splits` over $2{,}000$ template partitions, `score_draws`, `resolve_gates`, `contrasts_from`, and the 500×40 template bootstrap. It is a re-run of the shipped code, not a stand-in for it. The data are 50 simulated $6 \times 4 \times 69$-template Bernoulli cubes in each of four regimes, with WorkBench's action-required counts per template. *Null* has no model $\times$ form interaction. *Diffuse* is the precision simulator's alternating ramp at amplitude 0.10, the row the planning table used. *Bad cell* takes 0.20 from one model on the form with the smallest form effect, so κ is large in a cell no pooled default looks at. *Reversal* gives three models a gain on form 0 and a loss on form 1 and the other three the reverse, so the models disagree about the best form and a probe can pay. Every dataset also gets a second interval from a grouped resampler, in which every copy

of a template drawn more than once is assigned to one partition set. The leakage the reviewer asked about cannot occur there.

E120's production interval on simulated data with a known target (2,000 template partitions, 500 template resamples × 40 partitions, 50 datasets per regime, seed 20260919). Mean, SD and h.w. are across datasets; h.w. is the mean interval half-width. Coverage is of the across-dataset mean, the quantity a bootstrap of the estimator targets; nominal 95%. "excl. 0" is the share of the datasets whose interval does not contain zero. The grouped column is the second resampler, which keeps every copy of a template in one partition set.

regime	contrast	mean	SD	h.w.	coverage (Wilson)	excl. 0	grouped (Wilson)
null	default − probe	−0.0139	0.0094	0.0108	0.720 [0.583, 0.825]	0.52	0.720 [0.583, 0.825]
	keep − default	0.0069	0.0048	0.0062	0.800 [0.670, 0.888]	0.56	0.740 [0.604, 0.841]
	keep − probe	−0.0070	0.0056	0.0070	0.640 [0.501, 0.759]	0.24	0.700 [0.562, 0.809]
	κ gate − random gate	0.0000	0.0002	0.0032	1.000 [0.929, 1.000]	0.00	1.000 [0.929, 1.000]
diffuse	default − probe	0.0282	0.0171	0.0199	0.740 [0.604, 0.841]	0.70	0.820 [0.692, 0.902]
	keep − default	−0.0026	0.0131	0.0146	0.720 [0.583, 0.825]	0.32	0.700 [0.562, 0.809]
	keep − probe	0.0255	0.0067	0.0108	0.940 [0.838, 0.979]	0.98	0.920 [0.812, 0.968]
	κ gate − random gate	0.0029	0.0014	0.0072	1.000 [0.929, 1.000]	0.00	1.000 [0.929, 1.000]
bad cell	default − probe	0.0034	0.0202	0.0166	0.580 [0.442, 0.706]	0.42	0.600 [0.462, 0.724]
	keep − default	0.0028	0.0126	0.0112	0.620 [0.482, 0.741]	0.44	0.580 [0.442, 0.706]
	keep − probe	0.0061	0.0084	0.0085	0.660 [0.522, 0.776]	0.58	0.800 [0.670, 0.888]
	κ gate − random gate	0.0053	0.0040	0.0064	0.880 [0.762, 0.944]	0.32	1.000 [0.929, 1.000]
reversal	default − probe	0.0413	0.0263	0.0241	0.680 [0.542, 0.792]	0.82	0.740 [0.604, 0.841]
	keep − default	−0.0008	0.0212	0.0175	0.620 [0.482, 0.741]	0.34	0.640 [0.501, 0.759]
	keep − probe	0.0405	0.0081	0.0138	0.940 [0.838, 0.979]	1.00	0.900 [0.786, 0.957]
	κ gate − random gate	0.0016	0.0015	0.0078	1.000 [0.929, 1.000]	0.00	1.000 [0.929, 1.000]

The real WorkBench cube: the locked interval and the grouped one. The rule fixed before the run prints both intervals when an end moves by more than 0.005. No locked interval is changed.

contrast	point	locked interval	grouped interval	shift
default − probe	−0.0108	[−0.0191, 0.0035]	[−0.0280, 0.0051]	0.0089 (both printed)
keep − default	0.0342	[0.0238, 0.0472]	[0.0219, 0.0498]	0.0026 (locked only)
keep − probe	0.0234	[0.0115, 0.0432]	[0.0053, 0.0389]	0.0062 (both printed)
κ gate − random gate	0.0020	[−0.0033, 0.0086]	[−0.0051, 0.0122]	0.0036 (locked only)

What came back. Coverage of the across-dataset mean runs from 0.580 to 0.940 on the three contrasts between rules, against a nominal 95%. The worst cell is default − probe on the bad cell, 0.580, Wilson [0.442, 0.706]. The κ gate − random gate contrast covers at 1.000 in three regimes and 0.880 on the bad cell, because its point estimate barely moves. The reason is width: on the bad cell the across-dataset SD of default − probe is 0.0202 and the mean half-width of its interval is 0.0166, so the interval is narrower than one standard deviation of the quantity it brackets. The grouped resampler does not repair this. On that same cell it reads 0.600, and its mean half-width is 0.0146 against the production interval's 0.0166: narrower, not wider. Template copies inside a partition are therefore not what makes the interval optimistic.

The rule, and one clause of it that was wrong. The script fixed its reading rule before the run: the intervals are nominal 95%, and they are acceptable if coverage is at least 0.90 in every regime for the two contrasts the main text rests on (default − probe and keep − default) and the null false-positive rate is at most 0.10; otherwise the paper prints the measured coverage beside every E120 interval and says "no detected advantage", never "not separated" or "equivalent". The verdict block reads `acceptable: false`, worst headline coverage 0.58. One clause of that rule was misspecified, and the correction belongs here rather than in the numbers. The clause assumed the true contrast is zero when there is no model × form interaction, and it is not: a 40-episode probe is noisy, so a fixed default beats it even on an additive surface, and the null regime's across-dataset mean of default − probe is −0.0139, not 0. The shares of null intervals that do not contain zero, 0.52 for default − probe and 0.56 for keep − default, are therefore power against a real nonzero difference, and this paper does not report them as a false-positive rate. Coverage is the criterion that decides, and coverage fails.

The real cube. The second table above runs the grouped resampler once on the real WorkBench cube and prints its interval beside the locked one. Two ends move by more than the 0.005 the rule set: default − probe by 0.0089 and keep − probe by 0.0062. Two do not: keep − default by 0.0026 and the κ gate − random gate by 0.0036. For the first two the grouped intervals are: keep − probe, grouped [0.005, 0.039] against the locked [0.012, 0.043]; default − probe, grouped [−0.028, 0.005] against the locked [−0.019, 0.004]. Table 12 prints the locked intervals, each with

its measured coverage, and these two grouped intervals in its caption. The main text prints the one it discusses, pooled default − probe, with its coverage (§5.2); its grouped interval is printed here and in Table 12's caption. No locked interval is re-scored or replaced.

What this does not settle. Fifty datasets per regime is a small number. The worst cell's coverage of 0.580 carries a Wilson band of $[0.442, 0.706]$, wider than the gaps between most cells, so the table does not order them. What it settles is that coverage sits below the nominal 95% almost everywhere, not by how much. Four regimes are four shapes of interaction, not a sample of the shapes WorkBench could have. The measurement is of the interval, not of the point estimates: no locked E120 number moves because of it.

K.13 How much history does the pooled default need? (post hoc)

Everything in this subsection is **post hoc**. A separately validated conditional evaluation was added (App. L); the original split-averaged intervals remain poorly calibrated (App. K.12). A reviewer then asked for this analysis: reduce the history, keep the held-out templates and the incoming model's probe budget fixed, and keep fixed `json`, which needs no history, in the comparison, so that the pooled default's robustness can be told apart from the benefit of having much history. `experiments/review_20260923_hd/history_size.py` (23 September 2026, `data/review_20260923_hd/history_size.json`) reads only E120's cached cells through E120's own code and makes no API call; it edits no protocol and re-scores no locked verdict. Its docstring fixed the estimand, the intervals, their validation and the reading rule before it computed a number.

Design. The pooled default for incoming model B names the format with the best pooled history mean over the five other models. Here it reads the history of only k of them, $k = 1, \ldots, 5$, and every k-subset is used: $\binom{5}{k} = 5, 10, 10, 5, 1$, so 31 subsets per incoming model and 186 (B, subset) pairs, with no sampling. Everything else is E120's: its own $2{,}000$ template partitions (`draw_splits` under `numpy.random.default_rng(120)`), so each split's 49 held-out templates and B's 40-episode probe are the same at every k. Two rows were known before the run. At $k = 5$ the rule is the pooled default. At $k = 1$, averaged over the five one-model subsets, it is keeping A's format, because one model's best pooled format is that model's best history format. The script checks both identities on every split against E120's engine, and over all $2{,}000$ splits it reproduces E120's locked rule table and one-off calibration cost to their printed four decimals; otherwise it stops before writing. Only $k = 2$, 3 and 4 are new.

Estimand and reading rule. $R(k)$ is B's held-out regret under the default read on k models, task-weighted as in E120 and averaged over the splits, the six incoming models and the $\binom{5}{k}$ subsets. $D_{\text{json}}(k) = R(k) - \text{regret}(\texttt{json})$ is paired on the same split and the same B, and $D_{\text{probe}}(k)$ likewise against the 40-episode probe. The history cost is k models $\times$ 4 formats $\times$ the split's action-required history tasks, with those cells' provider USD. These are E120's rule-level estimand, expectations over its partition distribution, not quantities over task families. They carry no interval: the only template interval E120 has for a split-averaged regret covered as little as 0.58 of the time (§K.12), and a new one would need its own validation. The rule fixed before the run reads D_{json} against 0.01, the margin E120's locked X7 uses. $k_{\min}$ is the smallest k from which D_{json} stays at or below 0.01; $k_{\min} = 2$ reads "pooling suffices" and $k_{\min} \in \{3, 4, 5\}$ reads "needs $k_{\min}$ models".

What came back. $k_{\min} = 4$, so the verdict is "needs 4 models" (Table 19). In the words the script licensed: reading three other models' history, the pooled default trails fixed `json` by 0.013 in held-out regret, more than 0.01; from four models ($1{,}321$ history episodes) on it is within 0.01. Its standing depends on how much history there is; fixed `json`, which needs none, has regret 0.003. Four models' history costs USD 4.82 on average. With two models' history the default trails by 0.022, and with one, which is inheriting, by 0.037. Against the probe: from three other models' history on, the default's held-out regret is no higher than the 40-episode probe's (-0.0005 at three); at two it is 0.008 higher.

The shortfall differs by pair. At $k = 3$, 16 of the 60 (B, subset) pairs trail `json` by more than 0.01; at $k = 4$, 6 of 30; at $k = 5$, 1 of 6. At every k the largest shortfall is for incoming `gpt-5.6-luna`:

Table 19: **History size (post hoc).** The pooled-history default read on k of the five models other than B, on E120's 2,000 template-blocked splits, every k-subset. "hist. ep." and USD: the history cells the default reads, mean over splits and subsets, paid once and not per swap. Regret is task-weighted; −json and −probe are the paired differences in regret against fixed json and against the 40-episode probe of B. All of these are rule level and descriptive, with no interval. json share: the share of (split, B, subset) instances on which the rule names json. Last column: the conditional default-minus-json contrast with the choices frozen on E120's first partition. Generated from data/review_20260923_hd/history_size.json.

history read	hist. ep.	USD	regret	−json	−probe	json share	partition 0: −json
$k = 1$ model (keep A's format)	330	1.21	0.039	+0.037	+0.023	0.732	identical choices
$k = 2$ models	661	2.41	0.024	+0.022	+0.008	0.862	identical choices
$k = 3$ models	991	3.62	0.016	+0.013	−0.001	0.934	identical choices
$k = 4$ models	1,321	4.82	0.009	+0.006	−0.007	0.976	identical choices
$k = 5$ models (pooled history)	1,652	6.03	0.005	+0.003	−0.011	0.994	identical choices
fixed json	0	0	0.003	–	−0.014	1.000	–
probe B, 40 ep./swap	0	0	0.016	+0.014	–	0.744	–

0.214 reading gpt-4o-mini's history alone, 0.016 reading all five. With fewer models, one model's preference moves the pick more often: the default names json in 0.994 of instances at $k = 5$, 0.934 at $k = 3$ and 0.862 at $k = 2$, and at every k most departures are to pythonic (0.115 of instances at $k = 2$). Two readings of the same numbers, neither an interval: over splits, the 5th to 95th percentiles of D_{json} run from 0 to 0.046 at $k = 3$ and from 0 to 0.038 at $k = 4$, and the default trails json on 0.571 and 0.223 of splits; weighting each held-out template once instead of each task gives D_{json} of 0.0130 and 0.0064 at $k = 3$ and 4, the same verdict.

The argmax is the engine's own floating-point expression, which is what lets $k = 5$ reproduce E120. Where the pooled success counts tie exactly, rounding rather than the json-first order breaks the tie in 243 of the 120,000 instances at $k = 3$ and 48 of the 60,000 at $k = 4$ (and 7 of 12,000 at $k = 5$, in E120's engine too). A regret lies in $[0, 1]$, so no way of breaking those ties could bring $D_{\text{json}}(3)$ down to 0.01 or lift $D_{\text{json}}(4)$ above it.

The conditional contrast. As in App. L, the script also freezes every default from history cells on E120's first five partitions and scores default minus json and default minus probe on the 49 held-out templates, with the t and percentile intervals checked against a known target under the same two generators, 2,000 datasets each. On partition 0, the primary one, every default at every k names json for every incoming model and subset, and so does the probe. Both contrasts are therefore zero on every template by construction, and no interval is read (the table's last column). On partitions 1–4, a sensitivity that moves no verdict, some default departs from json on 4, 3, 2, 1 and 0 partitions at $k = 1$ to 5, and on each of them default minus json has a calibrated t interval above zero: where less history moved frozen defaults off json, they did worse on those held-out templates, on average over incoming models and subsets.

What this does not settle. The rule-level numbers are descriptive, and the 0.01 margin is a point comparison, not a test. The subsets are drawn from these five models, on one family and one date, and the per-pair spread suggests that which models' history is read matters as well as how many. No locked E120 number moves. What changes is how the pooled default's cost is stated: on WorkBench its standing rests on the history of at least four of the five other models, which fixed json does not need. This is one reason the paper no longer carries the recommendation the locked protocol names (App. K.15).

K.14 Cost

16,560 grid episodes cost USD 60.01 by provider-reported usage, present on every cell. The provider's balance page moved by USD 60.64 between the read before the first smoke run and the read after the last grid cell; the difference is the first 288-episode smoke run (USD 0.57 on the balance page) and the reserve model's 48 episodes in the second (USD 0.05); the second run's other 240 episodes were answered from the local request cache and did not reach the provider. Per episode: claude-haiku-4.5 USD 0.0130 (11,690 tokens, 5.6 calls), gpt-5.6-luna 0.0031,

`qwen3-32b` 0.0022, `deepseek-chat` 0.0016, `llama-3.3-70b` 0.0010, `gpt-4o-mini` 0.0007 (Table 11). A 40-episode probe of one incoming model costs 205 provider calls and USD 0.146 on average over the six; the one-off calibration of the fixed default and the gates costs 1,652 episodes and USD 6.03 once. A probe costs USD 0.146 per swap and the collection USD 60.01; against the pooled default the point estimates give no finite break-even for five of six models under the stated accounting, because the pooled default leaves the probe almost nothing to find.

K.15 THE ELEVEN VERDICTS

Table 20 lists every locked prediction with its observed value. Two confirmed, two gates passed, seven refuted. What each outcome licenses was written in the protocol before collection. After the collection, the inference validation (§K.12) measured the template interval's coverage 0.58 to 0.94 against a nominal 95%, and its own rule, fixed before it ran, governs every interval-based licence: each interval is printed with its measured coverage, and a contrast whose interval contains zero shows "no detected advantage", never an equivalence; with coverage below nominal, no interval is read as a bound. The main text therefore says the following. The probe's advantage over the pooled default, seen only on variants of our suite, is scoped as suite-only, and the interval is printed with its coverage (X3, interval includes zero). X3's clause that the main text and the abstract's policy sentence recommend the fixed default is executed in a narrower form. The abstract says that both deployable shared formats, fixed `json` and the pooled default, had lower observed regret than every probe we tested, and its last sentence asks a study to test its selection rule against a strong shared default; the main text says the same and what history each default needs. Neither recommends the pooled default over `json`, because of a comparison the licence did not foresee: fixed `json`, which the protocol fixed before collection, had lower observed regret than the pooled default (0.003 against 0.005) and needs no WorkBench history. X3's clause for an upper end below 0.03, an informative null that a 40-episode probe cannot buy three points, reads the upper end as a bound and is not used. The probe's advantage over inheriting is reported as measured on both families, on WorkBench below the 0.03 line (X2, refuted on the line, not on the sign, so X2's sentence that the probe did not beat keeping is not used). No sentence calls the probe cheap, and the break-even counts are reported as measured point estimates (X4). The paper stops offering κ as a gate and says no decision information from κ was detected at this precision (X5). For X6 the main text says only that no signal we tried, κ included, was shown to say when to spend a probe; this appendix reports that no advantage was detected for the κ gate over the reversal gate: the licence's "the reversal rate does the same job" would read an interval of unmeasured coverage as an equivalence, and its clause recommending the reversal rate is not executed either, because we did not detect an advantage for the reversal gate over the random gate (0.012 against 0.011). X7's sentence, that neither κ nor a simpler summary is a probe-spending rule, is worded in the main text at the scope of the rules tested: the tested κ gates did not show an advantage. A sentence about every possible rule would go beyond what was tested. The result table carries the harm column beside every rule (X8); X8's confirmation rests on an interval whose coverage was not measured, so the main text establishes no non-inferiority. X10's sentence, that the cheapest repair for a swap is pooling the existing models' history rather than probing, is reported as its comparison: the pooled default lowered regret against inheriting by 0.034 at no per-swap probing cost, and Table 12 ranks every rule by regret at its cost. The main text does not call pooling the cheapest repair, because fixed `json` costs nothing per swap either and had lower observed regret. X9's first clause held and its second did not: two forms are some model's held-out best (`json` for five, `pythonic` for one), and the inherited format is B's best form in 0.605 of swaps, more than half, so the licensed sentence is worded as measured and not as "one format served every model".

K.16 WHAT THIS CANNOT SHOW

One task family, templated by its authors, with 69 templates that bound the precision no matter how many instances are run. Six models on one routing provider on one date, so the result is a design and a date, not a property of a swap; one of the six is a reserve, substituted by a locked rule for a behaviour that is itself a harness effect. Four free-text forms and no schema-constrained form. A step cap of 10, a per-domain toolkit and a form-neutral tool block, so absolute scores are not comparable to WorkBench's leaderboard; these are main effects and lower every form together. A state oracle that scores a stopped episode incorrect by its authors' rule. Cost in provider dollars, not in what a correct task is worth to a team. Gates scored at one matched rate and one fixed threshold;

Table 20: E120's eleven locked predictions, their observed values, and the verdicts the locked analyzer wrote. "Interval" is the template-bootstrap 95% percentile interval of the named paired difference.

	locked prediction	observed	verdict
X0	oracle accepts all 690 gold lists; 2,760 rendered lists and 3,788 calls round-trip; smoke ends with zero API errors	690/690; 2,760/2,760; 3,788/3,788; 0 errors on 288 episodes	PASSED
X1	pooled correct on the 568 action-required tasks in $[0.10, 0.90]$ and no-parsed-call below 0.15	0.489; 0.016 (13,632 episodes)	PASSED
X2	regret(keep) − regret(probe) ≥ 0.03, lower > 0	$+0.023$, $[0.012,\ 0.043]$	**refuted**
X3	regret(pooled default) − regret(probe) ≥ 0.03, lower > 0 (the reviewer's question 2)	-0.011, $[-0.019,\ 0.004]$	**refuted**
X4	$N_B^* \leq 2{,}000$ for at least 4 of 6 incoming models	0 of 6; ∞ for five, 9,710 for one	**refuted**
X5	regret(κ gate) $<$ regret(random gate) at probe rate one half, upper < 0 (question 1)	$+0.002$, $[-0.003,\ 0.009]$	**refuted**
X6	regret(κ gate) $<$ regret(reversal gate) at probe rate one half, upper < 0	$+0.0001$, $[-0.003,\ 0.005]$	**refuted**
X7	the $\widetilde{\kappa} \geq 0.10$ rule beats the free default by 0.01 at under 0.75 of the probe's USD, or beats the probe by 0.01 at no higher USD	0.017 at USD 0.135 against 0.005 (pooled default) and 0.016 at USD 0.146	**refuted**
X8	harm rate of the success-chosen form at most 0.02 above the fixed default's, upper < 0.02	-0.0002, $[-0.007,\ 0.004]$	confirmed
X9	at least 2 forms are some model's best, and the inherited format is B's best in under half of swaps	2 forms (`json`, `pythonic`); inherited best in 0.605	**refuted**
X10	regret(keep) − regret(pooled default) ≥ 0.02, lower > 0	$+0.034$, $[0.024,\ 0.047]$	confirmed
completeness	at least 60 complete templates	69 of 69, NORMAL	PASSED

other thresholds are curves, not verdicts. A mean-matched null that its own calibration check ruled out, so $\widetilde{\kappa}$ here carries an interval and no test. It can show whether a fair, prospectively specified probe beats a strong fixed default here, at a counted cost, and whether κ moved that decision. It did not, and it did not. It cannot show that either transfers to another family.

Part III — Unified policy results

L Opportunity, selection error, and two conditional comparisons (post hoc)

Everything in this appendix is **post hoc**. It was computed on 22 and 23 September 2026, after every E120 outcome had been seen, from cached cells only and with no API call; no protocol is edited and no locked verdict is re-scored. Two scripts produce it: `experiments/review_20260923_hd/decision_margins.py` (the decision margins of Table 21 and the decomposition below, `data/review_20260923_hd/decision_margins.json`) and `experiments/review_20260923_hd/conditional_policy.py` (the conditional comparison, `data/review_20260923_hd/conditional_policy.json`). Split percentiles here describe how much a quantity moves across splits of one design; they are not intervals.

Every design's decision margins. Table 21 lists, for each design, the best shared format, each model's decision margin and where the interaction sits. The main text quotes three of its readings: WorkBench's one positive margin, its losing advantage against the deficit of that format, and the strict parser's six positive margins.

Opportunity minus selection error. Fix one split and its held-out table. Let h_0 be that table's best shared form, $\widehat{G}$ its oracle gap, $U(\pi)$ the incoming models' mean held-out success under rule π's choices, and $R(\pi)$ their mean regret against each model's best held-out form. Then $U(\pi) = \frac{1}{a}\sum_m \max_h p_{mh} - R(\pi)$ and $U(h_0) = \frac{1}{a}\sum_m \max_h p_{mh} - \widehat{G}$, so $U(\pi) - U(h_0) = \widehat{G} - R(\pi)$ exactly; the script checks it on every split of both designs. WorkBench uses E120's own 2,000

Table 21: **Decision margins (post hoc).** Designs as in Table 2. $\widehat{G}$ is the mean margin $\max_h p_{mh} - p_{mh_0}$; margin>0 counts the models it is positive for. Losing: among formats that stay worse than h_0 for their model, the largest interaction advantage $\gamma_{mh} - \gamma_{mh_0}$, then that format's deficit d_h (two numbers, not a ratio). SS, no gain: the share of the interaction's sum of squares in formats other than h_0 that do not beat h_0 for their model; it shows where the interaction sits, not a share of lost gain. Schema `json` B is one of Harness-Bench's two schema-guided formats. †Fails the parse check.

design	h_0	$\widehat{G}$	margin>0	largest margin (model, format)	losing: $\gamma_{mh}-\gamma_{mh_0}$ / d_h	SS, no gain
BFCL v3 multi-turn	`react`	0.008	4/7	0.023 (`phi-4`, `pythonic`)	0.037 / 0.040	0.59
Harness-Bench, six formats	`xml`	0.023	6/13	0.103 (`gemma-3-27b`, schema `json` B)	0.186 / 0.228	0.68
our suite, 4k-token budget	`pythonic`	0.096	3/7	0.442 (`phi-4`, `react`)	0.124 / 0.162	0.47
our suite, permissive parser†	`json`	0.132	6/7	0.519 (`mistral-small-24b`, `pythonic`)	0.165 / 0.184	0.47
our suite, typed arguments	`json`	0.038	3/7	0.115 (`llama-3.1-8b`, `react`)	0.118 / 0.137	0.66
our suite, reasoning models	`react`	0.074	5/7	0.192 (`qwen3-235b-thinking`, `pythonic`)	0.132 / 0.247	0.59
our suite, strict parser	`pythonic`	0.143	6/7	0.346 (`phi-4`, `react`)	0.115 / 0.135	0.41
our suite, strong models	`json`	0.000	0/4	0 (all best on h_0)	0.207 / 0.245	0.50
WorkBench	`json`	0.001	1/6	0.005 (`gpt-4o-mini`, `pythonic`)	0.140 / 0.281	0.69

splits (`numpy.random.default_rng(120)`, `draw_splits` and `score_draws` imported), and every $R(\pi)$ reproduces the regret E120 reports. Our suite uses the 2,000 exact $10/10/32$ partitions of `data/review_20260912/policy_reanalysis.json`; the keep-the-format, probe and pooled-probe rows reproduce that file, and the other suite rows are new. Costs follow E120's protocol on both designs, so the pooled probe default, which reads B's 40 probe cells, is charged 40 episodes. The suite's cells carry no priced usage, so no USD is given for it.

Table 22: **Opportunity minus selection error (post hoc).** $R(\pi)$: held-out regret. $U(\pi)-U(h_0)$: held-out improvement over the best shared form of the held-out table, equal to $\widehat{G} - R(\pi)$ on every split. "best": share of swaps on which the rule picks B's best held-out form (ties count). "ep.": B's episodes per swap. The oracle row's improvement is the held-out $\widehat{G}$. On WorkBench the pooled-history default also reads the other models' history, paid once. †The headline grid fails the no-parsed-call gate. Generated from `data/review_20260923_hd/decision_margins.json`.

	WorkBench (E120)				headline suite†			
rule	$R(\pi)$	$U(\pi)-U(h_0)$	best	ep.	$R(\pi)$	$U(\pi)-U(h_0)$	best	ep.
keep A's format	0.040	−0.037	0.61	0	0.191	−0.058	0.24	0
pooled-history default	0.005	−0.003	0.78	0	0.245	−0.113	0.15	0
fixed json	0.003	0.000	0.79	0	0.136	−0.003	0.28	0
pooled probe default	0.006	−0.004	0.76	40	0.156	−0.023	0.29	40
probe B, 3 tasks/format	0.022	−0.019	0.66	12	0.082	0.051	0.44	12
probe B, 5 tasks/format	0.020	−0.018	0.64	20	0.068	0.065	0.50	20
probe B, 10 tasks/format	0.016	−0.014	0.64	40	0.050	0.083	0.60	40
probe B, all probe tasks	0.010	−0.008	0.62	330	same as 10 tasks/format			
oracle (reads held-out labels)	0.000	0.003	1.00	0	0.000	0.133	1.00	0

Reading Table 22. On WorkBench the held-out $\widehat{G}$ averages 0.003, and no rule we ran beats the best shared format on average: $U(\pi) - U(h_0)$ is −0.003 for the pooled-history default, −0.014 for the 40-episode probe and −0.037 for keeping A's format. On our suite, which fails our own gate, the held-out $\widehat{G}$ averages 0.133 and the probe recovers 0.083 of it. The probe picks B's best held-out format about as often on both (0.64 of WorkBench swaps and 0.60 of ours). What differs is the gain on offer: `json` is already B's best held-out format in 0.79 of WorkBench swaps and in 0.28 of ours. The pooled-history default is our suite's worst rule (−0.113): three formats nearly tie there, so leaving B out moves its pick off `json` for 4 of 7 models. On WorkBench the interaction is not small, but 0.69 of its sum of squares lies in formats that do not beat `json` for their model, 0.30 in the `json` column, and 0.009 in formats that beat `json`; this split describes where the squared interaction cells sit, not a causal share of lost gain. App. L.1 extends the comparison to every design that passes the validity gate.

On WorkBench the held-out $\widehat{G}$ averages 0.0026 over the splits (split percentiles 0 to 0.0078), against 0.0009 on the full table, and `json` is the held-out h_0 in every split, so fixed `json` is h_0 by construction there. On our suite the held-out $\widehat{G}$ averages 0.133 (split percentiles 0.094 to 0.165), and `json` is

the held-out h_0 in 0.797 of splits. The 40-episode probe improves on h_0 in 0.99 of the suite's splits. On the full suite table, leaving the incoming model out of the pool gives a mean regret of 0.321 at the pooled pick against 0.132 at `json`: the mean over the other models shades the pick against the incoming model's own positive interaction, and with three forms near a tie that is enough to move it: on the full table `json`, `react` and `pythonic` have form means 0.467, 0.440 and 0.440, and leaving B out moves the pooled pick off `json` for 4 of 7 models.

The conditional comparison: estimand. The script's docstring states the estimand, the intervals, the validation and the reading rule, and all four were written before it computed a number. We fix one template partition by a rule chosen before looking: the first partition drawn by E120's locked seed, which is E120's own first split (10 history templates with 78 action-required tasks, 10 probe templates with one action-required probe task each, and 49 held-out templates with 410 tasks). For each of the six incoming models B we freeze the form each rule names, reading history and probe cells only: the pooled default (the best pooled history form of the five other models), the probe choice (B's best form on its 10 probe tasks), `json`, and, for each incumbent A, the inherited form (A's best history form). Ties go to the first form in `json`, `react`, `xml`, `pythonic`, as in E120. The primary contrast on held-out template t is the six-model mean of B's success under the probe choice minus under the default, and the estimate is its mean over the 49 templates, each template weighted equally. The target is conditional: it scores the frozen choices on this partition, not the rules, so it is not E120's locked contrast, which is task-weighted and averaged over $2{,}000$ partitions, and it says nothing about how often a rule picks well.

What was frozen, and what the reading rule allowed. On partition 0 the pooled default and the probe named `json` for all 6 incoming models, and so did every inherited form in all 30 swaps. For four of the six the probe reached `json` only through the tie rule. So all three contrasts are zero on every held-out template, and the script's rule R4 applies: identical choices, no interval read. On the held-out templates `json` was not the best form for three models, each by less than two points of success. That is a description of the held-out cells, not a result.

The interval, checked against a known target. History, probe and the frozen choices are held fixed; only the held-out templates are resampled, from the partition's own 49 with replacement, under two generators (independent Bernoulli tasks at each template's observed rates, and a bootstrap of each template's own tasks that keeps the correlation between forms), $2{,}000$ datasets each. Wherever a contrast is not identically zero, both intervals reached the pre-declared band $[0.93,\ 0.97]$ under both generators: the t interval's coverage ran from 0.9395 to 0.9545 and the percentile interval's from 0.9325 to 0.947. By the script's rule R3 the t interval is the one printed.

Sensitivity: E120's first five partitions. The script's rule R5 runs partitions 1–4 with the same code; they move no verdict. The probe equals the default for 6, 3, 6, 6 and 3 of the six models. Probe minus default is 0, -0.136, 0, 0 and -0.026 (SD 0.059), and the two non-zero values have calibrated t intervals $[-0.168,\ -0.104]$ and $[-0.043,\ -0.008]$, so 2 of the 5 partitions have a calibrated interval below zero. In short: with choices frozen on five template partitions, the probe does worse on two, with frozen-choice intervals below zero (simulated coverage 0.940–0.955), and picks the same formats on three. Each of the six departures from `json` had lower held-out success than `json`; those per-model values are descriptive, since coverage was checked only for the six-model mean. Inherit minus default is 0 on partition 0 and negative on the other four, with a calibrated t interval below zero in 4 of the 5; `json` minus default is zero in all five, because the pooled default is `json` for every model in every partition.

The five partitions in one table, and their ties. Table 23 lists the five partitions: how many of the six incoming models get the same frozen choice from the probe and the pooled default, the template-weighted contrast, its calibrated t interval where rule R2 allows one, and that interval's measured coverage under the two generators. What the intervals cover is narrow: the frozen choices of one partition, templates weighted equally, and the two generators above. They are not intervals for the rules' split-averaged regrets of Table 16, whose template intervals remain poorly calibrated (App. K.12). The last column is post hoc (24 September 2026; `experiments/review_20260924_hd/unified_policy.py`): it re-runs the frozen choices under each of the 24 orderings of the four formats. The identical choices of parti-

tion 0 hold under 8 of the 24 orderings, exactly those that rank `json` above both `react` and `pythonic` (`data/review_20260924_hd/derived_readings.json`). Under the other 16 some model's probe choice leaves `json`, and the contrast is then negative, down to -0.053; it is never positive.

Table 23: **The conditional comparison on WorkBench's first five template partitions (post hoc).** Same choice: incoming models whose frozen probe choice equals the pooled default's. Probe − default: the six-model mean of held-out success, templates weighted equally. The t interval is printed only where its coverage, measured under both generators, fell in $[0.93, 0.97]$. Last column: the contrast's range over the 24 orderings used to break ties. Generated by `experiments/review_20260924_hd/make_writer_outputs.py`.

partition	same choice	probe − default	calibrated t interval	its coverage	24 tie orders
0	6 of 6	0	identical choices	–	$[-0.053, 0]$
1	3 of 6	-0.136	$[-0.168, -0.104]$	0.940–0.951	$[-0.166, -0.136]$
2	6 of 6	0	identical choices	–	$[-0.001, 0.001]$
3	6 of 6	0	identical choices	–	$[-0.003, 0]$
4	3 of 6	-0.026	$[-0.043, -0.008]$	0.952–0.955	$[-0.026, -0.026]$

L.1 One comparison over every gate-passing design, and its ties (post hoc)

Everything in this appendix is **post hoc**: every outcome it reads had been seen, it reads cached cells only, makes no model call, edits no protocol and re-scores no locked verdict. `experiments/review_20260924_hd/unified_policy.py` writes `data/review_20260924_hd/unified_policy.json`, and its docstring fixed these choices before the first run. *Designs:* the seven designs that pass the validity gate in `data/review_20260915/design_gates.json`, plus WorkBench; the headline grid (the main text's permissive-parser design), which fails the gate, is computed the same way and printed as a separate sensitivity row, never pooled. It shows the same pattern as the three designs with a realized gain: the probe keeps 0.083 of its 0.133, so the positive case does not rest on it. *Endpoints:* each design keeps its own; BFCL is turn accuracy and Harness-Bench a graded score in $[0, 1]$, and both are labelled so wherever they are printed. *Splits:* WorkBench uses E120's own 2,000 template-blocked partitions; every other design uses 2,000 task-level partitions into 10 history, 10 probe and the remaining held-out tasks. *Rules:* E120's rule engine, called unchanged, with the same rules on every design; fixed `json` and the probe on all probe tasks run only on WorkBench, because a fixed format picked from these tables would not be deployable. The pooled default is read from the other models, so two incoming models can get different formats; its mean regret is above $\widehat{G}$ on every design of Table 3, so on average h_0 is also the stricter comparator here. The script checks Eq. (5) on every partition, and it reproduces E120's regrets, the headline grid's regrets in `data/review_20260923_hd/decision_margins.json` and every full-table $\widehat{G}$ of Table 2. Table 3 in the main text and Fig. 5 below print the result, and Table 24 adds the 3- and 5-task probes and the probe on all probe tasks. The designs share models and tasks; they are related configurations, not independent replications.

Ties. The engine takes the largest floating-point mean, so a tie goes to the first format in `json`, `react`, `xml`, `pythonic`. Because every WorkBench outcome is 0 or 1 and every format in one decision is scored on the same tasks, each decision's exact tie set comes from integer success counts. The script records every tied set before resolving it, then re-runs every rule under each of the 24 orderings of the four formats, one ordering for every rule in a run, and under uniform choice among the tied formats. Table 25 gives the result. Ties are common for small probes and rare for the pooled default. Floating-point rounding never creates a tie, but for the pooled default it breaks 7 of its 17 exact ties, which is why the engine's regret differs slightly from the exact `json`-first value. No ordering makes any probe beat the pooled default, and the orderings that put `json` first give the probe its lowest regret. After the first run had been seen, and labelled so in the docstring, the same machinery was run on every other design, keeping only the ranges. The last column of Table 3 turns them into the 10-task probe's gain over the best shared format: $\widehat{G}$ does not depend on how ties are broken, so that gain ranges over $\widehat{G}$ minus the range of the probe's regret (`experiments/review_20260924_hd/derived_readings.py`). Its sign is the same un-

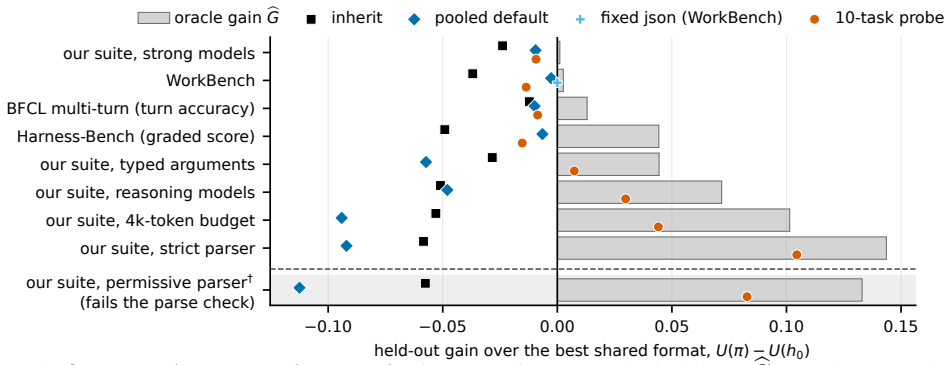

Figure 5: **Oracle gain and achieved gain (post hoc).** Bars: the held-out $\widehat{G}$. Markers: each rule's mean achieved gain over the best shared format chosen with hindsight, $U(\pi) - U(h_0) = \widehat{G} - R(\pi)$. Three of the eight designs have positive probe gains under every tie order. The typed-argument result ($+0.007$ with ties to `json`) is tie-sensitive. Values are in each benchmark's own units.

der every ordering on every design except typed arguments, where it runs from -0.011 to 0.011, and 0.000 when ties are broken at random. On WorkBench the 10-task probe scores below the pooled default, and the difference runs from -0.038 to -0.009 over the 24 orderings. On WorkBench, 35% of the 10-task probe's decisions are exact ties, and under all 24 tie orders every probe has higher regret than the pooled-history rule. The probe's lead over the pooled default keeps its sign under every ordering on every design except strong models, where it stays within 0.006 of zero either way.

The regimes, and how they were formed. The row groups of Table 3 apply one rule, written after the table had been seen and recorded in `experiments/review_20260924_hd/derived_readings.py`: a design shows a realized adaptation gain when the 10-task probe's mean achieved gain is above zero under the `json`-first rule and under every ordering of the design's formats; otherwise, limited adaptation opportunity when its held-out $\widehat{G}$ is below 0.02, and opportunity offset by selection error when it is not. Among these eight designs, the cut sits in the empty gap between the held-out $\widehat{G}$ values 0.013 and 0.044. A new design of App. M, P-typed, falls inside that gap: its split-averaged $\widehat{G}$ is 0.0230, 0.003 above the cut. So the gap describes these eight designs, not where the regimes separate. The grouping describes the measured outcomes of these designs at this budget, and the table generator applies the same rule, so the table and the text cannot disagree. App. M then tests the rule prospectively on five new designs with new models; the grouping of the eight designs stays post hoc.

Table 24: **The full comparison (post hoc).** Table 3 with every rule's regret, the 3- and 5-task probes and the probe on all probe tasks included. $\widehat{G}$ and $R(\pi)$ are means over 2,000 held-out splits. Last two columns: the 10-task probe's achieved gain over the best shared format, $\widehat{G} - R(\pi)$, with ties to `json`, and its tie-order range. BFCL is in turn-accuracy units and Harness-Bench in graded-score units. †Fails the parse check; a sensitivity row, not pooled with the others.

		selection error $R(\pi)$							10-task probe's gain	
			pooled	fixed	probe, tasks per format					every tie
design	$\widehat{G}$	inherit	history	json	3	5	10	all	$\widehat{G} - R$	rule
our suite, strong models	0.001	0.025	0.011	–	0.010	0.011	0.010	–	−0.009	[−0.028, −0.009]
WorkBench	0.003	0.040	0.005	0.003	0.022	0.020	0.016	0.010	−0.014	[−0.041, −0.012]
BFCL (turn accuracy)	0.013	0.025	0.023	–	0.025	0.023	0.022	–	−0.009	[−0.009, −0.007]
Harness-Bench (graded)	0.044	0.093	0.051	–	0.075	0.067	0.060	–	−0.015	[−0.018, −0.014]
our suite, typed arguments	0.044	0.073	0.102	–	0.043	0.040	0.037	–	0.007	[−0.011, 0.011]
our suite, reasoning models	0.072	0.123	0.120	–	0.082	0.063	0.042	–	0.030	[0.022, 0.035]
our suite, 4k-token budget	0.101	0.154	0.195	–	0.085	0.073	0.057	–	0.044	[0.038, 0.060]
our suite, strict parser	0.144	0.202	0.236	–	0.083	0.062	0.039	–	0.105	[0.091, 0.105]
our suite, permissive parser†	0.133	0.191	0.246	–	0.082	0.068	0.050	–	0.083	[0.075, 0.089]

Table 25: **WorkBench's ties (post hoc).** Exact ties: the share of a rule's 12,000 decisions whose best format is tied on integer success counts. Engine: the regret under E120's `json`-first floating-point rule. 24 orders: the range over every ordering of the four formats. Uniform: the exact expectation under uniform choice among ties. Last column: $U(\pi) - U(\text{pooled default})$ over the 24 orderings.

	exact	selection error $R(\pi)$			$U(\pi) - U(\text{pooled})$
policy	ties	engine	24 orders	uniform	24 orders
inherit	0.08	0.039	[0.039, 0.048]	0.044	[−0.042, −0.034]
pooled history	0.00	0.005	[0.005, 0.006]	0.005	–
fixed json	–	0.003	[0.003, 0.003]	0.003	[0.002, 0.003]
probe, 3 tasks	0.68	0.022	[0.019, 0.124]	0.068	[−0.119, −0.014]
probe, 5 tasks	0.53	0.020	[0.018, 0.083]	0.049	[−0.078, −0.013]
probe, 10 tasks	0.35	0.016	[0.014, 0.043]	0.028	[−0.038, −0.009]
probe, all probe tasks	0.08	0.010	[0.010, 0.013]	0.011	[−0.007, −0.005]

L.2 A CONDITIONAL CHECK OF THE STRONGEST POSITIVE CASE (POST HOC)

Everything in this subsection is **post hoc**. Every outcome of the strict-parser design had been seen, and Table 3 already reported its split-averaged gain. `experiments/review_20260925_hd/conditional_strict.py` writes `data/review_20260925_hd/conditional_strict.json`. Its protocol, `experiments/review_20260925_hd/PROTOCOL.md`, fixed the target, the split, the intervals, the simulation check and the reading rule, and it was committed alone before the script computed any number. The script reads cached cells only, makes no model call and changes no number reported elsewhere.

The question. Table 3 averages the probe's gain over 2,000 splits. Its split percentiles show how the gain moves from split to split; they are not an interval for any target. This check asks a narrower question. Take one split, fixed by rule, and freeze each model's format from its history and probe tasks. How sure can we be that these frozen choices beat the pooled default on held-out tasks like these? It is the WorkBench comparison above, with the task instead of the template as the unit.

The split and the frozen choices. The design has 7 models, 4 formats and 52 tasks, scored as task success. The split is the first of the 2,000 that App. L.1 draws for this design: the first permutation of `numpy.random.default_rng([10412, 1])`, with 10 history, 10 probe and 32 held-out tasks. No other split can replace it. For each incoming model B the script freezes two formats, reading history and probe cells only. The *probe choice* is B's best format on its 10 probe tasks per format (40 episodes). The *pooled default* is the best format of the six other models on the 10 history tasks. Ties go to the first format in `json`, `react`, `xml`, `pythonic`. No fixed format was named for this design before its outcomes existed, so there is no fixed-`json` contrast. On this split the pooled default is `pythonic` for all 7 models. The probe picks `json` for 2 models, `react` for 3 and `pythonic` for 2, so it agrees with the pooled default for 2 of the 7. Two of the seven probe decisions are exact ties.

The target. Let $y_t(B, h)$ be 1 if model B succeeds on held-out task t with format h, and 0 otherwise. The primary contrast on task t is $d_t = \frac{1}{7}\sum_B [\, y_t(B, \text{probe choice}) - y_t(B, \text{pooled default}) \,]$, and the estimate is the mean of d_t over the 32 held-out tasks. Tasks and models weigh equally. The target is the mean of d_t over held-out tasks like these, with the 7 models and the frozen choices held fixed. The secondary contrast replaces the pooled default with the best shared format of the held-out tasks, chosen with hindsight (`react` on this split). Its mean is this split's value of the achieved gain $\widehat{G} - R(\pi)$ of Table 3. That format is chosen on the tasks it is scored on, so the secondary contrast leans toward the shared format.

The intervals, checked against a known target. Two intervals are computed over the task values: the t interval and a percentile bootstrap with 2,000 resamples. To check them, the frozen choices are held fixed and only the held-out tasks are redrawn, 32 per simulated data set and 2,000 data sets per generator. The *task bootstrap* draws whole tasks with replacement, so it keeps the correlation between formats and between models on one task. The *Bernoulli* generator draws every outcome independently at the model's success rate with that format over the held-out tasks. One task has a

rate of 0 or 1, so WorkBench's per-template generator would add no noise here. The protocol's rule: an interval is *calibrated* when its measured coverage lies in $[0.93, 0.97]$ under both generators. A calibrated interval may be printed as a 95% interval for the conditional contrast, and when both are calibrated the t interval is printed.

What the split shows. The frozen probe choices scored 0.1607 above the pooled default on the held-out tasks, with a t interval of $[0.0894, 0.2321]$. Its coverage was 0.955 under the Bernoulli generator and 0.957 under the task bootstrap, so the interval is calibrated, and it lies above zero. The percentile interval, $[0.098, 0.228]$, is calibrated too. The task value d_t is not zero on 25 of the 32 tasks. Against the best shared format, the frozen probe choices scored 0.1473, with a calibrated t interval of $[0.0958, 0.1988]$; rounded, against that split's best shared format, chosen with hindsight, the same choices scored 0.147 higher, with a nominal 95% interval of $[0.096, 0.199]$ and measured coverage 0.953–0.959 (task bootstrap, Bernoulli). In simulation this secondary estimate averaged 0.136 (Bernoulli) and 0.140 (task bootstrap) against a true 0.147, so its bias works against the probe. Before it reads the split, the script checks that its loader reproduces this design's row of Table 3 to four decimals. The protocol licenses one sentence: "On the first declared split of the strict-parser design, the probe's frozen format choices scored 0.1607 $[+0.0894, +0.2321]$ above the pooled default on the 32 held-out tasks (post hoc)."

Ties and other splits. Both are described here, and neither moves the reading. Over the 24 orderings of the four formats, the primary estimate on this split runs from 0.134 to 0.161. The `json`-first rule gives the largest of these values, and no interval goes with the range. Table 26 runs the same code on the first five splits. The primary estimate is 0.161, 0.232, 0.321, 0.188 and 0.246 (SD 0.062). All five intervals are calibrated and above zero, and the first split (split 0 in the table), on which the reading rests, has the lowest estimate. The secondary estimate is 0.147, 0.138, 0.107, 0.121 and 0.094. On split 1 its interval is not calibrated (task-bootstrap coverage 0.926), so there it is a point estimate only. The other four are calibrated and above zero.

What this check does not show. It scores the frozen choices of one split. It is not an interval for the probe rule over fresh history and probe draws, so it is not an interval for the split average of Table 3. It is not a replication: the 52 tasks and 7 models are the ones that produced the positive case. It says nothing about new models, about other task families, or about the other two designs with a realized adaptation gain (reasoning models and a 4k-token budget), which share this suite. Its coverage was measured under the two generators above only. The Bernoulli generator ignores the differences between tasks, and the task bootstrap treats new tasks as draws from these 32.

Table 26: **The conditional check on the strict-parser design's first five splits (post hoc).** Same choice: incoming models whose frozen probe choice equals the pooled default. Estimates: the seven-model mean of held-out task success, tasks weighted equally. The secondary contrast is scored against the best shared format of each split's held-out tasks, chosen with hindsight. A t interval is printed only where its coverage, measured under both generators, fell in $[0.93, 0.97]$; its coverage is the range over the two generators. Last column: the primary estimate's range over the 24 orderings used to break ties, not an interval. The reading rests on split 0; splits 1–4 are a sensitivity. Generated by `experiments/review_20260924_hd/make_writer_outputs.py`.

		probe − pooled history (primary)			probe − best shared format (secondary)			
split	same choice	estimate	calibrated t interval	its coverage	estimate	calibrated t interval	its coverage	24 tie orders
0	2 of 7	0.161	[0.089, 0.232]	0.955–0.957	0.147	[0.096, 0.199]	0.953–0.959	[0.134, 0.161]
1	0 of 7	0.232	[0.150, 0.315]	0.948–0.953	0.138	not calibrated	0.926–0.934	[0.143, 0.232]
2	1 of 7	0.321	[0.231, 0.412]	0.951–0.952	0.107	[0.053, 0.161]	0.956–0.967	[0.237, 0.321]
3	0 of 7	0.188	[0.103, 0.272]	0.953–0.954	0.121	[0.051, 0.190]	0.946–0.967	[0.179, 0.196]
4	1 of 7	0.246	[0.163, 0.329]	0.948–0.953	0.094	[0.055, 0.132]	0.953–0.968	[0.161, 0.246]

L.3 Is the selection error a weakness of the plain probe? (post hoc)

This subsection and the next two are **post hoc**. Their protocol was committed alone before any code (`experiments/review_20260925c_hd/PROTOCOL.md`). The scripts read cached cells only, make no model call and change no frozen label (data: `data/review_20260925c_hd/`).

Before scoring anything new, they reproduce the plain probe's and the pooled default's $R(\pi)$ and every held-out $\widehat{G}$ of Table 3. At the 0.02 cut they re-read the eight frozen labels and every E127 label exactly.

Two stronger selectors. The *plain probe* is the 10-task probe of Section 5.2. It keeps the format with the best mean on the incoming model's 10 probe tasks, breaks exact ties by the format order (`json` first) and ignores the history. The two new selectors read the same data and use the history too. (1) The *history tie-break* breaks exact probe ties by the pooled default's history score instead of the format order. (2) The *shrunk probe* scores each format h by $s(h) = (n\, p_{\text{probe}}(h) + \lambda\, p_{\text{history}}(h))/(n + \lambda)$ with $n = 10$, and keeps the best score. Here λ is the weight of the history, counted in probe tasks: $\lambda = 0$ gives the history tie-break and $\lambda = \infty$ the pooled default. For each split and incoming model, λ is picked from $\{0, 5, 10, 20, 40, \infty\}$ by nested leave-one-model-out on the history cells alone, so no probe or held-out cell of the incoming model is read. Both selectors cost what the plain probe costs: 10 tasks per format for the incoming model, plus the history read once. Table 27 gives their selection error on the same 2,000 splits.

Table 27: **Two stronger selectors on the same data (post hoc).** Selection error $R(\pi)$ at 10 probe tasks per format, the mean over the 2,000 held-out splits of Table 3; the plain probe and the pooled default reproduce it. History tie-break: exact probe ties broken by the history. Shrunk probe: the probe pulled toward the history, with λ chosen by nested leave-one-model-out on history cells. Last column: the history tie-break's achieved gain $\widehat{G} - R$ with ties to the format order and under every ordering of its remaining ties; a range, not an interval. Row groups: the frozen labels. Generated by `experiments/review_20260925c_hd/make_selector_tables.py`.

		selection error $R(\pi)$				tie-break's gain
design	$\widehat{G}$	plain probe	history tie-break	shrunk probe	pooled history	$\widehat{G} - R$, range over tie orders
limited adaptation opportunity						
our suite, strong models	0.001	0.010	0.010	0.010	0.011	$[-0.009, -0.009]$
WorkBench	0.003	0.016	0.015	0.009	0.005	$[-0.013, -0.013]$
BFCL (turn accuracy)	0.013	0.022	0.021	0.022	0.023	$[-0.008, -0.008]$
opportunity offset by selection error						
Harness-Bench (graded)	0.044	0.060	0.059	0.050	0.051	$[-0.015, -0.015]$
our suite, typed arguments	0.044	0.037	0.039	0.054	0.102	$[0.004, 0.005]$
realized adaptation gain						
our suite, reasoning models	0.072	0.042	0.044	0.055	0.120	$[0.028, 0.029]$
our suite, 4k-token budget	0.101	0.057	0.050	0.062	0.195	$[0.051, 0.053]$
our suite, strict parser	0.144	0.039	0.050	0.056	0.236	$[0.093, 0.094]$

What the table shows. (1) Neither selector cuts the plain probe's R by half on any of the 8 designs. The largest cut is 0.010, by the shrunk probe on Harness-Bench. (2) Counting only changes of at least 0.005 (about one held-out task of one model), the history tie-break lowers R on 1 design (the 4k-token budget) and the shrunk probe on 2 (Harness-Bench and WorkBench). (3) Shrinking helps on the two designs where the pooled default already has lower R than the plain probe, Harness-Bench and WorkBench (WorkBench: 0.016 to 0.009), and hurts where adaptation pays (strict parser: 0.039 to 0.056); it also raises R on typed arguments and reasoning models. On those three designs of our suite the pooled default's R is 0.102 to 0.236. There the other models' shared format is a poor choice for the new model, so pulling toward it gives up part of the gain. (4) On the two designs where selection error offsets the opportunity, the stronger selectors leave at most a small gain. On Harness-Bench the shrunk probe still scores -0.005 against the best shared format. On typed arguments the history tie-break keeps a small gain, 0.004 to 0.005, under every tie order, where the plain probe's gain runs from -0.011 to 0.011. A pooled-history rule is only as good as its history. On WorkBench it reads every format on the other five models, and needs 4 of them to come within 0.01 of fixed `json` (App. K.13); on our suite, with 10 history tasks, it does worse than inheriting on 3 of its 5 variants (Table 3). So two stronger simple selectors that use the same data leave most of the selection error in place. We tried only these two, so this is not a limit for every selector. More probe tasks remain the lever that works (Table 24).

L.4 WHERE DOES THE PROBE GO WRONG? (POST HOC)

An example. Take two models on the strict parser. For `gpt-oss-20b`, the best two formats differ by 0.019 on the full table. The 10-task probe picks its best format on only 42% of the splits, but a miss costs 0.082 on the held-out tasks. For `phi-4`, the best format is 0.346 ahead of the second. The probe picks it on 98% of the splits, but the rare miss costs 0.495. A close call is missed often and cheaply; a clear best is missed rarely and at a high price.

The top-two gap. For each model, the *top-two gap* is its best full-table score minus its second best. It says how hard the model's best format is to find. The decision margin (Table 21) says how much the best format adds over the best shared one. The two answer different questions: a model can have a large margin and a close second best.

What we measured. We split the 58 model rows of the 8 designs at the median gap, 0.042, into close rows (at or below it) and separated rows (Table 28a). At 10 tasks per format, the probe misses the best format on 41% of the splits of a close row and on 22% of a separated row, and a miss costs 0.0801 against 0.1929. So close rows make 65% of the misses but 48% of the selection error: the probe's mistakes happen mainly where the top two formats are close, and there they are cheap. The same holds at 3 and 5 tasks. The other half of the selection error comes from fewer, costlier misses where one format is clearly best. Most of the gain comes from models whose best format is clearly ahead of the second and at least 0.02 better than the best shared one (Table 28b): these rows gain 0.106 per model, and the rows with the same margin but a close second best gain 0.025. The rank correlation between the gap and how often the probe finds the best format is 0.49 over all rows and 0.93 on the strict parser, in line with App. AA. The rows share models and tasks, and the gaps come from the same tables the regrets are scored on. So this is descriptive: it carries no p-value and is not a prediction for new models.

Table 28: **Where the probe goes wrong (post hoc).** (a) The 58 model rows of the 8 designs, split at the median top-two gap (0.042): close rows at or below it, separated rows above. Misses the best format: the mean share of the 2,000 splits whose choice is not among the model's best formats on the full table. Cost per miss: the mean held-out regret on those splits. (b) At 10 tasks per format, the same halves crossed with each model's decision margin; gain per model: the mean achieved gain over the split's best shared format. Descriptive: the rows share models and tasks. Generated by `experiments/review_20260925c_hd/make_selector_tables.py`.

(a)

probe tasks per format	top two formats	model rows	misses the best format	cost per miss	share of all misses	share of all selection error
3	close	29	47%	0.0799	54%	36%
	separated	29	40%	0.1861	46%	64%
5	close	29	45%	0.0783	58%	41%
	separated	29	33%	0.1842	42%	59%
10	close	29	41%	0.0801	65%	48%
	separated	29	22%	0.1929	35%	52%

(b)

	close top two		separated top two	
decision margin	rows	gain per model	rows	gain per model
at least 0.02	11	0.025	12	0.106
below 0.02	18	−0.013	17	−0.022

L.5 DOES THE GROUPING DEPEND ON THE 0.02 CUT? (POST HOC)

The regime rule of App. L.1 calls a design's opportunity limited when the probe keeps no gain and the held-out $\widehat{G}$ is below 0.02. We moved this cut to 0.01, 0.03 and 0.05 in two ways: on $\widehat{G}$ only, as the rule does, and also on the gain, so that a realized gain must reach the cut too (Table 29). The labels at 0.02 stay the paper's labels.

The eight designs. With the cut on $\widehat{G}$ only, designs move only between limited opportunity and opportunity offset by selection error. BFCL ($\widehat{G} = 0.013$) moves at 0.01. Harness-Bench and typed arguments ($\widehat{G} = 0.044$) move at 0.05. The labels are the same at 0.02 and 0.03. The three realized gains cannot move, because this rule asks nothing of the gain's size. If the gain must also reach the cut, only the strict parser keeps a realized gain at every cut.

The prospective comparison. This one does depend on the cut. With the cut on $\widehat{G}$ only, at 0.01 and 0.05, as at 0.02, reading 20 tasks did no better than copying the old label. At 0.03 the 20-task reading matched 4 of 4 suite designs and copying matched 3 of 4. The reason is one design: P-typed's held-out $\widehat{G}$, 0.022, lies between 0.02 and 0.03, while its source design's is 0.044. On BFCL the 20-task reading missed at every cut on $\widehat{G}$. We report the locked cut, 0.02.

An "uncertain" output. A future rule could say "uncertain" when its calibration tasks do not settle the label. Read this way with E127's reported intervals (post hoc, and not a hit), all 5 E127 labels are uncertain at 0.02: three achieved-gain intervals contain zero, and for the other two the oracle-gain interval crosses the cut. The labels are descriptive; the continuous $\widehat{G}$ and the achieved gain beside them are the result.

Table 29: **Moving the regime cut (post hoc).** (a) Each design's label at four cuts. lim.: limited adaptation opportunity; off.: opportunity offset by selection error; real.: realized adaptation gain. Bold: a label that differs from the frozen label at 0.02. Gain: the 10-task probe's achieved gain with ties to `json` (Table 3). "Cut also on the gain": a realized gain must be at least the cut. (b) E127: how many new designs' held-out regimes each reading matched at each cut, with the old labels re-read at the same cut. Generated by `experiments/review_20260925c_hd/make_selector_tables.py`.

			cut on $\widehat{G}$ only				cut also on the gain			
design	$\widehat{G}$	gain	0.01	0.02	0.03	0.05	0.01	0.02	0.03	0.05
(a) the eight designs, grouped by their label at 0.02										
limited adaptation opportunity										
our suite, strong models	0.001	-0.009	lim.	lim.	lim.	lim.	lim.	lim.	lim.	lim.
WorkBench	0.003	-0.014	lim.	lim.	lim.	lim.	lim.	lim.	lim.	lim.
BFCL (turn accuracy)	0.013	-0.009	**off.**	lim.	lim.	lim.	**off.**	lim.	lim.	lim.
opportunity offset by selection error										
Harness-Bench (graded)	0.044	-0.015	off.	off.	off.	**lim.**	off.	off.	off.	**lim.**
our suite, typed arguments	0.044	0.007	off.	off.	off.	**lim.**	off.	off.	off.	**lim.**
realized adaptation gain										
our suite, reasoning models	0.072	0.030	real.	real.	real.	real.	real.	real.	**off.**	**off.**
our suite, 4k-token budget	0.101	0.044	real.	real.	real.	real.	real.	real.	real.	**off.**
our suite, strict parser	0.144	0.105	real.	real.	real.	real.	real.	real.	real.	real.
(b) the prospective test (E127): matches with the held-out regime										
our suite, 20-task reading (of 4)			4	4	4	4	4	4	4	3
our suite, copying the old label (of 4)			4	4	3	4	4	4	3	3
BFCL, 20-task reading			miss	miss	miss	miss	miss	miss	hit	hit
BFCL, copying the old label			miss	hit	hit	hit	miss	hit	hit	hit

L.6 One common format against the pooled default (post hoc)

This analysis is **post hoc**. Its protocol was committed alone before any code (`experiments/review_20260925d_hd/PROTOCOL.md`). The scripts read cached cells only and make no model call (data: `data/review_20260925d_hd/`). Before scoring anything new, they reproduce $\widehat{G}$ and the $R(\pi)$ of the pooled default and of the 10-task probe on all eight designs (Table 3), and E129's primary gain. For E129, each format's mean on the history tasks was already printed when the protocol was written.

Why a second default. The pooled default is a selection rule, not one format: it leaves the incoming model out, so on 4% (WorkBench) to 88% (strict parser) of the splits it gives different models

different formats. A *common format* gives one format to every model. On each split it is the format with the best mean, over all models, on the split's history tasks. Unlike the pooled default, it also reads the incoming model's own history runs, so part of any advantage it has may come from that extra information. A gain counts as better or worse when it is at least 0.005 in size and more splits agree than disagree; otherwise there is no clear difference. This rule was fixed before the run.

The eight designs (Table 30a). (1) The common format, which also reads the incoming model's history runs, had lower regret than the pooled default on 6 of 8 designs and was never worse. It came within 0.024 of the hindsight best shared format on every design. On WorkBench it is `json` on every split, the paper's fixed `json`. (2) Against the common format the probe is better on 4 of 8 designs, the same four as against the pooled default. The probe gets a different label against the two defaults only on strong models: there it loses 0.009 to the common format but ties the pooled default, and under one of the two tie orders that tie becomes a loss. (3) Where the probe wins, its gain over the common format (which reads the incoming model's history runs) is smaller than over the pooled default: 0.126 against 0.197 on the strict parser, 0.069 against 0.138 with a 4k-token budget, 0.050 against 0.078 with reasoning models and 0.020 against 0.065 on typed arguments. The last label depends on the tie order: the gain runs from 0.004 to 0.028. The gain over the hindsight best shared format is smaller again (0.105, 0.044, 0.030 and 0.007).

The held-out BFCL families (Table 30b). On E129's frozen partition the common format, which again reads the incoming model's history runs, depends on the tasks that choose it. Chosen on the 25 history tasks it is `json`, ahead of `react` by 0.02 points. Against it the probe's gain was 0.29 points (95% interval $[-0.31,\ 0.89]$), the same conclusion as against the pooled default (0.06 points, $[-0.65,\ 0.78]$); this row is E129's contrast of the probe over fixed `json`. Chosen on all 35 calibration tasks or on the 40 older BFCL tasks, it is `react`. Against it the probe was 1.26 points worse ($[-2.19,\ -0.33]$) and the pooled default 1.32 points worse ($[-2.44,\ -0.21]$). This format makes the same choices as E129's selector (b+), whose contrast E129 printed without an interval. Every interval here is a t interval whose coverage passed E129's check (0.93 to 0.97 under both generators).

What this shows. The choice of default changes the size of the probe's gain, and on the held-out BFCL families the tasks used to choose the default change its sign. So a claim that probing helps, or does not, holds against a named default, not against defaults in general. No frozen label, regime or locked verdict changes.

Part IV — The prospective study, the new-template test and the held-out-family test

M THE THREE REGIMES, TESTED PROSPECTIVELY ON NEW DESIGNS (E127)

The main text reads eight designs into three regimes with a rule written after Table 3 had been seen (App. L.1). That reading explains measured outcomes. It does not show that the rule can tell the regime of a design nobody has measured yet. E127 asks one question: if we read the regime of a new design from 20 calibration tasks, does it match the regime we then read from held-out tasks collected afterwards? We saved the protocol, `experiments/127_prospective_regime/protocol.md`, in its own git commit before any new model ran a task, and the five predictions before any held-out call. Code: `experiments/127_prospective_regime/`; data: `data/e127_prospective_regime/`.

Five new designs. Each new design copies the configuration of one design the paper already reports and changes two things: the models, and the decoding seed (127, so every call is a fresh draw). Table 32(a) names the design each one copies. The first four are the suite designs. P-bfcl copies BFCL v3 multi-turn (its first 40 tasks, scored as turn accuracy). The suite designs use all 52 tasks of their suite. P-strict, P-4k, P-typed and P-bfcl run seven new chat models under four formats; P-strong runs four new strong models under `json` and `react`. That is $5{,}904$ cells in all. We checked

Table 30: **One common format against the pooled default (post hoc).** (a) Selection error $R(\pi)$ at 10 probe tasks per format, the mean over the 2,000 held-out splits of Table 3, and the paired gains: the first rule's mean held-out score minus the second's. Common format: one format for every model, chosen on the split's history tasks of all models, the incoming one included. Bold: better or worse under the label rule fixed before the run; *: the label changes under some tie order. (b) E129's frozen partition, in turn-accuracy points: the common format chosen on each declared set of tasks, its lead over the second format, its held-out regret, and the gains over it with their validated t intervals. Generated by `experiments/review_20260925d_hd/make_common_table.py`.

(a) The eight designs

		selection error $R(\pi)$			paired gain		
design	$\widehat{G}$	common format	pooled history	10-task probe	probe over common	probe over pooled	common over pooled
limited adaptation opportunity							
our suite, strong models	0.001	0.001	0.011	0.010	**−0.009**	0.000*	**0.009**
WorkBench	0.003	0.003	0.005	0.016	**−0.014**	**−0.011**	0.003
BFCL (turn accuracy)	0.013	0.022	0.023	0.022	0.000	0.001	0.001
opportunity offset by selection error							
Harness-Bench (graded)	0.044	0.045	0.051	0.060	**−0.015**	**−0.009**	**0.006**
our suite, typed arguments	0.044	0.057	0.102	0.037	**0.020***	**0.065**	**0.044**
realized adaptation gain							
our suite, reasoning models	0.072	0.092	0.120	0.042	**0.050**	**0.078**	**0.027**
our suite, 4k-token budget	0.101	0.126	0.195	0.057	**0.069**	**0.138**	**0.070**
our suite, strict parser	0.144	0.165	0.236	0.039	**0.126**	**0.197**	**0.070**

(b) The held-out BFCL families (turn-accuracy points, 64 held-out tasks)

common format chosen on	format (lead)	regret	probe over common	pooled history over common
25 history tasks (primary)	`json` (0.02)	2.60	0.29 [−0.31, 0.89]	0.22 [−0.18, 0.62]
all 35 calibration tasks	`react` (0.38)	1.05	−1.26 [−2.19, −0.33]	−1.32 [−2.44, −0.21]
BFCL's 40 older tasks	`react` (0.56)	1.05	−1.26 [−2.19, −0.33]	−1.32 [−2.44, −0.21]
for comparison, the probe over the pooled-history rule (E129's primary): 0.06 [−0.65, 0.78]				

each copied configuration without any model call: re-run on the cached responses, it reproduces every stored cell we could re-run (1,456 of 1,456 on the strict parser). The driver passes every collector parameter explicitly.

New models. A rule written before any candidate was called chose them from OpenRouter's public list of 25 September 2026: base, text-only, paid, general-purpose models that are not reasoning models, accept temperature 0, have at least 16,384 tokens of context, state no size below 7B, and had not run our suite; newest first. A completion price of at least USD 2 per million tokens put a model in the strong pool, because price was the only strength signal available before any outcome. The chat pool allowed at most two models per organisation and the strong pool one. We called each candidate at most once, in list order, and took the first seven chat and first four strong models that answered. Chat pool: ministral-14b-2512, ministral-8b-2512, qwen3-next-80b-a3b-instruct, qwen3-30b-a3b-instruct-2507, gpt-4.1-mini, gpt-4.1-nano and llama-4-maverick. Strong pool: palmyra-x5, kimi-k2-0905, gpt-4.1 and command-a (nova-premier-v1 failed its call: the provider reported the model at end of life). None of these models has a stored cell on our suite or on BFCL from before E127; two near cases are listed at the end of this appendix.

Split and rule. For each design, a seeded permutation of the sorted task ids gives 10 history, 10 probe and the remaining held-out tasks: 32 on the suite designs and 20 on P-bfcl. The three designs on the same task suite (P-strict, P-4k, P-strong) share one split, so no model met a held-out task of any design before the predictions were committed. The rule reads the 20 calibration tasks only. (1) Each model's format is its best format on its 10 probe tasks. Ties go to the first format in an order of the formats; the rule keeps all 24 orders (2 on P-strong), and the choices are then frozen. (2) The predicted oracle gain G_{pred} is the mean $\widehat{G}$ of 2,000 tables of held-out size drawn with replacement from the 20 calibration tasks. (3) The predicted achieved gain A_{pred} is the mean gain of the frozen choices over the best shared format on 2,000 such tables drawn from the 10 history tasks only, so the choices are not scored on the tasks that made them. (4) The paper's own grouping rule, imported unchanged from `experiments/review_20260924_hd/derived_readings.py`, turns the two into a regime: a realized adaptation gain if the gain is above zero with ties to `json` and under

every order; otherwise limited adaptation opportunity if G is below 0.02; otherwise opportunity offset by selection error. The held-out regime comes from the same rule on the held-out tasks: G_O is their $\widehat{G}$, and A_O is the frozen choices' gain on them. A design is a hit only when the two labels are the same. The reported interval of each prediction resamples the calibration tasks first and then the held-out-sized table, because the calibration tasks are themselves a sample.

Gate, completeness and time order. Every design passed the validity gate on its 20 calibration tasks, which fixed the scored set before any held-out call, and again on all its tasks. The pooled outcome and the no-parsed-call rate were 0.7335 and 0 (P-strict), 0.7253 and 0 (P-4k), 0.8510 and 0.0014 (P-typed), 0.7981 and 0 (P-strong), and 0.3665 and 0.0036 (P-bfcl, turn accuracy). All $3{,}504$ held-out cells were collected; no model was excluded and no task removed. The scorer reads the per-call logs and checks the order in time: each of the $10{,}539$ fresh calls of the calibration stage returned before the predictions were committed, each of the $15{,}532$ fresh held-out calls was sent after, and no call was served from an older cache. The held-out driver refused to run unless the committed predictions matched the sha256 stated in a dated addendum. E127 cost USD 2.88 in model calls.

Results. Table 32 gives each design. On the source designs, whose regimes were already known, the same rule matched the regime read from the held-out tasks of the same partition on 63% to 92% of random partitions on our suite and on 55% on BFCL; copying the known regime matched on 78% to 99% and on 81%. The held-out regime is read from one table of 32 / 20 tasks; on the source designs this one-table reading differed from the design's averaged regime on up to 22% of partitions. Two more results, in the protocol's fixed wording: The held-out G fell inside its reported interval on 4 of 4 suite designs and inside it on BFCL; the held-out achieved gain fell inside on 4 of 4 suite designs and inside on BFCL. On 4 of 4 suite designs but not on the BFCL design, the regime read from 20 tasks agreed with the paper's split-averaged reading of the same design over all its tasks. That split-averaged reading is the paper's own procedure (App. L.1) run on all tasks of the new design, so it is not prospective. On the five source designs, the same code gives the $\widehat{G}$, achieved gain and regime of their rows in Table 3 exactly. The protocol keeps two more results out of the paper, the regret of three decision policies on the held-out tasks and a one-level percentile interval; they are in `e127_scored.json`.

Table 31: **The prospective test at a glance.** Held-out gain of the frozen probe choices over the best shared format (ties to `json`); old, predicted and held-out regimes; P-bfcl in turn accuracy. More: Table 32.

design	source design	held-out gain	tie-order range	old regime	predicted	held-out
P-strict	strict parser	$+0.071$	$[+0.063, +0.080]$	realized	realized	realized
P-4k	4k-token budget	$+0.076$	$[+0.076, +0.085]$	realized	realized	realized
P-typed	typed arguments	-0.027	$[-0.036, +0.018]$	offset	offset	offset
P-strong	strong models	0.000	0.000	limited	limited	limited
P-bfcl	BFCL	-0.009	-0.009	limited	realized	limited

The miss, and the hit at the cut (post hoc). These checks were written after scoring (`posthoc_mechanism.py`, `e127_posthoc_mechanism.json`). They describe; they rescore nothing.

P-bfcl missed because a small gain on the calibration tasks turned negative on the held-out tasks. The prediction said realized adaptation gain because the frozen choices gained $+0.0100$ on tables drawn from the 10 history tasks, under every tie order; no probe choice was tied. Of the gain on the 10 history tasks themselves, $+0.0107$, one model made $+0.0071$: gpt-4.1-mini scored 0.050 higher in `pythonic` than in `react` there. On the 20 held-out tasks `react` was the best shared format. Five of the seven models had been sent to another format, and four of those five did worse there (ministral-14b -0.035, llama-4-maverick -0.023, gpt-4.1-mini -0.0125, qwen3-next-80b -0.0125); only gpt-4.1-nano gained ($+0.021$). So A_O was -0.0089 under every order, and G_O, at 0.0071, was below the cut: limited adaptation opportunity. The held-out label is stable: no single held-out cell, changed alone, alters it. The prediction sat near both cuts: its gain interval already spanned zero, and G_{pred} was only 0.003 above the cut.

Table 32: **E127: the regime read from 20 calibration tasks, and the regime of the held-out tasks collected afterwards.** (a) Source: the configuration each new design copies. Old: the regime the paper reads on that configuration, the copy baseline. Predicted: read from the calibration tasks. Held-out: read from the 32 (20 on P-bfcl) held-out tasks with the frozen probe choices. Split-avg.: the paper's own reading of the new design over all its tasks, with its mean held-out $\widehat{G}$; not prospective. Realized, offset and limited: realized adaptation gain, opportunity offset by selection error, limited adaptation opportunity. (b) G_{pred} and A_{pred}: the predictions, with their reported intervals; (min): the smallest value over the tie orders (24; 2 on P-strong). G_O and A_O: the same quantities on the held-out tasks, ties to `json`. Turn accuracy on P-bfcl, task success elsewhere.

(a) Regimes

design	source	old	predicted	held-out	split-avg. ($\widehat{G}$)	result
P-strict	strict parser	realized	realized	realized	realized (0.1064)	hit
P-4k	4k-token budget	realized	realized	realized	realized (0.1046)	hit
P-typed	typed arguments	offset	offset	offset	offset (0.0230)	hit
P-strong	strong models	limited	limited	limited	limited (0.0019)	hit
P-bfcl	BFCL	limited	realized	limited	limited (0.0115)	**miss**

(b) Oracle gain and achieved gain

design	G_{pred} [interval]	G_O	A_{pred} [interval] (min)	A_O (min)
P-strict	0.1267 [0.0670, 0.1875]	0.0982	+0.0848 [−0.0224, +0.1562] (+0.0413)	+0.0714 (+0.0625)
P-4k	0.1311 [0.0759, 0.1786]	0.0893	+0.0861 [−0.0314, +0.1473] (+0.0861)	+0.0759 (+0.0759)
P-typed	0.0250 [0.0000, 0.0714]	0.0223	−0.0568 [−0.1652, 0.0000] (−0.0568)	−0.0268 (−0.0357)
P-strong	0.0009 [0.0000, 0.0312]	0.0000	0.0000 [0.0000, 0.0000] (0.0000)	0.0000 (0.0000)
P-bfcl	0.0230 [0.0064, 0.0483]	0.0071	+0.0100 [−0.0179, +0.0286] (+0.0100)	−0.0089 (−0.0089)

P-typed's hit sits at the cut. Its G_O is 0.0223, only 0.0023 above 0.02, and three models make all of it (gpt-4.1-nano, qwen3-next-80b and qwen3-30b). One held-out cell moves a model's format mean by $1/32$, which can move G by up to 0.0045, more than the margin. Of its 896 held-out cells, 79 would each, changed alone, turn the label into limited adaptation opportunity, a miss. Its negative gain comes from ties on the probe tasks. Four models scored the same in `json` and `react` there, and the `json`-first order sent them to `json`; on the held-out tasks `json` trailed `react` for all four, by 0.031 to 0.156. No single held-out cell, changed alone, changes the label of P-strict, P-4k or P-strong.

So the empty gap in held-out $\widehat{G}$ around the cut 0.02 (App. L.1) holds for the eight designs only. P-typed's split-averaged $\widehat{G}$, 0.0230, lies inside it.

What E127 does not show. Five designs, three task families, one gateway, one day. The designs are related, not independent replications: P-strict, P-4k, P-typed and P-bfcl share the same seven models, and the three designs on one task suite share their held-out tasks. The models are new to these tasks, not new to the world: qwen3-next-80b-a3b-instruct was once called on our suite through a free route that returned only errors, and the thinking versions of two of the new models are among the reasoning models of our suite. The regimes of the source configurations were known, so some agreement comes from the configuration alone; this is why copying the old label is the baseline. BFCL was written by other people, but it is already one of the paper's eight designs, so P-bfcl is not a new benchmark. With four suite designs and one BFCL design, these are counts, not rates. Nothing here tests a new task family or a new benchmark.

N FROZEN CHOICES ON NEW TASK TEMPLATES (E135)

The paper's positive case is the strict-parser design, and its prospective copy with new models gained on held-out tasks of the same suite (App. M). But all 52 suite tasks use a small set of arithmetic patterns, so a reader can ask whether the frozen choices learned something about the models or only about these patterns. E135 asks one question: does a frozen selector keep its advantage on new tasks, written before any model ran them, outside the templates that produced the positive result? A negative answer was declared as useful as a positive one. The task specification (`task_spec.md`), the frozen choices (`frozen_choices.json`) and the protocol (`protocol.md`) were each com-

mitted before any E135 task was sent to a model; the protocol was committed alone. Code: `experiments/135_new_templates/`; data: `data/e135_new_templates/`; tables and the numbers the paper prints: `make_tables.py`. Gains are in success-rate points (success rate $\times 100$).

Scope, in the protocol's words. E135 keeps E127's strict-parser configuration (7 chat models, 4 formats, the suite's fictional database and tools) and changes only the questions: 96 tasks from 24 new templates whose arithmetic differs from every suite question. 11 templates add an operation the suite never uses, such as the money left over after buying whole units or the smaller of two prices, and 13 combine suite operations in new ways. Every format choice was frozen from E127's cells before any E135 task was sent to a model. Under json-first ties, the probe and the pooled-history selector chose the same format (json) for five of the seven models, so the comparison rests on two models: qwen3-30b-a3b (xml against json) and gpt-4.1-nano (react against json).

Where the sentences are printed. The protocol asks the main text for twelve sentences. They take about 30 lines, and five of them name experiments by code. So §6.1 prints two of them verbatim (the primary sentence with its no-interval sentence, and what a hit shows), states the scope and the two carrying models in plain words with the same numbers, and says what P2 predicted and that it missed, without the two gaps; Table 4 carries the gain, the hindsight gain, the test and the tie-order range. The same-number, mechanism, P1, serving and headroom sentences and the P2 gaps are printed only here. This appendix prints every licensed sentence verbatim (`addendum_20260925d_placement.md`).

The 24 templates. A template is a question pattern and an answer rule; its *shape* is the arithmetic of its answer (Table 33). The 11 new-operation templates use a remainder, rounding up, the larger or smaller of several values, or a rule with two branches, none of which any suite shape uses; the 13 others combine the suite's operations in a way no suite shape does. The question wrapper, the database (18 cities, 12 items), the two tools, the scoring rule, the budget of 6 replies and the strict parser are the suite's. Each template has 4 instances. An instance takes the first seeded draw whose answer is at least 2, whose template condition holds (for example, money left over is positive), whose branch alternates with the other instances in the five templates with two branches, and whose entities are new within the template; 21 of the 96 needed more than one draw. The templates need 1, 2, 3 and 4 lookups in 5, 12, 6 and 1 cases (mean 2.125 lookups per task, against 2.16 on the 32 held-out suite tasks).

Novelty is checked by function, not by string. Every shape is an expression. Each of the 52 suite shapes, bound to its own question, must give that task's answer, and each new shape must give its 4 tasks' answers. Two shapes with the same number of lookups count as the same shape when they agree on 200 random database-like draws under some matching of their looked-up values (every permutation) and of their question numbers (every one-to-one pairing). The check fails if a new shape equals a suite shape or another new shape. It also injects six disguised suite shapes, such as $y + x$ and $\mathrm{round}(y/x, 2)$, and fails unless it catches all six. A scripted agent, with no model and no network, solves all 96 tasks in all four formats under the strict parser within 6 replies (384 runs).

What the author knew. One author wrote the templates after reading E127's results, including that its gain came from two models, and knowing that the `react` and `pythonic` parsers cut a tool argument at its first ")". No template was tried on a model or changed after one. Two failure mechanisms were measured on E127's cells after the templates were written, and the templates were kept. The one effect of this knowledge we can count works against the probe: the new set has twice the suite's share of tasks whose answer has a grouping parenthesis (32 of 96, against 5 of the 32 held-out suite tasks), which hurt gpt-4.1-nano's `react` choice on the suite.

Frozen choices. Four selectors, all frozen from E127's committed strict-parser cells, with its 10 history tasks H and 10 probe tasks P. (a) The 10-task probe: each model's best format on P. (b) The pooled-history rule, the primary comparator: the best format of the other six models, pooled on H (240 episodes); the protocol calls it the pooled-history selector. (c) One common format, chosen once on H over all seven models. (d) Fixed `json`. Under ties to `json`, (b), (c) and (d) give `json` to every model, and (a) differs from (b) only for qwen3-30b-a3b (`xml`) and gpt-4.1-nano (`react`). Each of the 24 tie orders is applied to (a), (b) and (c) together; under them 2 to 7 models differ. Code checks that (a) equals E127's committed probe choices under all 24 orders, that (b) equals the

Table 33: **E135: the 24 templates.** Answer shape (x, y, z, w: looked-up values; q, q_1, q_2, B, k, X: numbers in the question); class (op: an operation no suite shape uses; comp: a new composition of suite operations); lk.: lookups; calc: needs a calc call; ( ): a grouping parenthesis; gain: the frozen probe over the pooled-history rule on the template's 4 tasks, in points (json-first ties).

#	template	answer shape	class	lk.	calc	( )	gain	#	template	answer shape	class	lk.	calc	( )	gain
1	change	`B-q*x`	comp	1	y	n	+7.1	13	split	`floor(B/2/x)+floor(B/2/y)`	comp	2	y	n	0.0
2	leftover	`B%x`	op	1	y	n	+10.7	14	sets	`floor(B/(x+y))`	comp	2	y	y	−14.3
3	rule1	`if(x>X,q1*x,q2*x)`	op	1	y	n	0.0	15	wavg	`round((q1*x+q2*y)/(q1+q2),2)`	comp	2	y	y	+7.1
4	discount	`round(q*x*(100-k)/100,1)`	comp	1	y	y	0.0	16	pctmax	`round(k*max(x,y)/100)`	op	2	y	n	+7.1
5	ceil	`ceil(B/x)`	op	1	y	n	+25.0	17	maxcost2	`max(q1*x,q2*y)`	op	2	y	n	+7.1
6	buythen	`floor((B-q*x)/y)`	comp	2	y	y	0.0	18	max3	`max(x,y,z)`	op	3	n	n	0.0
7	buydiff	`floor(B/x)-floor(B/y)`	comp	2	y	n	+3.6	19	range3	`max(x,y,z)-min(x,y,z)`	op	3	y	n	+7.1
8	pctsum	`round(k*(x+y)/100)`	comp	2	y	y	−7.1	20	top2sum	`x+y+z-min(x,y,z)`	op	3	y	n	+14.3
9	share	`round(100*x/(x+y),1)`	comp	2	y	y	−3.6	21	ratioprod	`round(x/y*z,1)`	comp	3	y	n	0.0
10	cheaper	`q*min(x,y)`	op	2	y	n	+10.7	22	ifpop	`if(x>y,q*z,q*w)`	op	3	y	n	+3.6
11	pctdiff	`round(100*(x-y)/y,1)`	comp	2	y	y	−3.6	23	avgprice3	`round(q*(x+y+z)/3,1)`	comp	3	y	y	+3.6
12	unitswap	`round(q*x/y,2)`	comp	2	y	n	+7.1	24	min4	`min(x,y,z,w)`	op	4	n	n	+7.1

paper's pooled-history rule under ties to `json` and (c) E127's pooled default, and that the gain of (a) over (b) on E127's 32 held-out tasks is 7.6 points, the first row of Table 4.

Run, gate and time order. The collector, strict parser, seed (127), decoding and budget are E127's; only the task builder changes. All 2,688 cells (7 models × 4 formats × 96 tasks) were collected in one pass per model, with no provider error, no missing cell and no excluded model or task (13,600 fresh calls). The scorer checked that every logged call, every pass start and every cell file is later than the protocol commit and the self-test commit. The validity gate passed: pooled success 0.567 and a no-parsed-call rate of 0 in every format. Mean success fell from E127's held-out tasks to the new tasks in every format (`json` 0.741 to 0.640, `react` 0.741 to 0.592, `xml` 0.746 to 0.574, `pythonic` 0.656 to 0.460), so the new questions are harder.

How many tasks, and what they can resolve. The protocol declared 3 points the smallest meaningful gain, less than half of E127's two held-out gains (7.1 and 7.6). The template is the unit (24 template means), because on E127's held-out tasks the between-template share of the variance of the paired difference was 0.39. With E127's variance components the template-level standard error at 24×4 is 0.0175: a template-level test detects about 5 points with 80% power (0.78) and cannot resolve 3 points (0.37). If both known failure mechanisms carried over unchanged, the gain would have been about 6.2 points.

The primary test, and why it has no interval. Before any call, four interval methods (t, bootstrap-t, BCa and percentile) were run on four synthetic 24×4 populations built from E127's held-out tasks, under two generators, and a lock rule took the first method whose coverage lay in $[0.93, 0.97]$ everywhere. None did: every method covered one population only 0.90 to 0.92 of the time. So no interval was locked, and P1 is scored by the exact one-sided sign-flip test over the 24 template means (all 2^{24} sign vectors), which is exact if the 24 template gains are independent and each symmetric about zero. In data simulated from ours with no gain but a few heavy losses, a p this small occurred in up to 6.9% of datasets (at most 1.6% when the templates were shaped like the 24 new ones), so $p = 0.0128$ overstates the evidence (App. Q). In planning, with true gains of 4.9 to 6.3 points, this test gave $p < 0.025$ in 57% to 69% of datasets, so a miss would have been weak evidence that the gain was gone. The protocol's main-text sentences, verbatim and in its order: On 96 tasks from 24 new templates, the frozen 10-task probe beat the pooled-history selector by 3.9 points (exact one-sided sign-flip test over the 24 template means, $p = 0.0128$). The advantage seen on the suite's templates carried over to these 24 new templates. We give no interval for this gain: in planning, no interval method reached its coverage target on every simulated population (appendix). Under json-first ties the gain is one seventh of two models' gains: qwen3-30b-a3b gained 19.8 points with xml over json (+56.3 on E127's held-out tasks) and gpt-4.1-nano gained 7.3 points with react over json (−3.1 on E127's). Under json-first ties the one common format chosen on E127's history tasks and fixed json are also json for every model, so the same number is the gain over them. Applying each of the 24 tie orders to both selectors, the gain ranged from 3.4 to 6.5 points. On the 88 tasks that need a calc call the gain was 3.9 points, and on the 8 that do not it was 3.6; on the 32 tasks whose answer has a grouping parenthesis it was −2.2, and on the other 64 it was 6.9. 27 of qwen3-30b-a3b's 41 failed json episodes contain an object with the tool name as the key ({"calc": ...}), the error seen

on E127's tasks (31 of 37). We predicted that the probe would beat the pooled-history selector on the new templates; it did (the exact sign-flip test gave $p = 0.0128$). Because the gain comes mainly from qwen3-30b-a3b's json error, this result shows that a model-level formatting error persists on new tasks; it does not show that the probe learns task-dependent preferences. We predicted that qwen3-30b-a3b's xml-over-json gap would be at least 0.30 larger on tasks that need a calc call than on tasks that do not; the gaps were 0.19 and 0.25, a difference of -0.06 (P2 miss). Serving was not pinned; on the 61% of model-task pairs where both formats were served by the same provider, the gain was 2.7 points. Against the best single format picked in hindsight on the new tasks (json), the probe's gain was 3.9 points; on E127's held-out tasks it was 7.1 points. The empirical headroom on the new tasks was 5.2 points, against 2.8 points (95th percentile 4.3) for tables with the format labels shuffled (E127's held-out tasks: 9.8).

The 61% is over the two carrying models' 192 (model, task) pairs, because the other five models add exactly zero: gpt-4.1-nano was served by one provider in every cell and keeps all 96 tasks, and qwen3-30b-a3b, served by five providers, keeps 22. On its 22 tasks qwen3-30b-a3b gained 11.5 points; with so few tasks this is noisy.

Intervals, printed as descriptive. The t interval, printed as descriptive because no interval method reached its coverage target on every planning population, was $[0.6, 7.1]$; on the E135 population it covered 0.938 and 0.948 of simulated datasets under the two generators. It would cover the sampling of templates like these 24 and of tasks within them, treating the 7 models and their frozen choices as fixed. Its coverage on the E135 population lies in the band, but the lock was decided in planning, so it stays descriptive; the bootstrap-t, BCa and percentile intervals are $[0.7, 7.2]$, $[0.9, 7.0]$ and $[1.0, 7.1]$. With templates fixed and tasks resampled within them, the 95% interval was $[1.9, 5.8]$ (validated). It covered 0.948 under both within-template generators; the stratified percentile interval, $[2.2, 5.5]$, covered 0.923 and is not validated. Its target is narrower than the primary's: new tasks from these same 24 templates, not new templates.

Spread by model and by template. Table 34 gives each model. Five models get the same format from (a) and (b), so they add exactly 0, and the gain is $(19.8 + 7.3)/7$. Across the 24 templates the gain ranged from -14.3 to 25.0 points; leaving out any one template gave overall gains from 3.0 to 4.7 points. 14 templates gained, 6 were exactly 0 and 4 lost (Table 33). Leaving out the 8 percentage-answer tasks (share, pctdiff), the gain was 4.5 points.

Table 34: **E135: each model.** Frozen choices of the probe (a) and the pooled-history rule (b), json-first ties; the gain of (a) over (b) on the 96 new tasks, in points, with E127's 32 held-out suite tasks in parentheses; and the success rate of each choice on the new tasks and on the suite's held-out tasks.

model	(a) probe	(b) pooled history	gain of (a) over (b)	(b)'s format		(a)'s format	
			new tasks (suite)	new	suite	new	suite
ministral-14b-2512	`json`	`json`	0.0 (0.0)	0.792	0.906	0.792	0.906
ministral-8b-2512	`json`	`json`	0.0 (0.0)	0.688	0.844	0.688	0.844
qwen3-next-80b-a3b-instruct	`json`	`json`	0.0 (0.0)	0.771	0.875	0.771	0.875
qwen3-30b-a3b-instruct-2507	`xml`	`json`	+19.8 (+56.3)	0.573	0.281	0.771	0.844
gpt-4.1-mini	`json`	`json`	0.0 (0.0)	0.760	0.875	0.760	0.875
gpt-4.1-nano	`react`	`json`	+7.3 (−3.1)	0.333	0.656	0.406	0.625
llama-4-maverick	`json`	`json`	0.0 (0.0)	0.562	0.750	0.562	0.750

The three locked predictions. P1, that the probe beats the pooled-history rule on the new templates, hit, and P2, the mechanism prediction, missed (both sentences above). The regime label read on the new tasks was realized adaptation gain (P3 hit); on E127's held-out tasks it was realized adaptation gain. P3 uses the paper's own label rule (App. L.1): the label is kept when the gain over the hindsight format is above zero both with ties to `json` and under the least favourable tie order (3.9 and 0.9 points).

Why P2 missed (post hoc). These counts were computed after scoring (`posthoc_e135.py`, `e135_posthoc.json`); they describe and re-score nothing. P2 split the tasks by whether the reference plan needs a calc call, but the model does not follow that plan. On the 8 tasks that need none (`max3`, `min4`), qwen3-30b-a3b in `json` made a calc attempt in 6 of 8 episodes, 4 of them

with the tool name as the key. On `min4` it tried to add up the values it had looked up, a call it did not need, and repeated the malformed call until its 6 replies ran out (2 of 4 tasks); `xml` solved all 8. So the no-calc gap (0.25) rests on 2 tasks and contains the very error P2 assumed it could not contain. The calc gap was small too (0.19), because the error became rarer. The tool-name key was qwen3-30b-a3b's first calc attempt on 29 of the 96 new tasks (30%), against 17 of E127's 32 held-out tasks (53%). It never appeared on a task with one lookup (0 of 20 new tasks; 0 of 11 suite tasks), and it appeared on 29 of the 76 new tasks with two or more lookups, against 31 of 41 suite tasks. We do not know why the rate halved; the error appeared at all five upstream providers that served this model's `json` cells. This is why its gain fell from 56.3 to 19.8 points: its `json` success rose from 0.281 to 0.573, while `xml` went from 0.844 to 0.771.

gpt-4.1-nano's `react` choice went from −3.1 to +7.3 points. The parser mechanism carried over: on the 32 grouping-parenthesis tasks `react` solved 6 and `json` 12, and 12 of its `react` cells had a calc argument cut at the first ")". The sign flipped because its `json` fell further: `json` success halved (0.656 to 0.333), and 49 of its 64 failed `json` episodes are wrong final answers, not format errors. Over all models, 235 `react` and 212 `pythonic` cells of 672 had a cut calc argument (E127's suite: 74 and 50 of 364), as expected with twice the share of grouping-parenthesis tasks.

Headroom and its floor. The empirical headroom $\widehat{G}$ is a maximum of noisy means, so it is biased upward. The floor comes from 1,000 tables in which the four format outcomes of every model and task are permuted at random. The measured 5.2 points lies above the floor's 95th percentile (4.3), by 0.9 points.

Cost. The probe costs each incoming model 40 episodes (4 formats × 10 tasks); the pooled-history selector reads 240 episodes of the other six models and costs no new episode. E135 cost USD 0.60 in model calls.

What E135 does not show. The questions are new, but the database, the two tools, the question wrapper and the scoring rule are the suite's, so E135 tests new arithmetic, not new domains, tools or benchmarks. Five of the seven models get the same format from both selectors, so the result is about two models, mainly one model's formatting error. The models and their frozen choices are E127's, so this is a new-task test, not a new-model test. One author wrote the templates after reading E127's results. Serving was not pinned, and on the pairs served by one provider the gain was smaller. A rerun with a corrected parser and one fixed provider per model is in App. P. The gain is about half of E127's, and at 24 templates a gain of 3 points could not have been resolved.

O Does the parser make the gain? An audit of cached replies (post hoc)

The problem. Our `react` and `pythonic` parsers read a tool argument with the pattern `[^)]*`, so they stop at the first ")". This holds for the permissive parser and for the strict parser. A valid call such as `Action: calc((1+2)*3)` reaches the calculator as `(1+2`, and the calculator returns an error. The `json` and `xml` parsers have no such cut. So a reader can ask how much of a format effect comes from the model and how much from our harness. The typed argument design uses its own parser, which reads to the end of the line and has no cut.

What we did. We wrote a corrected parser that matches parentheses and quotes, and re-read every cached reply of three strict-parser designs with it: the strict-parser design (1,456 cells), its copy with new models (1,456) and the new-template test (2,688). We did not re-read the 4k-token, reasoning or strong-model designs or their copies, which also use the strict parser. No model was called. The protocol (`experiments/review_20260926_hd/PROTOCOL.md`) was committed alone before any count was made; the analysis is post hoc, because the data had been seen. Code: `fixed_parser.py`, `test_fixed_parser.py` and `parser_audit.py` in the same directory; data: `data/review_20260926_hd/parser_audit.json`.

Checks. The corrected parser passes 17 contract tests: nested parentheses, ")" inside quotes, escaped quotes, multiline values, typed arguments and the first-action rule. Replaying each cell's

recorded replies through the old parser and the unchanged tools gives the stored success, number of steps and final answer in every cell of all three designs. On the 24,994 replies that contain none of the constructs above, the two parsers read every reply the same way.

Four kinds of failure. Each reply is put in one class: (i) the model wrote an invalid call; (ii) the model wrote a valid call and our harness misread it; (iii) the call was read correctly, but the tool rejected its arguments; (iv) every call was read and ran, and the task still failed. A failed episode is counted under the first problem it met. Table 35 gives the counts.

Table 35: **The first-")" cut in three strict-parser designs (post hoc).** Cells with a calc argument cut at its first ")"; failed episodes, with those whose first problem was a harness misparse in parentheses; success as measured; and bounds on success under the corrected parser (the replay is exact up to the first reply whose tool response would change; after it the outcome is set to failure or to success).

design	format	cells	cut at ")"	failed (misparse first)	success as measured	[lower, upper]
new templates	`json`	672	0	242 (0)	0.640	[0.640, 0.640]
	`react`	672	235	274 (129)	0.592	[0.528, 0.769]
	`xml`	672	0	286 (1)	0.574	[0.574, 0.574]
	`pythonic`	672	212	363 (122)	0.460	[0.406, 0.622]
suite, new models	`json`	364	0	85 (0)	0.766	[0.766, 0.766]
	`react`	364	74	92 (41)	0.747	[0.723, 0.841]
	`xml`	364	0	91 (0)	0.750	[0.750, 0.750]
	`pythonic`	364	50	120 (38)	0.670	[0.657, 0.766]
strict-parser design	`json`	364	0	193 (0)	0.470	[0.470, 0.470]
	`react`	364	61	183 (40)	0.497	[0.494, 0.533]
	`xml`	364	0	228 (0)	0.374	[0.374, 0.374]
	`pythonic`	364	24	179 (12)	0.508	[0.508, 0.533]

What the counts show. The cut occurs only in `react` and `pythonic`. On the new templates it hits every model, from ministral-8b (49 `react` and 48 `pythonic` cells) to gpt-4.1-nano (15 and 13). It is a handicap of these two formats that all models share, larger for models that write grouped arithmetic. Not every cut matters: on the new templates the corrected argument runs in 292 of 411 cut `react` replies and 282 of 383 cut `pythonic` replies; in the rest it fails anyway, because the model used something the calculator does not support, such as `max`, `round` or `//`. On the strict-parser design most cuts change nothing: the corrected argument fails the same way in 105 of 124 cut `react` replies. The one `xml` cell with a misparse on the new templates has an escaped "<" in a calc argument; decoding it does not change the tool's error.

The two carrying models. qwen3-30b-a3b's `xml` and `json` cells have no cut, so its gain of 19.8 points on the new templates is exact under the corrected parser. Its `json` failures are its own: in all 27 new-template failures whose first problem is an invalid call, the call uses the tool name as the key. gpt-4.1-nano's `json` has no cut and its `react` has 15 cut cells, all in failed episodes, so the correction can only raise its `react` gain. So the cut works against the frozen choices, not for them.

Bounds, not corrected results. An episode changes after a corrected tool response, and we do not know what the model would have written next. So a replay can only bound the gains. With the frozen choices, the gain over the pooled-history rule lies between 3.9 and 5.7 points on the new templates (3.9 as measured) and between 7.1 and 7.6 points on the 32 held-out suite tasks of the new-model copy (7.6 as measured). The replay also cannot redo the choices. The probe's choice stays the same for 6 of 7 models; qwen3-30b-a3b's `react` could tie its `xml` on the 10 probe tasks, and the tie order would then pick `react`. With that choice the new-template gain would lie between -0.3 and 4.9 points. Only a rerun can settle this, which is App. P.

P A RERUN WITH A CORRECTED PARSER AND FIXED PROVIDERS (E136)

Question. When every format is parsed faithfully and each model is served by one fixed provider, do the frozen choices of the new-template test (App. N) still beat `json` on its 96 tasks? The protocol (`experiments/136_parser_rerun/protocol.md`) was committed alone, after the audit

above and before any E136 code or model call, with three predictions. It tests the frozen choices only: the probe and the pooled-history rule were not rerun.

Design. The two models whose frozen choice is not `json`: qwen3-30b-a3b (frozen choice `xml`) and gpt-4.1-nano (`react`); the other five models choose `json` under both selectors and add exactly 0. Four formats, the 96 tasks and the first run's system prompts, tools, seed (127), temperature 0, 400 output tokens and 6 replies per episode. The corrected parser above replaces the strict parser for all four formats and keeps the first-action rule; nothing else in the agent changes. Each model is pinned to one provider with fallbacks off: Alibaba for qwen3-30b-a3b, which five providers had served in the first run, and OpenAI for gpt-4.1-nano, its only provider in the first run. The format order rotates over tasks, so each format runs first on 24 tasks. Replies go to a new cache; none is read from the first run.

Checks before and during the run. Before any call, the rerun's loop with the old parser reproduced all 2,688 first-run cells from their cached replies, and a scripted client showed that the corrected parser was in force (calc arguments written as `((expr))` reached the tool whole; the old parser cut them in every `react` and `pythonic` cell). All 768 cells (2 models $\times$ 4 formats $\times$ 96 tasks) were collected in one pass, with no provider error. Every one of the 1,805 qwen3-30b-a3b calls was served by Alibaba and every one of the 1,976 gpt-4.1-nano calls by OpenAI. The run cost USD 0.12.

The three predictions, in the protocol's words. All three hit. With a parser that reads nested parentheses and quotes, and one fixed provider per model, the frozen choices still gain 7.7 points over json on the 96 new-template tasks (E135: 3.9). gpt-4.1-nano's react choice does not lose its gain when react calls are parsed faithfully (25.0 points; E135: 7.3). qwen3-30b-a3b's xml choice keeps a gain of 29.2 points over json, above 10 points. (The main text writes "first run" for E135, as the wording addendum `addendum_20260926_wording.md` allows; nothing else changes.)

Table 36: **E136: the two carrying models, first run and rerun.** Success in points on the same 96 tasks (first run: strict parser, provider not pinned; rerun: corrected parser, the provider in parentheses), the change, and each format's gain over `json`. *The frozen choice. Last row: the frozen choices' gain averaged over all 7 models.

model (rerun provider)	format	first run	rerun	change	gain over `json`, first run	rerun
qwen3-30b-a3b (Alibaba)	`json`	57.3	58.3	1.0	–	–
	`react`	52.1	76.0	24.0	−5.2	17.7
	`xml`*	77.1	87.5	10.4	19.8	29.2
	`pythonic`	41.7	47.9	6.2	−15.6	−10.4
gpt-4.1-nano (OpenAI)	`json`	33.3	29.2	−4.2	–	–
	`react`*	40.6	54.2	13.5	7.3	25.0
	`xml`	11.5	11.5	0.0	−21.9	−17.7
	`pythonic`	38.5	46.9	8.3	5.2	17.7
all 7 models	frozen choices				3.9	7.7

What changed, and what we cannot separate. gpt-4.1-nano kept its provider. Its `xml` success did not change and its `json` success fell by 4.2 points, while its `react` rose by 13.5 points (Table 36). qwen3-30b-a3b changed provider. Its `xml`, which the cut never touched, rose by 10.4 points, so part of its larger gain comes from the new provider or from rerunning, and the rerun cannot tell which. The parser and the route change together, so the table describes the two changes jointly; it does not split them. As descriptive t intervals over the 24 template means, the rerun's gain was $[4.2, 11.2]$ points over all models, $[11.4, 47.0]$ for qwen3-30b-a3b and $[6.8, 43.2]$ for gpt-4.1-nano.

Post hoc readings. These were computed after the scored result was seen (`posthoc_e136.py`, `e136_posthoc.json`). gpt-4.1-nano's `xml` gave the same result on every task in both runs. The fix was used: in 27 `react` and 16 `pythonic` cells of qwen3-30b-a3b, and in 9 and 12 of gpt-4.1-nano, the old parser would have read a reply differently. Had qwen3-30b-a3b chosen `react`, the choice the audit left open, its gain would be 17.7 points and the gain over all models 6.1. All three rerun gains lie above the audit's replay bounds; those bounds held the first run's replies and its `json` results fixed, and the rerun changed both.

What E136 does not show. It reruns frozen choices, not the selection rule, so it says nothing about whether the probe would choose the same formats under the corrected parser. It covers 2 models and 96 tasks. It cannot say whether the parser or the provider caused a change. The first run's numbers and the audit's bounds stay in the paper beside it.

Q DOES THE SIGN-FLIP TEST KEEP ITS LEVEL? (POST HOC)

What the test assumes. The new-template test (App. N) scores its primary result by the sign-flip test over the 24 template means $d_1, \ldots, d_{24}$: the p-value is the share of the 2^{24} sign patterns whose sum is at least the observed sum. Its null hypothesis is that the 24 template gains are independent and each is symmetric about zero (a loss of any size is as likely as a gain of the same size); they need not share one distribution. Under this null, flipping the sign of any d_t does not change its distribution, so all 2^{24} sign patterns are equally likely given the sizes $|d_t|$, and the p-value is exact. A mean of zero does not give symmetry, and decoding at temperature 0 adds no coin flip that would make the two formats interchangeable. So symmetry has to come from the population of templates. Independence is assumed, not tested; the inference treats the 7 models and their frozen choices as fixed.

The check. We reused the planning simulator of the new-template test and drew 10,000 datasets of 24 template gains from each of 57 null worlds whose mean gain is exactly 0 but which need not be symmetric (`experiments/review_20260926_hd/SIGNFLIP.md`, committed alone before the script; the script `signflip_null.py`; data `data/review_20260926_hd/signflip_null.json`). The worlds come from the four planning populations built from the suite's held-out tasks, from the 24 new templates themselves, recentred or re-weighted to mean zero in both skew directions, from three-point stress mixes, and from three symmetric worlds that check the code. Before anything new was computed, the script reproduced the 24 template means and $p = 0.0128$. The rule for what to print was fixed in `SIGNFLIP.md` before the script ran.

Table 37: **How often the sign-flip test rejects when there is no gain (post hoc).** 10,000 datasets of 24 template gains per null world. Rejection rate at the one-sided 2.5% level, and the share of datasets whose p-value is at most the observed 0.0128.

null world (mean template gain exactly 0)	scenarios	rejection rate at 2.5%	share with $p \leq 0.0128$
symmetric (implementation check)	3	0.7% to 1.9%	at most 0.9%
shaped like the 24 new templates, both directions	6	1.7% to 2.9%	at most 1.6%
suite held-out populations 0 and 3, both directions	12	1.0% to 3.1%	at most 1.7%
suite populations 1 and 2, as seen (a few heavy losses)	6	3.2% to 8.6%	at most 6.9%
suite populations 1 and 2, mirrored (a few heavy gains)	6	0.5% to 1.1%	at most 0.5%
three-point stress mixes, a few large losses	12	3.0% to 21.6%	at most 20.6%
three-point stress mixes, a few large gains	12	0.0% to 0.7%	at most 0.3%

What it shows. Table 37 gives the rates. The test keeps its level when the template gains are symmetric and when they are shaped like the 24 new templates: then a p as small as ours occurred in at most 1.6% of datasets. It rejects too often when a few templates lose heavily among many that gain a little, as in two of the four populations built from the suite's held-out tasks: there a p as small as ours occurred in up to 6.9% of datasets. The 24 observed gains are not shaped like that: 6 are 0, 14 positive and 4 negative, the largest loss is 4 in 28 model-task pairs, and the sample skewness is +0.28. But a rare heavy loss need not show up among 24 templates, and no test can rule it out from these data. A t test does no better (up to 8.6% in the worlds built from data). So, as `SIGNFLIP.md` requires, the paper prints beside the primary result that under skewed null worlds built from our data a p as small as ours occurred in up to 6.9% of datasets, so the evidence is weaker than its nominal p. The locked verdict and $p = 0.0128$ are unchanged.

R A FROZEN SELECTION TEST ON HELD-OUT TASK FAMILIES (E129)

The realized adaptation gains of §5.2 come from related designs of our suite, and the prospective study (App. M) gave new models task families the paper already reports. E129

asks one question on task families that nothing in the paper had used: when one of 7 models is treated as the incoming model, with the other 6 as history, does choosing its format with a 10-task probe beat a deployable default built from the other models' history? A negative answer was declared as useful as a positive one. The protocol, `experiments/129_heldout_families/protocol.md`, was committed alone before any E129 model call; the frozen choices and predictions, `predictions.json`, were committed alone before any held-out call. Code: `experiments/129_heldout_families/`; data: `data/e129_heldout_families/`; tables and the numbers the paper prints: `make_tables.py`. Gains and regrets are in turn-accuracy points (turn accuracy $\times 100$).

Scope, in the protocol's words. E129 uses one benchmark, BFCL v3 multi-turn, and 99 tasks built on two APIs (`TravelAPI`, `VehicleControlAPI`) that no earlier design in this paper used, with 7 chat models and 4 formats. A task family is a task's set of APIs. Formats were chosen on 35 tasks from 4 families and scored on 64 tasks from the 6 other families. Both sides use both main APIs, and 83% of the held-out gold calls use a function that also occurs in the calibration tasks. Each model was treated in turn as the incoming model; all seven had run BFCL's 40 earlier tasks, and λ was chosen on those cells, including the incoming model's own. This is one case, not a sample of benchmarks, families or models. (The note of Table 5 prints the same sentence with "This test" for "E129".)

Population, and a correction. BFCL v3 multi-turn has 200 base tasks. BFCL builds each around one main stateful API (`GorillaFileSystem`, `TradingBot`, `TravelAPI` or `VehicleControlAPI`), sometimes with one helper API. The paper's BFCL design and E127's P-bfcl ran the loader's first 40 tasks. The loader sorts task ids as strings, so these are ids 0, 1, 10–13 and 100–133: 34 `TradingBot` tasks and 6 `GorillaFileSystem` tasks. The brief for E129 said that all 40 use `GorillaFileSystem`; they do not, and the protocol corrected this before any call. No earlier design showed a model `TravelAPI` or `VehicleControlAPI`. The population is every task built on these two APIs: 100 tasks, minus task 180, removed before any call because one of its non-final turns has an empty gold list, a request shape no earlier design sent. That leaves 99 tasks, 366 turns and 10 task families. A task family is a task's set of APIs.

Split by family. Whole families go to one side. The families are grouped by main API (`TravelAPI`: 5 families, 49 tasks; `VehicleControlAPI`: 5 families, 50 tasks). In each group, a seeded permutation of the family names is walked, and a family joins calibration when that brings the group's calibration total closer to 17 tasks. Calibration is 4 families and 35 tasks (`TravelAPI`, `MathAPI+TravelAPI`, `MathAPI+VehicleControlAPI`, `TwitterAPI+VehicleControlAPI`); a second seeded permutation splits it into 10 probe tasks (5 per group) and 25 history tasks. The held-out side is the 6 other families: 64 tasks and 246 turns (Table 40b). The split reads no model output.

What was held out, and what was not. No held-out turn has word-Jaccard 0.5 or more with any calibration turn (the largest is 0.43). But both main APIs appear on both sides: 347 of the 416 held-out gold calls (83%) use a function that also occurs in a calibration gold call, and 15 of the 44 held-out functions never do. Two of the 64 held-out tasks start from the same state as a calibration task. `MessageAPI` and `TicketAPI` occur only in held-out families here; they were helpers in 14 of the 40 older tasks, which selector (c) reads.

Models, collector and stages. E127's 7 chat models (App. M), the 4 formats and the collector of the BFCL design, the gold history rendered in the target format, temperature 0 and at most 700 tokens. The seed is E127's (127), so the only difference from P-bfcl is the tasks; a request audit showed that none of the 10,248 requests was already cached. A cell is one model, format and task; its outcome is turn accuracy, averaged over tasks. Stage 1 ran 980 calibration cells (3,360 calls); stage 2 ran 1,792 held-out cells (6,888 calls). No model, cell or task was excluded. The scorer's time-order checks passed with no violation: every fresh calibration call returned before the predictions were committed, every fresh held-out call was sent after, and no call was served from a cache. E129 cost USD 1.468 in model calls.

Selectors. Each of the 7 models is in turn the incoming model; the other 6 are its history. Table 38 lists the selectors. (a) Fixed `json`. (b) The shared default, the primary comparator: the format

with the best pooled score of the 6 history models on the 25 history tasks. It is the pooled-history rule of Table 1: the history changes with the model left out, so (b) gave `json` to five models and `react` to two. The protocol's sentences call it the shared default, and we keep that wording where we quote them. (b+) The same on all 35 calibration tasks (descriptive). (c) The same on the 40 older tasks, from E127's cells, with no new call. (d) The 10-task probe: the incoming model's best format on the 10 probe tasks, ties broken `json`, `react`, `xml`, `pythonic`. (e) As (d), but tied formats are ordered by (b)'s pooled means. (f) The history-shrunk probe: the format with the largest $(K\,p_P + \lambda\,p_{\text{hist}})/(K + \lambda)$, with $K = 10$ probe tasks, p_{hist} (b)'s pooled mean and $\lambda = 20$. λ was chosen before any E129 call, on E127's P-bfcl cells (the same 7 models on the 40 older tasks), as the value in $\{0, 5, 10, 20, 40, \infty\}$ with the largest mean held-out turn accuracy over 1,000 random $10/10/20$ splits; 20 beat the plain probe there by 0.4 points. Those cells include each incoming model's own. Every choice was frozen under both scorers, and (d)'s under all 24 tie orders.

Table 38: **E129: every selector on the 64 held-out tasks.** Regret: the mean over the 7 models of the model's best held-out format minus the chosen format, in turn-accuracy points, under scorer B (primary) and scorer A (the collector's own). Choices are scorer B's. H: the 25 history tasks; P: the 10 probe tasks; old: the 40 older BFCL tasks of E127. The last row is chosen on the held-out tasks themselves, so its selection procedure is not available when deploying; a deployable rule can still pick the same format. Its regret is the held-out oracle gain $\widehat{G}$.

selector	reads	choice (7 models)	new episodes	regret, B	regret, A
(a) fixed `json`	nothing	`json` ×7	0	2.60	2.53
(b) pooled history, calibration families (primary comparator)	600 H episodes	`json` ×5, `react` ×2	0	2.38	2.30
(b+) pooled history, all calibration tasks (descriptive)	840 H∪P episodes	`react` ×7	0	1.05	1.83
(c) pooled history, BFCL's 40 older tasks	960 old episodes	`react` ×6, `pythonic` ×1	0	0.71	0.71
(d) 10-task probe	40 own P episodes	`json` ×4, `react` ×3	40	2.31	2.24
(e) probe, history tie-break	(d) + (b)'s 600	`json` ×3, `react` ×4	40	2.10	2.02
(f) history-shrunk probe, $\lambda = 20$	(d) + (b)'s 600	`json` ×3, `react` ×4	40	2.10	2.02
best shared format in hindsight (not deployable)	the held-out tasks	`pythonic` ×7	—	0.51	0.51

Two scorers. Scorer A is the collector's own. It compares string arguments after stripping spaces and quotes, and an `xml` attribute is always a string. Written in their natural typed form, the gold calls fail scorer A in `xml` on 102 of the 246 held-out turns (against 5 of 170 turns on the older tasks), and in `json` on one held-out turn. So, before any call, selection under scorer A risked partly selecting a parser. Scorer B, fixed before any E129 call, reads a number, list or dict written as text the same way in all 4 formats, and reads the `json` reply with a full decoder; it still requires the right function names, argument names and values. Scorer B is primary. On the held-out cells, 15 of the 6,888 turn verdicts differ between the two (14 in `xml`, 1 in `pythonic`); in each, B marks the turn correct and A does not. The primary gain is the same under both scorers to six decimals, and so is its reading.

Gates. The paper's validity gate (pooled outcome in $[0.10, 0.90]$, no-parsed-call rate below 0.15) passed on the calibration cells (0.1749 and 0.0071) and on the held-out cells (0.2249 and 0.0089), both under scorer B. So the intervals are printed and the predictions are scored. By format, the share of held-out cells with no parsed call was 3.3% in `json`, 0.2% in `xml` and 0 in `react` and `pythonic`.

Contrasts and intervals. Table 39 gives every contrast. The unit is the held-out task, and a gain is a paired mean over the 64 tasks, averaged over the 7 models. The interval treats the 7 models and their frozen choices as fixed; with 6 families, no interval over families is attempted. An interval is printed only when its coverage, measured in simulation against a known target, lay in $[0.93, 0.97]$ under both generators: resampled tasks with binomial turns, and a bootstrap of tasks and turns (2,000 datasets each; Monte Carlo SE about 0.005). The primary result is printed in §6.2. Keeping the default chosen on the old BFCL tasks gave a held-out regret of 0.71 points, against 2.38 for the default chosen on the calibration families. The probe's gain over that old default was -1.61 points (95% interval $[-2.56, -0.66]$). Against the best shared format picked in hindsight on the held-out tasks, a reference no deployable rule can reach, the probe's gain was -1.80 points. (This is the protocol's sentence. More exactly, no deployable rule can use its selection procedure, which reads the held-out outcomes; a deployable rule can still pick the same format, as (c) did for one of the seven models.) With history tie-breaking, the probe's held-out regret was 2.10 points against 2.31

for the plain probe. With history shrinkage ($\lambda = 20$, chosen on the old BFCL tasks), the probe's held-out regret was 2.10 points against 2.31.

Table 39: **E129: every contrast.** Gain in turn-accuracy points over the 64 held-out tasks, averaged over the 7 models; scorer B unless named. Coverage of the t and percentile intervals in simulation, under the binomial-turn / task-and-turn-bootstrap generators; the band is $[0.93, 0.97]$. The hindsight contrasts re-choose the best shared format in every resample and are descriptive.

contrast	gain	printed 95% interval	coverage, t	coverage, percentile	status
primary: (d) over (b)	0.06	t $[-0.65, 0.78]$	0.949 / 0.954	0.936 / 0.940	both in band
(d) over (a) fixed json	0.29	t $[-0.31, 0.89]$	0.950 / 0.953	0.942 / 0.933	both in band
(d) over (c) old default	−1.61	t $[-2.56, -0.66]$	0.952 / 0.948	0.943 / 0.937	both in band
(e) over (b)	0.28	t $[-0.30, 0.85]$	0.936 / 0.955	0.928 / 0.934	t in band only
(f) over (b)	0.28	t $[-0.30, 0.85]$	0.955 / 0.950	0.943 / 0.937	both in band
(d) over (b), scorer A	0.06	t $[-0.65, 0.78]$	0.951 / 0.961	0.944 / 0.935	both in band
(d) over hindsight best	−1.80	percentile $[-2.82, -0.94]$	none	none	descriptive
(e) over hindsight best	−1.59	percentile $[-2.54, -0.77]$	none	none	descriptive
(f) over hindsight best	−1.59	percentile $[-2.54, -0.76]$	none	none	descriptive
(d) over (b+)	−1.26	none	none	none	descriptive

The four locked predictions. All four hit; none missed. *P1* ("the 10-task probe will not be shown to beat the deployable default"): We expected the probe not to be shown better than the default; its interval did not lie above zero, so the expectation held. This hit is not evidence that the probe and the default are equivalent. P1 was expected to hold: on random splits of the older tasks it held on 90% of splits, so the hit adds little on its own; the estimate and its interval carry the result. *P2* said that the E127 regime rule, applied to the calibration families, gives the regime read from the held-out families. The regime read from the 35 calibration tasks was limited adaptation opportunity and the regime read from the 64 held-out tasks was limited adaptation opportunity; they matched. Copying P-bfcl's measured label (limited adaptation opportunity) matched. This is one design, so a match or a miss is one count, not a rate. P2 therefore cannot tell the rule apart from copying the old label. *P3a* and *P3b* (history tie-breaking, and history shrinkage, have held-out regret no higher than the plain probe) are one comparison: (e) and (f) made the same choices, and they differ from (d) on one model only, llama-4-maverick (23.65 in `react` against 22.14 in `json`; $1.51/7 = 0.22$ points of regret).

The regime reading. On the 64 held-out tasks, $\widehat{G}$ is 0.51 points and the frozen probe's gain over the hindsight best shared format is −1.80: limited adaptation opportunity. From the calibration tasks the rule predicted $G = 1.03$ (reported interval $[0.22, 2.68]$) and an achieved gain of −0.64 (−1.12 to +0.07 over the tie orders): the same label. P-bfcl's own reading under scorer B, on the 40 older tasks, is 0.90 and −0.89: the same label again. The predicted interval for G crossed the 2-point cut; the measured value sits in its lower part.

Spread by model and by family (descriptive). Table 40 gives both. Four models' probes chose the default's format, so they add exactly 0 to the primary gain. The other three split: gpt-4.1-nano gained 3.18 points, gpt-4.1-mini lost 1.22 and llama-4-maverick lost 1.51. Across the 6 held-out families the probe's gain ranged from −0.93 to 0.77 points; leaving out any one family gave overall gains from −0.13 to 0.23 points.

Serving providers. gpt-4.1-mini's formats were served by Azure, OpenAI; without gpt-4.1-mini the gain was 0.28 points (descriptive). gpt-4.1-nano's formats were served by Azure, OpenAI; without gpt-4.1-nano the gain was −0.46 points (descriptive). llama-4-maverick's formats were served by DeepInfra, DigitalOcean, Novita, Parasail; without llama-4-maverick the gain was 0.33 points (descriptive). qwen3-30b-a3b-instruct-2507's formats were served by Alibaba, DekaLLM, Nebius, SiliconFlow, StreamLake; without qwen3-30b-a3b-instruct-2507 the gain was 0.07 points (descriptive). qwen3-next-80b-a3b-instruct's formats were served by Alibaba, DeepInfra, Google, Parasail; without qwen3-next-80b-a3b-instruct the gain was 0.07 points (descriptive). These five values are recomputed; the scorer filled them wrongly, as the next paragraph explains.

A defect in the scorer's provider sentences. The protocol asks, for each model whose four formats were not all served by one upstream provider, for the primary gain without that model. The scorer instead dropped all five such models at once, which leaves the two ministral models, whose choices

under (b) and (d) agree, so it printed 0.0 for every model. That number is not the one the protocol asks for. The correct values come from the scored per-model gains, $(7 \times 0.0632 - g_m)/6$; `make_tables.py` recomputes them and checks them against `posthoc_e129.json`. The paragraph above and the last column of Table 40a print them; they are descriptive. The two GPT models count as mixed only because Azure served 2 and 1 of their 984 held-out calls. No locked number is affected.

Table 40: **E129: spread by model and by held-out family (descriptive).** (a) Each model's choice under the pooled-history rule (b), the old default (c) and the probe (d), its best held-out format, the probe's gain over (b), and (*, post hoc) the primary gain with that model left out. (b) The probe's gain over (b) in each held-out family, with its task count. Turn-accuracy points, scorer B.

(a) By model

model	(b)	(c)	(d)	best held-out	(d) over (b)	without it*
ministral-14b-2512	`json`	`react`	`json`	`pythonic`	0.00	0.07
ministral-8b-2512	`json`	`react`	`json`	`react`	0.00	0.07
qwen3-next-80b-a3b-instruct	`json`	`react`	`json`	`xml`	0.00	0.07
qwen3-30b-a3b-instruct-2507	`react`	`react`	`react`	`react`	0.00	0.07
gpt-4.1-mini	`json`	`react`	`react`	`json`	−1.22	0.28
gpt-4.1-nano	`json`	`react`	`react`	`pythonic`	3.18	−0.46
llama-4-maverick	`react`	`pythonic`	`json`	`pythonic`	−1.51	0.33

(b) By held-out family

family	tasks	(d) over (b)
MessageAPI+VehicleControlAPI	9	−0.93
MessageAPI+TravelAPI	10	−0.55
TravelAPI+TwitterAPI	10	−0.24
TicketAPI+VehicleControlAPI	2	0.00
VehicleControlAPI	19	0.50
TicketAPI+TravelAPI	14	0.77

Cost. The probe costs each incoming model 40 episodes (4 formats × 10 tasks, 160 calls, USD 0.0230 on average); the shared default costs no new episode and reads 600 episodes of the other six models. The probe's cost ranged from USD 0.0084 (ministral-8b) to 0.0631 (gpt-4.1-mini) per incoming model. For all 7 incoming models it comes to 280 episodes, 1,120 calls and USD 0.161.

Why the fresh default lost to the old one (post hoc). These numbers were computed after scoring (`posthoc_e129.py`); they re-score nothing. Pooled over the 7 models, `react` leads on the 40 older tasks (38.80 against 38.24 for `pythonic` and 34.68 for `json`) and on the 10 probe tasks. The 25 history tasks are the only task set where `json` leads when all 7 models are pooled, by 0.02 points (16.52 against 16.50 for `react`). (b) pools the other six models; for the five models it gave `json`, `json` led `react` there by 0.02 to 0.66 points. So (b) chose `json` for five models on small margins. On the held-out tasks `pythonic` leads (23.68), then `react` (23.13), and `json` is 2.10 points behind `pythonic`. The old default chose `react` or `pythonic`. Each selector's choice was the model's own best held-out format for few models: 1 of 7 for the probe, 2 for (b) and 3 for (c). This explains one comparison on one benchmark. It is not evidence that a default chosen on older tasks carries over.

What E129 does not show. It is one benchmark, 10 families, 7 models and 4 formats, run on one day; the interval covers tasks from these 6 families, not new families. Both main APIs appear on both sides. The incoming models are simulated: all seven had run the 40 older tasks, and λ and (c) use those cells. Each turn is scored with the gold history, so this is next-call accuracy, not end-to-end task success. P1 was expected to hold. Most of all, the measured oracle gain on these tasks was small ($\widehat{G} = 0.51$ points), so E129 tests the paper's warning, that a probe must beat a strong default, and does not add a case where adapting pays on held-out families.

Part V — Statistical validation

S STATISTICAL RIGOR: UNCERTAINTY, SIGNIFICANCE, AND ROBUSTNESS (E11)

This appendix quantifies the uncertainty behind the headline numbers and answers three robustness questions a careful reader should ask. Some paragraphs below serve the archival five-component instrument (App. AR), not the current argument: Proposition 3(i) on the drift index D, the regression-prediction and silent-drift paragraph, 'Why these component metrics?', 'Does the drift index depend on its weights?', the LLM-as-judge baseline (E12), the localization check and the abstraction check (E19). No current claim rests on them.

One table of what each number is. Table 41 names, for every statistic and every check the paper leans on, the response it reads, the formula and the script that produces it, the null or generator, the resampling unit, and the one conclusion it licenses. The first six rows carry the current argument: the oracle gain, the selection error, the achieved gain, the prospective study, the new-template test and the held-out-family test. The next three check the new-template test: the parser audit, the rerun with a corrected parser and the sign-flip level check. The tenth covers the post hoc checks of the selection error. The self-check `experiments/review_20260912/statistic_contract.py --self-check` evaluates each producer on noiseless toy grids where the answer is known by hand.

Which collection these numbers came from, and what they look like on the newest one. Every live-suite uncertainty result below was computed in June on the collection App. E later retired: the same fixed design, 7 models $\times$ {JSON, ReAct} $\times$ 52 tasks, but stitched from `data/live` and `data/live_paid`. On that collection κ was 0.466. The re-collection put the same design at 0.219, so an interval printed below can sit *disjoint* from the value the paper now headlines, and a reader deserves to know that before reading them. We therefore re-ran this appendix's analysis, unmodified, on the single-provenance grid (E101, `data/e101_stats_single_provenance.json`; the script takes the collection as an environment override so nothing about the analysis changed). What survives and what does not:

on the live suite	June (stitched, below)	24 Aug (single-provenance)
κ (raw success)	0.466	$\mathbf{0.2187}$
task-bootstrap 95% CI	[0.33, 0.60]	[0.1477, 0.3212]
mean-matched no-interaction null, z	10.5	$\mathbf{9.7}$
draws of 5,000 reaching the observed value	0	0
range over five response scales	0.152–0.466	0.0545–0.2237

The claim this appendix exists to defend survives: on the newest collection the observed κ still sits far above a null that matches the cell means and removes only the interaction (z=9.7, no draw of 5,000 as large), so the mean-variance-coupling explanation is refuted on current data as well as on the retired collection. What does not survive is the June threshold prediction that a variance-stabilized κ would exceed 0.25: on the newest collection the three success scales give 0.217–0.224, so that prediction fails. The numbers below are left as they were computed, with this table as their provenance.

Proposition 3 (Consistency)**.** *Under A1–A3 with tasks drawn i.i.d. from $\mathcal{D}$: (i) each plug-in component estimator $\widehat{\delta}_k$ converges almost surely to its population counterpart as the task sample grows, and hence $D \to \sum_k w_k \delta_k$ for any fixed weights; (ii) in the crossed design of §3.1 with tasks as blocks, the empirical variance shares $(S_M, S_{\mathcal{H}}, S_{M\mathcal{H}})$ converge to their population values as the block grows, so κ is consistent. (Proof sketch in App. S; for near-zero true drift the absolute-difference estimators retain a finite-sample upward bias, quantified empirically in App. S.)*

Proof sketch for Proposition 3. (i) Outcome and recovery components are absolute differences of sample means of bounded i.i.d. variables, so the strong law gives a.s. convergence, and continuity of $|\cdot|$ transfers it. Trajectory and tool-selection components are Jensen–Shannon divergences of empirical categorical distributions over finite vocabularies; empirical frequencies converge a.s. (SLLN), and JS divergence is continuous on the simplex, so the plug-in estimates converge. The tool-argument component is a sample mean of bounded per-pair distances (SLLN). A finite weighted sum

statistic or check	response	formula / estimator (producer)	null or generator	resampling unit	what it may conclude
oracle gain $\widehat{G}$ (Eq. (2))	the design's own score	mean best-format score minus the best shared format's, on cell means (`kappa_identity.py`, `unified_policy.py`)	none	none (full table); 2,000 held-out splits	the most a per-model choice adds on that table; not a population gain; the cut 0.02 is in the design's units
selection error $R(\pi)$ (Eq. (5))	same	mean held-out regret of the rule's choices (`unified_policy.py`)	none	the same splits	what the rule loses of $\widehat{G}$ at this budget; not an interval
achieved gain $\widehat{G} - R(\pi)$	same	paired held-out difference against h_0, split average and tie-order range (`derived_readings.py`); frozen choices on one split (`conditional_strict.py`)	frozen-choice t intervals, coverage simulated	splits; tasks within a split	a split average; a frozen-choice interval covers one split's choices only
E127 (App. M)	same	held-out gain A_O of frozen probe choices; regimes by the same rule (`score.py`)	predictions committed first	one held-out table	counts only; new models on reported task families
E135 (App. N)	task success	template-weighted paired gain of the frozen probe choices over the pooled-history rule (`e135_score.py`, tables by `make_tables.py`)	exact one-sided sign-flip test over template means; no interval met its planning band	the 24 template means	new arithmetic, same database and tools; rests on two models
E129 (App. R)	turn accuracy, scorer B	paired held-out gain of frozen choices over the shared default and other selectors; regret (`e129_score.py`, tables by `make_tables.py`)	t interval printed only if its simulated coverage is in $[0.93, 0.97]$ under two generators	the 64 held-out tasks	one benchmark, 6 families, 7 models; limited opportunity, so no positive case
parser audit (App. O)	task success	cached replies re-read by a corrected parser; each episode replayed up to the first changed tool response (`parser_audit.py`)	none (post hoc)	none	how often the harness misread a valid call; bounds on gains, not corrected gains
E136 (App. P)	task success	paired gain of the frozen choices over `json`, corrected parser, one pinned provider per model (`e136_score.py`)	predictions committed first; scored on point estimates	the 96 tasks	the frozen choices only; parser and provider change together
sign-flip level check (App. Q)	E135's template means	rejection rate of the sign-flip test on zero-mean nulls (`signflip_null.py`)	57 null worlds, symmetric and skewed	24 templates, 10,000 datasets each	how far $p = 0.0128$ can be trusted if template gains are skewed
selectors, close calls, regime cut (Apps. L.3–L.5)	same	$R(\pi)$ of two more selectors on the same splits; each model's top-two gap and cost per miss; labels at four cuts (`run_analyses.py`, `threshold_labels.py`)	none (post hoc)	the same splits; model rows	whether two simple selectors leave the selection error; where misses happen; descriptive
primary κ (§3.1); 0.501, repeat 0.421	task success in $\{0, 1\}$, averaged over tasks per cell	$\mathrm{SS}_{M\mathcal{H}}/(\mathrm{SS}_M + \mathrm{SS}_{M\mathcal{H}})$ on the $a \times b$ table of cell means	none (descriptive)	tasks, bootstrap	a share of success variance in this finite design; positive floor at the null
$\widetilde{\kappa}$, ladder rungs; 0.487 headline	binary success on our ladder; graded outcome in $[0, 1]$ on the external and frozen ladders	EMS-corrected sums of squares from the two-way fit (tasks as replicates; residual: task outcome minus its cell mean, pooled over cells), truncated at 0	mean-matched no-interaction null (E64)	tasks	the same share with the noise inflation reduced (the corrected null mean stays positive); the rung comparisons
index 0.563 (E82; reported first)	binary success	variance-component ratio from the task-blocked fit; weights the interaction by $b/(b-1)$	—	tasks	history only; not the primary
Freedman–Lane p (E85)	task success	task-blocked fit (model, format, task, their two-way terms; three-way term as error); permute residuals of the fit without $M \times \mathcal{H}$	rejection rate 0.055 on Gaussian grids (E88); 0.027–0.062 on an exact outcome-scale null (E119)	grids	the headline interaction is not permutation noise
E103 / E109 rejection rates	binary	same test	additive-*logit* grids: a small outcome-scale interaction is present (median $\max\lvert\gamma\rvert$ 0.017)	grids	sensitivity of the test to that generator; not a type-I rate
E119 coverage	binary	one-sided 95th task-bootstrap percentile of $\max\lvert\gamma\rvert$ and of raw κ	exact outcome-scale additive surface plus one interaction shape	datasets	$\max\lvert\gamma\rvert$'s percentile covered in every feasible regime; raw κ's did not;

preserves a.s. convergence. (ii) With i.i.d. replicates in a balanced design, each cell/marginal/grand mean converges a.s.; the sums of squares are continuous functions of these means (plus a residual term whose mean converges by SLLN), so the shares—ratios with a.s.-convergent, eventually-positive denominators—converge to their population values, and so does $\kappa = V_{M\mathcal{H}}/(V_M+V_{M\mathcal{H}})$ whenever $V_M+V_{M\mathcal{H}} > 0$. The caveat is the finite-sample upward bias of $|\Delta|$-type estimators at near-zero true drift (Jensen); E2's γ=0 condition measures it empirically (recovered $S_{M\mathcal{H}} \approx 0.03$), and it vanishes asymptotically. □

Is the multi-model interaction real? For the E9 design (seven fully-covered models × two harnesses × 52 tasks, success response), a task-level bootstrap (B=2,000) gives $\kappa = 0.466$ with 95% CI $[0.33, 0.60]$—bounded well away from zero. A permutation test that randomly exchanges the two harness labels within each (model, task) cell (B=2,000) rejects harness-exchangeability at $p = 5 \times 10^{-4}$; since the harness *main-effect* share is negligible ($S_{\mathcal{H}} = 0.001$), the rejection is attributable to the model×harness interaction. The interaction is a statistically solid property of these models, not sampling noise.

Is κ valid for a binary success response? κ is an ANOVA-style variance *partition*, but the success response is Bernoulli, so the Gaussian/homoscedastic F-test would be inappropriate. We never use it: κ is reported as a descriptive variance share (its sums-of-squares definition needs no distributional assumption), its significance comes from the distribution-free permutation test above, and its uncertainty from the distribution-free task bootstrap—none of which assume normality. As an independent, distribution-appropriate check we also fit a logistic GLM, logit $\Pr(\text{success}) \sim$ model × harness, and test the interaction by likelihood ratio against the additive model: $\chi^2(6) = 79.9$, $p = 3.8 \times 10^{-15}$, with a deviance-based interaction share of 0.51—in close agreement with the ANOVA $\kappa = 0.47$. We are careful not to overstate that agreement as corroboration: the ANOVA variance share and the GLM deviance share are computed from the *same* 7×2 table of cell proportions, so they are re-expressions of one another rather than independent lenses, and one cell (`llama-3.1-8b` at 0.00 under ReAct) is separated, so the saturated GLM's MLE is not finite. The permutation test and the mean-matched null below are the genuinely independent checks; the GLM is reported as a consistency check only.

The EMS correction is near-vacuous in this design—we say so (E33). We previously offered $\widehat{\kappa}$=0.466 → $\widetilde{\kappa}$=0.464 as evidence that the headline is not a bias artifact. It is weak evidence *here*. With b=2 harnesses, $\nu_M = a-1 = 6$ and $\nu_{M\mathcal{H}} = (a-1)(b-1) = 6$ are equal, so the correction subtracts the same quantity from numerator and denominator; the derivative of $\widetilde{\kappa}$ is proportional to $\mathrm{SS}_{M\mathcal{H}} - \mathrm{SS}_M$, which is near zero because the shares are nearly equal (S_M=0.101, $S_{M\mathcal{H}}$=0.088). The invariance is thus structural, not earned: realized movement is 0.002 here versus 0.067 for the three-harness design (b=3, where $\nu_{M\mathcal{H}}$=2ν_M), a 33× difference. We therefore retain the EMS check only for E16 and rest the E9 defense on the mean-matched null above, which *is* a resampling check for this exact design and is the null-floor evidence a careful reader should demand.

Could κ be an artifact of Bernoulli mean–variance coupling, or of response selection (E32)? A natural objection is that cell variance $p(1-p)$ is mechanically tied to the cell mean, so location–scale coupling might manufacture an interaction. It cannot enter this way: $\kappa = \mathrm{SS}_{M\mathcal{H}}/(\mathrm{SS}_M+\mathrm{SS}_{M\mathcal{H}})$ is a function of the 14 cell *means* alone—the replicate count r multiplies both sums of squares and cancels—so the within-cell variance never reaches κ through the residual term. We verify this numerically: the identity-scale cell-mean κ equals the raw Bernoulli κ to 10^{-12}. Coupling can therefore act only through sampling noise in the cell means, which we quantify directly with a **mean-matched no-interaction null**: fit an additive (model + harness, no interaction) model to the observed cell means, simulate Bernoulli data from those fitted probabilities with the same 52 tasks per cell, and recompute κ (5,000 draws). The null gives $\kappa = 0.065 \pm 0.038$ (95th percentile 0.135, max 0.344) against an observed 0.466: z=10.5, $p < 2 \times 10^{-4}$. The interaction is not manufactured by the response's variance structure. Nor is it an artifact of choosing a response, though it *is* response-dependent, so we report the range rather than one number: $\kappa = 0.466$ (raw success), 0.451 (arcsine-$\sqrt{}$ cell success), 0.453 (logit, empirical-Bayes shrunk), 0.184 (step count), 0.152 (tool-call count); each of the five beats its own mean-matched *no-interaction* null, with 0/5000 null draws reaching the observed value (z=10.1–21.5, and z=14.3–17.8 under the stricter null that preserves cross-cell task correlation). We rest this on the nulls rather than on the bootstrap intervals: the raw

η^2 ratio is a ratio of non-negative sums of squares, so an interval that "excludes zero" says nothing on its own. Variance stabilization leaves the success-response κ almost unchanged, and the honest summary is that the interaction is large on task *outcomes* and moderate on *process* counts.

How certain are the regression-prediction and silent-drift results? For E10, across 30 random grouped train/test splits (held-out *tasks*), the regression-prediction AUC is 0.752 with 95% range $[0.71, 0.79]$. That figure is nonetheless *not* evidence of a useful predictor. Three checks bound it. (i) Under stricter grouping (E29) it falls to 0.56 (leave-one-old-model-out) and 0.49 (leave-one-pair-out). (ii) Two trivial baselines beat it outright (E31): $1-$(the new model's marginal success rate under that harness) gives 0.786, a leave-one-task-out per-pair base rate 0.761. (iii) An exact decomposition of the pooled AUC (E32) shows *why*: 99.1% of its $928{,}400$ ranking comparisons are *between* $(old, new, harness)$ cells and only 0.86% are within one, and five of the nine features are literally constant inside a cell yet carry 83% of the fitted importance—so the pooled number is almost entirely a between-pair statistic, not the triage quantity. Measuring the triage axis directly (within-cell task ranking on out-of-fold scores, same-fold blocked to remove a fold-assignment artifact that depresses the naive estimate to 0.515) gives $\mathbf{0.641}$, CI $[0.600, 0.676]$, over 256 blocks; metadata-only and both trivial baselines sit at exactly 0.500 on this axis by construction. The honest conclusion is therefore two-sided: the previously headlined 0.75 is withdrawn, and what survives is a *modest* within-pair signal (0.64) carried entirely by incumbent-behaviour features (App. AM). For E7, an instance-level bootstrap over the 500 Verified instances gives a `gpt-4`→`gpt-4o` flip rate of 0.168, CI $[0.14, 0.20]$, against a resolve-rate delta of 0.008, CI $[0.00, 0.04]$: the behavioral-drift CI lies entirely above the score-drift CI, so "behavior changes far more than the score" holds with non-overlapping uncertainty.

Confirmatory vs. descriptive tests (multiplicity). An earlier version of this paper claimed a two-member confirmatory family. That was wrong, and we correct it: the pre-registered family has **six** members—E9 family-A interaction, E14-P1 family-B interaction, E14-P2 cross-family transfer, plus the locked predictions for E13 (debiasing), E15 (CI coverage) and E19 (abstraction granularity). Three carry p-values and three are decision-rule predictions reported by verdict. Applying Holm–Bonferroni across the three tested members: the headline interaction survives ($p{=}5 \times 10^{-4}$, $p_{\text{Holm}}{=}0.0015$), as does cross-family transfer ($0.0034 \to 0.0068$); family-B does not (0.49), which is the refutation we already report. Of the decision-rule members, E13 was *partially* confirmed (56% bias reduction, not the locked $\geq 60\%$, at a $\leq 5\%$ MAE cost), E15 *half* confirmed (nominal coverage at $\kappa{=}0.47$, under-coverage near zero), and E19 *narrowly missed* ($r{=}0.73$ at $K{=}3$ against a locked 0.8). So the headline survives proper multiplicity control, and three of six locked predictions did not fully hold—stated plainly rather than absorbed into a smaller family. Everything else (SWE-bench κ from single-cell 2×2 grids, the cross-class flip pool) is descriptive; where we previously read a descriptive CI inferentially we now say only that the interval excludes the value, not that a test was passed.

Why these component metrics (§AR)? Each metric is chosen for boundedness and comparability rather than uniqueness: JS divergence is symmetric, bounded in $[0, 1]$, and defined even when the two action/tool distributions have disjoint support (where KL diverges); normalized edit distance is alignment-free and length-normalized for argument strings of differing length; and the outcome/recovery rate-gaps are the natural $[0, 1]$ contrasts for their quantities. These are an interpretable instantiation, one of several admissible (e.g. Wasserstein or Shapley-interaction variants); App. AJ verifies the profile is low-cross-talk under this choice.

Does the drift index depend on its weights? The drift index uses uniform component weights by convention (§AR); weights may instead encode application priorities. Re-evaluating E1 ground-truth recovery under 500 random Dirichlet weight vectors yields recovery correlation $r(w) \geq 0.89$ for *every* draw (median 0.96; Fig. 8, left): any reasonable weighting of the components yields an index that tracks true drift, so the uniform choice is a presentation convention, not an assumption anything rests on. For *interpretation* the component profile and κ carry the weight; D is a convenient headline summary.

Finite-sample bias: theory check and debiasing (E13, Proposition 4).

Proposition 4 (Finite-sample null bias and a moment-corrected estimator)**.** *For the paired success-rate difference $\widehat{\Delta}$ over n i.i.d. tasks with $\widehat{\delta}_{out} = |\widehat{\Delta}|$: at true drift δ=0, $\mathbb{E}[\widehat{\delta}_{out}] = \sigma\sqrt{2/\pi}\,(1+o(1))$ (folded-normal mean, by the CLT), and since $\mathbb{E}[\widehat{\Delta}^2]=\delta^2+\sigma^2$, the moment-corrected $\widetilde{\delta}_{out} = (\max\{0, \widehat{\Delta}^2 - \widehat{\sigma}^2\})^{1/2}$ is consistent and built on an unbiased estimate of δ^2.*

With paired Bernoulli outcomes (p=0.5, 2,000 replicates per cell), the measured null bias of the raw $|\widehat{\Delta}|$ estimator matches the folded-normal prediction $\sigma\sqrt{2/\pi}$ within $\sim$1.5% at every $n \in \{25, \ldots, 400\}$ (e.g. 0.0284 vs. 0.0282 at n=400; Fig. 6, left)—the bias is fully understood, not an unexplained artifact. The moment-corrected $\widetilde{\delta}_{\text{out}}$ cuts the null bias by $\sim$56% and halves null MAE, at the cost of a small negative bias at small positive drift and a $\leq$5% MAE increase at $\delta \geq 0.3$ (Fig. 6, right). Per our locked protocol, this *partially* confirms the prediction (we predicted $\geq$60% bias reduction and no MAE cost): the correction is a bias–variance trade-off, recommended specifically for near-zero-drift certification.

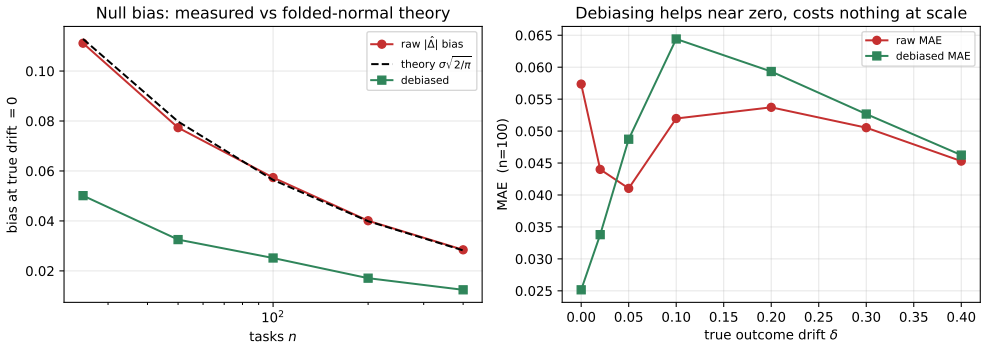

Figure 6: E13: (left) null bias of the raw estimator matches folded-normal theory; the moment-corrected estimator roughly halves it. (right) MAE vs. true drift at n=100: debiasing helps near zero and costs $\leq$5% at larger drift.

What does an LLM-as-judge baseline give on real data (E12)? The natural alternative to a structured estimator is to show a strong LLM both trajectories and ask "how similar is the behavior?" We ran this on 102 real E9 pairs (judge: `llama-3.3-70b`, disjoint from the judged pairs; two substitutions $\times$ both harnesses). The judge's dissimilarity correlates moderately with our drift index (r=0.58, ρ=0.58)—it sees *that* behavior changed—but its correlation with the individual components is diffuse (0.07–0.45): a single holistic score cannot say *what* changed. Structurally, a pairwise judge also provides no model$\times$harness attribution (κ requires a designed comparison across harnesses) and no phase localization, and judge scores inherit known LLM-judge biases (Gu et al., 2024). Judges are complementary sanity checks, not substitutes for decomposition.

Are the κ confidence intervals calibrated (E15)? We report bootstrap 95% CIs for κ; a reviewer should ask whether they cover the truth. On a generative (7 models $\times$ 2 harnesses) grid with a *known* population κ, we draw finite Bernoulli task samples (300 trials) and measure empirical coverage of the task-bootstrap interval. In the regime where we make interaction claims—non-trivial κ, e.g. the family-A κ=0.47—coverage is at nominal: **0.947** at both n=52 (true κ=0.47) and n=120 (true κ=0.34). Near $\kappa \approx 0$ the interval *under-covers* (0.16–0.39) and the point estimate is upward-biased (0.14 for true 0.03), the same boundary/folded-normal effect as Proposition 4: a bounded ratio cannot be symmetrically estimated at its floor. This is why we never assert a near-zero κ from its CI; instead the harness-exchangeability *permutation test* adjudicates significance there—and it behaves correctly, returning p=0.49 (no interaction) for the near-zero wiki family (App. AN) rather than a false-precision interval. Locked prediction (coverage in $[0.90, 0.97]$ at $n \geq 52$): confirmed for non-trivial κ, refuted near zero—reported as is.

Two finite-sample caveats we do not hide. (a) The corrected $\widetilde{\kappa}$ clips each corrected sum of squares at zero ($\max\{0, \cdot\}$), which introduces a small *boundary* bias when the true effect is near

zero; we therefore use $\widetilde{\kappa}$ as a debiased *point* check and treat the permutation test as the arbiter of significance (it makes no such correction). (b) The consistency of the JS- and edit-distance-based components (Prop. 3) is asymptotic; like the $|\Delta|$ estimators, plug-in divergences of *empirical* distributions carry a positive finite-sample bias at small per-task step counts, in the same direction as the $|\Delta|$ null bias of Prop. 4. Both caveats reinforce the same protocol: read aggregate/variance quantities and significance tests, not single-sample low-replicate point estimates, as the things claims rest on.

We validate this on a balanced 4×3 simulator with a *known* population κ (residual-free cell means) and r replicates, 2,000 Monte-Carlo reps per point. At true $\kappa{=}0$ the raw estimator carries a large null floor that shrinks with replicates—$\widehat{\kappa} = 0.35, 0.24, 0.14, 0.078, 0.041$ at $r = 2, 4, 8, 16, 32$—whereas the corrected $\widetilde{\kappa}$ is 3–4$\times$ smaller and near zero ($0.14, 0.078, 0.038, 0.019, 0.009$); across the full true-$\kappa$ grid the mean absolute bias at $r{=}4$ drops from 0.072 (raw) to 0.015 (corrected), and both estimators converge to the truth as r grows (Fig. 7). Applying the correction to the headline numbers shows they are *not* bias artifacts: the seven-model E9 interaction, estimated from 52 task-replicates, barely moves ($\widehat{\kappa}{=}0.466 \to \widetilde{\kappa}{=}0.464$). The correction subtracts $\mathbb{E}[\mathrm{SS}_{M\mathcal{H}}] = (a{-}1)(b{-}1)\sigma^2$ from the numerator; the null floor above (e.g. 0.041 at $r{=}32$) is how large that leaves $\widehat{\kappa}$ *when the true* κ *is* 0, but E9's true κ is large, so the same additive term is a small fraction of $\mathrm{SS}_{M\mathcal{H}}$ and the ratio barely moves—not a contradiction with the floor but its complement. The three-harness E16 value, from only 12 tasks, corrects more visibly ($0.442 \to 0.375$) yet stays large and significant under the permutation test ($p{=}5\times10^{-4}$). This explains the E15 near-zero over-estimate mechanically and lets us report $\widetilde{\kappa}$ as the debiased point estimate; we retain κ as the headline where ample replicates make the two coincide. E36 (App. AP) tests this mechanism directly and confirms it: at $b{=}4$ the degrees of freedom differ (6 vs. 18) yet with 52 replicates the correction still moves κ only $0.394 \to 0.371$. Replicate count, not the number of harness levels, is what makes the correction small—which is what this appendix argued and which corrects the mechanism we had asserted in Limitation T2.

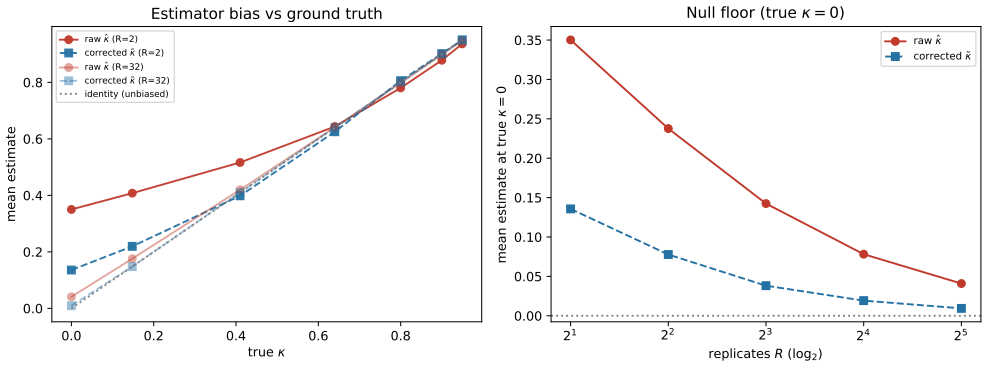

Figure 7: E18: (left) mean estimate vs. true κ for the raw $\widehat{\kappa}$ and the EMS-corrected $\widetilde{\kappa}$ at the fewest ($R{=}2$) and most ($R{=}32$) replicates; the raw estimator sits above the identity line, the corrected one tracks it. (right) the null floor at true $\kappa{=}0$ vs. replicate count: the raw estimator's floor ($0.35 \to 0.04$) is 3–4$\times$ the corrected estimator's, and both vanish as $R \to \infty$.

Does localization survive benign reorderings? Phase localization assumes aligned step order. Applying k random adjacent transpositions to *both* trajectories (benign order noise) degrades localization accuracy gracefully from 0.91 (fresh sample; E3's full-grid accuracy is 0.896) ($k{=}0$) to ≈ 0.61 at $k{=}4$ (Fig. 8, right)—still roughly twice the $1/3$ chance level. Phase attribution is usable under moderate reordering but should be read with care for heavily reordered traces; the component decomposition itself is order-insensitive by construction.

Is the result an artifact of the abstraction granularity? (E19) The trajectory and tool-selection components score divergence over an abstract action/tool vocabulary; a reviewer should ask whether the granularity of that vocabulary—a logging choice—drives the findings. We test the sharp version of the question. Coarser logging trivially shrinks divergence *magnitude* (data-processing inequality), so the real question is whether the drift *ranking* survives: reusing the E1 ground-truth scenarios,

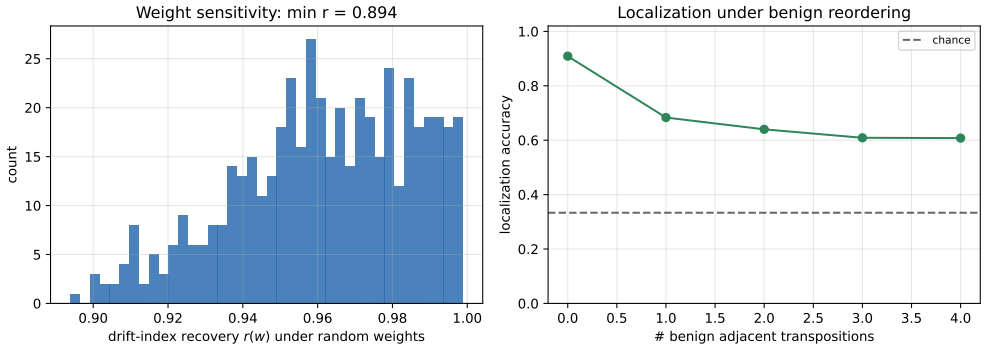

Figure 8: E11: (left) drift-index ground-truth recovery under 500 random component weightings—never below r=0.89; (right) phase-localization accuracy under increasing benign reordering noise.

we collapse the 6-tool vocabulary onto K buckets via random surjective maps (5 per K) and re-measure tool-selection recovery against the *native*-granularity injected drift. At native granularity (K=6) recovery reproduces E1 exactly (r=0.937); under coarsening it degrades *monotonically and gracefully*—$r = 0.84\pm0.03$ at K=4, 0.73 ± 0.05 at K=3, and 0.62 ± 0.05 even when six tools are collapsed to two—never to the chance level of 0 (Fig. 9). Our locked prediction ($r\geq0.8$ for $K\geq3$) holds at K=4 but is narrowly missed at K=3 (0.73), reported as is. The reading is therefore not "granularity-invariant" but "graceful resolution knob": finer logging sharpens recovery, yet the drift ranking is preserved under realistic coarsening rather than being an artifact of one chosen vocabulary.

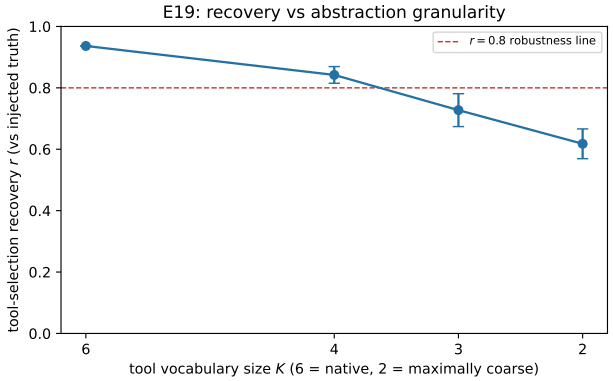

Figure 9: E19: tool-selection recovery r (vs. the native-granularity injected drift) as the 6-tool vocabulary is collapsed onto K buckets (mean$\pm$std over 5 random surjective maps). Recovery is exact at K=6, stays above the r=0.8 line at K=4, and degrades gracefully—never to chance (0)—under aggressive coarsening.

Future work. Shapley interaction indices as an alternative to ANOVA for $V_{M\mathcal{H}}$; signed and bias-corrected component estimators; richer recovery semantics; and large-scale application to public multi-model trajectory dumps on standard agent benchmarks.

T THE 52 TASKS ARE BLOCKS, NOT REPLICATES (E82, CONFIRMATORY)

`two_way_anova` takes an array shaped (model, harness, replicate) and treats the third axis as independent replicates inside each cell. In our live grid the third axis is the *same* 52 *tasks in every cell*. That is a blocked design, not independent replication, and an outside reviewer was right to say so: task difficulty, model $\times$ task and harness $\times$ task all land in the residual, and the EMS correction then subtracts that residual as if it were noise.

Proposition 2 says the hypothesis holds "only approximately". A warning is not a measurement. This appendix measures it.

What we did. Same grid, no new data: 7 models $\times$ 4 protocol forms $\times$ 52 tasks, one run per cell, success as the response. We fit the fully crossed model with **task as a third factor** — main effects for model, harness and task, and all three two-way interactions. In symbols, for model M, harness (format) $\mathcal{H}$ and task t,

$$Y(M, \mathcal{H}, t) = \mu + \alpha_M + \beta_{\mathcal{H}} + \eta_t + \gamma_{M\mathcal{H}} + (\alpha\eta)_{Mt} + (\beta\eta)_{\mathcal{H}t} + \varepsilon, \tag{6}$$

where Y is the task outcome, η_t the task effect, $(\alpha\eta)_{Mt}$ and $(\beta\eta)_{\mathcal{H}t}$ the model $\times$ task and format $\times$ task effects, and ε the error; the other terms are those of §3.1. With one observation per cell the three-way term cannot be separated from error, so $\mathrm{MS}_{M\mathcal{H}T}$ *is* the error mean square. Every variance component is then formed against that error term instead of against a residual that has task variation buried in it. The protocol, with all four predictions, was committed before the script ran.

Where the variance actually sits. As a share of the total sum of squares:

	model	harness	**task**	$M \times \mathcal{H}$	$M\times$**task**	$\mathcal{H}\times$task	error
share of $\mathrm{SS}_{\text{total}}$	0.120	0.022	**0.115**	0.121	**0.199**	0.071	0.352

Two of these were being called noise. The task main effect is 11.5% of the total, and model $\times$ task is 19.9% — larger than any other single term. In plain terms: which model wins depends heavily on which task you give it, and the two-way analysis was charging all of that to the residual and then subtracting it.

The headline survives; the blocked number is a different index, and we say so. Blocking on task moves the reported value from $\widetilde{\kappa}$=0.487 to $\mathbf{0.563}$. That direction was predicted before running, from the algebra rather than from the data: the two-way residual is larger, the correction subtracts $\nu\,\mathrm{MS}_{\text{resid}}$ from each effect, and $\nu_{M\mathcal{H}}$=18 against ν_M=6, so an inflated residual penalises the interaction term three times as hard as the model term. All four locked predictions were confirmed:

	prediction	verdict
P1	$\kappa_{\text{blocked}} \geq \kappa_{\text{two-way}}$	confirmed (0.563 vs 0.487)
P2	$\kappa_{\text{blocked}} > 0.20$	confirmed
P3	task main effect $> 10\%$ of $\mathrm{SS}_{\text{total}}$	confirmed (11.5%)
P4	task-clustered 95% interval excludes 0	confirmed, and nearly worthless (below)

P4 is a coin flip, and we measured how much of one. This repository refuses claims that rest on an interval excluding zero, because the *raw* η^2-style κ is a ratio of non-negative sums of squares: it cannot reach 0, so "its interval excludes zero" is true whatever the data say. That is E51's locked P2, and it is a real defect. Our first instinct was to void P4 for the same reason. That was wrong. κ_{blocked} is built from EMS-corrected components floored at 0, so it *can* be exactly zero, and inheriting a refusal rule without checking whether its reason applies is its own kind of error.

So we measured it instead of arguing about it. On 100 synthetic grids with model, harness and task main effects, noise, and **no** interaction, the median κ_{blocked} is 0.000 — and "the interval excludes zero" still holds $\mathbf{49}\%$ of the time. P4 can fail. It just fails about half the time when it should always fail, so it is **not a test** worth resting anything on, and confirming it establishes almost nothing.

The task-clustered interval is still worth reporting as *precision*: resampling whole tasks — all 28 cells of a task move together, because the task is the unit that repeats — gives $[0.466, 0.664]$. A cell-level bootstrap would break the blocking this appendix is about and report a narrower interval for the wrong reason.

What the 0.563 is, and what it is not. A reviewer worked through the algebra and we confirm it (`experiments/review_20260912/statistic_contract.py`, self-checked on eight fixtures). E82's index divides each corrected sum of squares by its degrees of freedom and by its cell count and reads the result as a variance component: $\hat{V}_M = \max\{0, (\mathrm{MS}_M - \mathrm{MS}_{err})/(bn)\}$,

$\hat{V}_{M\mathcal{H}} = \max\{0, (\mathrm{MS}_{M\mathcal{H}} - \mathrm{MS}_{err})/n\}$, $\kappa_{\mathrm{E82}} = \hat{V}_{M\mathcal{H}}/(\hat{V}_M + \hat{V}_{M\mathcal{H}})$. Algebraically $\kappa_{\mathrm{E82}} = b\,\mathrm{SS}^{\star}_{M\mathcal{H}}/((b-1)\,\mathrm{SS}^{\star}_{M} + b\,\mathrm{SS}^{\star}_{M\mathcal{H}})$, where $\mathrm{SS}^{\star}$ is the error-corrected sum of squares. So it weights the interaction term by $b/(b-1) = 4/3$ relative to the model term. In a noiseless grid with equal model and interaction energy the finite-design share is $1/2$ and κ_{E82} is $4/7$, however many tasks are added. On the headline grid, replacing the two-way residual by the blocked error term moves the corrected share from 0.487 to 0.491; the change of normalization adds the rest, to 0.563. About 95% of the $0.487 \to 0.563$ movement is the normalization, not the blocking. The divisors $(a-1)$ and $(a-1)(b-1)$ are those of a random-effects reading in which models and formats are drawn from populations; our design fixes both sets, so we make no such reading. We therefore report the plain finite-design share, κ=$\mathbf{0.501}$ (task bootstrap $[0.409,\ 0.603]$; repeat grid 0.421, $[0.338,\ 0.514]$; leave-one-model-out jackknife SE=0.21; `data/review_20260912/headline_intervals.json`). Under the harness-label permutation within each (model, task) — E36's null, retired as a test because it also removes the harness effects, kept here as a floor diagnostic — that plug-in share has a median of 0.07 on this grid and a 97.5th percentile of 0.12, so 0.501 is far above its floor. We keep $\kappa_{\mathrm{E82}} = 0.563$ (repeat 0.481) as the descriptive index under which E82's P1–P4 and E118's H1–H4 were scored; those verdicts are not re-scored. The two-way corrected share, 0.487 (repeat 0.406), is the sensitivity the ladder uses.

The bar that does mean something (E85). The question P4 was reaching for is whether the interaction mean square is above the blocked error at all — if it is not, $V_{M\mathcal{H}}$ floors at 0 and $\kappa_{\mathrm{blocked}}$ is exactly 0. It is: the task-blocked test gives $F(18, 918)$=17.48 with permutation p=0.0005.

Our first null permuted harness labels *within each (model, task) cell*. That holds the model means, the task means and every model $\times$ task cell mean exactly as observed — but it also destroys the harness main effect and harness $\times$ task, together 9.3% of the total sum of squares. We disclosed that as "anti-conservative by at most 9.3% of SS", and a reviewer pointed out that this is not a type-I error bound. It is not. A null that removes more than the term being tested has a false-positive rate nobody has measured.

So we replaced it and measured the replacement. **Freedman–Lane**: fit every term *except* model $\times$ harness, permute the residuals of that reduced fit, add them back, recompute F. The reduced fitted values never move, so both main effects, model $\times$ task and harness $\times$ task are preserved *exactly*, and only the term under test is randomised.

null	preserves	mean F under H_0	type-I at nominal 0.05
label permutation (first attempt)	model, task, $M\times$task	—	not measured
Freedman–Lane (reported)	**all but** $M\times\mathcal{H}$	$\mathbf{0.991}$	0.090 (superseded)

An F-ratio should centre on 1 when the effect is absent, and this one does: mean 0.991 over $2{,}000$ draws.

Is the test calibrated, or only persuasive? (E88) Our first check used 100 no-interaction grids, measured a type-I error of 9.0% at a nominal 5%, and we reported that it leaned liberal. A reviewer replied that an acceptance band is not calibration and asked for at least $1{,}000$ grids with an interval on the rate. That is the right ask. We ran it, and **we were wrong about the direction**:

permutation scheme	type-I at nominal 0.05	Wilson 95%	p on the real grid
Freedman–Lane	0.055	$[0.043,\ 0.071]$	$\mathbf{0.0005}$
ter Braak	0.050	$[0.038,\ 0.065]$	$\mathbf{0.0005}$
ter Braak, as we first wrote it	0.000	$[0.000,\ 0.004]$	1.000

Both intervals contain 0.05: on Gaussian grids with no interaction the test is calibrated, and a second scheme built on a different residual gives the same answer. The 9.0% from the first check was sampling noise — with 100 grids its own standard error is 2.2% — so E88's own locked prediction, **P1**, that a larger run would show Freedman–Lane liberal, is **refuted**.

E88's third row is ours to own. Ter Braak permutes the *full*-model residuals; our first version added them back to the *full* fitted values, which leaves the model $\times$ harness effect inside every null draw, so the null sits on top of the observed statistic and the test rejects 0 of $1{,}000$ times. That is an

implementation defect, not a property of the method. Adding them to the reduced fit instead gives the second row. Both are in the released record, because a scheme that never rejects looks like a conservative test until you notice why.

Those grids are Gaussian, and our response is not, so we measured that too. Every null grid above draws normal effects. The response the headline decomposes is per-task success — one Bernoulli trial per cell, with the same 52 tasks in every cell, so the task-related dependence is real. A calibration measured in a Gaussian regime licenses a claim about that regime, and a reviewer was right to say so. E103 repeats the same schemes, unchanged, on binary grids built from additive logits with no model×harness term, at the grand success rate of the headline grid (0.4073) and at both design sizes we report on. One caveat first, and it is also a reviewer's: the test reads the outcome scale, and a sigmoid of additive logits is not additive there. On E103's first headline grid the outcome-scale cell means carry a largest interaction of 0.028 before any Bernoulli draw, and over 1,000 fresh draws of the same generator the median is 0.017 at the headline size and 0.006 at the ladder size (`data/review_20260912/generator_diagnostics.json`). So these grids are a small-interaction regime, not an exact null, and the rates below are the test's sensitivity to that regime, not its type-I error. Freedman–Lane rejects **0.062** of these grids at the headline size 7×4×52 (Wilson [0.0487, 0.0787]) and 0.055 at the ladder size 5×2×52 (Wilson [0.0425, 0.0709]); both intervals contain 0.05, and both are under the 0.10 ceiling the protocol declared before the run (`data/e103_binary_calibration.json`). So on a binary response close to the headline's, with a small outcome-scale interaction present, the test rejects at about its nominal rate. That is a statement about this scheme at these two sizes on that generator; a null with exactly zero outcome-scale interaction is not measured here, and the p-values computed elsewhere name their own null where they appear. Power against a declared alternative is measured at the frozen design size in App. V.

So we built a null on the right scale, and measured what the percentile is worth (E119). E103's grids come from additive logits, and the paragraph above says why that is a sensitivity rather than a type-I rate. The fix is a null that is additive on the *outcome* scale, which is the scale the test reads. E119 builds one: draw centred model, format, task, model×task and format×task effects, add them, shrink the sum by one common factor so every probability stays inside $[0.01, 0.99]$, and set $p = p_0 + fz$. There is no model×format term, no sigmoid and no clipping, and the script *asserts* that the mean surface's interaction is zero to 10^{-12} before it samples a single outcome. Outcomes are drawn with a shared per-task normal, so the 5×2 (or 7×4) cells that share a task can be correlated (ρ=0.3) without changing any cell's probability; different tasks stay independent. The protocol is `experiments/119_bound_calibration/protocol.md`, locked with no data in the tree, and it fixed seven predictions before the run; five held and two did not. One boundary on that provenance: git shows the protocol committed before the analyzer and the analyzer before any result, but the configuration file was committed together with the result, so "configuration written before execution" rests on the analyzer writing it first, not on an independent record; that commit also added a 10^{-9} tolerance at the support boundary after the smoke run discarded grids sitting exactly on it, which the protocol's "outside $[0.01, 0.99]$" does not exclude.

The reported test is calibrated on this null. Over 24 regimes — two designs, two base rates (0.10 and 0.4073), three model-effect scales, two task correlations, 1,000 grids each — Freedman–Lane rejects between **0.027** and **0.062** of no-interaction grids at a nominal 0.05, and its Wilson interval contains 0.05 in **23** of the 24. That is the calibration claim this paper leans on, and it is now measured on the scale the statistic lives on.

The mean-matched null is a different story, and it is ours to own. The same run scores E64's mean-matched null, the test the frozen rungs use: it rejects between 0.000 and 0.001 of the same grids. A test that almost never rejects cannot tell a reader much by not rejecting, which is exactly why App. V reports percentiles of the frozen interaction instead of reading anything into its p-values.

Does the 95th percentile cover? 500 datasets proposed per cell, 499 task bootstraps each, on the $5 \times 2 \times 31$ design, with a known interaction of 0, 0.02, 0.05 or 0.10 added to the mean surface. Because nothing is clipped, the realized interaction equals the declared one exactly, which E104's generator could not say. The table below is the p_0=0.4073, independent-task cell; every cell of every run, with its feasible and proposed counts and Wilson interval, is printed at the end of this appendix

(Table 42), rendered from the result file by `make_e119_table.py` so that no summary here can drift from it.

true $\max\|\gamma\|$	0	0.02	0.05	0.10
coverage, $\max\|\gamma\|$ percentile	1.000	**1.000**	**1.000**	**1.000**
coverage, raw κ percentile	1.000	0.90	**0.42**	**0.02**
datasets that stayed in support, of 500	500	252	169	51

The first column is free: the true value is zero and the statistic cannot be negative, so anything covers. Of the 16 planned cells, four are these zero-effect cells, four could not be run (at p_0=0.10 no interaction of 0.05 or more fits inside the support: 500 of 500 proposals left it), and eight are informative. The other three settings do not all agree with the one shown: $\max|\gamma|$'s coverage is 1.000 in all eight, but the ratio's at 0.02 is 0.90 here, 0.89 with correlated cells, and 0.015 and 0.000 at p_0=0.10, where the true ratio is already 0.96.

Two things follow, and the paper says both. **In outcome points the percentile covered**: over the eight informative cells (1,054 datasets) and the four trivial ones it was at or above the truth in every dataset, which is looser than 95% needs. **As a ratio it did not**: coverage falls to 0.42 and then 0.02 as the true interaction grows, because a bigger interaction with the same small model effect pushes the true ratio towards one while noise holds the observed ratio down. That is E109's small denominator arriving from the other side, and it is why this paper reports the absolute interaction beside every ratio. What this licenses was decided before the run: C3 and C4 held and C5 did not, and the protocol says that if any coverage prediction fails, the word "bound" stays out. So the paper calls all of these percentiles and prints the coverage beside them. Treating the two statistics differently is a reading we formed after the result; we state it here as that, and the locked verdicts stand as scored (C5 refuted).

Three limits on the coverage numbers. Feasibility is conditional on this generator and its draws: a cell with no feasible dataset is unestimated, not a rate of zero and not a proof that the amplitude is impossible at that mean. At 0.10 only 51 (and 50 with correlated cells) of 500 proposals stayed inside, the ones whose nuisance happened to be small, so those coverages of 1.000 carry Wilson lower limits of 0.930 and 0.929. And the response is binary with one interaction shape; the frozen rungs are graded, so this is evidence about the procedure, not a guarantee for those data. Power on this null is also lower than E104's clipped generator suggested: at a realized 0.10 the test rejects in 15 of 27 feasible datasets (0.56, Wilson [0.37, 0.72]; 273 of 300 proposals excluded), on a binary response at this design size, with cells drawn independently. The protocol planned power at both task correlations; the analyzer, committed before the run, only implemented ρ=0. That is a deviation from the protocol and we record it as one. We then ran the missing arm, post hoc and under its own seed (`data/review_20260915/e119_power_rho03.json`): with correlated cells the test rejects a realized 0.10 in 15 of 28 feasible datasets (0.54, Wilson [0.36, 0.70]), so the arm the analyzer skipped reads as the arm it ran. C7 stays scored on the locked run.

A ratio has two ways of being large, and we measured the second one (E109). A reviewer wrote that a κ near one can reflect a small model-main-effect *denominator* rather than a large interaction, and listed near-zero denominators among the things our inference had never been audited for. That is right, so we swept it: binary grids at the headline size, built as E103 builds them, with the model-effect scale σ_M moved from 0.8 (E103's value) down to 0, where the models are exchangeable and the denominator is noise. One caveat carries over from E103: these grids are additive in the logit, so their outcome means carry a small interaction (median $\max|\gamma|$ 0.017 over 1,000 fresh draws at this size, `data/review_20260912/generator_diagnostics.json`), and the rows labelled "no interaction" below mean no *injected* interaction. The rejection rate in the first row is this generator's rate at that small interaction, not a type-I error on an exact null; E119 measures the exact null. Protocol `experiments/109_small_denominator/protocol.md`; artifact `data/e109_small_denominator.json`.

model-effect scale σ_M	0	0.1	0.2	0.4	0.8
rejection rate, Freedman–Lane (none injected)	0.0650	0.0550	0.0525	0.0475	0.0625
median $\widetilde{\kappa}$ (no interaction)	0	0	0	0	0
95th pct of *raw* κ (no interaction)	**0.9147**	0.8696	0.8026	0.6137	0.3293
median $\widetilde{\kappa}$, interaction held fixed	0.7362	0.7376	0.6606	0.4486	**0.2074**
median $S_{M\mathcal{H}}$, interaction held fixed	0.0329	0.0333	0.0339	0.0323	0.0305

Three things, in order of how much they should change a reader's mind.

The reviewer's mechanism is real and large. Hold the injected interaction pattern fixed on the logit scale and shrink the model axis, and the same injection reads as $\widetilde{\kappa}$=0.2074 or $\widetilde{\kappa}$=0.7362 depending on the denominator — a 3.5× swing. On the outcome scale the interaction is not exactly invariant, but nearly: $S_{M\mathcal{H}}$, the absolute share, sits between 0.0305 and 0.0339 across the whole sweep. That is the argument for reporting the absolute interaction beside the ratio, which this paper does, and it is why the last row matters more than the fourth.

The raw estimator is unusable at a small denominator. With *no interaction at all* and exchangeable models, the raw κ's 95th percentile is **0.9147**. The bias-corrected estimator's median is 0 at every σ_M. This is the clearest reason we report the corrected value.

The test's rejection rate does not move with the denominator. On this generator Freedman–Lane rejects between 0.0475 and 0.0650 of the grids with nothing injected, including at σ_M=0. That is a sensitivity of this generator, read beside E119's exact-null rates of 0.027–0.062; it says a small denominator breaks the ratio, not the p-value.

And where does the headline sit? On the 7×4×52 grid, S_M=0.120 and $S_{M\mathcal{H}}$=0.121: the numerator is as large as the denominator, so the headline's $\kappa \approx 0.5$ is not a compressed-denominator artifact. We report those two shares here so a reader can place any κ in this paper in the right regime rather than taking our word for it.

One locked prediction we wrote badly. E109's W3 said the median $\widetilde{\kappa}$ would *rise* as σ_M falls, and the scorer tested strict monotonicity across all five points. It came out 0.7362 at σ_M=0 against 0.7376 at σ_M=0.1 — an inversion of 0.0014 between two points where the model axis is noise either way — so **W3 is refuted**, and it is recorded refuted. Its fallback does not apply: that fallback was written for "the ratio does not respond to the denominator", and the ratio responds by a factor of three and a half. A knife-edge monotonicity test over five points, two of them indistinguishable, was a badly specified prediction. We are saying so rather than reading the result as the confirmation it nearly was.

What this does not fix. Blocking repairs the *residual*, not the population question. The 7 models and 4 protocol forms are still fixed finite levels chosen by us, so both 0.487 and 0.563 remain descriptive shares of a finite design. The leave-one-model-out jackknife (SE=0.22, App. E) is still the honest uncertainty on the model axis, and nothing here replaces it. What changed is narrower and worth stating plainly: the objection to the replicate assumption was correct, and correcting it does not weaken the result.

E119, every cell. Table 42 prints each regime of the three runs from the result file: what was proposed, what stayed in support, successes over the feasible denominator, and the Wilson interval. It is rendered by `experiments/119_bound_calibration/make_e119_table.py`, and `verify.sh` fails if the committed table differs from what the result renders.

U DECODING-NOISE FLOOR: WHEN IS DRIFT ATTRIBUTABLE? (E17)

A measurement is only meaningful above its noise floor. For model-substitution drift the relevant floor is *within-model* sampling variation: if the same model, sampled twice, drifts as much as two different models, a single sample cannot attribute drift to the substitution. We measure this directly. For three models we draw *two independent stochastic samples* (temperature 0.7, different seeds, seed included in the request and the cache key) of the *same* model on a shared 12-task JSON harness, compute the trajectory decomposition's *self-drift*, and compare it to cross-model drift on the identical tasks.

Run A: rejection rate on an exactly additive outcome-scale null (1,000 grids per regime, 199 permutations, α=0.05)

design	p_0	σ_M	ρ	FL rejections	Wilson	mean-matched rate
headline 7x4x52	0.1	0.0	0.0	55/1000	[0.042, 0.071]	0.000
headline 7x4x52	0.1	0.0	0.3	54/1000	[0.042, 0.070]	0.000
headline 7x4x52	0.1	0.1	0.0	41/1000	[0.030, 0.055]	0.000
headline 7x4x52	0.1	0.1	0.3	44/1000	[0.033, 0.059]	0.000
headline 7x4x52	0.1	0.8	0.0	58/1000	[0.045, 0.074]	0.000
headline 7x4x52	0.1	0.8	0.3	61/1000	[0.048, 0.078]	0.000
headline 7x4x52	0.4073	0.0	0.0	53/1000	[0.041, 0.069]	0.000
headline 7x4x52	0.4073	0.0	0.3	58/1000	[0.045, 0.074]	0.000
headline 7x4x52	0.4073	0.1	0.0	62/1000	[0.049, 0.079]	0.000
headline 7x4x52	0.4073	0.1	0.3	40/1000	[0.030, 0.054]	0.000
headline 7x4x52	0.4073	0.8	0.0	27/1000	[0.019, 0.039]	0.000
headline 7x4x52	0.4073	0.8	0.3	44/1000	[0.033, 0.059]	0.000
ladder 5x2x31	0.1	0.0	0.0	50/1000	[0.038, 0.065]	0.000
ladder 5x2x31	0.1	0.0	0.3	48/1000	[0.036, 0.063]	0.000
ladder 5x2x31	0.1	0.1	0.0	46/1000	[0.035, 0.061]	0.000
ladder 5x2x31	0.1	0.1	0.3	50/1000	[0.038, 0.065]	0.000
ladder 5x2x31	0.1	0.8	0.0	47/1000	[0.036, 0.062]	0.000
ladder 5x2x31	0.1	0.8	0.3	44/1000	[0.033, 0.059]	0.000
ladder 5x2x31	0.4073	0.0	0.0	53/1000	[0.041, 0.069]	0.000
ladder 5x2x31	0.4073	0.0	0.3	49/1000	[0.037, 0.064]	0.000
ladder 5x2x31	0.4073	0.1	0.0	45/1000	[0.034, 0.060]	0.000
ladder 5x2x31	0.4073	0.1	0.3	55/1000	[0.042, 0.071]	0.000
ladder 5x2x31	0.4073	0.8	0.0	55/1000	[0.042, 0.071]	0.001
ladder 5x2x31	0.4073	0.8	0.3	40/1000	[0.030, 0.054]	0.000

Run B: coverage of the one-sided 95th percentile (5×2×31; 500 datasets proposed per cell, 499 task bootstraps each)

p_0	ρ	true max $\lvert\gamma\rvert$	feasible/proposed	max $\lvert\gamma\rvert$ covered (Wilson)	raw κ covered (Wilson)	status
0.1	0.0	0.0	500/500	500/500 [0.992, 1.000]	500/500 [0.992, 1.000]	trivial
0.1	0.0	0.02	65/500	65/65 [0.944, 1.000]	1/65 [0.003, 0.082]	informative
0.1	0.0	0.05	0/500	—	—	not estimated
0.1	0.0	0.1	0/500	—	—	not estimated
0.1	0.3	0.0	500/500	500/500 [0.992, 1.000]	500/500 [0.992, 1.000]	trivial
0.1	0.3	0.02	70/500	70/70 [0.948, 1.000]	0/70 [0.000, 0.052]	informative
0.1	0.3	0.05	0/500	—	—	not estimated
0.1	0.3	0.1	0/500	—	—	not estimated
0.4073	0.0	0.0	500/500	500/500 [0.992, 1.000]	500/500 [0.992, 1.000]	trivial
0.4073	0.0	0.02	252/500	252/252 [0.985, 1.000]	227/252 [0.858, 0.932]	informative
0.4073	0.0	0.05	169/500	169/169 [0.978, 1.000]	71/169 [0.348, 0.495]	informative
0.4073	0.0	0.1	51/500	51/51 [0.930, 1.000]	1/51 [0.003, 0.103]	informative
0.4073	0.3	0.0	500/500	500/500 [0.992, 1.000]	500/500 [0.992, 1.000]	trivial
0.4073	0.3	0.02	242/500	242/242 [0.984, 1.000]	216/242 [0.847, 0.926]	informative
0.4073	0.3	0.05	155/500	155/155 [0.976, 1.000]	49/155 [0.248, 0.393]	informative
0.4073	0.3	0.1	50/500	50/50 [0.929, 1.000]	5/50 [0.043, 0.214]	informative

Run C: power at a realized interaction (5×2×31; 300 datasets proposed per cell, 199 permutations; cells drawn independently: the ρ=0.3 arm the protocol planned was not run)

p_0	ρ	true max $\lvert\gamma\rvert$	feasible/proposed	FL rejections (Wilson)	mean-matched	status
0.1	0	0.02	53/300	6/53 [0.053, 0.226]	0/53	run
0.1	0	0.05	0/300	—	—	not estimated
0.1	0	0.1	0/300	—	—	not estimated
0.4073	0	0.02	148/300	16/148 [0.068, 0.168]	0/148	run
0.4073	0	0.05	85/300	13/85 [0.092, 0.244]	0/85	run
0.4073	0	0.1	27/300	15/27 [0.373, 0.724]	0/27	run

Table 42: **E119 accounting.** A cell with zero feasible datasets is unestimated under this generator, not a zero rate. Run C's ρ=0.3 arm was in the protocol and not in the locked analyzer; a post-hoc, unscored run of it is in `data/review_20260915/e119_power_rho03.json`. FL: Freedman–Lane, the test the paper reports.

Result. At temperature 0.7, per-task self-drift and cross-model drift are *comparable*: mean self-drift 0.235 vs. mean cross-model drift 0.213 (ratio 0.91); the same-model outcome-flip rate (0.17–0.42) approaches the cross-model rate (0.33–0.50). For these small models at high temperature, a single sample per model is therefore *insufficient* to separate a substitution effect from decoding noise at the per-task level.

Implication — the method's valid operating regime. This bounds, rather than refutes, the claims, and dictates the protocol we already use:

- **Temperature 0 (our live studies E5/E9/E16): the floor, measured (E32).** Greedy decoding is deterministic *in principle*, but hosted temperature-0 serving is not bit-reproducible (batching/kernel nondeterminism), so we measure the residual floor rather than assert it. Mining the released response cache for calls identical in every argument *except the seed* yields 1,227 reseed groups / **1,983 seed-pairs** across 18 models. Exact byte-for-byte agreement is 0.824, i.e. a **raw text floor of** 0.176 (cluster-bootstrap CI $[0.152, 0.201]$)—*not* near zero, and only $0.75\times$ the temperature-0.7 self-drift of 0.235 measured in E17. Crucially, almost all of that variation is *harness-invisible*: when both samples parse, the extracted tool call is identical in 99.91% of pairs, so the floor that actually propagates into our components is the **parsed-call floor of** 0.076 (CI $[0.059, 0.093]$; conformance-flip floor 0.061). Restricted to the E9 *models* the parsed-call floor is 0.093—but that restricts by model key, not by *call type*, and is superseded below. Per-cell drift estimates in E5/E9/E16 should therefore be read against a $\sim$0.06–0.09 floor, roughly 3–$4\times$ quieter than temperature 0.7 but not zero. Heterogeneity we disclose rather than average away: only 8 of 18 models reach ≥ 0.90 exact agreement, and `gpt-oss-20b`—an E9 model—has a parsed-call floor of 0.305, *above* both other measured floors, so its per-cell attributions carry no usable margin and we do not rely on them.
- **Aggregate / variance quantities.** The interaction κ is a variance share over the *entire* (model×harness×task) design, not a per-task point estimate; the permutation tests (App. S) compare against a within-design null. These remain valid under noise, which averages out across the design.
- **High temperature requires replication.** If a deployment runs at temperature > 0, per-task drift must be denoised by averaging over k samples (the floor shrinks as $1/\sqrt{k}$); reporting single-sample per-task drift at high temperature is not supported.

We also flag the corresponding caveat for the SWE-bench study (App. AI): those are single-sample leaderboard submissions, so the 16.8% `gpt-4`→`gpt-4o` flip rate is an upper bound that includes whatever sampling noise each submission carries. We state its status precisely: App. AI reports it *calibrated against the independence null computed from the same two marginals* rather than against zero, and a common noise inflation raises the observed flip rate and that null together, so the calibrated "48% of the way to independence" is more stable under decoding noise than the raw rate. It remains an upper bound, and no per-submission floor is recoverable from public leaderboard data; the noise-controlled weight sits with the temperature-0 live studies (E5/E9/E16). The same single-sample caveat applies to the WebArena (App. AJ) and τ-bench (App. AK) per-task decompositions, which are likewise single observations at unstated temperature: their per-task component and flip magnitudes are upper bounds that include within-model sampling noise, so the cross-class *pattern* (not any single pair) is what we rely on. Stating this floor explicitly is, we believe, what a measurement contribution owes its readers.

The floor above was measured on the wrong population; the in-domain one is $2.4\times$ larger (E43). Every reseed source feeding the 0.08 figure—`E28d_chains`, `E27_broad`, `E27_seeds`, `E27_string`, `E30_reasoning`, `E28_chains`, `E26_pilot`—is a single-turn or short protocol/handoff call. *None* comes from the live agent loop whose per-task claims the floor is used to defend, and our earlier restriction "to the E9 models" restricted by model key rather than by call type, which hid the transfer. We therefore measured the floor where the claims live, by reseeding the live grid at `seed=1` (the seed is part of the cache key, so this is a fresh draw) and pairing episodes across draws.

Over the complete design—728 paired episodes, all seven models—**episode-outcome disagreement is** 0.220, action-sequence disagreement 0.669, and tool-sequence disagreement 0.527. Per-model outcome floors span 0.038 (`mistral-small-24b`) to 0.337 (`gpt-oss-120b`), so a sin-

gle pooled number was always a poor summary of this quantity. Per-model outcome floors span 0.038 (`mistral-small-24b`) to 0.429 (`gpt-oss-20b`, n=7), so a single pooled number was always a poor summary. No model is seed-insensitive, so this is a genuine resample floor rather than a provider ignoring the seed—and that holds however the provider treats the seed, since two calls at temperature 0 are two samples from a nominally deterministic configuration and their disagreement is exactly the quantity wanted.

The consequence is asymmetric and we state it plainly. Any *single-sample, per-task* statement about the live suite carries a $\sim$19% same-model floor, not 8%, and over half of same-model reseeds take a materially different path. But κ is a variance share over 728 episodes rather than a per-task statement, so episode-level resample noise averages down instead of propagating, and the flip-at-score-parity results come from the public-log studies with their own separately reported floors. Net, this strengthens the case for reporting the aggregate interaction and weakens per-task live claims.

The first test-retest of the headline, and a limit on per-task attribution. The same reseed gives the first test-retest κ has ever had: on the balanced grid (52 tasks complete in both draws), $\kappa = 0.466$ at seed 0 and 0.523 at seed 1, a difference of $\mathbf{0.057}$, with bootstrap CIs $[0.3231, 0.595]$ and $[0.3967, 0.6317]$. So the headline reproduces across a fresh draw to within about 0.06—less than the 0.113 the permissive parser contributed (App. F). That draw is not same-period, and an earlier version of this paragraph said it was: seed 0 was collected in May and June, seed 1 in August. Two larger sources have since been measured and neither was anticipated here. On the single-provenance grid, harness-factor size spans $\mathbf{0.284}$ rather than the 0.089 the stitched grid showed (0.204 at b=2 to 0.487 at b=4, App. AP), and *re-collecting the same design three months later* moved the two-form estimate by $\mathbf{0.26}$ because one model's tool-use compliance changed under an unchanged alias (App. E). So the reseed is the *smallest* of the four perturbations we can now quantify, not a bound on them, and the largest one does change which design carries the headline. We state that rather than leaving the 0.06 to imply a stability it does not establish. Per-model conclusions are a different matter: the parser change alone moves 2 of 7 best-interface assignments, and E47 (App. AA) shows those movers are the models whose top two interfaces are nearly tied. We separate the two claims deliberately, having once let an aggregate supersession stand next to a per-model sentence it no longer supported (App. AO).

The floor is measured between collections, and the gap does not explain it. A reader can reasonably ask whether 0.220 is decoding noise or ten weeks of drift. We can answer it with data already here. Every pair of our collections that shares cells disagrees on a fifth to a quarter of paired episodes, and the pair closest in time is not the smallest: 0.2198 across the ten-week pair, 0.2541 across five days, and 0.2759 across twelve weeks (E99, `data/e99_ladder_provenance.json`). Shrinking the gap does not shrink the disagreement. So 0.220 is not a pure decoding floor, and it is also not an artefact of the gap: what it bounds is the same cell run again later, which is the comparison a practitioner actually makes.

One qualification belongs here rather than only in a limitation. The in-domain floor (0.220) is comfortably below the *median* cross-model per-task flip rate on this suite (0.461 over 42 pairs), a $2.1\times$ margin. But the *minimum* is 0.157—*below* the floor. For the most similar model pairs, per-task outcome differences on this suite are not distinguishable from same-model resample noise. Per-task attribution is therefore licensed for a typical pair and not for the closest ones, which is one more reason the paper's claims use quantities averaged over tasks ($\widehat{G}$, $R(\pi)$ and the achieved gain), not per-task comparisons.

Part VI — Sensitivity analyses

V A FROZEN REPLICATION ON NEW MODELS (E92)

Every reviewer of this paper has made the same point, and they are right. Our evidence history is long. We have repaired a parser, changed the null, changed the estimand, changed which model term the variance goes to, and reported all of it. A ledger of corrections shows honesty; it does not show

that the last set of choices was not tuned to the answer. The fix is one collection where every choice was locked first, on models chosen before any cell ran.

How new the models are, exactly. The protocol's criterion was routable at temperature 0 and **absent from the thirteen of E67**, the external design whose middle rung contradicts ours. All six meet it. Two of the six — `llama-3.1-8b` and `gpt-oss-20b` — do appear in our own seven-model design, so "new" means new to the external substrate, not unseen by us. The other four had no part in the ladder on our suite, but three of them appear in other designs of this paper: `qwen3-32b` and `gpt-5.6-luna` among WorkBench's six models, and `mistral-medium` among the strong models. We say so because the protocol's gloss on F1, "models that had no part in forming it", is true of those four and not of the two.

What was frozen, and how you can check it. The protocol is `experiments/92_frozen_factorial/protocol.md`, and the `protocol_sha256` in the result file is its hash *as locked*, at commit `c38376f5`, which contains no data. The shipped file hashes differently because Amendment 1 was appended to it a commit later. So the check a reader runs is `git show c38376f5:.../protocol.md | shasum -a 256`, compared against the hash in the data, with the data commits later still.

Frozen before collection:

- **the substrate** — Harness-Bench at one pinned commit, a benchmark we did not build;
- **the tasks** — the same 32 that E67 pinned, copied unchanged, no re-selection;
- **the models** — six, *none* of them among E67's thirteen (two are in our own seven-model design, see above);
- **the forms and the budget** — four forms, temperature 0, the same token limit;
- **the analysis** — `analyze_e64.py`, byte-for-byte the script that produced the earlier ladder, with no edit to the script. Its command line differs: `--dir` points at the new cells, `--out` writes a new file, and `--models` names the five that Amendment 1 leaves measurable rather than the six locked here;
- **the rungs and the predictions** — the three rungs verbatim, plus four new predictions F1–F4 written down before a single call was made.

The protocol also says what we were not allowed to do: "no re-selection of models, tasks, forms or budget is permitted to rescue it". That sentence is the point of the exercise, and it cost us something below.

One model could not be measured, and that is itself a result. Of the six models, `command-r` completed 32 of 32 tasks on both free-text forms and 0 **of** 32 **on both constrained forms**. We diagnosed it directly rather than guessing from the sweep: its route returns **HTTP 400** when the request carries the provider's `json_schema` response format. The same one-line probe succeeds on two other models in the set, so this is a property of that route, not a bug of ours.

We did not replace it. The protocol forbids re-selection, and a model whose route rejects the schema mechanism has no third-rung datum to score — it is missing by capability, not a zero. It is reported as unmeasurable on the ladder, the way lifelines was reported unmeasurable in the companion paper rather than scored, and Amendment 1 records the decision *before* any κ was computed.

This matters beyond bookkeeping. Our advice is to evaluate a schema as a candidate. One of the six models the protocol named *cannot be given a schema at all* through its route. So the advice has a precondition, and we state it: check that your provider will carry a schema before you plan around one.

The other gap was ours to fix, and we fixed it the boring way. `mistral-medium` failed intermittently on all four forms. It passes the same probe, so the failures were transient API errors, made worse by a detail of the harness: one failed request aborts a whole ten-step episode, so a small per-request error rate becomes a large per-cell one. We re-ran the collector, which skips cells that already exist, until the gaps closed. Same models, same forms, same tasks, same budget, same script — only a slower request interval. Completing a collection the protocol already specified is not choosing a new one.

V.1 WHAT THE FROZEN DESIGN FOUND: NO LADDER, BECAUSE NO INTERACTION

The primary analysis is the unchanged script. Its output, on the five models that carry the ladder (31 common tasks; one task dropped because two oracles raised while grading):

rung	forms	κ	S_{MH}	S_M	S_H	no-call	gate	p
L1 free text	json / react	**0.000**	0.0019	0.330	0.003	0.12	passed	1.00
L2 one side	json_a / react	**0.003**	0.0102	0.261	0.014	0.14	passed	0.36
L3 both	json_a / json_b	**0.000**	0.0036	0.263	0.000	0.18	**failed**	1.00

The interaction is not there to remove. On the 13-model design the same substrate gave κ=0.11 $\rightarrow$ 0.39 $\rightarrow$ 0.00. On these five it gives 0.00 $\rightarrow$ 0.00 $\rightarrow$ 0.00. The reason is in the two variance columns. Against the 13-model reference at L1, the interaction S_{MH} fell from 0.025 to 0.002 — thirteen times smaller — while the model main effect S_M rose from 0.105 to 0.330. Both moved. The five models span mean scores from 0.15 to 0.68, so almost all the variance is *between models*. Each model's row moves less across the four forms than the models move apart: the widest row is `gpt-oss-20b`, 0.156 on react against 0.372 on json_a, and even that is smaller than the gap between the weakest and strongest model. The rows do not all slope the same way — `gpt-oss-20b` gains under a schema while `gpt-5.6-luna` loses on json_b — which is why the interaction is small rather than zero.

The ratio is fragile; the null is not. Those two columns are computed on the five ladder models, and one caveat belongs beside them. `command-r` has no L2 or L3 datum, but it has both free-text forms on all 31 common tasks — 62 cells. Amendment 1 drops it from every rung, not only from the two its route cannot reach. Put it back at L1 and the interaction share is 0.007, not 0.002, so the fall against the 13-model reference is about *four* times rather than thirteen, and S_M is 0.318 rather than 0.330. What does not move is the verdict: κ(L1) is 0.000 on five models and on six. This recomputation is post hoc and is not the frozen script's output — that script intersects tasks across all four forms, so it cannot run on six models at all. It is in `experiments/92_frozen_factorial/posthoc_commandr_l1.py` and `data/e92_commandr_l1_posthoc.json`.

Every locked prediction about the ladder was refuted. P3 (the κ ladder), P4 (the absolute ladder), P5 (both drops separated from zero) and P6 (no manufactured interaction): all refuted. P2, the validity gate, **failed at L3**. P0 and P1 — the budget removed truncation, and schema violations stayed under the locked limit (the schema was requested with `strict` off, so this is measured adherence, not enforced decoding) — confirmed, so the instrument worked; it is the phenomenon that did not appear.

P6 deserves its own sentence, because "refuted" understates what the check did: the locked truncation watch flagged *all three* rungs CONFOUNDED. It fires on tiny rates — mean truncation is 0.003, 0.009 and 0.011 across the rungs — and at L1 it flags a rung whose κ is 0.000, where there is no interaction for truncation to manufacture. So we report the flag as locked and do not lean on it: on a rung with no interaction the rule cannot distinguish a confound from nothing at all, which is the same near-vacuity we report for the EMS correction at b=2 (App. S).

The four predictions this experiment added fared the same way:

- **F1** (κ(L3) $<$ κ(L1) on new models): **refuted**. Both are 0.
- **F2** (holds in $\geq$ 6 of 8 Harness-Bench workflow classes): **refuted**. It holds in **two**: *Office & Business Communication* (0.10 $\rightarrow$ 0.17 $\rightarrow$ 0.00, 4 tasks) and *SRE, DevOps & Release Ops* (0.04 $\rightarrow$ 0.00 $\rightarrow$ 0.00, 4 tasks). In the other six, κ is 0 at every rung. Every class holds 3 to 5 tasks, so a zero there is what the bias correction returns on a handful of tasks as much as it is a finding. The split is computed per rung, on the tasks that rung's two forms both cover, so one class compares 4 tasks at L1 against 3 at L2 and L3 and the class totals are 32, 31 and 31. Per the protocol, the claim narrows to those two, named.
- **F3** (at least one class is non-monotone): **confirmed**, by Office & Business Communication, where the middle rung is the highest — the same shape the 13-model design showed overall.

- **F4** (leave-one-model-out SE on $\kappa(\mathrm{L1})$ exceeds 0.10): **refuted**. The SE is $\mathbf{0.0}$: drop *any* one of the five models and $\kappa(\mathrm{L1})$ is still exactly 0. We had predicted large model uncertainty and did not get it. We then wrote that this was "precision about zero", and that was wrong. The bias-corrected estimator is floored at zero, so a small effect gives zero on every leave-one-out fit by construction; that jackknife says nothing about precision. What the frozen data say about size is measured below and does not rest on the floored estimator.

How big could the interaction be and still look like this? (E104) A test that does not reject is not a measurement of absence. So we asked the question the other way round: what sizes are still compatible with the frozen data, and what sizes would we have caught? The protocol is `experiments/104_frozen_bounds/protocol.md`, locked with no data in the tree, and the artifact is `data/e104_frozen_bounds.json`. Two answers, one from the data and one from the design.

From the data. Resample the 31 common tasks with replacement, 5000 draws, and read off the 95th percentile. A reviewer asked the right question about that percentile — does it cover? — and we have now measured it, on a binary simulation (E119, App. T). The answer differs by quantity. For the largest interaction cell $\max|\gamma|$, in outcome points, the percentile was at or above the true value in every dataset of every feasible regime, eight with a real interaction and four with none. For the *ratio* κ the same percentile covers the truth 1.000, 0.90, 0.42 and 0.02 of the time as the true interaction grows in the cell shown, and as little as 0.000 elsewhere. The protocol said, before the run, that if any coverage prediction failed the word "bound" stays out, and one did (C5). So every number in the table below is a percentile, its measured coverage is printed in App. T, and reading the two statistics differently is our interpretation after the result, not the protocol's license. Two more reasons to keep the word out: the simulation is binary with one interaction shape and these rungs are graded scores (69% of outcomes lie strictly between 0 and 1), and at the largest amplitude only 51 and 50 of 500 proposed datasets stayed in support, whose Wilson lower limits (0.930, 0.929) do not reach 0.95. We report the raw share, the corrected share, and — because a ratio is hard to feel — the largest single interaction cell $\max_{m,h}|\gamma_{mh}|$, where $\gamma_{mh} = \bar{Y}_{mh} - \bar{Y}_{m\cdot} - \bar{Y}_{\cdot h} + \bar{Y}_{\cdot\cdot}$ is what one model gains or loses on one form beyond what the model and the form separately predict. It is in outcome points on Harness-Bench's graded $[0, 1]$ score, it is not a ratio, and it is not floored. It is also not a model's raw movement between the two forms, which includes the form's common effect and is larger: 0.096 on rung 1 and 0.217 on rung 2 for the model that moved most, against interaction cells of 0.029 and 0.066 (`data/review_20260915/frozen_movement.json`). An additive surface can move every model by 0.2 between forms and have an interaction of exactly zero. (The artifact's own `reading` field and prediction U2 still say "bound"; they were written before E119 and are shipped as written.)

Each cell is the point value with the task-bootstrap 95th percentile in brackets.

rung	gate	κ raw	κ corrected	$\max\|\gamma\|$	RMS γ
L1	passed	0.0056 (**0.0512**)	0.000 (0.0271)	0.0288 (**0.0734**)	0.0149 (0.0456)
L2	passed	0.0378 (0.1090)	0.0029 (0.0801)	0.0663 (0.1134)	0.0356 (0.0630)
L3	**failed**	0.0137 (0.0682)	0.000 (0.0298)	0.0333 (0.0794)	0.0218 (0.0479)

Read the first rung: the raw share is 0.0056 with a 95th percentile of $\mathbf{0.051}$, the corrected share is 0 (0.027), and the largest interaction cell is 0.0288 outcome points (0.0734): in 95% of the resampled grids, no cell sits more than about $\mathbf{7}$ points of the graded score away from what its model and its form separately predict. Compare like with like. The corrected percentile 0.027 is below 0.1127, the corrected value this same rung takes on the external substrate in E67, and far below 0.4747, the value it takes on our own suite; the raw percentile 0.051 is below E67's raw 0.1926. Rounded, the raw share reads 0.006 and 0.038 on the two rungs that passed their gate, with task-bootstrap 95th percentiles $\mathbf{0.051}$ and $\mathbf{0.109}$; these are percentiles, not bounds (App. T). So the frozen run reads far below the ladder at this rung. We stop short of calling the two incompatible, and now for a measured reason rather than caution: these two numbers are ratios, and E119 shows the ratio's percentile misses the true value once there is a real interaction to miss. The outcome-points row is the one whose percentile covered in that simulation.

One disclosure about U1. The prediction was locked against 0.4596, and that number is retired: it is E61's external first rung, which the JSON-parser defect touched, and E67 supersedes it (App. AX).

Locking a comparison against a superseded value was our mistake, and we do not re-score the prediction to hide it — U1 is scored as written, and it is confirmed. The reason that is not luck is the paragraph above: the percentile also sits below both values that are current, on both scales, so the verdict does not depend on the stale threshold. The third rung's row is printed for completeness and carries no claim: its no-parsed-call gate failed, and a rung that fails its gate is invalid, not an informative zero.

From the design. How large an interaction would this design have caught? We build grids that keep the observed surface — grand mean, model marginals, form marginals, residual pool — and add an alternating-sign interaction scaled to a declared $\max|\gamma|$, then run the same test. 300 grids per size, 200 null draws each, $\alpha = 0.05$.

Two things about this generator matter for reading the table, and a reviewer found both. The mean surface and the residual sums are both clipped to $[0, 1]$, so the interaction present in the generated outcomes is smaller than the declared one: averaged exactly over the residual pool, a declared 0.10 realizes 0.082, a declared 0.05 realizes 0.042, and a declared 0 still carries 0.003 (`data/review_20260912/generator_diagnostics.json`). And the residuals are drawn i.i.d. from one pool, so the grids carry no task dependence beyond the additive surface. E119 answers the same question with a generator that has neither the clipping nor the mismatch — nothing is clipped, and the realized interaction equals the declared one exactly — on a binary response at this design size. Its power run drew every cell independently: the protocol planned a correlated arm for power and the locked analyzer did not run it, a deviation we record; a post-hoc run of that arm, unscored, reads the same (App. T). There the test rejects a realized 0.10 in **15** of **27** feasible datasets (0.56, Wilson $[0.37, 0.72]$), not 0.99; the other 273 of the 300 proposed datasets left the support and were excluded, so this is power conditional on the draws that generator can make. The two numbers are not in conflict: they are two generators, one graded and clipped, one binary and exact. The honest summary is that the power of this design depends on the response and the generator, and we report both rather than the flattering one.

declared $\max\|\gamma\|$	0.05	0.10	0.15	0.20	0.30
power, the test we report	0.477	**0.993**	1.000	1.000	1.000
power, Freedman–Lane	0.473	0.993	1.000	1.000	1.000

So against this generator the test catches a declared 0.10-point interaction 99 times in 100 and is a coin flip at half that. That is the power of one clipped, alternating-sign, pooled-residual simulation. It is not a guarantee for every 0.10-point interaction the real outcome distribution could carry, and it says nothing about the coverage of the percentiles above. Both schemes give the same power to within 0.4 points, so within this simulation the number does not depend on the permutation scheme.

And the decision, on this substrate. The one decision result the paper keeps is that measuring the incoming model beats keeping the outgoing model's format. We ran E70's comparison on the frozen L1 grid. The locked run (seed 104) drew a 31-task bootstrap, removed the duplicates and split what was left, so its splits averaged 6.3 history, 6.3 probe and 7.2 held-out tasks and the probe made 12.6 incoming-model episodes per swap, not the 20 the text priced; a reviewer found this and we confirm it. On those draws keeping the incumbent's format costs $\mathbf{0.0372}$, measuring the incoming model costs $\mathbf{0.0430}$, the oracle costs 0, the paired difference is -0.0057 with interval $[-0.0865, 0.0633]$, and the probe wins in 41% of draws. Prediction U5 said the probe would win and is **refuted**; the locked fallback says the abstract names its scope, and it now does. We then ran the exact budget post hoc: 2,000 partitions of the 31 tasks into exactly 10 history, 10 probe and 11 held-out tasks, all 20 ordered pairs of the five models, the same held-out cells for every policy, seed 10412 (`data/review_20260912/policy_reanalysis.json`). Keeping the format costs 0.0305, probing costs 0.0383, a fixed default costs 0.0290; keep minus probe is -0.0078 with split percentiles $[-0.0631, 0.0464]$, and the probe is the lower in 37% of partitions. Resampling the models as well gives -0.0085; resampling the held-out tasks inside each partition gives -0.0080. So on this grid a 20-episode probe shows no advantage over keeping the format. That is not equivalence: the percentiles permit an advantage of several outcome points either way, and they are split sensitivity, not coverage. One reading fits both results: where the interaction reads near zero, as it does here, a probe has little to find; where it is large, as on our suite, the probe pays for itself. That is a reading, not a mechanism we have tested, and E120 contradicted it on a third family: on WorkBench $\widetilde{\kappa}$ reads 0.38, a 40-episode probe beats keeping the format by $+0.023$ ($[0.012, 0.043]$),

and it shows no detected advantage over a default pooled from the deployed models' history (default minus probe -0.011, $[-0.019, 0.004]$, an interval whose measured coverage is below nominal; the point estimates give no finite break-even for five of six models; App. K).

Why the third rung failed its gate, named. The gate requires a no-parsed-call rate below 0.15; L3 recorded 0.18, which is 57 no-call cells of the 310 the rung scores. Every count here is on those same 31 common tasks. Fifty of the 57 belong to one model: under the constrained schema, `gpt-5.6-luna` replies {`"done": true`} at step 0 — before making a single call — on 50 of its 62 constrained episodes (0.71 on json_a, 0.90 on json_b), and on none of its free-text ones. The mechanism is visible in the schema: it declares no `required` fields and carries a top-level `done` boolean, and the prompt says to reply {`"done": true`} when finished. That reply is therefore schema-valid and prompt-sanctioned at any step, including the first, and one model takes the exit before doing any work. `gpt-oss-20b` is the mirror case: without a schema it fails to produce a parseable call in 0.61 and 0.55 of episodes (19 and 17 of 31); with one, 0.10 and 0.13. So the schema helps one model and lets another quit, and the outcome column sees neither, because both models score low either way.

We report L3 as a rung that failed its gate. P2, the per-rung validity gate, is one of the locked predictions, and the protocol forbids re-selection of models to rescue a refutation, so we do not remove the model that made it fail.

What this does to the paper's claims. Fallback (b) was written for this outcome and says what happens: it goes in the main text and the abstract, and the ladder claim is scoped to the suite we built. So:

- The ladder $0.47 \to 0.27 \to 0.11$ is a fact about *our* suite and *our* five models. It reversed at the middle rung on 13 outside models and is flat at zero on five more. Three designs, three shapes. It is not a general law and the paper no longer presents it as one.
- "Specify the call fully and the interaction is near zero" survives in the weakest possible form here: it was zero before specification too. The frozen design confirms the paper's *other* claim instead — that κ is a share within the design you ran — with a pre-registered design at the bottom of the range, 0.00.
- The advice acquires two preconditions we can now state from data: the route must carry a schema at all (`command-r` returns HTTP 400), and the model must keep working under one (`gpt-5.6-luna` exits at step 0). Neither is a property of the model alone; both are properties of the model *under this harness*, which is the paper's subject.

One correction to our own secondary analysis, disclosed. The first version grouped tasks by the benchmark's free-form *tags* rather than its `class:` field, produced three families from eleven tasks, and scored F2 on that. It now joins to the eight classes the protocol names, read from each task's own metadata at the pinned commit and committed as data. F2 was refuted under both groupings; the numbers above are from the correct one.

W THE SPECIFICATION LADDER AND THREE OTHER SENSITIVITY READINGS

This appendix holds four sensitivity readings of κ: the specification ladder on our suite, its re-collection on Harness-Bench's 13 models, the across-days measurement, and BFCL. They are not the paper's story. The main text keeps one sentence where a licence requires it: the ladder does not replicate prospectively (E68's fallback), and comparisons between collections are bounded by run-to-run noise, not by date (E111's fallback). The frozen external reading itself is in App. V.

The specification ladder, a sensitivity of our suite. Rung 1 compares two free-text formats (JSON against ReAct), rung 2 gives one side a schema, rung 3 both. On our suite the schema is requested with `strict` off, so rungs 2 and 3 are schema-guided output, not enforced decoding (App. X). Every ladder value is $\widetilde{\kappa}$ on that rung's two formats; over the headline's four formats it reads 0.487. On our suite $\widetilde{\kappa}$ reads $0.47 \to 0.27 \to 0.11$, and what measuring the incoming model saves over keeping the format falls with it, $+0.141 \to +0.011 \to +0.000$ (each rung's five-model collection). Not a denominator artifact: S_{MH} drops too, $0.117 \to 0.067 \to 0.030$, even though



Figure 10: **The ladder is a reading of our suite, not a law.** Left: $\widetilde{\kappa}$ by rung on our suite ($0.47 \rightarrow 0.27 \rightarrow 0.11$), on Harness-Bench's 13 models ($0.11 \rightarrow 0.39 \rightarrow 0.00$), and frozen ($0.00 \rightarrow 0.00$, third rung gate-failed). Middle: S_{MH}. Right: what measuring buys on ours (Apps. X, V).

the model main effect rises, $S_M = 0.129 \rightarrow 0.174 \rightarrow 0.212$, which would lower κ on its own. Those are three collections, not one sweep: rung 1 is May and June's, and rung 2 pairs an August schema-guided column against rung 1's own ReAct column, so a serving-stack change sits inside that rung as well as between them; against the newest unconstrained arm the first step is $0.23 \rightarrow 0.18$ (App. X). The same JSON-vs-ReAct pair reads 0.204 in the headline collection and 0.47 as rung 1 (another collection, same EMS correction): a 0.27 gap between two readings. Two layers sit under the ladder: κ on protocol adherence alone is 0.566, and with adherence held fixed κ on success is 0.445 (App. AS). Conditioning on adherence is a stratification, not an intervention, so this is a description of the ladder and not a measured cause. The schema is unavailable for 2 of 7 models and reverses the best format for 2 of the 5 that can use it (Apps. AP–X; variants in Apps. I, G).

Across substrates, re-collected. We re-collected the same three rungs on Harness-Bench's own tasks and oracle (Yao et al., 2026): 13 models $\times$ 4 formats $\times$ 32 tasks, 1,662 of 1,664 cells collected, 30 tasks present for every (model, format) pair. The ladder goes $0.11 \rightarrow 0.39 \rightarrow 0.00$: giving *one* side a schema triples the interaction (paired interval $[-0.41, -0.12]$; the free-text pair alone reaches only p=0.08), and giving both sides one takes it to zero, $[0.24, 0.51]$. Our suite reads the opposite at that rung ($0.47 \rightarrow 0.27$). Both steps sit within reach of the day-to-day movement of 0.13 measured below (the Harness-Bench interval's near edge sits at it; our 0.20 step has no interval). In the cells, a schema helps 7 of the 13 models and hurts the other 6: it sorts them rather than aligning them. Over all four formats $\widetilde{\kappa}$=0.22, lower than our $\widetilde{\kappa}$=0.49, and the gap's interval includes zero (Apps. Y, Y.6).

What else the frozen replication shows. With the sixth model added back at rung 1, $\widetilde{\kappa}$ stays 0.00, and the fall against the 13-model design is $3.7\times$, not $13\times$. The model main effect is three times the 13-model design's (0.105 there against 0.330 here, rung 1) while S_{MH} fell $13\times$ (0.025 against 0.002). The largest interaction cell is not a model's raw change between formats, which includes the format's common effect (App. V). The test itself catches a 0.10-point interaction in 15 of 27 feasible datasets (0.56), and the mean-matched null these rungs are tested against rejects none of it. The gate failed because the schema declares no required field and has a top-level `done`: one model replies {`"done": true`} at step 0 on 50 of its 62 constrained episodes, and a done-only reply has no tool call.

Across days. On three Harness-Bench collections (ten models, 27 tasks), one day moved rung-1 $\widetilde{\kappa}$ by 0.133 and twelve days by 0.131 against the reference as collected, and 0.172 and 0.254 against the parser-repaired reference (App. Y.8); by the locked fallback, E108 measured a design that is unstable between runs, not drift over an interval, and every cross-collection comparison in this paper is bounded by run-to-run noise rather than by date (Apps. Y.7, Y.8).

Where every model gets the same schema. A pre-registered run on BFCL v3 multi-turn (Patil et al., 2025) reads $\widetilde{\kappa}$=**0.00** on turn accuracy, its secondary endpoint (task bootstrap $[0.00, 0.13]$; the task-success endpoint failed its gate at pooled 0.018), with S_{MH} at 0.008. Read it as a near-zero estimate here, not as a measured zero: its null almost never rejects and 0.13 is not nothing (Apps. T, J).

A schema is a candidate, not a fix. Our schema-guided rungs read lower, and on Harness-Bench giving one side a schema was the worst case; that is an observation, not a cause. A schema can add a failure κ cannot see, so check success and the no-parsed-call rate beside it.

X DOES A SCHEMA-GUIDED JSON FORM REMOVE THE INTERACTION? (E44)

The null these p-values come from, and what happens under the one we validated. The p-values in the tables below come from permuting harness labels *within* each (model, task) cell. App. T retired that scheme for the headline: it removes the harness main effect and harness×task along with the interaction, and a null that removes more than the term being tested has a false-positive rate nobody has measured. That objection bites hardest here, because the schema-guided rung has a harness main effect of 0.115, thirty-eight times the unconstrained 0.003 — precisely the term the retired null destroys. So we recomputed all three rungs under Freedman–Lane, the replacement whose false-positive rate App. T measures at 0.055 against a nominal 0.05 (E102, `data/e102_ladder_null.json`). No permutation of 2,000 reached the observed F at any rung ($F = 26.51$, 20.07 and 11.13), so all three sit at the floor $p = 0.0005$ — the same value the retired scheme reported. The ladder's significance does not depend on the null we retired, and now we know that rather than assuming it.

Which collection each rung came from, and what that costs. E44 ran in June, when `data/live` and `data/live_paid` were the only unconstrained collection we had. The schema-guided column was collected on 6 August, and the suite was collected twice more after that. Nobody re-derived this ladder against the newer data, so its first step compares a June JSON column against an August constrained column while the June ReAct column stands on both sides. Anything that changed in the serving stack between June and August lands inside that step, and App. E shows something did change: one model's JSON success moved from 0.077 to 0.788.

We recomputed the ladder's own estimator, unchanged, on every collection in this repository that carries the columns it needs (E99, post hoc, `data/e99_ladder_provenance.json`):

unconstrained arm	May/June (as published)	19–22 Aug	24 Aug
rung 1, unconstrained	$\mathbf{0.4747}$	0.5223	0.2257
rung 2, schema-guided	$\mathbf{0.2666}$	0.1584	0.1834

Every value is bias-corrected, the convention the published rungs use, so the first column reproduces them exactly. Only the unconstrained arm varies across the columns: the constrained column is the single 6 August collection in all three, because it is the only one we have. The direction holds in all three — the constrained column has the lower share whichever collection supplies the unconstrained arm. The size does not. The published first step is the largest of the three, and with the newest unconstrained arm it is 0.2257 against 0.1834, which is almost flat. So the ladder's *shape* is supported and its *first step* is not separable from collection drift. We report the locked rungs first because they are what the protocol locked, and we report this table beside them because it is what a reader needs to know how much of that step to believe.

Grammar- and schema-constrained decoding is the standard production remedy for a model failing a protocol's syntax, so a reviewer will reasonably ask whether our per-scaffold recommendation is obviated by a configuration change. We measured it. The answer is genuinely mixed and we report it that way, including a prediction of our own that the data refute.

Design and validity gate. We re-ran the JSON level with the provider's `json_schema` response format (an object restricted to the protocol's own keys), unchanged prompts, and paired it against the existing unconstrained ReAct level—the same b=2 design as the headline. Pre-registered gate, evaluated first: schema-guided tool-call adherence is $\mathbf{0.996}$ with a 0.000 **parse-failure rate**. We should be exact about what that shows: the schema is *requested* through the gateway's structured-output field with `strict` set to false, so what is established is that the models satisfied it, not that the decoder compelled them to. Models satisfy it by calling tools rather than by answering immediately. Only 5 of 7 models could be included: both `gpt-oss` models return `content: null` under a schema, with all output routed to `reasoning`. That exclusion is part of the answer: a remedy unavailable for 2 of 7 models cannot substitute for measurement. Table 43 reports the constrained design, Table 44 its per-model cells, Table 45 the payoff under constraint, and Table 46 the two-equally-constrained comparison.

Table 43: E44: with the JSON level constrained the interaction reads 44% lower, not zero, while the harness *main* effect grows 44-fold. Both κ columns are bias-corrected, as in the ladder.

JSON level	κ	95% CI	perm. p	$S_{\mathcal{H}}$
unconstrained	0.475	$[0.314, 0.613]$	5×10^{-4}	0.003
schema-guided	$\mathbf{0.267}$	$[0.182, 0.389]$	5×10^{-4}	$\mathbf{0.115}$

The objection substantially lands. κ reads 0.267 on the constrained column against 0.475 on the free-text one, 44% lower. The two columns were collected on different dates (June and August) with the ReAct column shared, so this is an association across collections, not a measured effect of constraining; on the newest unconstrained arm the gap is 0.23 against 0.18 (the E99 table above). Our locked $\kappa > 0.25$ is cleared, but only marginally and with a CI reaching down to 0.182, so we report it as a marginal pass. What survives is still a real interaction ($p{=}5 \times 10^{-4}$). One reading fits the cells: the JSON column is lifted almost uniformly on the constrained collection, which moves variance from the interaction into a harness *main* effect ($S_{\mathcal{H}} : 0.003 \to 0.115$); with mixed dates we cannot say the constraint alone did that.

We predicted the wrong direction, and the reason is worth stating. From App. AS we had $\kappa(\text{success} \mid \text{adherence}) = 0.445$, above the unconditional 0.394, and predicted that removing the adherence layer would leave roughly that higher value. This **conflates conditioning with intervening**. The conditional quantity is the interaction among trajectories that *happened* to adhere; constraining the decoder does not select that subset, it *changes the population*, making models adhere that otherwise would not—and those newly-adhering trajectories mostly succeed. `phi-4` goes from emitting parseable JSON in 19 of 52 episodes to 52 of 52, and its JSON success from 0.078 to 0.961. A conditional decomposition predicts what an intervention leaves behind only when the intervention selects rather than shifts; ours shifts.

Table 44: E44 per-model success. For `mistral-small-24b` and `phi-4` the *best interface reverses* once structured output is enabled.

model	JSON uncon.	JSON constr.	ReAct	optimum
`llama-3.1-8b`	0.235	0.392	0.000	JSON → JSON
`llama-3.3-70b`	0.471	0.529	0.627	ReAct → ReAct
`mistral-small-24b`	0.275	$\mathbf{0.922}$	0.706	**ReAct → JSON**
`phi-4`	0.078	$\mathbf{0.961}$	0.412	**ReAct → JSON**
`qwen3-14b`	0.745	0.922	0.333	JSON → JSON

What the objection does not touch: two optima reverse. For 2 of 5 models the best protocol reverses. A practitioner who measured unconstrained and then enabled a schema would be running the *wrong* interface for both. Constrained decoding therefore does not remove the need to measure which harness suits which model; it changes the answer, adding the decoding configuration as a further axis of the same phenomenon. Accordingly we restrict the strongest form of our recommendation to free-text protocols, and for constrained ones we recommend re-measuring rather than assuming an earlier measurement transfers.

And the practitioner payoff does not survive it (E45). The paper's practical claim rests not on κ's magnitude but on the measured payoff of acting on it, so we re-ran E41's selection procedure verbatim on this grid—identical models and tasks, the only difference being whether the JSON level is constrained.

The payoff shrinks by a factor of 13 and its interval touches zero, so our locked prediction that it would survive above 0.02 is **refuted**. The mechanism is the harness main effect again, not a vanished interaction: constraining lifts JSON almost uniformly, so the single best fixed interface suits 4 of 5 models (against 3 of 5 unconstrained) and *which* interface is best flips from ReAct to JSON. A fixed default therefore captures nearly all of the available benefit.

Table 45: E45: held-out gain of per-model interface selection over the best *fixed* interface, 400 random splits. The payoff is large for free-text protocols and indistinguishable from zero once one protocol is schema-guided.

	unconstrained			schema-guided		
probe K	gain	CI	splits >0	gain	CI	splits >0
5	$+0.121$	$[0.030, 0.187]$	98.8%	$+0.007$	$[0.000, 0.017]$	45.8%
10	$+0.141$	$[0.091, 0.195]$	99.8%	$+\mathbf{0.011}$	$[0.000, 0.019]$	65.2%
26	$+0.140$	$[0.092, 0.215]$	100%	$+0.016$	$[0.000, 0.031]$	82.0%

We state the consequence plainly because it is the one that most weakens this work, and we pre-committed to reporting it: **for a practitioner using structured output, measuring which interface suits which model buys almost nothing on this suite.** The strong practical recommendation belongs to free-text protocols.

What is *not* established is the case that matters most in production: we constrained *one* of two protocols. Two *equally* constrained alternatives—a JSON schema against native function calling—are untested here, and the uniform lift that kills the payoff in our design would not obviously arise between them. That is the experiment that would settle the practical claim, and we name it rather than assume its outcome.

Two *equally* constrained protocols: the decisive test (E46). E45's collapse could have been an artifact of comparing a constrained protocol against an unconstrained one, so we ran the test that distinguishes those: Schema A (our shape, {`"tool","args"`}) against Schema B (the OpenAI function-calling shape, {`"name","arguments"`}), both delivered through the same mechanism. Adherence is 0.996 and 1.000, so the asymmetry is genuinely gone.

Table 46: E46: with compliance equalized, the interaction is small and the payoff does not return. Leave-one-model-out shows the residual interaction is carried by `phi-4`.

model	Schema A	Schema B	contrast	design	
`llama-3.1-8b`	0.404	0.462	-0.058	κ	$\mathbf{0.125}$
`llama-3.3-70b`	0.538	0.442	$+0.096$	CI	$[0.063, 0.219]$
`mistral-small-24b`	0.923	0.942	-0.019	p	5×10^{-4}
`phi-4`	$\mathbf{0.962}$	0.577	$+\mathbf{0.385}$	$S_{\mathcal{H}}$	0.009
`qwen3-14b`	0.923	0.904	$+0.019$	κ w/o `phi-4`	0.015

Our mechanism check **confirms** ($S_{\mathcal{H}}$=0.009 against E44's 0.115: equalizing constraint does remove the main effect) while both substantive predictions are **refuted**. κ is 0.125, below our locked 0.20 and carried by one model of five. The held-out payoff is $+0.0003$ (CI $[-0.029, 0.010]$, positive in 34% of splits): measuring which schema suits which model buys nothing on average. Removing the main effect did not rescue the payoff, because what remains is too small and too concentrated.

What this means for the paper's central claim. Reading the three designs together—κ=0.475 free-text, 0.267 with one side constrained, 0.106 with both, all bias-corrected—**most of what our headline interaction measures is protocol *compliance*, not protocol *form***. Equalize compliance and roughly three quarters of it goes.

This does not make the phenomenon unreal. Compliance differences are large (κ on adherence alone is 0.566, App. AS) and squarely deployment-relevant: `phi-4` emits parseable JSON in 19 of 52 episodes unconstrained and 52 of 52 constrained. Nor is form irrelevant—`phi-4` still swings 0.385 on the *shape* of a schema at full compliance both ways. What it removes is the general framing. The claim the evidence supports is the narrower one we now make: the interaction is large on free-text protocols and largely a compliance effect; per-model protocol selection pays there ($+0.10$) and not under structured output, neither against an unconstrained alternative ($+0.011$) nor between two constrained ones ($+0.000$).

The four rungs are four different experiments. The main text quotes three rungs measured on one substrate and then a fourth point from BFCL, and it is worth being explicit that these are not one controlled sweep with four settings. The first two points are the two arms of a single same-substrate intervention (E44). The third varies only the schema *shape* on that same substrate (E46). The fourth is external BFCL, where κ sits *below* its own no-interaction null (p=1.00) — undetectable rather than measured-as-zero, which is a weaker statement than a measured zero. We read the first three as the intervention and the fourth as triangulation.

Y DOES THE LADDER HOLD ON SOMEONE ELSE'S TASKS? (E60, E61)

> **Read this first.** Everything in §Y.2 and before it was measured through a JSON parser of ours that rejected valid tool calls whose arguments carried a JSON payload — which is most of what Harness-Bench asks a model to write. On the failures whose full reply we stored, 43–50% were valid calls we discarded (App. AX, C8). A rejected call changes what the model is told next, so those episodes could not be repaired by re-reading them; the whole collection was run again. **§Y.5 is the result that stands.** The earlier sections are kept because the chain of what we measured, how it broke, and what replaced it is the part a reader can check.
>
> **Second, and separate.** E60 and E61 below also used a 600-token output budget. That budget turned out to cut off schema-obedient replies mid-file, which made the middle rung uninterpretable. We collected the whole thing again at a budget where that stops happening, and the fresh run disagrees with E61 about the first step. **E64 (§Y.2) supersedes E61**, and the paper's external claim is E64's. E60 and E61 are kept because the chain matters: what we measured, why it was wrong, and what replaced it.

The main result of this paper is a ladder with three rungs. Let both sides write the tool call however they like, and the model and the harness interact a lot. Force one side to fill in a fixed form, and the interaction drops. Force both sides, so the only thing left that differs is what the fields are called, and almost nothing is left. We measured that on our own tasks with five models. The obvious question is whether it was a fact about tool calls or a fact about our tasks.

So we ran the same three rungs on Harness-Bench (Yao et al., 2026), which we did not write, and with 13 models instead of five.

What the three rungs are. A rung is a comparison between two ways of asking for the same tool call. The tools never change. Only the wrapper does.

Rung	first way	second way	what is pinned down
L1	`json` (free text)	`react` (free text)	nothing
L2	`json_a` (fixed form)	`react` (free text)	one side
L3	`json_a` (fixed form)	`json_b` (fixed form)	both sides; only the field names differ

At L3 the two ways are as close as two ways can be. One asks for `tool` and `args`, the other asks for `name` and `arguments`. Same tools, same task, same model. If the interaction were really about the wrapper, this is where it should die.

The first attempt failed its own gate, and the gate was ours. Before looking at any rung, E60 checked that the fixed form was actually being enforced. We wrote that check as: at least 90% of replies should come back as a tool call we can read. It came back at 82%, so E60 reported no ladder at all. That verdict stands and is in the ledger as refuted.

The check was the wrong check. A reply can follow the form perfectly and still fail to parse, because it ran out of room half way through. Harness-Bench tasks ask the model to write files, and a file written inside a JSON string is long. Here is a real reply, cut off exactly where our budget ran out:

```
{"tool":"write_file","args":{"path":"in/schema.sql",
 "content":"DROP TABLE IF EXISTS customers;\n
 CREATE TABLE customers (\n  customer_id TEXT PRIMARY KEY,\n
region
```

That reply obeyed the form. It just stopped. Counting it as a broken form blames the model for our budget.

We can tell the two apart, because the API says which happened. In a pilot of 102 failures across eight models, 94 of 95 failures on the fixed form were `finish_reason="length"`, meaning the reply was cut off. On free text `react`, only 1 of 7 was. So on the fixed forms almost every failure is about length, not about form.

The corrected check. E61 re-states the gate as the thing it was meant to measure: how often a reply actually breaks the form, by being prose, or prose wrapped around some JSON. The threshold is 2%. Cut-off replies are counted and reported separately, not charged to the gateway.

form	breaks the form	cut off by our budget	steps
`json_a`	0.29%	14.5%	3,802
`json_b`	0.24%	14.8%	3,824

Three replies in a thousand are prose. That is what this gate measures and all it measures. It is worth being exact about the word: we request a schema through the gateway's structured-output field with `strict` set to *false*, so this is a schema the model is *asked* to follow, not one the decoder makes it follow. The gate also says nothing about whether the JSON is syntactically valid or whether our own parser can read it — and the second of those turned out to matter enormously (App. AX, C8).

This is a re-analysis, and we say so. E61 uses the same cells E60 collected. Same 13 models, same 32 tasks, same 600-token budget, same code that fetched them. It is one re-analysis with a corrected gate, pre-registered before it was run. It is **not** an independent repeat of E60, and it must not be described as one. Against the claim this paper makes, though, it is an outside test: tasks we did not write, scored by their own checker, with 13 models rather than five.

The one thing that could make a re-specified gate dishonest is choosing it after seeing the answer. That did not happen, and it is checkable rather than a promise. E60's analysis stops at the gate and returns immediately. Its released file contains `"levels": {}` and the words `kappa`, `S_interaction` and `p_vs_null` appear nowhere in it. No interaction number had been computed on this data when the corrected gate was written. The three ladder predictions were carried over word for word.

What we found. The ladder holds.

	κ, the share		S_{MH}, the raw amount	
rung	their tasks	our tasks	their tasks	our tasks
L1 nothing pinned	0.460	0.475	0.040	0.117
L2 one side pinned	0.217	0.267	0.028	0.067
L3 both sides pinned	0.000	0.106	0.003	0.030

Both columns fall, on both measures, on tasks we did not write. At L1, p=0.011 against the mean-matched null and p=0.00025 under the task-clustered sign-flip, with a task bootstrap interval of $[0.244, 0.621]$. At L2, p=0.035 and p=0.00025, interval $[0.046, 0.421]$. At L3 the raw κ is 0.031 and the bias correction takes it to zero, with p=1.0: there is nothing left to find. That is the end of the ladder, and on their tasks it ends lower than on ours.

We also checked our own instrument, and it did not come out clean. Cut-off replies are not spread evenly. On L2 they hit `json_a` 18 times more often than `react` (13.1% against 0.74%). A model that writes long files loses more steps on the JSON side than a terse model does. That is a model-by-harness interaction we created with our own token budget, and it would look exactly like the thing we are trying to measure.

So we measured it, using the same decomposition on the cut-off rate itself. We fixed the rule in advance: a rung where the cut-off rate carries at least as much interaction as the outcome does is labelled *confounded* and cannot carry a claim on its own.

rung	κ of the outcome	κ of the cut-off rate	verdict
L1	0.460	0.298	clean
L2	0.217	0.420	**confounded**
L3	0.000	0.000	flagged, but see below

L2 is a real problem. At that rung our budget carries more interaction structure than the outcome does, and we do not think L2 can stand alone.

L3's flag is a tie at zero. The rule fires on "greater than or equal", and both numbers are exactly zero, so it fires where there is no interaction in either quantity to confuse. Cut-off replies at L3 are symmetric across the two columns (13.3% against 13.5%), so they cannot manufacture a difference between them. The rule as we wrote it cannot tell *confounded* from *nothing to confound*. That is a flaw in our rule. We report the verdict as the rule produced it and describe the flaw here rather than quietly fixing it afterwards, and the ledger carries the check as a failed gate.

The check that answers the confound. We fixed this test in advance too. Sort the tasks by how often replies get cut off, take the least-affected third, and run the ladder again there. Sorting is on tasks, not on individual replies, so we are not selecting on the outcome. On those 11 tasks, where cut-off replies are almost absent (rate ≤ 0.034):

$$\kappa:\ 0.404\ \rightarrow\ 0.165\ \rightarrow\ 0.000 \qquad S_{MH}:\ 0.067\ \rightarrow\ 0.050\ \rightarrow\ 0.007$$

The ladder is still there with the confound taken away. So the fall is not something our token budget manufactured.

Two models were dropped, for a reason that has nothing to do with the answer. Our client sent the schema on one provider's path and silently dropped it on another's. Two models routed through the second path, so their "constrained" cells were ordinary free-text calls carrying a constrained label. Nothing errored. We found it by asking each model the same question twice, once with the schema and once without, and getting back byte-identical replies including prose and a code fence the schema forbids. Those 31 cells are quarantined with the diagnosis rather than deleted, the client now refuses a constrained request it cannot enforce instead of answering it, and two tests pin both halves. We checked every directory holding constrained cells: the main result routes every model through the path that worked, which matches its 99.6% adherence.

What this does and does not show. It shows the ladder is not a fact about our tasks. On 32 tasks written by someone else, scored by their checker, with 13 models, the interaction falls as the call gets pinned down, and it falls on the raw amount as well as the share, and it still falls when the tasks most affected by our token budget are removed. It does not show the ladder holds for every agent framework, every tool set, or every task type; L2 on its own is confounded by our budget; and the whole thing is one re-analysis of one collection, not a repeat of it.

Y.1 Is the ordering real? Same tasks, and uncertainty on the drops (E62)

A reviewer raised two problems with how the ladder above is reported. Both are right, and both are about the reporting rather than the data.

Problem one: the three rungs do not use the same tasks. L1 uses 32 tasks, L2 uses 31, L3 uses 27. A task drops out of a rung when any of the 13 models has a missing cell in either column of that rung. So the three numbers are measured on three slightly different sets of tasks, and some of the fall could be the tasks changing rather than the constraint changing.

The fix is to use only the 27 tasks that all three rungs share. Then every rung is measured on the same tasks:

	L1	L2	L3
κ, as published (32/31/27 tasks)	0.460	0.217	0.000
κ, same 27 tasks throughout	0.434	0.163	0.000
S_{MH}, same 27 tasks throughout	0.040	0.027	0.003

It still falls, on both measures. So the unequal task counts were not producing the ordering.

Problem two: an interval on each rung says nothing about the drop between two rungs. E61 gives each rung its own bootstrap interval. That is the wrong interval for this claim. The rungs are measured on the same tasks, so they move together; if a resample of tasks happens to be easy, every rung moves at once. Two separate intervals throw that away.

The right test resamples the tasks once and recomputes *all three* rungs on that same resample, then looks at the difference. We did that 5,000 times:

drop	point	95% interval	fraction of draws where the drop is positive
L1 − L2	0.272	[−0.029, 0.429]	0.959
L2 − L3	0.163	[0.012, 0.379]	0.981

The whole ladder is strictly decreasing in 94.0% of draws (90.8% for the absolute interaction).

What we now say, and what we stop saying. The second drop is separated from zero. The first is not: its interval reaches just below zero, at −0.029. So on this external suite we report the first drop as **directionally consistent but not interval-separated**. The point estimate does not change. The claim about how sure we are does.

We fixed that wording rule in the protocol before computing the interval, precisely so the decision was not made after seeing which side of zero it landed on.

This matters less for the paper's advice than it might look. The advice is to pin the call down all the way. That is the second step — from a half-specified call to a fully specified one — and that is the step whose paired difference interval excludes zero.

Does one model carry the ordering? No. Drop each of the 13 models in turn and recompute the common-task ladder. It is strictly decreasing in all 13 refits, in κ and in the absolute interaction. Any one model can be removed and the pattern stays.

Scope. E62 re-analyses E61's cells. It adds no data. It can only qualify what E61 reported, not strengthen it, and it is not counted as an independent confirmation.

Y.2 The fresh run, at a budget that stops cutting replies off (E64)

E61 had two fair problems. It re-analysed cells rather than collecting new ones, and its middle rung was confounded by our own token budget. Both are fixed the same way: collect again, with the gate already frozen, and give the model enough room that it stops running out.

How much room is enough is a measurement, not a preference. We asked the API directly, using the stop reason it reports for every reply:

budget	steps cut off	
600 tokens	18–22%	what E60 and E61 used
1800 tokens	19.7%	no better — models simply write more
4000 tokens	0.0%	

So 4000, applied to all four formats equally. Giving it only to the constrained ones would swap one confound for another. Everything else is held exactly as E61 had it: same 13 models, same pinned tasks, same step limit, temperature 0. 1,662 of 1,664 cells were collected; 30 tasks are complete in all three rungs and every number below uses those same 30.

The result, and it is not what E61 reported.

rung	κ	S_{MH}	vs. its own no-interaction null
L1 nothing pinned down	0.316	0.050	p=0.002
L2 one side given a schema	0.320	0.053	p=0.001
L3 both sides given a schema	0.000	0.004	p=1.000

drop	point	95% paired interval	fraction of draws positive
L1 − L2	−0.004	[−0.179, 0.147]	0.489
L2 − L3	0.320	[0.195, 0.484]	1.000

The first step is not small here. It is absent. Giving one side a schema changes κ by -0.004, and the interval on that change is centred on zero. The whole ladder comes out strictly decreasing in 48.9% of draws, which is a coin flip.

The second step is large and certain. Pinning down both sides, so that the two harnesses differ only in whether the fields are called `tool`/`args` or `name`/`arguments`, takes κ from 0.32 to 0.00.

What we can say about why, and what we cannot. What is measured is that `json` and `json_a` behave alike on this suite: per-model outcomes correlate at r=0.65 and their rates of malformed replies track each other. So comparing `json_a` against `react` is nearly the same comparison as `json` against `react`, and it gives nearly the same answer. At L3 both columns are JSON, whatever difficulty that carries is shared, and what is left is the naming — which is worth nothing.

Why the schema changes so little here, when it changes a great deal on our own tasks, we do not know. We had an explanation and it is refuted; §Y.3 is the test that killed it.

BFCL. On BFCL v3 every model is handed the same schema and κ is 0.00 (App. J), as at L3. Because the specification trend did not replicate consistently across settings (App. V), this match does not isolate schema specification as the cause of either near-zero estimate.

What we got wrong about our own instrument, again. Two things, both reported as the locked rules produced them.

The pre-registered instrument check (P6) watches the truncation rate and it *fails* at all three rungs. The flag is empty. Truncation now happens on 0.08–0.26% of steps, and the interaction share of something that rare is noise — the rule cannot tell "confounded" from "too rare to measure". This is the third time a rule of this shape has misfired for us, and the pattern is always the same: a ratio computed on a quantity that has almost stopped occurring.

Truncation did not leave without a replacement. Malformed JSON took its place as the failure that hits one side harder: 8.5% of `json_a` steps against 0.10% of `react`'s. A post-hoc check on that rate flags all three rungs. But when we test it directly, it does not carry the result: the correlation between how often a model writes broken JSON and how much worse it does on the JSON side is -0.05 at L1 and -0.16 at L2. We report the flag *and* the test that undercuts it.

Where this leaves the paper's claim. On our own suite the interaction falls at both steps, $0.47 \to 0.27 \to 0.11$. On a suite we did not write it falls only at the second, $0.32 \to 0.32 \to 0.00$. The part that survives on both, and on BFCL, is the second step. That is what the paper claims, and the first step is now reported as something we see on our own tasks and could not reproduce on someone else's.

Y.3 The explanation we had for that, and the test that refuted it (E65)

When the first step failed to replicate we had a tidy reason ready. A schema constrains the *wrapper* of a tool call — that there is a `tool` field and an `args` field — and says nothing about what goes inside it. On our own tasks the inside is tiny: the arguments an agent writes have a median of 21 bytes. On Harness-Bench the agent has to write a whole file in there, and the benchmark's own

expected answers have a median of 995 bytes. So, the story went, constraining the wrapper helps when the payload is small and does nothing when the payload is large.

That story was invented after seeing both results. Stories invented after the fact are the ones most likely to be wrong, so we tested it.

A better test than a third benchmark. The obvious move is to find a third suite that is not about writing files. That would confound payload size with everything else about a new benchmark — its tasks, its scorer, its difficulty, its tools. Harness-Bench already contains both kinds of task, so splitting *it* by payload size holds the models, the formats, the scorer and the collection fixed and varies only the thing the story is about.

We split on the byte size of each task's own `ground_truth.json`, which ships with the benchmark and states what the answer has to contain. It is written by the benchmark's authors and cannot be influenced by which model or format we ran. We deliberately did *not* split on anything measured from model behaviour — reply length, malformed rate, truncation — because those come after the outcome we were about to analyse.

It does not hold.

tasks	κ free	κ one pinned	first step	95% interval
15 smallest (200–981 B)	0.282	0.312	-0.030	$[-0.226,\ 0.126]$
15 largest (1010–4730 B)	0.344	0.224	$+0.121$	$[-0.400,\ 0.435]$

The first step does not come back where payloads are small. The point estimates run the other way, and the difference between the two halves is -0.15 where the story needed it positive. Sharpening the contrast to the ten smallest against the ten largest makes it worse for us, not better: -0.017 against $+0.257$.

The proposed mechanism is missing too. If large payloads were the problem, replies should break more often on the large half. They break slightly *less*: 8.3% against 9.6%.

What this does and does not establish. Every interval above spans zero, and the protocol said in advance that they would — 15 tasks a side, when 30 already gave $[-0.179, 0.147]$. So this is not evidence of a reversed effect either. It is powered to check a direction, and the direction is not the one the story predicted.

The honest position is therefore smaller than the one we started with: **our suite shows a first step, an external suite does not, and we cannot say why.** We are not replacing the refuted explanation with a second guess. A third suite is still worth running, now with the extra requirement that it should not be chosen to fit a hypothesis we have already failed to support.

Y.4 Is the disagreement just our small model pool? (E66)

Our own ladder uses five models. The external one uses thirteen. Before saying anything about why they disagree, the dull explanation has to be ruled out: that five models cannot tell the two answers apart.

Read this as a bound on sampling, not as a measurement. The subsets below are drawn from E64's cells, the collection this appendix retires because its κ came through the broken parser. What it answers — how much a five-model pool can wander — does not depend on the parser being right, because every subset is scored the same way. No number in it is claimed as a result about the world.

So we took the external cells, where the first step is -0.004 on all thirteen models, and enumerated *every* five-model subset — all 1,287 of them — recomputing the first step each time.

five-model subsets of a suite whose full answer is -0.004	
median first step	$+0.023$
interquartile range	0.202
5th–95th percentile	-0.415 to $+0.199$
full range	-0.943 to $+0.906$
fraction reaching our suite's $+0.208$	$\mathbf{4.0}\%$

Two things follow, and they point in opposite directions. Our first rung is not something a five-model pool hands you routinely: a suite with no first step produces one as large as ours about one time in twenty-five. So the difference is not explained by *which* models we happened to run — which is the only thing this resampling varies. It holds the task sample, the scorer and the substrate fixed, so it cannot tell us the two suites differ for any reason beyond model composition. That narrower statement is the one we make.

But five models is a wide instrument for this particular quantity. Its interquartile range here is 0.202, which is about the size of the effect we are measuring with it, and individual five-model subsets range from -0.94 to $+0.91$. So the first rung on our own suite should be read as a real but loosely-pinned number, not a precise one. We report it that way.

Y.5 The result that stands: re-collected behind a working parser (E67)

We collected the whole thing again — $1{,}662$ of $1{,}664$ cells, same 13 models, same tasks, same four forms, same budget — with the parser repaired. The analysis is not new: we ran E64's own analyzer, committed before E64's cells existed, pointed at the new directory. Writing no new analysis code is the point, because it leaves no room to shape the analysis around data already in hand.

The repair worked. This is worth showing because it says how much of the earlier appendix was our own instrument:

form	adherence before	after	malformed before	after
`json_a`	0.893	$\mathbf{0.983}$	0.0876	$\mathbf{0.0008}$
`json_b`	0.894	$\mathbf{0.980}$	0.0863	$\mathbf{0.0005}$
`react`	0.980	0.974	0.0010	0.0029

`react` does not move, which is exactly what has to happen: it never used the JSON parser. The "malformed JSON" that §Y.2 built an argument around was about 99% ours.

And the sweep does not fall.

rung	κ	S_{MH}	vs. its own null
L1 nothing pinned down	0.113	0.025	p=0.085
L2 one side given a schema	$\mathbf{0.395}$	0.060	p=0.00025
L3 both sides given a schema	0.000	0.010	p=1.000

step	point	95% paired interval
one side pinned, vs. neither	-0.282	$[-0.412,\ -0.115]$
both pinned, vs. one	$+0.395$	$[\ 0.243,\ \ 0.511]$

Giving *one* side a schema roughly triples the interaction here ($0.1127 \rightarrow 0.3947$), and that paired difference interval excludes zero on the negative side. Giving both sides one takes it to zero. The sweep is strictly decreasing in 0.06% of bootstrap draws — it is not a ladder in either direction. Note also that the free-text pair on its own does not clear a 0.05 bar (p=0.085): what is clearly present on this suite is the interaction that appears *once one side is constrained*.

Why the middle rung goes up, checked rather than asserted. If a schema raises the interaction because models differ in how well they *use* one, then the per-model gap between the constrained form and ReAct should be more spread out than the gap between free-text JSON and ReAct. It is: 0.157 against 0.111. And the plain version of the same fact is easier to read:

model	free-text JSON	given a schema	
`claude-haiku-4.5`	0.485	0.093	the schema wrecks it
`hermes-3-70b`	0.341	0.465	the schema helps it

Across the 13 models a schema helped 7 and hurt 6. Handing everyone the same schema does not make them behave alike; it sorts them. That is interaction, which is why constraining one side raises κ rather than lowering it. (This explanation is *exploratory* — it was formed after seeing the result. The prediction it makes about spread was stated before being computed, and it held, but that is one check and not a pre-registration.)

What this does to the paper's claim. Our suite and this one disagree about the middle rung, in opposite directions, and both measurements are now made with sound instruments. So the honest claim is narrower than the one we set out with:

> Specify the call **fully** — so that two harnesses differ only in what the fields are called — and the interaction drops to 0.11 on our suite, 0.00 on Harness-Bench and 0.00 on BFCL v3 (App. J). Specify *half* of it and the result depends on the substrate: it helped on ours and hurt here.
>
> We are careful not to call the remainder negligible. We declared a margin in advance — the residual counts as negligible if the 95% interval on what per-model measurement *buys* sits below 0.02 held-out success — with a guard that the margin must *fail* at the unconstrained rung, or it is too loose to mean anything. The guard failed (App. AB.4), so by our own rule the label is dropped, including on the substrate where the margin passed. What is left is the measurement: at the fully specified rung, measuring per model buys -0.0003 on our suite and -0.0004 externally, each with an interval about two to three points wide. A size, with its width, and no verdict attached.

The instrument checks are empty now, and the rule that reads them is still wrong. P6 is REFUTED at all three rungs. Truncation now runs at 0.0016–0.0029 and κ of the truncation rate reads 1.0 off a couple of steps in a single cell. That is the fourth time this rule shape has misfired for us, always the same way: it takes the interaction share of a quantity that has nearly stopped occurring. A ratio needs its denominator to exist, and the rule needs a floor on the base rate before its verdict means anything. We report it as the locked rule produced it and say plainly that it is empty.

Y.6 THE FOUR-FORM EXTERNAL GRID, REDONE (E69)

App. AG reported $\kappa = 0.553$ over four protocol forms on Harness-Bench and the paper cited it for two rounds as external replication. Its `json` column came through the broken parser, so the number is retired. Here it is again, with `xml` and `pythonic` collected behind the fix and combined with E67's `json` and `react` — 13 models, 29 tasks complete in all four columns, everything gathered after the repair. It is worth naming this for what it is: a **repaired partial extension**, not a fresh independent replication. Two of its four columns are E67's cells and only two were collected for it. Every column is post-repair, which is what makes the four-form number usable, but the grid was not gathered in one pass.

	S_M	$S_{\mathcal{H}}$	$S_{M\mathcal{H}}$	κ	
our suite (b=4)	0.120	0.022	0.121	0.487	
external, repaired (b=4)	0.089	0.075	0.043	**0.222**	p=0.003, CI $[0.176, 0.363]$

A prediction we had recorded as refuted lands on the right side again. E58 pre-registered that external κ would be *smaller* than our own suite's. Measured through the broken parser it came out larger — 0.553 against 0.487 — and we reported the prediction as refuted and wrote an appendix explaining the model-axis compression that seemed to account for it. With a parser that works the point estimate is on the predicted side: 0.222 against 0.487.

Be careful about how much that is worth. Bootstrapping tasks and models on both grids, the difference is $+0.266$ with a 95% interval of $[-0.189, 0.679]$, so the *ordering* is not established — two grids of seven and thirteen models cannot resolve a gap this size, and 86% of draws pointing the right way is not the same as a result. What the repair changed is the *sign*: a pre-registered prediction sat on the wrong side of ours under a broken instrument and sits on the right side under a working one. That is evidence about the instrument, which is what we claim, and it is a check we could not have run had we quietly dropped a prediction that "didn't work" instead of recording it.

No single form is safe there either, though it is closer than on our suite. Three distinct forms are somebody's best: `xml` for ten of the thirteen models, `json` for two, `react` for one. On our own suite the best form is spread more evenly. So that benchmark has something nearer to a default interface than ours does, which is worth saying because it is the direction that makes our headline look *less* general, not more.

Missing data. Three tasks are dropped because a cell is absent or its oracle raised while grading. The artifact lists each one with the reason, under the rule adopted in App. AX: an oracle crash is missing data, not a score of zero.

Y.7 The same design, re-run today: the calendar moves it (E108)

A reviewer put two of our problems together into one. Our own ladder's first two rungs come from collections about two months apart, so the constraint effect and whatever changed between the collections cannot be told apart. And the frozen replication changed the model pool *and* the date on the same external substrate, so its disagreement with E67 cannot be pinned on the pool. Both are the same missing number: **how much does the calendar move this design when the models are held fixed?**

So we re-ran E67's design. Same Harness-Bench commit, same fixed task list, same four forms, same 4000-token budget, temperature 0, ten steps, and `analyze_e64.py` called as a subprocess so nothing in the comparison could change how the ladder is computed. Only the date changed. The protocol is `experiments/108_contemporaneous/protocol.md`, locked with no data in the tree, and the artifact is `data/e108_contemporaneous.json`; the 1280 new cells are in `data/hbench_e108`.

Which models, by a rule written before the run. `hermes-3-70b` is no longer served, so it is dropped and named. Of the remaining twelve we took those whose completion price is at most $\$1.00$ per million tokens, a cap declared in the protocol because re-running a 1280-cell design costs money and we would rather declare a cap than quietly pick cheap models afterwards. That excludes `claude-haiku-4.5` and `mixtral-8x22b`, both named. Ten models remain, and *August's own cells are re-analysed on the same ten*, so both sides of every comparison share a model list.

rung	$\widetilde{\kappa}$		pooled outcome	
	August	today	August	today
1: `json` | `react`	0.3115	**0.4223**	0.2422	**0.4161**
2: `json_a` | `react`	0.2538	**0.4282**	0.2539	**0.4258**
3: `json_a` | `json_b`	0.0000	0.0187	0.2627	**0.5266**

Raw, uncorrected, the same three rungs read $0.3428 \to 0.2980 \to 0.0345$ in August and $0.4327 \to 0.4389 \to 0.3217$ today, and the absolute interaction share $S_{M\mathcal{H}}$ reads $0.0540 \to 0.0437 \to 0.0041$ against $0.0729 \to 0.0658 \to 0.0146$. The third rung is where raw and corrected part company: 0.3217 raw against 0.0187 corrected. That is the bias correction doing exactly what App. T's sweep says it is for — with nine interaction degrees of freedom and a graded response, $\mathrm{SS}_{M\mathcal{H}}$ of that size

is what noise alone produces — and it is why we read the corrected column. A reader who prefers the raw ratio should read the whole raw row, not the one cell that suits an argument.

What the four locked predictions did. **T1 confirmed**: every rung passes the same validity gate today. **T2 confirmed**: today's rung-1 $\widetilde{\kappa}$, 0.4223, is inside August's task-bootstrap interval $[0.18, 0.4744]$ on these ten models. **T3 refuted**: the order changed. August falls $0.31 \to 0.25 \to 0.00$; today it reads $0.42 \to 0.43 \to 0.02$, so the first step is gone and the middle rung is the highest. **T4 refuted, and not narrowly**: the pooled outcome roughly *doubled* at every rung — $0.24 \to 0.42$, $0.25 \to 0.43$, $0.26 \to 0.53$ — on the same models and the same tasks.

So the calendar is not a caveat here, it is an effect. Ten model names, one task list, one scorer, one analysis script, and success moved by 17 to 26 points. Whatever is behind a model name changed while the name did not, which is the thing this paper is about, arriving on our own instrument. The consequences, stated plainly:

- **No comparison between two of our collections can attribute a difference to design alone.** That includes our own first two rungs, about two months apart, and it includes the frozen replication's disagreement with E67 — which is confounded by date as well as by model pool. We had called this a limitation; it is now a measurement.
- **The corrected endpoint survives the calendar; the ladder does not.** With both sides constrained the *corrected* share is near zero in both collections (0.0000 and 0.0187). We say *corrected* on purpose: the raw share at that rung went $0.0345 \to 0.3217$ and the absolute share $0.0041 \to 0.0146$, so the endpoint claim rests on the correction, not on the raw ratio. The monotone fall survives nothing: the step from rung 1 to rung 2 is gone.
- **We do not let the success change stand in for the κ change.** They moved together here, but they are different quantities and T4's fallback says to report both, which is what the table does.

What E111 later did to this section. Running the same design once more, one day after this one, separated the two halves of the paragraph above. The *success* change is an interval effect: one day moves it by at most 0.0180 against 0.1739 to 0.2639 over twelve days. The κ change is not: one day moves κ at rung 1 by 0.1203, more than the 0.1108 twelve days moved it. So read the κ ladder above as a description of two collections and not as a measurement of an interval (App. Y.8). No number here is re-scored; what changes is what it supports.

And the model list matters as much as the date. The published thirteen-model August ladder is $0.1127 \to 0.3947 \to 0.0000$. The ten-model comparison uses a different August collection: `data/hbench_fresh`, E64's cells of 27 August, collected before the parser repair (unparsed step share 0.11 on the schema forms against 0.02 after it). On those cells the ten models read $0.3115 \to 0.2538 \to 0.0000$; on the repaired collection the same ten models read $0.153 \to 0.186 \to 0.000$ (`data/review_20260912/collection_comparison.json`). So the difference between the thirteen-model and the ten-model ladder mixes the model list with the collection, and we do not attribute it to the model list alone. We report all the ladders rather than the two that agree, because a shape this sensitive to which models and which collection you happened to have is not a law, and saying so is the paper's own headline claim applied to the paper's own evidence.

Both sides replay from the shipped cells. `replay_external_ladder.sh` now regenerates the August and the today ladders from `data/hbench_fresh` and `data/hbench_e108` with the same script and the same ten-model list, and byte-compares each against the released file. Adding that check found a defect worth naming: the analyzer records which cells made a task unusable by iterating a *set* of form names, and Python randomises string hashing per process, so that one diagnostic list's order was not reproducible while every number in the file was. We pin the hash seed in the two places that run the analyzer rather than patch a script the frozen replication's protocol committed to running byte-for-byte.

What this does not do. It is one re-run at one later date, so it measures *this* interval and not drift in general. It is not by itself the fresh frozen study with interleaved runs, repeated-run variability

and held-out tasks that a reviewer asked for. That study is App. Y.8, and it changes what this one supports.

A correction to how we described this interval. And one about which collection is the reference: the directory read by E108, E111 and E117 is `data/hbench_fresh`, E64's collection, not E67's `data/hbench_fixed`; the two were collected on the same day, before and after the parser repair. The protocols for E108 and E111, and an earlier version of this paper, called the reference collection *June* and the gap to the re-run *ten weeks*. Both are wrong. Every shipped cell carries the moment it was written, and those say the reference collection ran on **27 August 2026**, the re-run on **8 September** and the repeat on **9 September**: **twelve days**, then one (`data/collection_dates.json`, derived by `experiments/117_collection_dates/`). No *Harness-Bench* collection predates 25 August; E117 read those directories and no others. The confusion came from the *live* suite, whose collections really do run from May and June to August — that is a different suite, its statements stand (App. U), and "twelve days" was never a fact about it.

We have not edited the protocols: they are the pre-registration and they say what they said. What changes is the label, not the test. Every prediction here compares a short interval against a longer one, and it is scored the same way whether the longer one is twelve days or ten weeks. It does change the reading: **one day moving κ as much as twelve days did is a weaker statement than one day matching ten weeks**, and the corrected sentence is the one that now appears in the main text.

Y.8 The same design again, one day later: success drifts, κ does not (E111)

E108 has an obvious hole and this experiment is the hole. E108 re-ran the external design twelve days after August and found that everything moved. But nobody can tell from E108 alone whether twelve days did that or whether the design simply gives a different answer every time you run it. So we ran it again **one day later**, with nothing changed.

What is new, and what is not. Everything is E108's: the same ten models, the same 32 tasks, the same four forms, the same budget, the same scorer, and `experiments/68_fresh_ladder/analyze_e64.py` called unmodified. Three things are new, and they are the three a reviewer asked for.

1. **Interleaved.** Every collection before this one walked the work list model by model, so one model's cells sit together in time and a provider hiccup lands on one model. Here the work list is shuffled once with seed 111, so the pairs are spread across the window.
2. **Repeated.** The comparison against E108 *is* the repeated-run measurement: same design, about one day apart.
3. **Split.** The task list was split by seed 111 into a development half and a held-out half *before* anything was collected.

The protocol is `experiments/111_fresh_frozen/protocol.md`, committed with no data in the tree; the artifact is `data/e111_fresh_frozen/e111_fresh_frozen.json` and the 1278 cells are in `data/hbench_e111`. Two of the 1280 are missing: the provider answers `api_error` for `qwen-2.5-72b` on two tasks, every time we ask. E108's cells were written between `2026-09-08T06:05` and `08:42`; these between `2026-09-09T06:11` and `12:14`, the later end being the second pass for the cells that failed first time. All three rungs pass the validity gate.

The result, in one table. $\widetilde{\kappa}$ per rung, and how far it moved.

rung	August	E108	E111	moved in twelve days	moved in one day
1 free text both sides	0.3115	0.4223	0.5426	0.1108	**0.1203**
2 one side constrained	0.2538	0.4282	0.4384	0.1744	0.0102
3 both sides constrained	0.0000	0.0187	0.0000	0.0187	**0.0187**

F2 is refuted. We predicted the one-day move would be smaller than the twelve-day move at every rung. At rung 1 it is *larger* — 0.1203 against 0.1108 — and at rung 3 the two are equal to four decimals. Only rung 2 behaves as predicted.

The same three collections on the same tasks. The rows above use each collection's own usable task set (30, 31 and 29 tasks), so a step mixes a date with a change of tasks. A reviewer asked for the matched comparison, and it is now the one the main text quotes; the rows above are kept as sensitivity rows and are not re-scored. On the 27 tasks usable in all three collections (`data/review_20260912/collection_comparison.json`; five excluded tasks are listed there with the missing cell that excludes each), rung 1 reads $0.2967 \to 0.4279 \to 0.5604$ (steps 0.1313 and 0.1325), rung 2 reads $0.2252 \to 0.3773 \to 0.4521$ (0.1521, 0.0748), and rung 3 reads $0.0000 \to 0.1179 \to 0.0000$ (0.1179, -0.1179; the corrected share is floored at zero on two of the three dates, so those steps have a hard edge). Pooled success at rung 1 reads $0.242 \to 0.407 \to 0.427$. Resampling the 27 tasks jointly across the three collections (2,000 draws, seed 20260912) gives 95% percentile intervals of $[-0.19,\ 0.41]$ and $[-0.07,\ 0.33]$ for the two rung-1 steps. Those intervals describe task resampling within these fixed collections; they say nothing about other dates, providers or seeds. The reference collection is E64's, collected before the parser repair; with E67's repaired cells in the reference slot (26 tasks common to all four collections) the twelve-day step at rung 1 is 0.2535 and the one-day step 0.1723.

Pooled success is a different story, and it holds up. F3 confirmed. One day moved the pooled outcome by 0.0180, 0.0114 and 0.0007 at the three rungs, against 0.1739, 0.1719 and 0.2639 over twelve days. So the doubling E108 reported is a real change over that interval. It is κ, not success, that a single day can move as much as twelve days.

What we do about it. The protocol fixed this in advance, so we follow it rather than argue with it.

- The main text does not say the calendar *causes* the move. It reports what the three collections read, and says that date, provider state and seed all changed or are unrecorded between them, so it attributes the difference to none of them.
- The size is the finding: one day moved rung-1 κ about as much as twelve days did, and both moves have task-resampling intervals that span zero. Pooled success is the quantity that moved cleanly, 0.165 over twelve days against 0.020 over one.
- E108's own claim is scoped the same way: its κ ladder comparison stands as a description of two collections, not as a measurement of an interval.
- Nothing is re-scored. E108's numbers are what they were; what changes is what they support.

The held-out half agrees with the development half, and neither is precise. F4 confirmed. The protocol splits the 32 tasks by seed 111 into 16 and 16 *before* collecting. An earlier version of this analysis split the 29 that survived attrition instead, which makes the assignment depend on which cells came back; the split below is the pre-registered one, intersected with what survived, which realises as 14 and 15 tasks. At rung 1, $\widetilde{\kappa}$ is 0.3736 on the development half and 0.6246 on the held-out half, and the held-out value sits inside the development half's task-bootstrap 95% interval. That interval is $[0.1946,\ 0.8908]$. It spans most of the range κ can take, which is the honest reading: fifteen tasks do not pin this quantity down, and a split-half agreement this wide is weak evidence of anything. Correcting the split moved both halves by more than 0.2 — which is the same instability this appendix is about, arriving in the split itself.

What this still does not do. It is two runs, not many. Two runs one day apart bound run-to-run spread from below, not from above: the true spread could be larger. And it says nothing about whether a third run tomorrow would land near either of these.

Y.9 The headline grid collected again, twenty-one days later (E118)

Every stability result above is on some other collection. The headline grid, 7 models $\times$ 4 formats $\times$ 52 tasks, had never been run twice, so a reviewer's sharpest objection stood: the headline κ had no test-retest on its own grid. This experiment is that test. The protocol,

`experiments/118_headline_repeat/protocol.md`, was committed before any cell was collected, and the analysis script was committed before the grid was complete; the result file records the protocol's hash.

Design. Everything is `data/live_grid4`'s: the same seven models, four formats, 52 tasks, 400-token budget, collector and analysis (E82, unmodified). One thing differs, and it is the point: the collector's decoding seed is part of the LLM cache key, and the first grid was collected with no seed, so a re-run with no seed would replay the cached replies byte for byte. This run passed `--seed 118`, which forces every cell to be a fresh draw. The seed is also sent to the provider in the request, so the two runs differ in that one request parameter as well as in date; at temperature 0 it should be inert, and we did not verify that separately. All 1,456 cells were collected (6 that hit an API error on the first pass were collected on the collector's resume pass the same day). The reference grid was collected on 22 August 2026; git records its 1,456 cells in five commits that day. The repeat was written between 03:29 and 13:29 Toronto time on 12 September 2026, 21 days later. Those dates are frozen in `collection_provenance.json` with their evidence basis, the analysis reads them from that file, and the repeat's shipped manifest carries one UTC timestamp per cell that agrees with them to the second. A replay from the archive with every file's modification time reset, under UTC and under Toronto time, reproduces every statistic, every verdict, both dates and the 21-day interval byte for byte; the verifier runs that replay on every run.

Result. The finite-design share, the statistic of Eq. (1), reads 0.4211 against the first run's 0.5009, a difference of −0.0798; its task-bootstrap 95% interval is [0.3380, 0.5143], the first run's is [0.4089, 0.6027], and each point lies inside the other's interval (`data/review_20260912/headline_intervals.json`, E36's recipe). The predictions below were locked on E82's variance-component index (App. T), which reads 0.4814 against the first run's 0.5626, a difference of −0.0812; its task-bootstrap 95% interval is [0.3905, 0.5786], and each run's point estimate lies inside the other's interval (the first run's is [0.4663, 0.6644]; the repeat point sits 0.015 above that interval's lower edge). The two statistics move by the same amount. Pooled success moved from 0.4073 to 0.3826, by −0.0247. The 28 (model, format) cell means rank-correlate across the two runs at Spearman 0.955. At the cell level, 18.1% of the 1,456 outcomes flipped between the runs, below the same-model re-run floor of 0.22 measured on single-turn re-runs in App. U and consistent with it: two collections of the same models on the same tasks disagree on about a fifth of outcomes and on almost nothing at the cell-mean level.

Predictions, locked. H1, the repeat inside [0.4663, 0.6644]: **confirmed**, by 0.015. H2, $|\Delta\kappa| < 0.10$: **confirmed**, by 0.019; that threshold is tighter than the 0.12 one day moved a two-format rung, so this prediction could have failed. H3, pooled success moves by under 0.05: **confirmed** (0.0247). H4, the 28 cell means at $\rho \geq 0.80$: **confirmed** (0.955). The reporting rule was locked with them: 0.563 is not re-scored, and the repeat is printed beside it wherever the headline is reported as a result (Table 49, the caption of Table 2, and Apps. E and T; the abstract, the contributions, the main text's prose and the conclusion no longer report the headline).

Paired over all 52 tasks. Resampling whole task IDs jointly across the two grids (2,000 draws, seed 20260912) gives $\Delta\kappa_{\mathrm{E82}} = -0.081$ with a 95% percentile interval of [−0.165, 0.002]; the finite-design share moves −0.080, [−0.161, 0.000] (`data/review_20260912/collection_comparison.json`). This is a summary of two fixed collections, not an interval over future runs. The reference grid's date is 22 August by git history and 24 August by file time; neither is an API request time, and the repeat's dates are file write times.

What this changes, and what it does not. It bounds what the paper's own day-to-day claim means on the paper's own grid: three weeks moved the headline share by 0.08, less than the 0.12 one day moved rung-1 κ on Harness-Bench (App. Y.8), and the repeat point lies inside the first run's interval. So on this design the headline reproduces within its own interval across two runs 21 days apart. Two collections are two draws: they bound run-to-run spread from below, not from above, a third run could land outside either interval, and a point inside another run's interval is not an equivalence test.

Z WHAT POPULATION DOES κ DESCRIBE? (E54, DESCRIPTIVE)

Requested by an external reviewer, who objected that κ reads as an intrinsic per-swap quantity when it is a property of a finite design. This appendix is **descriptive**: no prediction was locked, and none of its numbers support a claim not already established elsewhere. It exists to bound interpretation.

The design. 7 models $\times$ 4 protocol forms $\times$ 52 tasks, task success as the response, a 400-token budget. Bias-corrected $\kappa = 0.4874$ on the full design, $p < 10^{-3}$ against a mean-matched no-interaction null.

No single model carries it. Dropping each model in turn:

dropped model	κ without it	Δ
`gpt-oss-120b`	0.4617	-0.0257
`gpt-oss-20b`	0.5463	$+0.0589$
`llama-3.1-8b`	0.6438	$+0.1564$
`llama-3.3-70b`	0.5004	$+0.0130$
`mistral-small-24b`	0.4399	-0.0475
`phi-4`	0.3272	-0.1602
`qwen3-14b`	0.4911	$+0.0037$

The largest single-model influence is 0.1602 (`phi-4`). Tasks matter even less: over a leave-one-task-out jackknife the largest shift is 0.0174, the standard deviation of the shifts is 0.0077, and only 0 of 52 tasks move κ by more than 0.02.

But the harness *levels* carry it a great deal — which is the point. Dropping one protocol form at a time moves κ by -0.008, -0.070, -0.106, -0.033 (JSON, ReAct, XML, pythonic respectively). More starkly, the six possible *two*-form sub-designs of the same grid span 0.204 to 0.505, a factor of two and a half:

two-form sub-design	κ
`react vs. xml`	0.505
`json vs. pythonic`	0.358
`json vs. xml`	0.307
`react vs. pythonic`	0.302
`xml vs. pythonic`	0.271
`json vs. react`	0.204

The pre-registered headline pair, JSON vs. ReAct, is the **smallest of the six** (0.204 against 0.505 for the largest). On the earlier stitched grid it was the largest, and we said so; the single-provenance re-collection inverts that ordering (App. E). Our locked choice was therefore the conservative one, and we report the range wherever the headline appears rather than only the design we happened to pre-register.

What this licenses. κ supports a *group-level* statement about a design like the one measured, and a deployment gate that follows from it: re-validate when the scaffold changes. It does not support a per-swap reading, and that is not a caveat we are asserting — we tested it. Selecting which tasks to re-test by their contribution to the interaction recalls true regressions *worse than random* over 168 swaps (E50; App. AW).

AA HOW OFTEN DOES A SMALL PROBE PICK THE WRONG FORMAT, AND WHAT DOES A WRONG PICK COST? (E47)

The current selection-error results are in Section 5.2 and App. L.1. This appendix measures one ingredient they rest on: how often a small probe picks the wrong format, and what a wrong pick costs. It was written for an earlier practitioner claim, now archival, that a 10-task probe per model beats the best fixed interface by $+0.102$ held-out success (App. AS). Two facts sit in tension with it.

First, a probe that small must often pick the wrong interface. Second, we have direct evidence that it does: our own 12-task probe (App. AO) mis-assigned 2 of 7 models relative to the 52-task grid. Either the payoff is fragile, or being wrong is cheap. We had never measured which, so we did.

Design. On the complete $7 \times 4 \times 52$ grid we draw $R{=}2000$ random probes of size $K \in \{5, 10, 12, 26\}$. For each we record whether the probe's argmax interface matches the full-grid choice (*agreement*), and what the choice costs in success measured *only on the* $52{-}K$ *tasks the probe never saw* (*held-out regret*), so a probe never grades its own homework. We report every quantity on **both** substrates—the strict one-action-per-step grid (App. F) and the permissive grid that E16 and E36 live on—because P4 compares against E16, and choosing the substrate after seeing the result would be exactly the freedom this paper argues against. Table 47 gives agreement and regret at each probe size.

Table 47: E47: probe agreement with the 52-task choice, and the held-out success it costs. Agreement climbs with K on both substrates and in both designs; regret stays small throughout. The $b{=}2$ design is the one the headline κ uses.

	$b{=}4$, strict		$b{=}4$, permissive		$b{=}2$, strict	
K	agreement	regret	agreement	regret	agreement	regret
5	0.655	0.061	0.514	0.063	0.816	0.044
10	0.757	0.036	0.638	0.047	0.910	0.020
12	0.780	0.033	0.667	0.045	0.927	0.016
26	0.871	0.022	0.815	0.035	0.987	0.004

Agreement rises with K (P1, confirmed). Monotone on both substrates and in all three designs.

Is the probe unreliable at the size we recommend? (P2, straddles the line—reported as such.) We locked "pooled agreement at $K{=}10$, $b{=}4$ is below 0.75." It is 0.638 on the permissive substrate (**confirmed**) and 0.757 on the strict one (**refuted**), so our own threshold falls between the two measurements. We therefore decline to claim either verdict and state the measurement: a 10-task probe choosing among four interfaces reproduces the full-grid choice roughly 64% of the time on the single-provenance grid. In the two-interface design it is better on the same substrate, 0.832 (0.910 under strict parsing). We name the substrate each time: pairing a four-form number from one grid with a two-form number from the other is how the main text previously read "64% among four, 91% among two".

Being wrong is cheap (P3, confirmed). Mean held-out regret at $K{=}10$ is 0.036 (strict) and 0.047 (permissive), and the worst single model costs 0.067. Set against the $+0.103$ the probe buys, the price of its mistakes is about a third of its benefit—so the payoff survives an argmax that is wrong roughly 30% of the time. This is the reconciliation the tension called for: measuring pays not because the probe is accurate but because its errors are inexpensive.

Why, mechanistically. Per-model agreement is governed by the gap between the best and second-best interface: Spearman $\rho{=}+0.93$ (strict) and $+0.89$ (permissive) between that margin and agreement. `gpt-oss-120b`, whose top two interfaces differ by 0.019, is picked correctly only 38% of the time—and loses just 0.057 for it. `mistral-small-24b`, with a 0.212 margin, is picked correctly 88% of the time. A probe cannot resolve interfaces it cannot distinguish, and precisely then the choice barely matters. Ties are common (22–25% of draws at $K{=}10$) and are a genuine practitioner situation rather than an error; agreement among untied draws is 0.795 (strict) and 0.740 (permissive).

The mechanism predicts instability under a *different* perturbation. If margin is what governs whether an interface assignment is resolvable, it should also predict which assignments move when we change something other than probe size. It does, partially. Going from the permissive grid to the strict one (App. F)—a change in parsing, not in sample size—moves 2 of 7 assignments, and the mover with the smallest permissive margin is `gpt-oss-120b` at 0.019, and every model with a margin above 0.13 is stable. We state this as the directional check it is rather than a clean

rank ordering: `phi-4` has a small margin (0.058) and does *not* move, so margin bounds which assignments are fragile without determining them.

Was our own 12-task mis-assignment a freak? (P4, confirmed.) On the like-for-like permissive substrate, a $K{=}12$ probe over three interfaces mis-assigns 1.71 of 7 models in expectation, and $P(\geq 2 \text{ flips}){=}0.56$. E16's 2 flips were an ordinary draw, not bad luck—which is why App. AO reports its per-model assignments as unresolvable rather than merely unlucky.

What this changes. It converts a reviewer's fair objection—"a 10-task probe cannot possibly pick right"—into a measured statement: it often does not, and that is priced in. It also gives a practitioner the dial. If a wrong interface choice is expensive in their setting, $K{=}26$ buys agreement 0.871 and regret 0.022; if they are choosing between two interfaces rather than four, $K{=}10$ already gives 0.910. Limitation 4 states what remains: reliability is a property of how separated the interfaces are on *this* suite, and a suite whose interfaces sit closer together would show lower agreement—though, by the same mechanism, also lower regret.

AB Is measuring per model actually worth it? (E63)

> **Two of the three parts of this appendix are superseded, and the numbers in them are kept only as a record.** §AB.1 used an incumbent averaged uniformly over all four formats, which is not a policy anyone runs; §AB.2 redoes it properly and its numbers are the ones we stand behind. The κ-gate comparison just below scored regret with the probe free, where a gate cannot win; §AB.3 prices the probe and is the answer that stands.

The paper's advice has two halves. Treat a schema as a candidate and check what it did. Measure per model where the call stays free text. The second half is a decision rule, and until now we had only ever compared it against one alternative: use the single format that works best on average. A reviewer pointed out that this is too easy a comparison, and that the advantage over the other things a practitioner might do had never been isolated. That is fair, so here is the harder comparison.

The decision, stated plainly. You are about to deploy a model. You must pick one call format and stick with it. Before deciding you are allowed to run the model on a few tasks — a *probe* — in each format, and look at the results. Which rule should you use to pick?

How we score a rule. Not by how well it does, but by how much it *loses*. For each model there is a best format, which we can see on the held-out tasks after the fact. **Regret** is how far below that a rule lands. Zero is perfect. Lower is better.

We score by regret rather than by "gain over the fixed default" because gain over the default flatters every rule whenever the default happens to be a bad one. Regret cannot be gamed that way.

7 models, 4 formats, 52 tasks, from the single collection the headline uses. Split the tasks into a 10-task probe and 42 held-out tasks, 400 times.

The rules we compared.

rule	what it does	regret
random	pick a format at random	0.195
best average format	the format with the best pooled probe score	0.153
best average rank	rank the formats inside each model, take the best average rank	0.143
abstain (ours, refuted)	*measure only when the probe margin beats its noise*	0.127
least churn	the format where the new model behaves most like the old one	0.079
κ gate	measure per model only when probe $\kappa \geq 0.20$	0.050
measure per model	**per-model best on the probe**	**0.048**
oracle	the best format, known after the fact; not a usable rule	0.000

What this says. Measuring per model is the best rule you can actually run. It beats the best average format by 0.105 regret, a rank-based version by 0.095, and a rule that picks whichever format keeps

behaviour closest to the outgoing model by 0.031. Every probe-based rule beats picking at random. The probe costs 40 task-runs per model at a 10-task probe, and that cost is the price of the 0.105.

Two things we got wrong, reported as wrong. First, we invented an *abstaining* rule for this experiment: measure per model, but only switch away from the safe default when the probe's margin is bigger than its own noise. It sounds prudent. It is worse than simply measuring — 0.127 against 0.048 — because the threshold makes it abstain on 86% of decisions, so it mostly just plays the default and inherits the default's regret. We fixed that threshold in the protocol before running, so the honest move is to report the rule as refuted rather than to loosen the threshold until it wins. It is dropped from the paper's advice.

Second, and this one matters more. Using κ itself as a gate — measure per model only when the interaction looks big enough to be worth it — gets 0.050, and simply always measuring gets 0.048. At a 26-task probe they are identical, 0.034 each. The gate never fires differently, because on this suite κ is above the threshold almost every time. So **we cannot show that κ adds decision value over just probing every model.** κ earns its place in this paper as the thing that tells you *whether an observed change transfers to another harness*, which is a different question from *which format to deploy*. On this substrate it does not additionally earn its place as a deployment gate, and we say so.

Scope. This is a re-analysis of the headline grid. It compares our own recommendation against alternatives on one suite of 52 tasks with 7 models. It does not show the ranking of rules holds elsewhere.

AB.1 The same question as a real swap, and the cost of doing nothing (E68, superseded)

Superseded by §AB.2, which gives the incumbent a format that suited it rather than one drawn uniformly. The numbers below are the record of what we measured first.

The comparison above treats a model on its own. But this paper is about *substitution*, and a reviewer pointed out that the churn rule there had an invented incumbent — whichever model happened to score best on the probe — which is not a thing anyone does. Framing it as a real swap fixes that, and it also exposes a rule we had never measured even though the first paragraph of this paper describes it.

The scenario. Model A is deployed, running on some format. You replace it with model B. What format does B run on? There are 42 ordered pairs of our seven models and four formats A might have been using, so 168 scenarios. Regret is measured against B's own best held-out format.

what you do	regret	runs	
keep the format A was using	**0.193**	0	the do-nothing default
use the best format on average	0.152	40	ignores the incumbent
pick the format closest to A's behaviour	0.135	40	churn, real incumbent
measure B on 10 tasks per format	**0.048**	40	
best possible for B	0.000	—	not a rule

Doing nothing is the worst option on the board. That is the finding, and it is not the one we expected to have to make explicitly. Keeping the format is not a neutral choice that leaves things as they were; it is worse than ignoring the incumbent completely and putting the new model on whichever format is best on average (0.193 against 0.152). Measuring the new model on ten tasks per format costs 40 task-runs and takes the regret to 0.048, about a quarter of doing nothing.

This is the number behind the sentence this paper opens with. "Same prompts, same tools, same output format" sounds like the careful thing to do. On this suite it is the most expensive of the four options.

One caution about how far that goes. The penalty for inheriting a format does correlate with how badly that format suited the outgoing model, but only at $r{=}0.28$. We had expected the harm

to concentrate where a team had tuned the format to the model it was replacing, which would have been a neat story; at that strength we report the correlation and not the story.

And this still does not show that κ *specifically* improves the decision. What it shows is that measuring beats not measuring. §AB above shows that gating on κ does no better than always measuring, and that limitation is unchanged.

AB.2 THE SAME TEST WITH AN INCUMBENT THAT WAS DEPLOYED SENSIBLY (E70)

A reviewer objected to how §AB.1 set the scene. It averaged every incumbent over all four formats, which quietly turned "keep the format you had" into "keep a format picked at random". Nobody runs that. If the outgoing model had been deployed with any care, it is sitting on a format that worked for *it*, and inheriting that might be far less bad than we made it look.

That is a fair objection to the design, so we redid it. Model A is now placed on the format that maximised A's own success on a 10-task history split, kept disjoint from both the probe and the scoring set. Then A is replaced by B.

what you do	regret	95% interval	runs per swap
keep the format A was using	0.192	[0.087, 0.320]	0
use one fixed default	0.158	[0.045, 0.285]	0
pick the closest-behaviour format	0.122	[0.038, 0.234]	40
measure B on 10 tasks per format	**0.049**	[0.000, 0.121]	40

The objection was reasonable and the answer did not move. Giving the incumbent its own best format changes the number from 0.193 to 0.192. The reason is worth stating plainly, and it is the most useful sentence in this appendix:

> A format chosen carefully for one model is the right format for its replacement only **22**% of the time.

So "keep everything else the same" fails not because the old format was bad, but because being right for the old model carries almost no information about being right for the new one.

What the reviewer's second point costs us, which is real. §AB.1 reported percentiles across task splits with the seven models held fixed, and called them intervals. They are not: they say nothing about which models we happened to have. E70's draws re-split the same 52 tasks *and* resample the seven models with replacement; a reviewer checked, and every interval in this section, the table above included, does so (a draw holds 4.6 distinct models and 35.9 ordered pairs on average, not 42). An earlier version of the main text called these intervals "models fixed", which was wrong. With the models fixed and all 42 ordered pairs scored on each of 2,000 exact 10/10/32 splits (seed 10412, `data/review_20260912/policy_reanalysis.json`), the regrets are 0.191 (keep), 0.156 (fixed default) and 0.050 (probe), and keep minus probe is 0.140 with split percentiles [0.07, 0.21]; that is split sensitivity on one design, not coverage. The paired advantages on E70's own draws:

probing, compared with	advantage	95% interval
keeping the format you had	+0.143	[0.026, 0.269]
one fixed default	+0.109	[−0.017, 0.231]
closest behaviour	+0.073	[−0.030, 0.201]

Only the first survives. Probing beats doing nothing with an interval clear of zero. Probing beats a well-chosen fixed default in 95% of draws and beats the churn rule in 92%, but neither interval excludes zero on seven models. **So the claim we make is the narrow one: measuring beats not measuring. That measuring beats a good default is the direction of the evidence, not a result.**

And the same discipline applied to the whole ranking. A reviewer pointed out that we had been writing "keep-format is the worst of four", which turns a point ordering into an inferential claim. All six pairwise gaps, with paired intervals:

comparison (positive = first has lower regret)		gap	95% interval
keep the format	vs. one fixed default	-0.034	$[-0.159,\ 0.120]$
keep the format	vs. closest behaviour	-0.070	$[-0.213,\ 0.071]$
keep the format	**vs. measure the new model**	$\mathbf{-0.143}$	$[\mathbf{-0.269},\ \mathbf{-0.026}]$
one fixed default	vs. closest behaviour	-0.036	$[-0.178,\ 0.094]$
one fixed default	vs. measure the new model	-0.109	$[-0.231,\ 0.017]$
closest behaviour	vs. measure the new model	-0.073	$[-0.201,\ 0.029]$

One of the six clears zero. We say that one and describe the rest as an ordering.

Uncertainty about which tasks, not just which split. The design above re-partitions the same 52 tasks every draw, so its interval says nothing about which tasks we happened to have. We also bootstrap the task pool, and partition the unique draws so that no task can land in both the probe and the scoring set. Letting one land in both would flatter exactly the probe-based rules we advocate. On those draws the surviving gap holds, but only just: $[-0.284, -0.000]$. Every other gap still spans zero. The unique-draw splits average 6.4 history, 6.4 probe and 20.3 held-out tasks, so this is a variable-budget comparison. That is the honest width of this result on seven models and 52 tasks.

And the cost column was wrong. We charged the fixed-default rule 40 task-runs per swap. Its probe is a one-off calibration reused by every later swap, so its marginal cost is zero and the 40 runs are reported once, separately. Only the two per-model rules pay 40 runs each time.

AB.3 Does κ tell you when a probe is worth paying for? No. (E72)

§AB found that gating on κ does about as well as always probing, and the paper has carried that as a limitation. It was not a fair test. It scored regret with the probe costing nothing, and if probing is free you should always probe — a gate can only ever lose ground by skipping one. κ was being asked to beat a rule it could at best tie.

So we asked the question a team with a budget asks: **is this probe worth running?**

The design. A decision unit is one (substrate, pair of harnesses). The author suite gives six pairs; the external grid gives fifteen, because it has both schema-constrained forms and so contains pairs where the two harnesses differ only in what the fields are called — exactly where a probe ought to be a waste. Loss is held-out regret plus λ per probe, swept over λ so no single price flatters us. The gate's threshold is fitted on one substrate and scored on the other, both directions.

The control is the entire point. A gate that simply probes less often already banks the saving, so `kappa_gate` has to beat a coin *matched on its own probe rate*.

It does not. The advantage over the rate-matched coin is $+0.00006$ in one direction and -0.00018 in the other, and neither interval clears zero.

And the direct version of the question is cleaner than our gate was. Our threshold search hit the top of its grid in both directions, so the "gate" had collapsed into "never probe" — a boundary hit, which is the signal to check an instrument rather than trust it. The cause was ours: probing buys 0.060 on our suite and 0.0033 externally, so the interesting prices sit below the resolution of the grid we swept.

That flaw does not matter for the gate result, and the question can also be asked without any gate, threshold or λ at all. Does a pair's κ predict how much probing that pair buys?

our suite: pair	mean κ	what probing buys
`react` | `xml`	0.498	0.067
`json` | `pythonic`	0.358	0.080
`json` | `xml`	0.303	-0.014
`react` | `pythonic`	0.298	0.132
`xml` | `pythonic`	0.261	0.006
`json` | `react`	0.197	0.090

Read the top and bottom rows together. The pair with the *highest* κ returns 0.067; the pair with the *lowest* returns 0.090. And `json | xml` sits in the middle of the κ ordering while probing it actively *hurts*. Nothing in that column tracks the other.

How strong is that, honestly? Weaker than we first wrote it. Our first version of this paragraph quoted a correlation of -0.04 computed across $1{,}800$ (pair, split) draws, and a reviewer pointed out that this is six pairs measured three hundred times each — the independent unit is the *pair*, not the split, and quoting $n{=}1800$ claims a precision the design does not have. At the unit level:

substrate	pairs	r	95% interval
our suite	6	$+0.04$	$[-0.92,\ 0.79]$
external	15	$+0.12$	$[-0.33,\ 0.63]$

The sign is the other way from the pseudo-replicated number, and both intervals span most of the range a correlation can take. **Six and fifteen harness pairs cannot resolve this question either way.**

So what we say is bounded. We can show no policy value for κ, and we cannot show there is none. The gate wins nothing against a rate-matched coin, and the direct correlation is unresolvable at the unit count we have. What we can say without hedging is that κ should not be *used* as a probe-spending rule on this evidence, and that this paper does not contain a κ-driven policy among its contributions.

κ is a share of model-pair mean-contrast variation inside one design, which is a statement about whether a result will *transfer*. Whether it is worth paying to find a better format is a different question, and we had been letting the two blur.

We asked it once more, at the level the advice is actually given (E106). Six and fifteen pairs inside one or two collections cannot resolve it. So we changed the unit: one design per *collection*, fourteen of them, and the design set fixed by a rule written in the protocol before anything was computed — every directory of ours that is a model $\times$ harness $\times$ task grid with at least three models, two harnesses and twenty-four common tasks, excluding by name the collections we had already declared bad. For each design we computed κ, and three summaries that need no decomposition at all, and what a held-out probe actually saves over keeping the incumbent's format (E70's policy, imported). Then one rank correlation per predictor against that saving. A reviewer found two defects in the design set: two of the fourteen are the parser-defect grids the paper had retired (`hbench_grid`, `hbench_grid_v2`; App. AX), and the BFCL design was scored on the whole-task endpoint that fails its gate rather than the per-turn endpoint that passes it. A second reviewer then found that the twelve designs we called "valid" after those two removals still held a grid the paper itself calls gate-failed (E52's frontier grid, no-parsed-call 0.1671), and asked for one rule. The rule is the rung gate's no-parsed-call threshold with BFCL's outcome band, plus the ledger's parser caveat, applied to every design including our own headline: not collected through the JSON parser the paper retired (correction C8), pooled outcome in $[0.10, 0.90]$, and no-parsed-call rate below 0.15 over the cells the synthesis reads (`data/review_20260915/design_gates.json`). Four author-suite grids fail the rate, the headline grid among them (0.159, the rate §5 discloses), together with `live_4h` (0.210), `live_seed1` (0.184) and the frontier grid. The pre-repair external collection `hbench_fresh` passes the rate (0.006) and is excluded anyway: the ledger says every prediction measured through that parser "is not evidence about the world until that collection is redone behind the repaired parser", a rejected call derails an episode without counting as a no-parsed-call, and `hbench_fixed` is that design redone. The locked table stands as written; the twelve-design rows are kept below as the all-design sensitivity; the seven designs that pass are the valid set.

predictor (fourteen designs)	Spearman ρ vs. what probing saves	95% interval
$\widetilde{\kappa}$, bias-corrected	$+0.411$	$[-0.14,\ 0.79]$
reversal rate (count the sign flips)	$+0.424$	$[-0.15,\ 0.77]$
cell-mean spread across harnesses	-0.051	$[-0.68,\ 0.56]$
harness main-effect share $S_{\mathcal{H}}$	-0.081	$[-0.69,\ 0.56]$

Prediction **V2** said κ would rank the saving at least as well as counting sign flips. It does not: 0.411 against 0.424. The gap is -0.013 with a design-clustered interval of $[-0.53,\ 0.44]$, and κ is the higher of the two in 49.7% of bootstrap draws (`data/e106b_correlation_interval.json`, post hoc, in its own file so E106's locked artifact stays as written). So the two are a tie, a tie is not the prediction we made, and the locked fallback would put the plainer rule in the main text: **count the sign flips**. E120 then scored that rule too, as a reversal gate against a random gate at a matched probe rate, and detected no advantage (0.012 against 0.011; App. K), so the main text carries neither rule: no advantage was detected for κ on this axis, and none for the plainer summary either.

predictor	seven gate-passing designs (valid set)		twelve designs (sensitivity)	
	ρ	95% interval	ρ	95% interval
$\widetilde{\kappa}$, bias-corrected	$+0.179$	$[-0.85,\ 0.96]$	$+0.497$	$[-0.15,\ 0.89]$
reversal rate (count the sign flips)	$+0.607$	$[-0.37,\ 1.00]$	$+0.624$	$[\ 0.05,\ 0.94]$
cell-mean spread across harnesses	$+0.036$	$[-1.00,\ 0.88]$	$+0.049$	$[-0.71,\ 0.64]$
harness main-effect share $S_{\mathcal{H}}$	-0.143	$[-1.00,\ 0.76]$	$+0.028$	$[-0.72,\ 0.66]$

Both post-hoc runs use one random stream per collection from seed 10612, tied ranks averaged (`scipy.stats.spearmanr`; E106 ranked with `argsort`, which breaks ties by position and moves the locked reversal-rate row from 0.424 to 0.429), and BFCL's `turns_correct/n_turns` with null or non-positive denominators excluded. Every design row that survives the gate is byte-identical between the two runs; only the set changed (`data/review_20260915_update/policy_reanalysis.json`, whose first version named the wrong manifest in one field and is kept beside it, and `data/review_20260912/` for the twelve). On the valid set the gap is -0.429 with a design-clustered interval of $[-1.32,\ 0.00]$, and κ is the higher in 0% of draws; on the twelve it is -0.128, $[-0.75,\ 0.37]$, 35%. So κ never ranks the saving better than counting sign flips as a point estimate, and on the valid set it ranks it worse. Neither predictor's own interval excludes zero on the valid set, and seven is an upper bound on the number of independent designs: four of the seven share the same 52 tasks, and 439 of `live_strict`'s cell files are byte-identical to files in `live_chat4k`. Dropping any one design moves ρ for $\widetilde{\kappa}$ between -0.31 and 0.37 and for the reversal rate between 0.37 and 0.83.

One out-of-sample point (E120). After this synthesis was locked and scored we collected one more design on tasks we did not write, WorkBench, 6 models $\times$ 4 forms $\times$ 568 tasks, and printed it in the synthesis's own format by E106's own measure without recomputing anything above (App. K, Table 17): $\widetilde{\kappa}$=0.384, reversal rate 0.400, payoff over inheriting $+0.027$, payoff against a fixed default -0.003. It is a high-κ design where probing pays over inheriting and not over a fixed default, the sign pattern of `bfcl_grid`, `hbench_fixed` and `live_strict_strong`; one point does not move a rank correlation over seven, and we do not add it to the locked table.

Two things not to over-read. On the locked fourteen and on the valid seven neither interval excludes zero, so we have not established that *either* predictor works — fourteen designs is not enough, exactly as the protocol said in advance, and seven overlapping ones are fewer. And although the two interaction-blind summaries sit near zero while the two interaction-aware ones sit higher, their intervals overlap heavily, so "you need to know something about the interaction" is a direction these data point in, not a finding they establish.

V3 is the one that stands on its own. It was locked as: designs with $\widetilde{\kappa} < 0.10$ have a mean saving below 0.02. Exactly one design qualifies — the BFCL grid, where every model is handed the same schema, $\widetilde{\kappa}$=0.00 — and its saving is $+\mathbf{0.0004}$ on the whole-task endpoint the locked run scored, and $+0.004$ on the valid per-turn endpoint (bias-corrected 0.00, raw share 0.14). Where the harness gives every model the same interface the interaction is not detected and a probe buys almost nothing. E104 reached the same place from the other direction on the frozen grid: the raw interaction share reads 0.006 (95th percentile 0.051), and measuring the incoming model showed no advantage over keeping the format. One design is one design, but the reading is the same in both, and it is the reading, not the correlation, that the advice rests on. V1 ($\rho > 0$) and V4 (the sign survives leave-one-design-out, ρ between $+0.264$ and $+0.500$) are confirmed as locked, on point estimates, and we quote them as directions rather than as relationships.

AB.4 Is what survives full specification actually negligible? We declared a margin and it failed its own guard (E73)

The paper used to say that once both sides of the call are specified, what is left is "small enough to ignore", with a 0.01 figure attached. A reviewer pointed out the obvious problem: that figure was picked after looking at the number it was describing. It was a description wearing the clothes of a test.

So we declared a margin first, in the unit a practitioner acts on:

> The residual counts as negligible if the 95% interval on **what per-model measurement buys** sits entirely below $\mathbf{0.02}$ held-out success — two points of task success, which does not repay a 40-run probe on any deployment we would recommend one for.

And with it a guard, because a margin that everything passes measures nothing: **the margin must fail at the unconstrained rung.**

rung	gain	95% interval	inside the margin
ours, fully specified	-0.0003	$[-0.029,\ 0.010]$	yes
ours, unconstrained	$+0.0771$	$[\ 0.020,\ 0.129]$	no
external, fully specified	-0.0004	$[-0.023,\ 0.033]$	no
external, unconstrained	-0.0056	$[-0.031,\ 0.007]$	**yes**

The guard failed, so the label goes. The last row is the problem. On the external suite the margin also passes where it should have failed, so on that substrate it cannot tell the rungs apart. Our locked fallback said that if the guard fails the framing is dropped everywhere, including where the margin passed, and that is what we have done: "small enough to ignore" is gone from this paper.

Why it failed is not the reason we guessed. We wrote the guard expecting that a failure would mean the margin was too loose. It is not — it fails cleanly at our own unconstrained rung, $[0.020, 0.129]$, which is what a working margin looks like. What actually happened is that on the external suite a probe buys almost nothing at *either* rung. That is consistent with §AB.3, where probing bought 0.0033 externally against 0.060 on our suite. A negligibility test cannot separate two rungs on a substrate where nothing is worth measuring anywhere.

And `external, fully specified` misses the margin only on interval width — upper bound 0.033 on thirteen models and thirty-two tasks — not on size. Its point estimate is -0.0004.

What we say instead. The measurement, with its width, and no label: at the fully specified rung, measuring per model buys -0.0003 on our suite and -0.0004 externally, each interval about two to three points wide. Readers can apply whatever margin their own deployment justifies. We are not supplying one, because ours did not survive its own guard.

AC Can this suite measure frontier models? (E52 — a non-result, reported)

The paper's headline grid is seven open-weight models, so we pre-registered adding four frontier chat models (`gpt-4o-mini`, `claude-haiku-4.5`, `gpt-5.6-luna`, `qwen3-32b`) to the *same* 52 tasks, the *same* four protocol forms and the *same* 400-token budget the headline κ is computed at. 832 cells. **The experiment returns a non-result by its own locked gate, and we report it rather than retuning the suite until it passes.**

What failed, and what did not. The pre-registered instrument check passed cleanly: the exemplar-copying rate is 0.000 in all four forms (the live suite uses concrete worked examples — see App. J for what happens when it does not). Pooled success is 0.6262, comfortably *inside* the $[0.15, 0.85]$ band, so the ceiling we worried about is not the problem. The gate failed on the no-parsed-call rate: 0.1671, just over the locked 0.15.

		no-parsed-call rate by form				
model	success	json	react	xml	pythonic	overall
`gpt-4o-mini`	0.808	0.000	0.000	0.000	0.000	0.000
`claude-haiku-4.5`	0.697	0.250	0.096	0.423	0.096	0.216
`gpt-5.6-luna`	0.524	0.115	0.038	0.038	0.000	0.048
`qwen3-32b`	0.476	0.327	0.173	0.654	0.462	0.404

Diagnosed before concluding. This project has twice mistaken its own instrument for a result (App. J), so the failure was chased down before it was interpreted. It is *not* a parser fault: the XML parser handles markdown fences, surrounding prose, single-quoted attributes and extra whitespace correctly on eight realistic variants. It is *not* truncation at 400 tokens, the failure mode that made reasoning models unmeasurable in App. G: the failing trajectories emit a well-formed `<answer>` tag and a final number, so they are complete rather than cut off. It is real model behaviour. On a suite of *fictional* entities whose values exist only behind the `db` tool, two of the four models skip the tool and fabricate an answer — `claude-haiku-4.5` returns "216250.0" on task `avg2` having made zero tool calls — and they do it at a strongly form-dependent rate (XML 0.279 against ReAct 0.077 pooled).

What we conclude, which is about our instrument. Per the protocol no κ is reported and the suite is not retuned. Under fallback (d) the two models failing the per-model parse gate would be excluded, which leaves two — too small a design to read an interaction from. The honest statement is that **this suite cannot measure these models**: it was built so that open-weight models must use the tool, and a model confident enough to answer fictional questions without looking anything up is outside what it can score. That bounds our instrument, not the truth about frontier models, and it is the reason the frontier evidence in this paper rests on the seven reasoning models of App. G, where the same budget and gates *are* met.

One practical note, recorded before collection. The batch route for `claude-haiku-4.5` ($0.50/M input rather than $1.00) returns `404: This model is only available through the Batch API`. An agent loop is synchronous — each step depends on the previous tool result — so a 24-hour batch endpoint cannot serve it. Anyone costing a study like this at batch prices should know that the discount is unavailable for agentic evaluation.

AD Extended comparison with the closest 2026 designs

Five 2026 preprints sit closest to this work, and each shares part of our setup. Two of them make our starting point unoriginal: that the best harness depends on the model, that model rankings reverse across harnesses, and that harness settings must be reported. We say so first. We then compare the six works on five questions (Table 48):

- *Interaction measurement.* Is the model–harness interaction reported apart from the two main effects?
- *Oracle opportunity.* Is the gain of a per-model choice over the best shared harness measured?
- *Finite-budget selection.* Is a rule that chooses from a few runs scored on held-out tasks?
- *Strong defaults.* Is that rule compared with a strong shared default, not only with keeping the old harness?
- *Transfer.* Are frozen choices tested on new models, new task templates or new task families?

Our contribution lies in the last four. The interaction measurement is a supporting tool: it describes sensitivity, and Proposition 1 shows why it does not decide whether adapting pays.

[†]Liu (2026) measures interaction *between scaffold components*, not between model and harness, so the column does not apply.

Interaction measurement: the phenomenon is established. Zhang et al. (2026) is the closest work and the most direct threat to our novelty. It is a position paper defending the "Binding Constraint Thesis": for long-horizon tasks across models of comparable frontier capability, harness con-

Table 48: The closest 2026 designs on the five questions. "Interaction": the model–harness interaction reported apart from the two main effects. "Opportunity": the gain of a per-model choice over the best shared harness. "Selection": a finite-budget rule scored on held-out tasks. "Defaults": that rule compared with a strong shared default. "Transfer": frozen choices tested on new models, templates or task families. Counts are as stated in each paper.

work	design	interaction	opportunity	selection	defaults	transfer
Zhang et al. (2026)	position paper + one decomp.	yes	no	no	no	no
Yao et al. (2026)	106 tasks, 5,194 traj.	no	no	no	no	no
Cho (2026)	6 models $\times 3$ harn., 24 tasks	no	no	no	no	no
Liu (2026)	2^5 components, 2 scales	n/a[†]	no	no	no	no
Ben Sghaier et al. (2026)	35 releases, model fixed	no	no	no	no	no
this paper	9 crossed designs, 3 transfer tests	yes	yes	yes	yes	yes

figuration governs more performance variance than model choice, so current protocols misattribute harness gains to model improvements. It supports this with a control-theoretic formalisation, a survey of published benchmarks and deployments, and a controlled variance decomposition showing harness-induced variance above model-induced variance, with model ranking reversals; it proposes a disclosure standard and a variance-decomposition protocol. Yao et al. (2026) runs 106 sandboxed, oracle-checkable tasks to 5,194 trajectories across model backends and harness configurations, and concludes that "agent capability should be reported at the model-harness configuration level rather than attributed to the base model alone". Cho (2026) crosses six models over four capability tiers and three harness conditions and finds harness sensitivity non-monotone in capability; our strong-model design reaches a compatible result from a different design (App. I), so we report it as corroboration, not discovery. These works show *that* the pairing matters, on larger and more realistic suites than ours. Our share κ measures *how much* of the variation in model gaps moves with the format within one design (§3.1). We use it only to describe sensitivity.

Oracle opportunity and finite-budget selection: the question none of the five asks. None of the five measures what choosing a harness per model would gain over the best shared harness, and none scores a selection rule on held-out tasks. Liu (2026) comes nearest in spirit. Its full 2^5 factorial over five scaffold components (planning, tools, memory, self-reflection, retrieval) on HotpotQA and GSM8K at two model scales finds that fully equipped agents can underperform subsets and that the best combination depends on the task and the scale: there is no configuration to default to. Its interaction is between scaffold parts; ours is between the model and the call format. We ask how much a per-model choice would gain (the oracle gain and its decision margins, §3) and how much of that gain a rule keeps when it chooses from a small budget (the selection error, §5.2).

Strong defaults. Ben Sghaier et al. (2026) fixes the model and follows 35 releases of a coding-agent harness on 50 SWE-bench Verified tasks, attributing quality changes to the harness alone. It is the mirror of our intervention: we fix the harness family and change the model. Both directions show that an upgrade can move behaviour whichever side changed. What our design adds is the comparison a team faces at a model swap: a per-model choice against a strong shared default. We keep three defaults apart (Table 1): the hindsight best shared format, a common format, and the pooled-history rule. On WorkBench, a strong default leaves almost nothing to find (§5.2); on three variants of our suite, a ten-task probe beats all three.

Transfer. None of the five freezes format choices and tests them on new models, new task templates or new task families. We test all three (§6), and the result bounds our own positive case: the gains seen on our suite carried over in part to new models and to new templates with the same tools, and a frozen probe showed no detected advantage over the pooled-history rule on new BFCL task families.

What we do not claim. We do not claim the phenomenon; the works above establish it. We do not claim a new principle: choosing a harness per model is algorithm selection (§2). Our claim is the link from the interaction to the oracle gain through decision margins, and measurements of when strong defaults and small probes succeed or fail.

Current results and the archival record. Everything above rests on current results. Earlier versions of this work framed the contribution as an interaction estimator: a finite-design share with a characterized null floor, a calibrated permutation test and bias-corrected intervals. Those tools, their external tests and their history are kept in the archival part of the appendix (Apps. AF, AG and AX), together with a pre-registered ledger of all 215 locked predictions, including the 67 that were refuted (App. AW). The history includes a failed external validity gate, a parser defect of ours, and a schema trend that did not replicate on new models. These records support the measurements; they are not the claim.

Archival record

AE About the archival record

The sections from here to the end record earlier framings of this project and the history of its instruments; they do not define the paper's current contribution. The current contribution is stated in Section 1: the link from the interaction to the oracle gain through decision margins, and a held-out test of selection rules against shared defaults. The record keeps five things, so a reader can check them: an external grid whose gate failed (App. AG); the five-component drift instrument the project began with (called *the trajectory decomposition* below), and the studies that used it, including the regression-triage study (Apps. AH–AV); the two estimands and paired drift (App. AF); the pre-registration ledger (App. AW); and the corrections to our own instruments (App. AX). Where a record limits a current claim, the current appendix that states the claim says so.

The four primary results of the earlier framing. Before the current argument, the paper named four primary results of its prospective studies. Table 49 keeps them, with the response, the population, the estimator, the source of uncertainty and the claim we do not make. They are records: the current argument is stated in Section 1, and its evidence is in Apps. B–M.

Table 49: The four primary results as the earlier framing named them. Only one of them is a paired quantity, paired drift, and it is not κ. Every row's "not" clause is a claim we do *not* make.

result	response	population	estimator (correction)	uncertainty	what it says, and does not
headline interaction	per-task success $(0/1)$	these 7 models $\times$ 4 formats $\times$ 52 tasks, one collection	plug-in interaction share κ of the finite design	task bootstrap; κ=0.501, $[0.409,\ 0.603]$; 0.421 on a repeat 21 days later	a share of *success* variance in this design. **Not** a decomposition of paired drift
frozen external design	graded outcome in $[0, 1]$	5 external models $\times$ 3 rungs, hashed first	the same script, byte-identical (bias-corrected $\widetilde{\kappa}$)	$\widetilde{\kappa}$ 0 and 0.003 on rungs 1–2; rung 3 failed its gate. Largest interaction cell 0.029 and 0.066 outcome points (95th percentiles 0.073, 0.113); percentiles, not bounds	the ladder did not replicate and the interaction reads near zero. **Not** "exactly zero"
swap decision	held-out regret	42 swaps of 7 models (suite); 30 swaps of 6 models, 568 action-required tasks of 690 (WorkBench)	regret of a fixed rule (none); $\widehat{G}$ beside it	suite: tasks re-split, models re-sampled; 1 of 6 gaps clears zero, touching it when the task pool is bootstrapped. WorkBench: template bootstrap; default $-$ probe -0.011, $[-0.019,\ 0.004]$, measured coverage 0.58–0.74 (App. K.12)	probing has lower regret than keeping the format on both; on WorkBench so does the pooled-history default, and the probe shows no detected advantage over it. **Not** a bound: coverage is below nominal
paired drift (companion)	outcome flip	SWE-bench `gpt-4`→`gpt-4o`, 500 instances	flip rate (none)	Wilson interval; 0.168	this, not κ, is the paired quantity. **Not** a variance share

AF Two estimands, and which component estimates which

An earlier version of this paper opened its method section with a paired expectation and then computed things that are not averages of a paired distance, and called them two readings of one equation. A reviewer said so. This appendix names the two quantities separately, gives them in full and maps every metric we report to one of them:

$$\Delta_{\text{pair}} = \mathbb{E}_{t,c}\big[\, d\big(A(\mathcal{H}, M_1, t, c), A(\mathcal{H}, M_0, t, c)\big)\big], \qquad \Delta_{\text{dist}} = D\big(P^{(1)}_{\mathcal{H}},\, P^{(0)}_{\mathcal{H}}\big), \tag{7}$$

where $P^{(m)}_{\mathcal{H}}$ is the law of $A(\mathcal{H}, M_m, t, c)$ over $t \sim \mathcal{D}$. For a bounded outcome Jensen orders them, $\mathbb{E}_t\big|Y^{(1)}_t - Y^{(0)}_t\big| \geq \big|\mathbb{E}_t Y^{(1)} - \mathbb{E}_t Y^{(0)}\big|$, with equality only if every change goes the same way. Silent drift is exactly this gap.

Paired drift, the companion question (moved here from the main text). The current main text does not use paired drift. Its estimand is this. Swap $M_0 \to M_1$ and hold $(\mathcal{H}, t, c)$ fixed. The two runs

τ_0 and τ_1 are paired, and their difference is caused by the substitution and its interaction with this harness, not by the task or the harness (an interchange intervention at the base model (Geiger et al., 2021)). The estimand is a paired distance

$$\Delta(\mathcal{H}, M_0 \to M_1) \;=\; \mathbb{E}_{t\sim\mathcal{D},\,c}\big[\, d\big(A(\mathcal{H}, M_1, t, c),\; A(\mathcal{H}, M_0, t, c)\big)\,\big], \tag{8}$$

for a divergence $d \in [0, 1]$; the outcome flip rate is its simplest instance. It is well defined under (A1) *harness invariance*, $\mathcal{H}$ byte-identical across the two runs; (A2) *context fixity*, c shared; and (A3) *no interference*, runs sharing no mutable state. The pairing holds exactly in the synthetic studies and only approximately in the temperature-0 live studies, since hosted greedy decoding is not bit-reproducible; we measure that floor (App. U). Paired drift says whether this swap changed what the agent did under this harness. It says nothing about whether the change depends on the harness.

Δ_{pair}: how much the same task changes. This is Eq. (8) exactly as written,

$$\Delta_{\text{pair}}(\mathcal{H}, M_0 \to M_1) \;=\; \mathbb{E}_{t\sim\mathcal{D},\,c}\big[\, d\big(A(\mathcal{H}, M_1, t, c),\; A(\mathcal{H}, M_0, t, c)\big)\,\big].$$

It is an average over *pairs*. Pairing is what makes it meaningful: the same task, the same harness, the same budget, one thing swapped. It answers "if I change the model, how much does a given item move?"

Δ_{dist}: how far the two behaviour profiles sit apart. Let $P_{\mathcal{H}}^{(m)}$ be the law of $A(\mathcal{H}, M_m, t, c)$ with $t \sim \mathcal{D}$. Then

$$\Delta_{\text{dist}}(\mathcal{H}, M_0 \to M_1) \;=\; D\big(P_{\mathcal{H}}^{(1)},\; P_{\mathcal{H}}^{(0)}\big)$$

for a divergence D between distributions. Nothing here is paired. It answers "do the two models behave the same *on average over the workload*?"

Why the difference is the paper's subject, not a technicality. For a bounded outcome,

$$\underbrace{\mathbb{E}_t\big|Y_t^{(1)} - Y_t^{(0)}\big|}_{\text{paired: item-level churn}} \;\geq\; \underbrace{\big|\mathbb{E}_t Y^{(1)} - \mathbb{E}_t Y^{(0)}\big|}_{\text{distributional: score gap}},$$

with equality only if every change goes the same way. Silent drift *is* that gap. A paper that reported only the right-hand side would call a swap a no-op while a sixth of the items changed answer (App. AI). So the two quantities are not competing definitions; the distance between them is the finding.

Which component estimates which.

metric	estimates	one pair?	in D?
outcome, aggregate score gap $\lvert\overline{Y^{(1)}} - \overline{Y^{(0)}}\rvert$	Δ_{dist}	no	**yes**
trajectory, JS between action-type distributions	Δ_{dist}	no	**yes**
tool selection, JS between tool distributions	Δ_{dist}	no	**yes**
recovery, $\lvert r^{(1)} - r^{(0)}\rvert$ over pooled error counts	Δ_{dist}	no	**yes**
tool arguments, mean NED over matched calls	Δ_{pair}	yes	**yes**
outcome, per-task flip rate $\frac{1}{n}\sum_i \mathbb{1}[\text{succ}_i^{(1)} \neq \text{succ}_i^{(0)}]$	Δ_{pair}	yes	no
action-sequence edit distance (§AR, E56 below)	Δ_{pair}	yes	no
per-pair recovery indicator	Δ_{pair}	yes	no

The last column is the one that was missing. **Five** metrics are in the composite D, and *one* of those five is paired. The other three paired metrics are counterparts we report beside their distributional partners; they are not summed into D. So D is four-fifths distributional: a summary, not an estimator of either estimand, which is why κ and not D is the primary output.

Which response does each κ decompose? Every one names it. The headline (App. T) uses **task success**. E56 below uses a **paired action-sequence edit distance**. The per-facet figures in App. D each use **one component**. No κ anywhere in this paper is computed on D.

And κ is neither. κ is a variance decomposition of a *chosen scalar response* across a model $\times$ harness design (§3.1). The headline uses task success. It does not decompose Δ_{pair} or Δ_{dist}. The rest of this appendix asks the one question that connects them: computed on a response that *is* a paired trajectory distance, is the interaction still there?

AF.1 Is the interaction visible on a trajectory-level response? (E56, exploratory)

An external reviewer objected that the earlier framing (App. AR) defined drift as a divergence over ordered trajectory pairs and validated it on the simulator (every component at $r \geq 0.895$, drift index r=0.97), while the headline κ is computed on a scalar *success* indicator that the divergence never touches. The objection is fair. This appendix asks the narrow question it licenses — is the interaction visible at all on a trajectory-level response? — and is careful about what the answer does *not* establish.

What we do *not* claim, and a correction. Our first attempt at this evaluated the five-component estimator on one pair at a time. That was wrong, and §AR says so in its own words: the trajectory and tool-selection components are aggregate over pairs, "a Jensen–Shannon divergence between action-type distributions being undefined for a single pair". The per-pair values that produced were not the declared estimator, and calling the result "κ on the formal estimand" was an overclaim; a second reviewer caught it. **Nothing here validates the aggregate five-component estimator on real data.**

The response, chosen to be well defined for a single pair. For each of the $\binom{7}{2}$=21 model pairs, each protocol form and each task, we take the normalized edit distance between the two models' *action-type sequences* under a byte-identical harness. This is a genuine trajectory-level divergence, it is defined pairwise, and it is the quantity §AR names for localization. The design is (model-pair $\times$ harness $\times$ task).

Result. κ=0.385 (bias-corrected $\mathbf{0.364}$), against a null mean of 0.005; p=0.001, computed plus-one over 1000 draws so it is bounded by the number of draws rather than reported as zero. Variance shares are S_{pair}=0.155, $S_{\mathcal{H}}$=0.013, $S_{\text{pair}\times\mathcal{H}}$=0.097.

The null took three attempts, and the first two were wrong in opposite directions. We report all three, because the same p=0.001 came out of a null we now consider invalid, and only an explicit calibration check told the versions apart. *(i)* Permuting each model's harness label independently. Under this scheme a pair's two endpoints agree on harness only $1/4$ of the time, so 75% of null cells compared one model under one harness against the other under a *different* harness, while every observed cell is a same-harness comparison — the null calibrated a different quantity than the statistic. It gave p=0.001: the right answer for the wrong reason. *(ii)* Resampling residuals in whole-pair blocks on the pair-level array, to respect each pair's internal dependence. This is not a null at all. Moving a pair's entire (harness $\times$ task) residual slab to another pair *relocates* its interaction pattern instead of removing it, so the interaction variance survives resampling; the null mean was 0.342 against an observed 0.364, and the false-positive rate was 0.000 at a nominal 0.05. A test that never rejects is not conservative, it is uninformative. *(iii)* What we report: pair and harness main effects held at their observed values, residuals redrawn *independently per cell*. The interaction is genuinely removed and every null cell stays a same-harness comparison. We accepted this version on its operating characteristics rather than its p-value: false-positive rate $\mathbf{0.050}$ against a nominal 0.05, and power $\mathbf{1.00}$ against a synthetic injected interaction of κ=0.477 — the same synthetic case on which version (ii) returned p=0.302. Both figures are emitted by the producing script on every run.

Two honest qualifications. First, the interaction is *smaller* on this response than on task success: 0.364 against 0.487. Behaviour diverges under protocol form, but less sharply than outcomes do. Second, the inference over endpoints is weak. A leave-one-model-out jackknife gives a standard error of $\mathbf{0.26}$ with values ranging 0.21–0.60 over the seven drops — and note that the reported null, destroying interaction cell by cell, does not itself model the dependence between pairs sharing an endpoint; the jackknife carries that, not the null — comparable to the estimate itself, and comparable to the 0.22 the headline carries on the same axis (App. E). We use a jackknife rather than a bootstrap

because resampling the seven endpoints *with* replacement duplicates models, and a duplicated model contributes a self-pair whose divergence is 0 by construction; that produced an interval which did not contain its own point estimate, and we discarded it rather than reporting it.

What this establishes. Narrowly: the headline is not purely an artifact of collapsing behaviour onto a success bit, because a trajectory-level pairwise divergence also carries a model×harness interaction that a same-harness, power-checked null rejects. It does not establish that the aggregate estimator is validated on real data, and it does not license a per-swap reading. The analysis is **exploratory**: no threshold was locked before it ran.

AG DOES κ REPLICATE ON A SUITE WE DID NOT BUILD? (E57)

> **This appendix is superseded.** Everything below was measured through the JSON parser that rejected valid tool calls carrying JSON payloads (App. AX, C8), so its `json` column is contaminated and the four-form κ computed over it does not stand. **We are not citing κ=0.553 as a result anywhere in this paper**, and the consistency checker fails the build if it appears outside a passage that says it is retired.
>
> What replaces it: E67 re-collected `json` and `react` on the same suite behind the repaired parser and finds the interaction present but much smaller, κ=0.146 (p=0.037), and E69 re-collects `xml` and `pythonic` so a four-form external grid exists that was gathered entirely behind the fix (App. Y). This section is kept because how the number was produced, and how it failed, is checkable; the number itself is not evidence.

Every κ reported so far is measured on a suite we built. That is the oldest unaddressed weakness in this paper, it cannot be fixed by editing, and App. AD names the experiment that would fix it. This is that experiment.

Substrate: not ours. We take the tasks and the scorer from Harness-Bench (Yao et al., 2026), at repository commit `1025086`. Its 106 tasks are sandboxed, oracle-checkable, and span eight workflow categories; we wrote none of them. Tasks are loaded through Harness-Bench's own loader, the workspace is staged and the prompt rendered by verbatim copies of its fixture and prompt helpers, and the response is its own `oracle_grade.score_workspace`'s `outcome_score`, reached through its own oracle runner. We wrote no scoring code. What is ours is the harness factor, a four-tool agent loop, and the estimator.

Design. 7 models × 4 protocol forms × 32 tasks = 896 cells. The model axis is the same seven as the headline grid and the harness factor is the same four protocol forms, which is what makes the two κs comparable at all. Tools are byte-identical across the forms (`read_file`, `write_file`, `list_dir`, `run_python`); only the syntax in which a call is requested differs.

Task selection is mechanical and was fixed before collection. Five eligibility rules were applied to all 106 tasks with no inspection of difficulty: exactly one prompt file; a non-empty fixtures directory; not one of the two tasks Harness-Bench itself grades with an LLM judge (`outcome_llm_weight`=0.9); no URL in the prompt; and not tagged `browser`, since we provide no browser tool. That leaves 90 eligible. From those, a deterministic class-stratified round robin over the eight workflow categories, lowest task id first, selects K=32 with 3–5 tasks per category. The list is pinned in the producing script. A failed validity gate does not license re-selection.

Three fairness decisions. Each would have manufactured an interaction if we had got it wrong, so each is stated and tested. *(i)* `write_file` needs multi-line content, and a form that cannot express a newline is crippled. If the crippling came from our parser rather than from the form we would be measuring the parser, so each form gets its idiomatic carrier: a JSON string value, a backslash-escaped string for ReAct and Pythonic, and a `<content>` *child element* for XML — which is what XML is for. A unit test asserts that all four parse a multi-line payload to the identical (tool, arguments). *(ii)* The worked example in every form calls a tool that does not exist in the tool set, so copying it verbatim is equally wrong under all four. App. J is why: a copyable placeholder was reproduced in 20.4% of XML replies there, at a rate that differed by form. *(iii)* The parser executes nothing. Arguments are read with an escape-aware pattern and an explicit unescape. Model-authored code runs only inside `run_python`, deliberately, jailed to the task workspace with a timeout.

Disclosed deviations, and why they cannot fake an interaction. Harness-Bench evaluates six full agent frameworks against eight frontier backends. We run four tools and seven non-frontier models, with a 10-step cap. Absolute `outcome_score` is therefore *lower* than its published numbers and our results are not comparable to its leaderboard. That is a **main effect**: a uniform capability reduction lowers every cell of every protocol form, and κ is a ratio of interaction variance to model-plus-interaction variance, so it cannot be manufactured by making every form worse. We also provide no browser and no network tool, which is why the eligibility rules exclude those tasks.

Inference carries every correction this paper has had to make to its own. Significance rests on a mean-matched no-interaction null, never on a bootstrap interval excluding zero — that test is unfalsifiable for the raw ratio, whose null floor is non-zero (App. J). The null redraws residuals independently per cell, because resampling them in blocks relocates the interaction instead of removing it and has no power at all (App. AF). Both the null's false-positive rate *and* its power are computed and reported, and the pre-registered rule is to treat the null as unusable, whatever p-value it produces, if its false-positive rate is far from nominal or its power is near zero. Intervals come from a task bootstrap and a leave-one-model-out jackknife; a with-replacement model bootstrap is invalid here because it duplicates models.

The analyser was validated before it saw real data. On a synthetic grid built with additive main effects and no interaction it returns $\kappa{=}0.000$ at $p{=}1.000$ and refutes the presence prediction; on one with an interaction injected at $\mathrm{sd}\,0.22$ it returns $\kappa{=}0.447$ at $p{=}0.00025$ with power 1.00 and confirms it. The single-provenance loader refuses to resolve a conflict by file order, verified by planting a duplicate cell with a different score.

Result: the validity gate FAILED, and we report the non-result. All 896 cells collected. Pooled `outcome_score` is $\mathbf{0.240}$, comfortably inside the pre-registered band $[0.10, 0.90]$ — the suite is neither trivial nor impossible for this harness. But the pooled *no-parsed-call* rate is $\mathbf{0.166}$, above the pre-registered threshold of 0.15 by 0.016. P1 therefore **fails**, and the protocol's locked fallback (a) applies: we report the measured difficulty and parse rate as a non-result, and we do *not* re-select tasks, change the tool set, raise the step cap, or drop a model to rescue it. **No κ is reported for this suite.**

Why it failed, which is the informative part. The failure is almost entirely one model of seven. Per-model no-parsed-call rates are `llama-3.3-70b` 0.000, `mistral-small-24b` 0.008, `llama-3.1-8b` 0.031, `qwen3-14b` 0.062, `gpt-oss-120b` 0.102, `phi-4` 0.102 — and `gpt-oss-20b` $\mathbf{0.859}$. In 86% of its cells it never produced a parseable call, because on these prompts the provider returns *no text at all*; we probed it directly and it emits a correct call on a short prompt at the same token budget, and returns nothing on the real task prompt at both 600 and 1600 tokens, so this is not a budget artefact of ours. Separating unparsed steps by cause across the whole grid, $1{,}749$ were empty provider replies and 998 were replies we refused. The empty ones are almost uniform across the four forms ($425/424/465/435$), i.e. a model-and-provider behaviour rather than a protocol-form effect; the refusals are strongly form-dependent (461 under JSON against 76 under ReAct).

What this does and does not establish. It does *not* establish that κ fails to replicate externally: the grid never became clean enough to ask. The external-validity question this appendix set out to answer therefore **remains open**, and we say so rather than reporting a number from a grid where one model's rows are 86% non-responses. What it does establish is a concrete bound on our instrument: a four-tool harness driven by this seven-model pool cannot produce a parse-clean grid on a realistic external suite, because one member of the pool does not reliably respond to these prompts at all. Answering the original question needs a model pool where every member can emit a parseable call on Harness-Bench prompts, or Harness-Bench's own six harnesses driving its own eight frontier backends. That is a collection problem, not an analysis problem, and it is the next experiment rather than this one.

The gate earned its place. Without it we would have reported a κ computed over a model axis containing a row of near-total non-responses, and the number would have looked publishable. Locking the gate in advance, at a threshold set by earlier experiments rather than by this grid, is what

turned a tempting result into a disclosed non-result. We note that the margin is small — 0.166 against 0.15 — precisely so a reader can see that we did not have to fail it, and did not move it.

E58: a screened pool, and the gate passes. E57's gate failed on one model that returned no text. Because its fallback forbids swapping that model, we pre-registered a *separate* experiment, E58, whose seventh pool member is chosen by a responsiveness screen — 8 cells, admit the first candidate with a 0.00 no-parsed-call rate, looking at nothing but whether a parseable call came back. The candidate list was amended once, before the screen ran and before any collection, because all six originally declared candidates had become unreachable (four registered against free endpoints since withdrawn, one model retired, one over quota); the amended list is every reachable non-reasoning registry model in *alphabetical* order, a mechanical order chosen so it cannot encode a preference. `claude-haiku-4.5` screened clean and was admitted. The pool is therefore mixed-tier, which we disclosed in advance as a change to the model *main* effect. E58 reuses E57's cells for the six retained models and collects only the seventh, so 768 of its 896 cells *are* E57 cells.

The gate now passes: pooled `outcome_score` 0.259, no-parsed-call rate $\mathbf{0.046}$ against the 0.15 threshold. Bias-corrected κ is $\mathbf{0.553}$ — *retired*, see the banner above, because the parser defect contaminates the `json` column — p=0.003 against the mean-matched no-interaction null, whose false-positive rate is 0.04 against a nominal 0.05 and whose power is 1.00. Task-bootstrap CI $[0.316, 0.794]$. **The interaction survives contact with tasks and a grader we did not write.**

But the *favourable* part of that result is an artifact, and we report it as one. The protocol's fallback (c) requires checking a surprise in our favour before believing it, and the check fails. P3 predicted κ would be *smaller* externally; it came out larger (the retired 0.553 against 0.487), so P3 is refuted — but not because the interaction is stronger. It is weaker:

	S_M	$S_{\mathcal{H}}$	$S_{M\mathcal{H}}$	κ
our suite (E55, b=4)	0.120	0.022	**0.121**	0.487
external (E58, *retired*)	**0.034**	0.070	**0.053**	0.553

The absolute interaction variance is *less than half* ours (0.053 against 0.121). κ is nonetheless higher because the model axis compressed by $3.5\times$: six of the seven models score within 0.024 of each other (0.227–0.252), with only `llama-3.3-70b` apart at 0.393. The leave-one-model-out jackknife makes the fragility explicit — dropping `llama-3.3-70b` sends κ to $\mathbf{1.00}$, because with it removed S_M is almost zero and the ratio is pinned at its ceiling by construction. So the honest statement is: *the interaction replicates externally and is smaller there; κ is larger only because the models are harder to tell apart on this suite.*

A limitation of our own estimand, found this way. κ is a ratio, so it is comparable across substrates only when the model-axis spread is comparable. Ours is $3.5\times$ the external one, and that alone accounts for the direction of the P3 refutation. We therefore report absolute variance components alongside κ wherever we compare across suites, and we note that the compliance ladder (§5) is *not* exposed to this problem: its absolute interaction variance falls $0.117 \to 0.067 \to 0.030 \to 0.008$ while its denominator rises over the first three rungs.

One claim weakens externally. P4 ("no default interface") is technically confirmed — two distinct forms are somebody's optimum — but only just: `xml` is best for *six* of seven models and has the highest pooled score (0.376 against `pythonic`'s 0.174), and the harness main effect (0.070) is the largest of the three terms here. On this suite there very nearly *is* a safe default, which is weaker than what our own suite shows, where every one of the four forms is somebody's optimum in 50% of task bootstraps. We report that as a limit on the "no safe default" claim rather than as support for it.

Is that null actually valid here? We broke our own circularity to find out. A reviewer objected that this null "violates support and task dependence", so its p-value and calibration lack a valid sampling interpretation. The objection is factually right about the data. HarnessBench `outcome_score` has a discrete, *task-specific* support — `001-file` admits only $\{0, 1\}$, `004-meeting-summary` only $\{0, 0.889, 1.0\}$, between 2 and 14 distinct values per task — and

task-level dispersion is comparable to within-task dispersion (sd 0.160 against 0.220). Our null instead adds a continuous residual to an additive fit and clips, which can produce values no task can score, and treats all 896 cells as exchangeable.

The sharper problem is one the objection implies and we state plainly: **our published calibration was circular**. It measured the false-positive rate on data drawn from the null's *own* mechanism, so a nominal rate was guaranteed by construction and told us nothing. We therefore rebuilt the check on data that carries exactly the two properties the null assumes away: additive truth with no interaction, using the observed model, harness and *task* effects, snapped to each task's real achievable support. Over 400 simulations at each of three dispersion levels, the published null's false-positive rate is $\mathbf{0.000}$ at every level — conservative, not inflated — while retaining power $\mathbf{0.99}$ and $\mathbf{1.00}$ against interactions injected at sd 0.15 and 0.30 on the same realistically-supported data. The simulated grids are not artificially clean: their dispersion (0.235–0.276) brackets the real grid's 0.299.

So the structural objection does not turn into an inflated error rate at this design. We do *not* conclude from this that the reported p-value is understated: on additive data the bias-corrected κ is usually truncated to zero, so a zero rejection rate partly reflects that truncation, and three conservative simulated processes do not establish tail validity in general. An earlier version of this paragraph did draw that conclusion and it was wrong.

A test that treats the task as the unit, since that is what the design replicates over. The right answer to the clustering half of the objection is a second test, not an argument. We fit the additive model and multiply each *task's* whole residual slab by an independent random sign. Within a task the correlation structure is kept, and across tasks a systematic model×harness pattern cancels. On the real grid this gives $\mathbf{p{=}0.0005}$.

That test respects clustering but **not** support. We measured it: 47.7% of its pseudo-outcomes land off the value set the task can actually score. An earlier version of this appendix claimed the opposite, and it was wrong. Its false-positive rate on realistic additive data is 0.01, but over 100 simulations, so the 95% interval is $[0.002, 0.055]$ — we report the interval rather than the point, because a rate estimated from one rejection in a hundred does not pin down much.

A support-preserving diagnostic, and why it is only that. To keep every value on the support we also run a within-block permutation: shuffle the *format* labels inside each (model, task) block. Every number stays exactly as observed and inside its own task, so support and clustering are both exact, and it gives $p{=}0.00025$. We report it as a **diagnostic and not as a test of the interaction**, because two things break that reading and both are ours to own. First, permuting format labels destroys the harness main effect as well as the interaction, so the null is the joint "no interaction *and* no harness main effect", and with $S_{\mathcal{H}}{=}0.070$ here a rejection can come from the main effect alone. Second, it permutes independently per (model, task), so one model's JSON column no longer lines up with another's. Cross-model format alignment is exactly what an interaction *is*, so scrambling it changes the estimand rather than nulling it. It tells us the result is not an artifact of values that cannot occur; it does not independently confirm the interaction.

Where that leaves the inference. The claim rests on two things that do test the interaction: the task bootstrap interval $[0.316, 0.794]$, which resamples the unit the design replicates over, and the task-clustered sign-flip ($p{=}0.0005$), which respects that clustering. The iid residual null agrees ($p{=}0.003$) and holds its level on data carrying the real support and task effects, and the within-block permutation says the result is not an artifact of impossible values. We do *not* claim these are three independent confirmations, because only two of them null the interaction specifically. The honest summary is narrower: the finding survives clustering, and it survives being tested on data that can only take values the tasks can actually score.

All numbers come from one frozen configuration — `check_null_validity.py` run with no flags, seed `20260826`, 400 simulations and 400 null draws, recorded inside the artifact itself. The number audit pins the four values quoted above and checks that the artifact's recorded configuration still matches the script defaults, so a run with different flags fails the build rather than silently replacing them.

AH SYNTHETIC EXPERIMENTS: FULL DETAILS

Experiment indexing. We index experiments E1–E99 for cross-reference; En maps to the appendix cited at its first mention in the main text.

Algorithm 1 Trajectory-decomposition estimation

Require: models $\{M_a\}$, harness variants $\{\mathcal{H}_h\}$, tasks $\{t_i\}$, reference model M_0

1: **for** each $(M_a, \mathcal{H}_h, t_i)$: run $A(\mathcal{H}_h, M_a, t_i, c)$; normalize to a TRAJECTORY

2: **for** each pair (M_0, M_a) under each $\mathcal{H}_h$: form paired trajectories and compute components $\widehat{\delta}_k$ (§AR) and drift index D

3: average the evaluation score over tasks in each $(M_a, \mathcal{H}_h)$ cell, giving the cell means $\widehat{P}$ and the shares $S_M, S_{M\mathcal{H}}$ (§3.1). (1) The raw κ is Eq. (1) on the cell means $\widehat{P}$. (2) The corrected $\widetilde{\kappa}$ comes from a two-way fit that treats the n tasks of a cell as replicates. Its residual, each task outcome minus its cell mean, pooled over cells, keeps task-to-task variation. From the model and interaction sums of squares subtract each term's degrees of freedom times this residual mean square (the EMS correction), set negatives to zero, and form the same ratio. (3) The interaction test uses a task-blocked fit: model, format and task effects and their two-way interactions, with the three-way term as the error. Its F compares the model×format mean square with that error, and its p-value comes from a Freedman–Lane permutation null that keeps every other term (App. T)

4: (4) $\widehat{G}$ is Eq. (2) on an observed table: here the full table, with the shared best form h_0 and each model's decision margin (Eq. (4))

5: on held-out splits: recompute h_0 and $\widehat{G}$ on each split's held-out table (not the full table of line 4), then each form-choosing rule π's selection error $R(\pi)$. (5) A rule's achieved gain is the paired held-out difference $U(\pi) - U(h_0) = \widehat{G} - R(\pi)$ of §3.3, averaged over the declared splits. On one partition fixed in advance, also the paired held-out difference between two rules' frozen choices, with an interval checked against a known target (§5.2, App. L)

6: align each pair (Needleman–Wunsch); aggregate per-phase mismatch ⇒ profile $(\phi_1, \ldots, \phi_P)$ (§AR)

7: extract incumbent (M_0) behavioral features + static metadata; fit f to predict high drift (App. AM)

Ensure: drift index, component profile, (S_M, κ), the oracle gap $\widehat{G}$ with the decision margins, each rule's $R(\pi)$ and $U(\pi) - U(h_0)$, one conditional policy comparison, phase profile, brittleness model

Line 3 is the current pipeline. Lines 4 and 5 add the adaptation value: the oracle gap and the margins from the same cell means, then, with h_0 and $\widehat{G}$ taken on each held-out table, each rule's held-out regret and improvement, and one policy comparison on a partition fixed before scoring (§3.2); both are post hoc here. The version we ran first fitted one two-way ANOVA on a replicate tensor and read every share from it; that pipeline gives the 0.501 raw share and the 0.487 corrected share. The 0.563 index came later, from E82's task-blocked variance-component fit (App. T), and is kept there as history. The gap between 0.487 and 0.563 is not the blocking: the index weights the interaction by $b/(b-1)$, and that weight makes 95% of it. The synthetic checks in this appendix (E1, E2, E6) exercise lines 2 and 6 on the composite index D; none of them computes κ on task success.

Specificity (E1). Injecting a single component at magnitude 0.8 and measuring all five yields the matrix in Fig. 11: mean on-diagonal response 0.42 vs. off-diagonal 0.037 ($11.4\times$). Components are axis-specific. (The ratio was $10.4\times$ before the tool-argument alignment fix of App. H: charging reordered repeated calls as maximal argument drift had been leaking signal into that column on every off-diagonal injection, so correcting the estimator *improved* specificity rather than weakening it.)

Coverage vs. single-scalar baselines (E6). Table 50 correlates each injected component with two natural baselines and with the trajectory decomposition. Outcome-only tracks only outcome, a trajectory-distance scalar tracks only trajectory; only the decomposition tracks every facet.

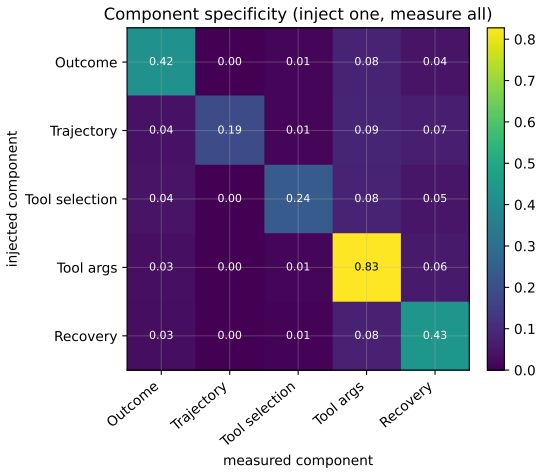

Figure 11: E1 specificity: inject one component (rows), measure all five (columns).

Table 50: Coverage: Pearson correlation between each injected component and each method's estimate.

Injected component	Outcome-only	Trajectory-only	**Decomposition**
Outcome	**0.95**	−0.07	**0.95**
Trajectory	−0.06	**0.77**	**0.99**
Tool selection	−0.03	−0.08	**0.88**
Tool arguments	−0.10	0.03	**0.99**
Recovery	0.07	0.10	**0.79**

Independent estimator and cross-regime recovery (E6). Trajectory drift injected via distribution mixing (JS ground truth) is recovered by a structurally different Needleman–Wunsch edit-distance estimator with $r \in [0.56, 0.87]$ across regimes (vs. $r \approx 0.99$ for the matched JS estimator), so recovery is not a formula tautology. Drift-index recovery across four regimes (varying tool count, obstacle rate, trajectory length): $r = 0.934, 0.960, 0.976, 0.823$.

Trajectory-phase localization (E3). Injecting substitutions confined to one phase (early/middle/late) and reading the alignment-based phase profile, measured drift peaks in the injected phase in 89.6% of $1{,}800$ tasks (100% for early; Fig. 12).

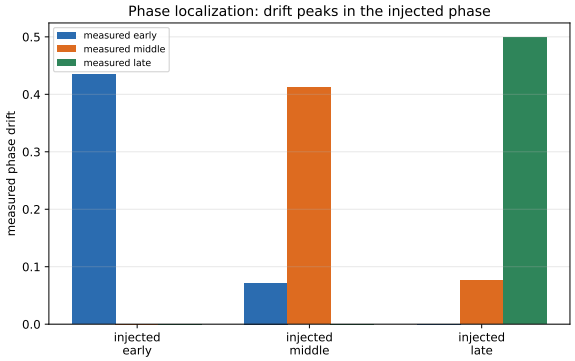

Figure 12: E3: measured per-phase drift grouped by the injected phase (localization accuracy 0.896).

Brittleness prediction (E4). Over 600 cells whose drift is driven additively by harness/task complexity and a declared model gap, a gradient-boosted classifier predicts high-drift cells with 5-fold

ROC-AUC 0.830 from features available before running the new model; using only incumbent-model behavioral features (no declared gap) still gives AUC 0.632 (Fig. 13). Both fell — from 0.855 and 0.671 — when the tool-argument estimator was corrected (App. H). The old estimator scored reordered repeated calls as maximal argument drift, which made the drift label partly a function of call ordering and therefore easier to predict than the drift itself is. We report the lower, correct figures.

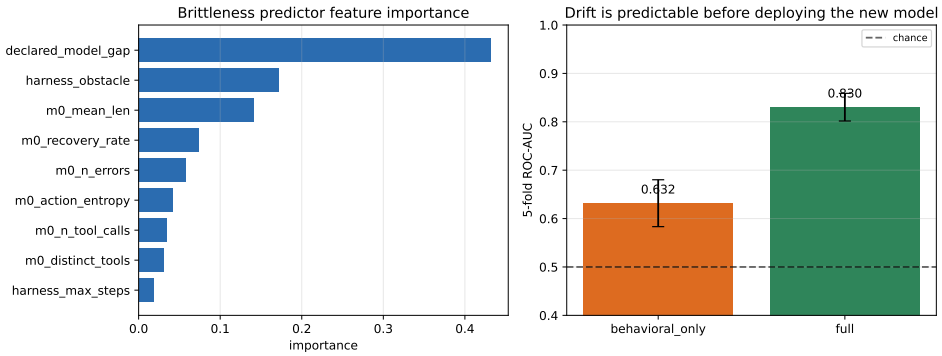

Figure 13: E4: (left) feature importance; (right) ROC-AUC for incumbent-behavior-only vs. full features.

Ground-truth recovery: table and interaction figures (E1, E2). Table 51 gives the per-component recovery; the figures below show it graphically.

Table 51: E1 ground-truth recovery across 240 scenarios: measured component drift vs. injected population ground truth. The drift index's calibration slope 0.99 is the one the main text quotes. Calibration slopes are near-ideal except for trajectory (0.78), whose drift is therefore compressed at large magnitudes.

	Outcome	Trajectory	Tool-select	Tool-args	Recovery	**Drift index**
Pearson r	0.957	0.994	0.937	0.999	0.895	**0.966**
MAE	0.035	0.024	0.024	0.007	0.051	**0.015**
Calib. slope	0.98	0.78	1.02	1.00	0.96	**0.99**

Three checks that the recovery is not built in. Three checks that the recovery is not built in. A score-delta baseline is *uncorrelated* with true process drift ($r{=}-0.02$): over the 13 silent-drift scenarios it reads 0.03 where the drift index reads 0.25. An LLM-as-judge similarity baseline tracks aggregate drift ($r{=}0.58$) but cannot attribute it to a component or the interaction. An *independent* edit-distance estimator sharing no code with ours recovers injected trajectory drift at $r \in [0.56, 0.87]$, with drift-index recovery holding across four regimes ($r{=}0.82$–0.98). Finite samples dilute absolute shares, so we read the ordering, not the level. Apps. AH, S.

AI SWE-BENCH VERIFIED (E7): FULL DETAILS

To test the trajectory decomposition on a recognized benchmark independent of our harness, we ingest public logs from the SWE-bench/experiments repository (Jimenez et al., 2024): per-instance outcomes for 11 leaderboard submissions, treated as (model, scaffold) cells over the full 500-instance Verified set (instances absent from a submission's lists are evaluated-but-unresolved, so our resolve rates match the official denominators). This is a real paired counterfactual at scale. We analyze outcomes only; the repository ships no trajectories for these submissions.

Score parity hides large behavioral drift. Under the fixed SWE-agent scaffold, resolve rates rise across model generations (`claude-3-opus` 0.158 < `gpt-4` 0.224 < `gpt-4o` 0.232 < `claude-3.5-sonnet` 0.336 < `claude-3.7-sonnet` 0.624). But the aggregate gap badly

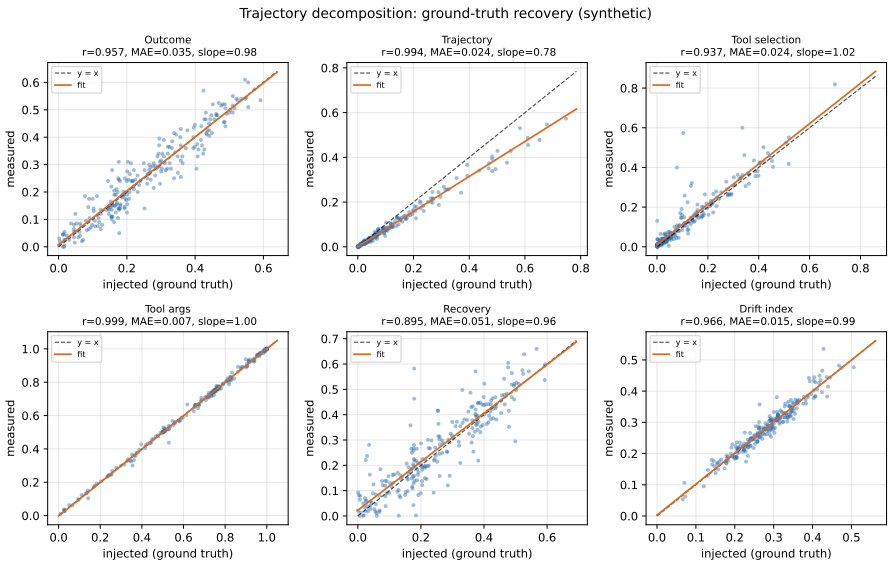

Figure 14: E1: measured drift vs. injected ground truth per component and for the drift index. Points lie on $y{=}x$; r, MAE, and slope annotated.

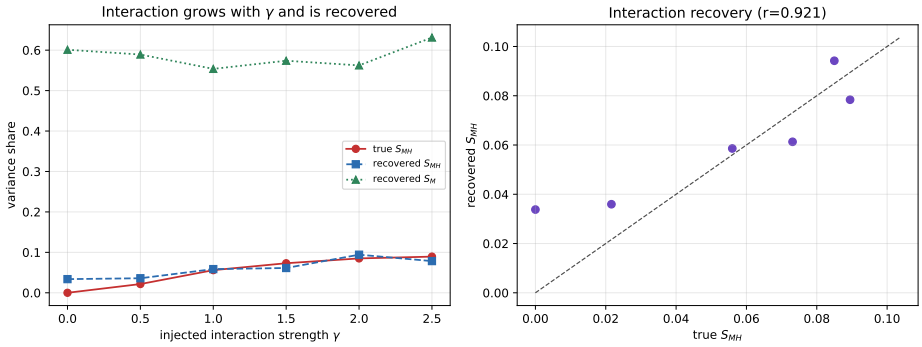

Figure 15: E2: recovered interaction share $S_{M\mathcal{H}}$ grows with injected strength γ and matches ground truth ($r{=}0.921$; it was 0.974 before the tool-argument correction of App. H).

understates the behavioral change: the per-instance *flip rate* (tasks whose outcome changes) is 0.15–0.48 across pairs and always exceeds the aggregate delta. Most strikingly, `gpt-4` → `gpt-4o` moves the resolve rate by only 0.008 (bootstrap 95% CI $[0.00, 0.04]$) yet **flips** 16.8% **of instances** (CI $[0.14, 0.20]$), with non-overlapping uncertainty (App. S). We do not report this as a $\sim$21$\times$ ratio: the flip rate bounds the score gap by construction and the denominator is near zero. Calibrated instead against the independence null $p(1{-}q) + q(1{-}p){=}0.352$, the observed 0.168 ($\phi{=}0.52$) places the swap 48% of the way from identical to statistical independence (E31). A practitioner judging this swap by its benchmark number would see a no-op; the agent solves a substantially different sixth of the benchmark. This is the silent drift of App. AH, on real frontier models and a standard benchmark.

A real interaction spectrum. For models appearing under multiple scaffolds we estimate the interaction directly (Table 52). The {`gpt-4o`, `claude-3.5-sonnet`} $\times$ {SWE-agent, Agentless} grid is nearly additive ($\kappa{=}0.005$): both models gain $\sim$0.16–0.17 from Agentless, so the substitution generalizes across these scaffolds. The {`claude-3.5-sonnet`, `claude-3.7-sonnet`} $\times$ {SWE-agent, tools} grid shows a larger, scaffold-dependent effect ($\kappa{=}0.103$): the 3.5→3.7 upgrade adds $+0.29$ resolve under SWE-agent but only $+0.14$ under tools. `claude-3.5-sonnet` alone swings from 0.336 (SWE-agent) to 0.530 (OpenHands), a strong harness main effect. Next

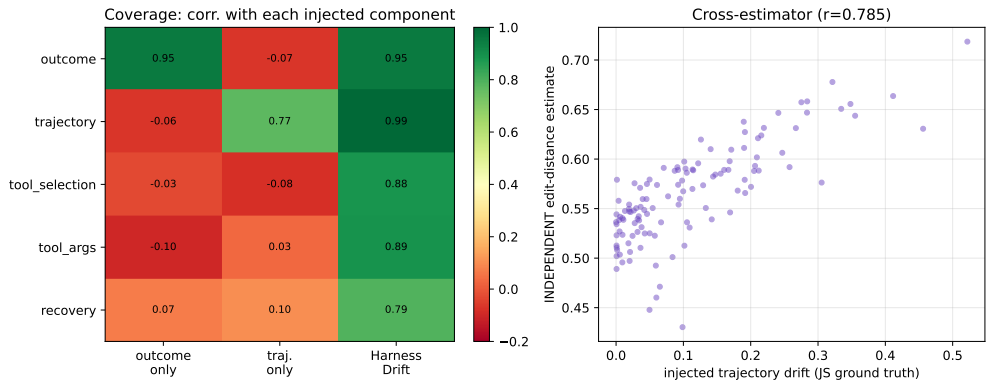

Figure 16: **Why not just watch the outcome?** *Left:* correlation between each injected component and what three estimators recover. An outcome-only estimator sees the outcome component (0.95) and nothing else (-0.10 to 0.07); a trajectory-only estimator sees trajectory drift (0.77) and nothing else. The trajectory decomposition recovers all five (0.79–0.99). The red columns are the reason a single number is not enough. *Right:* an *independent* edit-distance estimator that shares no code with ours tracks injected trajectory drift at r=0.785, so the recovery is not an artifact of one implementation.

Table 52: E7 model×scaffold interaction on SWE-bench Verified (per-instance resolved, n=500). Cell values are resolve rates.

Grid	cells (resolve rate)	S_M	$S_{M\mathcal{H}}$	κ
{gpt-4o, claude-3.5-sonnet}×{sweagent, agentless}	.23/.39, .34/.51	0.014	0.000	0.005
{claude-3.5-sonnet, claude-3.7-sonnet}×{sweagent, tools}	.34/.49, .62/.63	0.046	0.005	0.103

to the live study, where the interaction over seven fully-covered models reaches $\kappa \approx 0.47$, this is a *brittle↔robust spectrum*: mature coding scaffolds on repository tasks are largely harness-robust, interface-sensitive tool-use suites are brittle. We originally read this contrast as *capability*-driven (frontier vs. weak models); a pre-registered strong-model replication on our own suite does not support that reading (App. I), so we attribute the contrast to the task/scaffold regime and treat the capability account as untested. These SWE-bench κ values come from 2×2 grids with one resolve rate per cell, so they have no residual degrees of freedom: we read them as *descriptive* of the ordering, not as tested point estimates, and the live study (App. AL) supplies the replicated, permutation-tested κ.

AJ Full decomposition on real trajectories: AgentRewardBench (E8)

AgentRewardBench (Lù et al., 2025) runs the identical `GenericAgent` (BrowserGym) harness with four different LLMs—`gpt-4o`, `claude-3.7-sonnet`, `llama-3.3-70b`, `qwen2.5-vl-72b`—on shared WebArena tasks and releases the full trajectories (actions, errors, reward). This is a real, fixed-harness model substitution *with trajectories*, so the trajectory decomposition produces the *complete* component decomposition of each substitution, not just outcomes. We use the n=30 shared tasks whose trajectory files for all four models fall under a download-size cap.

Table 53 and Fig. 18 report success rates and the component decomposition of each substitution relative to `gpt-4o`. The substitutions rewire *how* the agent acts along several axes at once: outcome flips reach 23% of tasks (`gpt-4o`→`llama-3.3-70b`); tool-selection drift appears where models prefer different action types (0.18 for `claude-3.7-sonnet`); and—on these harder tasks, which trigger real action errors—*recovery drift activates* for most pairs (0.00, 0.36, 0.36, 0.50, 0.50, 0.86; one pair shows none at all): models differ markedly in whether they correct after a failed action, and

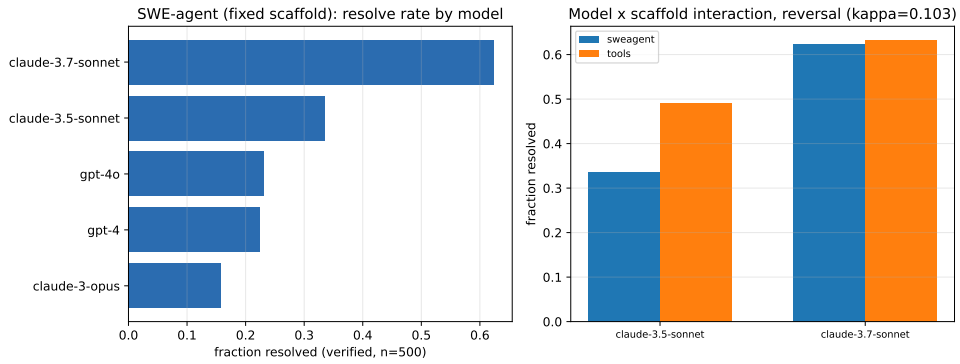

Figure 17: E7 (SWE-bench Verified, real logs): (left) resolve rate by model under the fixed SWE-agent scaffold; (right) a model×scaffold cell grid whose interaction the trajectory decomposition quantifies.

at n=30 this estimate is the least stable of the five. All five components are thus exercised on real web agents, with the caveat that recovery rests on only 2–5 error steps per model at n=30 and is flagged low-confidence.

Table 53: E8 (AgentRewardBench, WebArena, fixed GenericAgent harness, n=30 tasks): success rates and component decomposition of the three `gpt-4o`-anchored substitutions (success 0.43), shown for brevity; the refined argument metric and the cross-class pool (App. AK) use all $\binom{4}{2}$=6 pairwise contrasts among the four models. "args (raw)" is the naive flat-zip estimator, kept for contrast; "args†" is the refined aligned+semantic tool-argument drift (see below). The two are computed by separately named fields so the contrast cannot silently collapse when the canonical registry entry changes, which is how a stale version of this table survived one such change.

Substitution (gpt-4o →)	succ.	flip	outcome	traj.	tool-sel	args (raw)	args†	recovery
`claude-3.7-sonnet`	0.43	0.13	0.00	0.00	0.18	0.97	0.30	0.50
`llama-3.3-70b`	0.33	0.23	0.10	0.00	0.02	1.00	0.35	0.00
`qwen2.5-vl-72b`	0.33	0.10	0.10	0.00	0.09	0.91	0.26	0.36

Recovery is not reliably estimable at this n. The recovery rate is (#recoveries)/(#errors), but on these n=30 WebArena tasks the error-event denominator is tiny: only 2–5 tasks per model contain any error step (`gpt-4o` 2, `claude-3.7` 2, `llama-3.3` 2, `qwen` 5; 2–22 total error events). The recovery-component values in Table 53 therefore rest on a handful of events and should be read as low-confidence; we rely on the recovery component only in the synthetic ground-truth study and larger runs, and report the other four components as the substantive real-data decomposition here.

Components are axis-specific on real data, too (E23). A fair concern is that the 10× component specificity of E1 (App. AH) is measured on a simulator *built* to be axis-specific. We therefore test it where it matters: we compute each component *per task* on the real WebArena pairs—all $\binom{4}{2}$=6 model pairs × 30 tasks = 180 task-level observations, using the paper's own estimators—and report the empirical inter-component Pearson correlation. Among the four well-populated components (outcome, trajectory, tool-selection, aligned tool-arguments) the mean absolute off-diagonal correlation is **0.07** (max 0.10): the components carry *largely independent* signal on real agents, corroborating the simulator's specificity rather than relying on it. (Recovery is non-zero on only 10% of these tasks—too sparse to assess, consistent with the denominator caveat above.)

Refined tool-argument metric. On web actions the naive tool-argument drift (0.91–1.00 above) conflates two effects: volatile element-ids/pixel-coordinates that differ trivially across page states, and *trajectory misalignment* (the models take different actions, so position-i arguments do not correspond). Stripping element-ids alone barely helps (0.90–0.98), because misalignment dominates.

Our refined estimator (§AR, `tool_arg_drift_aligned` with the `web_semantic_args` normalizer) instead pairs the two trajectories' tool calls by minimum-cost assignment *within* each tool name and compares arguments only on matched same-tool calls (genuinely comparable actions), with element-ids removed. Two earlier descriptions of this estimator appeared here — alignment by tool name alone, then order-preserving alignment — and both were defective on repeated same-name calls; App. H records why. This isolates argument drift from trajectory/tool-selection drift and reduces the estimate to a meaningful 0.26–0.45 across the six substitutions (it was 0.45–0.65 under the name-only alignment)—the models do target different elements and type different text even on matched actions, but far less than the naive metric implies. We report the refined value for web agents; synthetic and SWE-bench results are unaffected.

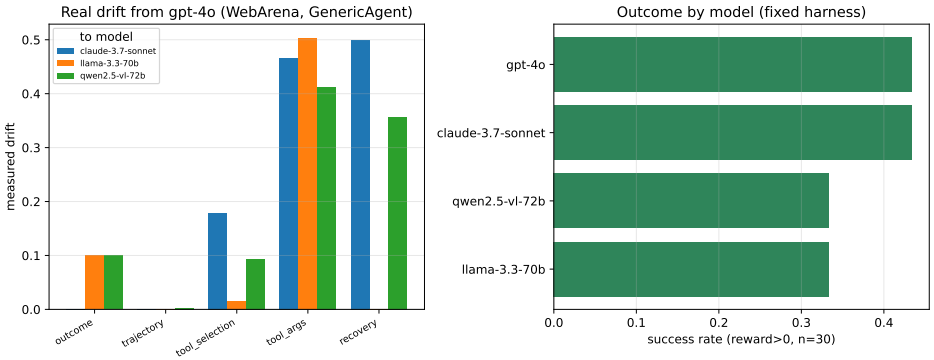

Figure 18: E8: (left) component drift of each substitution from `gpt-4o` on WebArena under the fixed GenericAgent harness (n=30); (right) success rate by model. The trajectory decomposition yields the full decomposition on real agent trajectories.

Drift that outcome-level churn cannot express (E31). Item-level churn is defined on outcomes, so it is blind by construction to a swap that reaches the *same* outcome by a different route. We measure that residue directly on these trajectories: for each model pair, among tasks where both models agree on the outcome, we count those whose action sequences diverge substantially (normalized edit distance ≥ 0.5).

pair (n=30 shared tasks each)	same outcome	… different path	% of all tasks
`claude-3.7-sonnet` vs. `gpt-4o`	26	7	23.3
`claude-3.7-sonnet` vs. `llama-3.3-70b`	23	11	36.7
`claude-3.7-sonnet` vs. `qwen2.5-vl-72b`	25	16	53.3
`gpt-4o` vs. `llama-3.3-70b`	23	11	36.7
`gpt-4o` vs. `qwen2.5-vl-72b`	27	16	53.3
`llama-3.3-70b` vs. `qwen2.5-vl-72b`	26	13	43.3
median	—	—	**40.0**

The median is 40% of *all* tasks and 48.9% of same-outcome tasks (range 23.3–53.3% over the six pairs). No outcome-level statistic—the flip rate included—can see any of it. An earlier version of this analysis returned 0.0 for every pair; that was an extractor defect (these parsed trajectories key actions under `actions`/`fn`, not `steps`/`action`), not a negative result, and it is recorded as such in our development log.

AK Generalization to a tool-agent-user agent class (τ-bench, E20)

The trajectory decomposition's real-data evidence spans *coding* (SWE-bench, App. AI) and *web navigation* (WebArena, App. AJ). To test generality across a qualitatively different agent class, we apply the full decomposition to **τ-bench** (Yao et al., 2024), a benchmark for *tool-agent-user* interaction: a customer-service agent converses with a simulated user and calls database-backed domain

tools (airline, retail), graded by a programmatic reward. We use the public in-repository historical trajectories for two models—`gpt-4o` and `claude-3.5-sonnet` ("new")—run under one fixed τ-bench harness, and form the `gpt-4o`→`sonnet` substitution on each domain (trial 0 per shared task; n=50 airline, n=115 retail). The analysis is entirely offline (no API calls).

Table 54: Trajectory decomposition of the `gpt-4o`→`claude-3.5-sonnet` substitution on τ-bench under one fixed harness. The aggregate reward *gap* is small, yet the per-task outcome *flip* rate is several times larger (silent drift), and the drift localizes to outcomes and tool arguments — in that order on retail (0.11 vs. 0.05) and the reverse on airline (0.08 vs. 0.09), a ranking that only became visible once the argument estimator was corrected twice (App. H); the pre-correction table put tool arguments far ahead on both. †tool-argument drift is the alignment-based estimator (App. AJ); the naive value is inflated by call misalignment (0.94 airline, 0.97 retail).

		success				component drift				
Domain	n	`gpt-4o`	`sonnet`	gap	flip	outcome	traj.	tool-sel.	tool-arg†	recovery
Airline	50	0.42	0.50	0.08	**0.36**	0.08	0.01	0.05	0.09	0.00
Retail	115	0.62	0.73	0.11	**0.27**	0.11	0.00	0.01	0.05	0.00

Three findings (Table 54, Fig. 19). *(i) Per-task churn exceeds the score gap.* The substitution's aggregate reward gap is small (0.08 airline, 0.11 retail), yet the per-task outcome *flip* rate is 36% (airline) and 27% (retail)—bidirectional churn the aggregate score nets out, as on SWE-bench and WebArena. *(ii) The decomposition localizes the change* to *tool arguments* and *outcomes*, with near-zero action-type and recovery drift: the two models follow a similar high-level rhythm (comparable step counts, 27.7 vs. 24.8 airline) but pass different arguments to the same tools. The alignment-based argument estimator (App. AJ) de-inflates the naive value from 0.94/0.97 to **0.09/0.05**, confirming that refinement generalizes beyond web actions. (These were 0.28/0.16 under the name-only alignment; the assignment-based matching of App. H accounts for the rest, because τ-bench trajectories call the same few tools repeatedly and that is exactly the case name-only alignment mishandled.)

(iii) A same-model negative control (E21) shows these public logs are high-temperature, so the per-task flip is only partly attributable to the substitution. τ-bench ships *four* stochastic trials per task per model; the same-model across-trial outcome-flip rate—an in-pipeline *null floor*—is 0.293 (airline, `gpt-4o`; range 0.22–0.38 over six trial pairs) and 0.226 (retail), with `claude` similar (0.269/0.233). The cross-model flip (0.36/0.27) is therefore only $\mathbf{1.23}\times/\mathbf{1.19}\times$ the within-model floor. Bootstrapping the *excess* (cross-model flip minus the mean same-model floor, averaged over both models) over tasks (E29, 10k resamples) puts it at 0.079, 95% CI $[-0.036, 0.197]$ (airline) and 0.040, CI $[-0.038, 0.120]$ (retail): **the attributable excess is not distinguishable from zero at either domain**—at this benchmark's sampling temperature, the per-task flip is consistent with pure decoding noise (the regime App. U (E17) flags). We report this transparently: it *bounds* the τ-bench silent-drift magnitude to within the noise floor, and methodologically shows the trajectory decomposition can *measure its own noise floor on real logs*. Clean per-task attribution requires the temperature-0 protocol of our live studies (E5/E9/E16), where the floor is near zero. We therefore present τ-bench as a *decomposition demonstration*—the five-axis profile and its localization to tool arguments are informative regardless of the flip magnitude—rather than as evidence that the substitution drives the flip.

With only two public models, no harness variants, and a high sampling temperature, we report τ-bench as a *decomposition demonstration with a measured noise floor* on a third agent class—*not* a model×harness κ and *not* a clean substitution-attributed flip rate. The method and decomposition carry over to a third, structurally distinct class; clean attribution of the drift magnitude there awaits temperature-0 re-collection.

Cross-class synthesis: how far behavioral churn outruns score drift. We pool every real model-swap pair we measured across the three agent classes: 10 on SWE-bench (all $\binom{5}{2}$ pairwise contrasts among five models, App. AI), 6 on WebArena (all $\binom{4}{2}$ contrasts among four models, App. AJ), and 2 on τ-bench, for 18 pairs (Fig. 20). *These pairs are not independent*—they reuse the same models and instances within each class—so we read the pool descriptively, not as an i.i.d. sample. A clarification first: for paired binary outcomes the per-task flip rate $(a{+}b)/n$ *necessarily* upper-bounds the absolute aggregate-score gap $|a{-}b|/n$ (where a, b count the two flip directions), with equality iff all flips

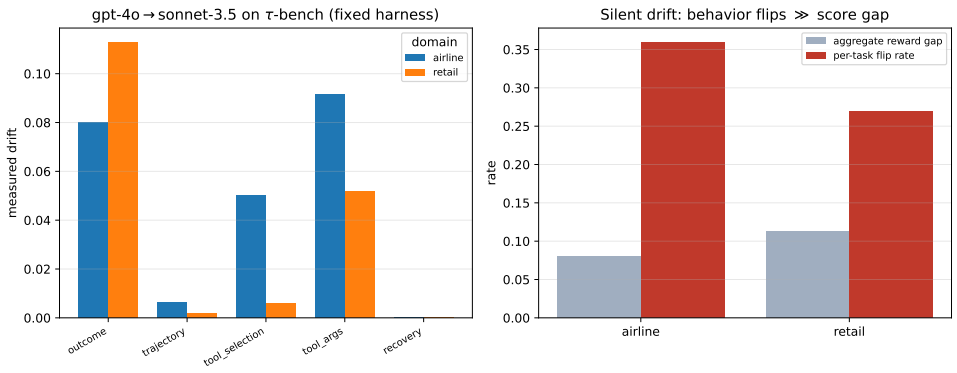

Figure 19: E20 (τ-bench, tool-agent-user). (left) component decomposition of the `gpt-4o→sonnet` substitution per domain (tool-argument shown as the inflated naive value; see Table 54 for the aligned estimate). (right) the per-task outcome flip rate exceeds the aggregate reward gap in both domains, though a same-model control (E21) shows much of the flip is sampling noise at this benchmark's temperature.

go one way. So "flip $\geq$ gap" is an algebraic identity, not a finding; the empirical content is *how far* the flip rate exceeds the gap. Rather than lean on that ratio—whose denominator can be arbitrarily small—we calibrate every pair against its *independence null* $p(1-q)+q(1-p)$, the flip rate expected if the two models erred independently (E33). Across all 18 pairs the observed flip sits at a **median** 55% **of the way from identical** (0) **to independent** (range 21–82%), and *no* pair reaches or exceeds its independence null. Two models that a leaderboard calls equivalent therefore agree far more than chance but far less than a drop-in replacement—a bounded, interpretable statement that the flip/gap ratio cannot make. Empirically the flip rate (0.10–0.48, median 0.23) is a *median* $2\times$ the gap and up to $21\times$ (`gpt-4→gpt-4o`, where the gap is 0.008 so the ratio is large but unstable). The informative regime is score parity: the three swaps with gap < 0.02—including two WebArena pairs with *identical* aggregate scores—still flip 13–17% of individual outcomes, and the one pair at the identity boundary (WebArena `gpt-4o→qwen`, flip $=$ gap $= 0.10$) has all flips one-directional. Per-task flip rates carry wide bootstrap intervals at these sample sizes (e.g., τ-bench airline 0.36, 95% CI $[0.24, 0.50]$, n=50; the score-tied WebArena pairs 0.13, CI $[0.03, 0.27]$, n=30), so no single pair is decisive; the cross-class *pattern*—not any one swap—is the claim. A crucial caveat bounds how much of this pattern is attributable to the substitution rather than to sampling: these public logs are single-sample (SWE-bench, WebArena) or high-temperature (τ-bench), and the comparison above is against the *score gap*, not against the *within-model noise floor*. Where we could measure that floor—τ-bench, with four trials per model (E21)—the cross-model flip is only $\sim 1.2\times$ the same-model self-flip, so on that benchmark much of the per-task flip is decoding noise. The honest reading: the cross-class synthesis shows the silent-drift *phenomenon* (per-task churn the aggregate score hides) is visible across structurally distinct classes, but rigorous *attribution* of its magnitude to the model substitution comes from the controlled temperature-0 live studies (E5/E9/E16), where the noise floor is near zero; the public-log pairs are corroborating, not the attribution the claim rests on.

A mapping defect in the *trajectory* component, now fixed and bounded (E35/E37). τ-bench is tool-agent-*user*, and our six-token vocabulary has no token for an utterance to the simulated user, so we originally normalized every content-only assistant message to `plan`. An independent labeler agreed with us $100/100$ on WebArena but only $74/100$ here (App. AU). Re-mapping such a message to `answer` when it hands the turn back to the user—using message ordering only, no model and no new token—raises agreement here to $94/100$. The defect was *relatively* large: it accounted for 59% of the airline trajectory component ($0.0064 \to 0.0026$) and 67% of retail's. But it is *absolutely* negligible, because that component is ≤ 0.006 while `tool_args` is 0.09: the largest shift in any reported quantity is 0.0038, and the outcome, tool-selection, tool-argument and recovery components and both flip rates move by exactly zero. The comparator here was 0.94 in an earlier revision, which is the naive flat-index value this paper elsewhere describes as a bug; the bound is stated against the corrected 0.09, and it still holds — 0.0038 against 0.09 rather than against 0.94

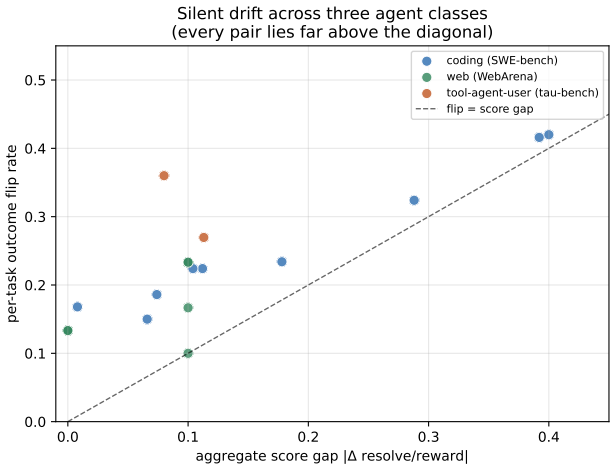

Figure 20: Cross-class synthesis: per-task outcome flip rate vs. aggregate score gap for all 18 real model-swap pairs (dependent: pairwise contrasts within three agent classes). The diagonal (flip = gap) is a *lower bound* that holds by construction for paired binary outcomes; the vertical distance above it is the bidirectional churn the aggregate score nets out. Score-tied pairs at the left edge still flip 13–17% of outcomes at near-zero aggregate gap.

— because the shift itself is on the trajectory axis and did not move. The demonstration-only status of this study therefore rests where it did—on the noise-floor argument above, which E37 does not touch.

AL EXPANDED MULTI-MODEL LIVE STUDY (E9)

To test the interaction at scale on real models, we ran the live tool-use harness (App. D) across *eight* models spanning six families—including two within-family scale pairs (`llama-3.1-8b`/`llama-3.3-70b`, `gpt-oss-20b`/`gpt-oss-120b`)—under both interface variants (ReAct, JSON) on a 52-task verifiable suite. Seven models completed all 52 tasks; `gemini-2.0-flash` completed 18 (provider cap). The entire study cost **$0.17** in API spend, illustrating that counterfactual drift attribution is cheap to run.

The optimal harness is model-dependent. Table 55 gives success rates. The best interface *flips across models*: four models score higher under JSON (`gemini-2.0-flash`, `gpt-oss-120b`, `llama-3.1-8b`, `qwen3-14b`) and four under ReAct (`gpt-oss-20b`, `llama-3.3-70b`, `mistral-small-24b`, `phi-4`), with swings as large as 0.44 (`mistral-small-24b`: 0.27 JSON vs. 0.71 ReAct—an outright reversal) and 0.42 (`qwen3-14b`: 0.75 vs. 0.33). A model-substitution decision made under one harness therefore need not transfer to another—precisely the failure mode κ is designed to flag.

A robust multi-model interaction. Fitting the two-way ANOVA over the seven fully-covered models $\times$ 2 harnesses $\times$ 52 tasks (response: task success) gives S_M=0.10, $S_{M\mathcal{H}}$=0.09, and an interaction ratio κ=**0.47** (task-bootstrap 95% CI $[0.33, 0.60]$; permutation p=5×10^{-4}, App. S)—i.e. nearly half of the model-attributable variation in success is harness-dependent. The all-eight-model design over the 18 commonly-covered tasks gives a consistent κ=0.36; on the action-count response κ=0.15. This is a large, real model$\times$harness interaction estimated over many models, complementing the small-sample within-family estimates of App. D and confirming at scale that "which model is best" is not harness-invariant.

Table 55: E9 success rates (fraction solved) under each harness. Seven models on n=52 tasks; `gemini-2.0-flash` on n=18. Better harness per model in bold; it splits 4/4.

Model	JSON	ReAct	prefers
`gemini-2.0-flash` (n=18)	**0.94**	0.89	JSON
`qwen3-14b`	**0.75**	0.33	JSON (Δ0.42)
`llama-3.3-70b`	0.48	**0.63**	ReAct
`gpt-oss-120b`	**0.38**	0.29	JSON
`mistral-small-24b`	0.27	**0.71**	ReAct (Δ0.44, reversal)
`gpt-oss-20b`	0.25	**0.31**	ReAct
`llama-3.1-8b`	**0.25**	0.00	JSON
`phi-4`	0.08	**0.40**	ReAct

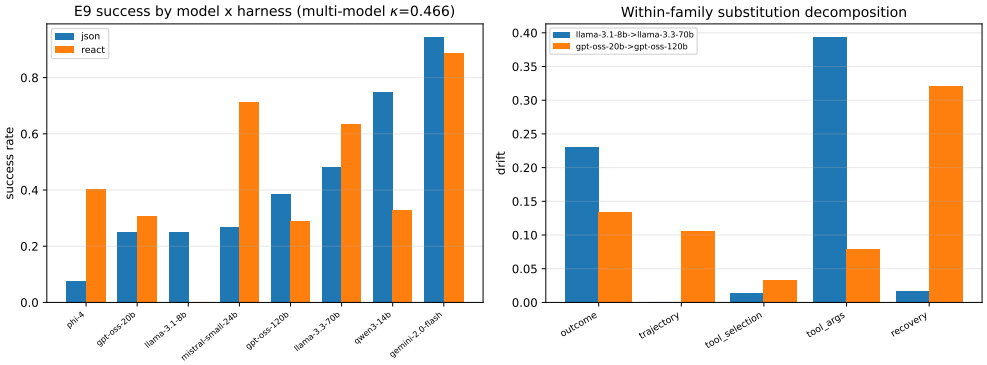

Figure 21: E9: (left) success by model under each harness across eight models (multi-model κ=0.47 on success over the fully-covered models); (right) component decomposition of the within-family scale upgrades.

AM Real-data brittleness: predicting regressions before deployment (E10)

§AH validated brittleness prediction synthetically (AUC 0.83; it read 0.86 before the tool-argument correction of App. H). Here we test it on the real eight-model data (App. AL). We pose brittleness in its most actionable, well-posed form: for a task the *old* model currently solves, will the *new* model *regress* (fail) after the swap? This target is directional (old→new $\neq$ new→old) and per-task, unlike the symmetric drift index, giving many labeled examples (1,944 old-solved cells across the eight models, two harnesses, and task suite). The base regression rate is 0.57.

All features are available *before* running the new model: the old model's behavior on that task (steps, tool calls, errors, recovery rate), its overall success under the harness, and static model-pair metadata (parameter-scale ratio, same-family flag, harness). The cross-validation grouping determines *what* the AUC measures, so we report three schemes of increasing difficulty (E29):

CV scheme	full-model ROC-AUC	generalizes to…
grouped by **task**	$\mathbf{0.75 \pm 0.02}$	an unseen *task*, model-pair already sampled
leave-one-**old-model**-out	0.56 ± 0.14	an unseen incumbent model
leave-one-**pair**-out	0.49 ± 0.06	an entirely unseen (old, new) pair

This is the honest scope of the result, and it corrects an over-claim in an earlier draft. When the model-pair has already been observed on *some* tasks, regressions on its *remaining* tasks are predictable before running them (grouped-by-task AUC 0.75, 95% range $[0.71, 0.79]$ over 30 splits, App. S). Two later re-analyses cut that number down and we report them here rather than only in the summary. First, 0.5 is the *wrong* baseline for it: 99.1% of the comparisons that AUC scores are *between* model-pairs rather than within one, and trivial baselines reach **0.786** on the same task—so the 0.75 is not above the right chance line, it is below it (E31). On the within-pair axis, which is the

one triage actually needs, AUC is $\mathbf{0.641}$ $[0.600, 0.676]$ where every baseline is pinned at 0.500 by construction: a real but modest signal. Second, the signal is carried mainly by the pair-level scale ratio (metadata-only 0.71; incumbent-behavior-only 0.56), so it does *not* extrapolate to an unseen pair: leave-one-pair-out AUC is 0.49, at chance. The practically supported claim is therefore *task interpolation for a calibrated pair*—given a sample of the new pair's behavior, triage which of the incumbent's currently -passing tasks the swap will break—not zero-shot brittleness prediction from metadata alone. The real task-interpolation AUC is also lower than synthetic (0.86), as expected—real drift is noisier and the cheap features are imperfect. (An earlier symmetric drift-index target was inconclusive at this scale; the directional regression target is both better-posed and what practitioners care about.)

Operational metrics (E22): actionable in the calibrated-pair regime. AUC is threshold-free; a deployment screen needs precision at a usable operating point. Using out-of-fold *grouped-by-task* probabilities over all $1{,}944$ examples (i.e. the calibrated-pair regime above, *not* the unseen-pair regime), the predictor attains PR-AUC 0.79 ($1.39\times$ the 0.57 base rate). Operated as a top-k triage: flagging the riskiest 10% of tasks yields **precision** 0.91 (recall 0.16); the top 20%, precision 0.85 (recall 0.30); the top 30%, precision 0.79 (recall 0.42). So a team re-testing only the top-flagged tenth of its suite *for a model-pair it has already partly run* catches a real regression 9 times out of 10—a concrete, favorable triage trade-off within that regime.

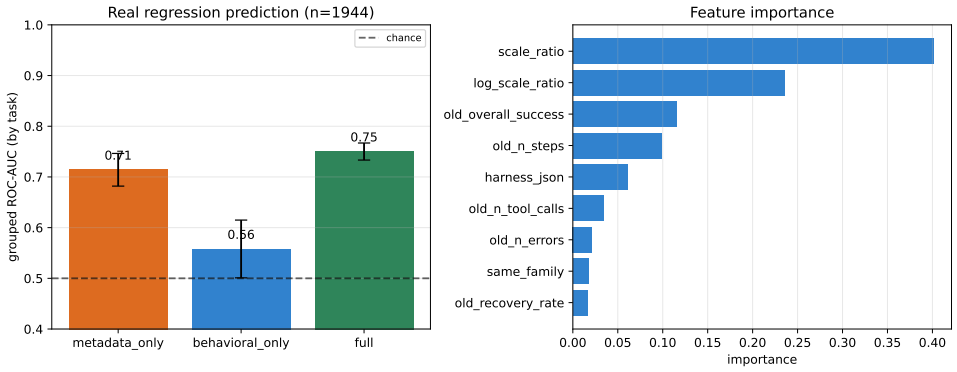

Figure 22: E10: (left) grouped-by-task ROC-AUC for predicting real regressions of a *calibrated* model-pair on its unseen tasks; (right) feature importance (pair-level scale ratio dominates, which is why the signal does not extrapolate to an unseen pair; see E29 CV table).

AN A SECOND TASK FAMILY: GENERALIZATION AND CROSS-FAMILY TRANSFER (E14)

All live studies so far use one task family (arithmetic/`db`-lookup). To test whether the findings generalize, we built a structurally different family: **wiki retrieval**, a fictional 12-page encyclopedia with tools `search`$(q)\to$ids and `read`$(id)\to$page plus `calc`; the 16 verifiable tasks require multi-hop reading (find a page, extract a numeric field, combine) rather than pure arithmetic. We ran the same seven-model pool under both ReAct and JSON harnesses (data cached to `data/live_wiki`). We *pre-registered* (`experiments/15_family2/protocol.md`) three predictions and report each verdict.

P1 (interaction present) — REFUTED, informatively. On family B the model$\times$harness interaction is *weak and not significant*: $\kappa_B = 0.06$ (bootstrap 95% CI $[0.03, 0.22]$), permutation $p = 0.49$, and 6 of 7 models simply prefer ReAct. This contrasts sharply with family A's strong, significant interaction ($\kappa = 0.47$, $p<10^{-3}$) and matches the SWE-bench coding-scaffold regime ($\kappa \approx 0$). The interaction's *magnitude is task-family-dependent*—not a universal constant—and the trajectory decomposition correctly reports it as near-zero where it is near-zero, with a permutation test that does not over-claim. That a pre-registered confirmatory prediction is refuted, and reported as such, is the intended behavior of a measurement instrument.

P2 (cross-family transfer) — CONFIRMED, the key result. We train the directional regression predictor (§AM) *only on family A* (arithmetic; 1,602 examples—the subset of E10's 1,944 cells from the seven models also run on family B; `gemini-2.0-flash` was delisted mid-study) and evaluate it on family B (wiki; 354 examples), which it has never seen. It predicts family-B regressions at **ROC-AUC** 0.68 (task-bootstrap 95% CI $[0.61, 0.75]$; lower bound $0.61 > 0.5$ chance, clearing the locked 0.60 bar; the class prevalence is 0.69, distinct from the AUC null of 0.5). Because family B reuses the *same* seven-model pool, this is a cross-*task-distribution* transfer for known model-pairs—the features that predict which currently-solved tasks a swap will break carry signal across task families, not merely within one—and it is complementary to the leave-one-pair-out result (E29): the signal generalizes to new *tasks* of a known pair, not to a new pair. Modest but non-null evidence that the brittleness mechanism is a property of model substitution, not of a single benchmark (Fig. 23).

P3 (preference shift) — CONFIRMED (exploratory). Best-harness preference is family-dependent: family A splits 4/4 JSON/ReAct, whereas family B is 6/7 ReAct. A model's best interface is therefore contingent on *both* the model and the task distribution—reinforcing that "best model" and "best harness" are not transferable defaults.

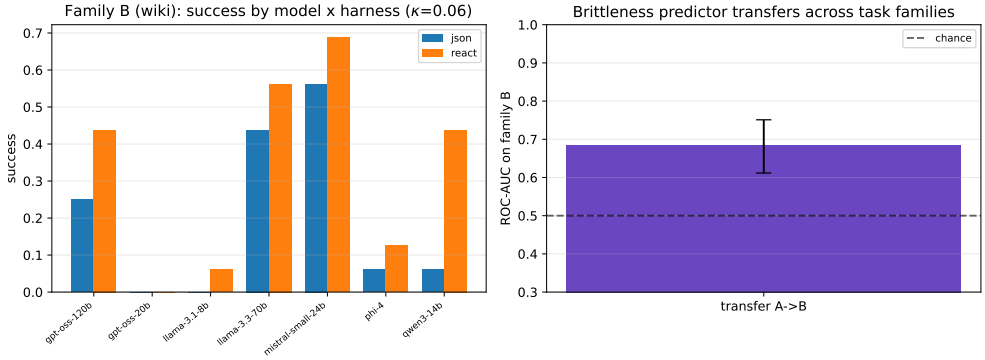

Figure 23: E14: (left) family-B (wiki) success by model×harness—interaction is weak here (κ_B=0.06, n.s.); (right) the brittleness predictor trained on family A transfers to family B (AUC 0.68, CI $[0.61, 0.75]$), well above chance.

AO THREE HARNESS INTERFACES: THE INTERACTION IS NOT A TWO-LEVEL ARTIFACT (E16)

A two-level (ReAct/JSON) design could in principle exaggerate an interaction. We therefore added a *third* interface to the arithmetic family—an XML tool-call protocol (`<act tool="db">Zorin</act>`, answers as `<answer>N</answer>`)—and re-ran the seven fully-covered models under all three harnesses ({ReAct, JSON, XML}). The model×harness interaction *persists* at three levels: $\kappa = 0.44$ on the 12-task subset (task-bootstrap 95% CI $[0.24, 0.64]$), permutation $p = 5 \times 10^{-4}$, with a negligible harness main effect ($S_{\mathcal{H}} = 0.02$)—so the variance is genuine interaction, not a global "XML is harder" shift.

This probe is superseded, and we keep it because of what it got wrong. At full 52-task coverage the same seven models under the same three interfaces give κ=0.377 (App. AP), so the 12-task estimate was optimistic; we report the 52-task value throughout. The conclusion that survives is the three-way split of the optimum. What does *not* survive is any per-model reading: going from 12 to 52 tasks **flips** 2 **of** 7 argmax assignments (`gpt-oss-120b` XML-best → JSON/XML tied at 0.385; `gpt-oss-20b` ReAct-best → XML-best). In particular, the sharp within-family contrast this probe appeared to show—`gpt-oss-120b` best under XML while its smaller sibling `gpt-oss-20b` was *worst* under XML (0.17)—**reverses**: at 52 tasks XML is `gpt-oss-20b`'s *best* interface of the three (0.327 vs. ReAct 0.308, JSON 0.250). We had drawn a non-monotone-within-family conclusion from 12 tasks, and it did not survive.

We can now say how ordinary that failure was rather than guessing: a 12-task probe over these three interfaces mis-assigns 1.31 of 7 models in expectation, with $P(\geq 2 \text{ flips})=0.40$ (E47, App. AA). So this probe was not unlucky—12 tasks simply cannot resolve per-model interface choice here, and the same appendix shows what can. That measurement exists because this correction prompted it.

On this subset the optimal interface spreads across *all three* options (Table 56), and that qualitative split does hold at 52 tasks—but with a different assignment (App. AP, Table 58). The claim a practitioner can rely on is the one that is stable under both measurements: the optimum is genuinely split across interfaces, so there is no interface to default to. *Which* interface wins for a *given* model is not resolvable from a 12-task probe.

Table 56: E16 success rate per model under three harness interfaces (n=12 shared tasks). Best per model in bold. Retained as the historical probe: the 52-task grid in Table 58 supersedes it and disagrees on 2 of 7 rows.

Model	ReAct	JSON	XML	best
`gpt-oss-120b`	0.42	0.50	**0.75**	XML
`gpt-oss-20b`	**0.50**	0.42	0.17	ReAct
`llama-3.1-8b`	0.00	**0.33**	0.00	JSON
`llama-3.3-70b`	**0.67**	0.58	0.50	ReAct
`mistral-small-24b`	**0.92**	0.50	0.42	ReAct
`phi-4`	**0.42**	0.25	0.00	ReAct
`qwen3-14b`	0.25	**0.92**	0.50	JSON

AP FOUR HARNESS INTERFACES: THE INTERACTION IS NOT A TWO-LEVEL ARTIFACT (E36)

The headline interaction is estimated on a two-level harness factor, which invites two objections: that a b=2 design can exaggerate an interaction, and that b=2 is exactly where our own EMS bias correction cannot act (Limitation T2). We answer both by raising the harness factor to **four** levels on the identical 52-task suite and the identical seven models, adding a fourth protocol form—`pythonic`, CodeAct-style function calls (`db("Zorin"), answer(N)`)—and completing the XML level from the 12-task subset of App. AO to all 52 tasks. The design is complete: $7 \times 4 \times 52$ with no missing cells. This experiment was pre-registered with four locked predictions; one is refuted and we report it as such. Table 57 gives the interaction at each harness-factor size.

Table 57: E36: the interaction across harness-factor sizes, all on the *same* 52 tasks and the same seven models, so b is the only thing that changes. κ is stable; the degrees of freedom differ at b>2, yet the EMS correction still moves little—a sample-size fact, not a b=2 artifact.

b	interfaces	κ	$\widetilde{\kappa}$ (EMS)	ν_M	$\nu_{M\mathcal{H}}$	$S_{\mathcal{H}}$
2	JSON, ReAct	0.219	0.204	6	6	0.001
3	+ XML	0.466	0.455	6	12	0.028
4	+ `pythonic`	**0.501**	0.487	6	**18**	0.022

The interaction is not a two-level artifact — but P1's locked test fails anyway (REFUTED). At b=4, $\kappa = 0.501$ (bias-corrected 0.487; task-bootstrap 95% CI $[0.390, 0.595]$; $p < 10^{-3}$ against a mean-matched no-interaction null). The *substantive* worry P1 was written to address — that the interaction is an artifact of having only two harness levels, and would shrink once more were added — is answered decisively in the opposite direction: adding levels *raises* the estimate. But P1 was locked as "κ at b=4 lies inside the b=2 interval", and on the single-provenance grid that interval is $[0.143, 0.319]$, so 0.501 falls outside it. **The locked test is refuted**, and we report it as refuted rather than rewriting it into the claim it happens to support. Its failure and P3's are the same fact seen twice: κ depends strongly on how many harness levels you measure, so an interval computed at b=2 does not bound an estimate at b=4.

But κ is *not* stable across harness-factor size (P3, REFUTED on the single-provenance grid). We locked "κ is stable across harness-factor size" and, on the stitched grid, confirmed it: the sequence read $0.466/0.377/0.394$ and spanned only 0.089. On the re-collected grid (App. E) it reads $0.204/0.455/0.487$ and spans $\mathbf{0.284}$, with the b=4 estimate falling *outside* the b=2 interval $[0.125, 0.309]$. **The prediction is refuted.** The direction is interpretable — JSON and ReAct are the two forms this suite's models handle most similarly, so the pair that isolates them has the least interaction to find, and each additional form adds a genuinely different demand — but we locked stability and did not get it, and the practical consequence is that a κ quoted without its harness-factor size is close to meaningless. Note also that the b=3 estimate is not comparable to App. AO's 0.44, which used a 12-task subset; we report the 52-task value throughout.

Every interface is somebody's optimum (P4, confirmed, with its stability measured). On the single-provenance grid all four forms are optimal for at least one model: JSON for `qwen3-14b`, ReAct for `gpt-oss-20b`, `llama-3.3-70b` and `phi-4`, XML for `gpt-oss-120b`, and `pythonic` for `llama-3.1-8b` and `mistral-small-24b`. We also resampled the pattern rather than resting on the point estimate: over 4000 task bootstraps at least *three* distinct forms win in 99.85% of draws, but all four in only 50.5% (App. E). The claim a practitioner should take away is that no interface is a safe default, not that exactly four are needed.

Table 58: Task success per model under the four interfaces on the single-provenance grid (n=52 tasks each; the same cells as App. E). Best per model in bold. Until 19 September this table printed the stitched June collection's cells beside prose about the single-provenance grid; those cells are retired to App. AX.

Model	JSON	ReAct	XML	`pythonic`	best
`gpt-oss-120b`	0.442	0.250	**0.462**	0.327	XML
`gpt-oss-20b`	0.212	**0.308**	0.173	0.135	ReAct
`llama-3.1-8b`	0.192	0.000	0.000	**0.269**	`pythonic`
`llama-3.3-70b`	0.481	**0.635**	0.462	0.500	ReAct
`mistral-small-24b`	0.404	0.712	0.250	**0.923**	`pythonic`
`phi-4`	0.788	**0.846**	0.038	0.269	ReAct
`qwen3-14b`	**0.750**	0.327	0.596	0.654	JSON

On the single-provenance grid the largest single swing is `phi-4`, from 0.038 under XML to $\mathbf{0.846}$ under ReAct—the same model on the same 52 tasks, differing only in protocol form. `llama-3.1-8b` scores exactly 0 under both ReAct and XML. The saturation control passes (cells span 0.000–0.923), and the pre-registered degenerate-cell fallback applies to `llama-3.1-8b`: excluding that model *raises* κ to 0.644, so the zero cells are not what produces the interaction. The most influential model is `phi-4` in the other direction—dropping it lowers κ to 0.327—which is the same model whose behaviour changed between our two collections (App. E), and we flag the coincidence rather than leaving a reader to find it.

A refuted prediction, and what it changes (P2). We predicted that once $\nu_M \neq \nu_{M\mathcal{H}}$ the EMS correction would become substantially informative, moving κ down by 0.03–0.20. The degrees of freedom do differ at b=4 (6 vs. 18) and the movement is nonetheless only 0.014 ($0.501 \to 0.487$ on the single-provenance grid; 0.023 on the stitched one). This is **refuted**, and it revises our own Limitation T2: the correction's small movement is a consequence of having 52 task-replicates, not of b=2. The substantive conclusion is unchanged and slightly strengthened—the corrected $\widetilde{\kappa} = 0.487$ is far above the κ=0 bias floor, so the interaction survives bias correction rather than being an η^2 artifact—but the mechanism we asserted for the correction being uninformative was wrong.

Protocol-adherence rates. Because a newly added interface's failures are otherwise ambiguous between genuine protocol-following failure and a parser stricter than what models emit, we report the fraction of trajectories that never produced a parsed tool call, on exactly this 7×4 grid: JSON 0.154, ReAct 0.192, XML 0.181, `pythonic` 0.239. `pythonic` is the highest but within the range the other three span, so its parser is not an outlier; App. F shows enforcing one action per step cuts all four to 0.04–0.07 (cf. Limitation T4).

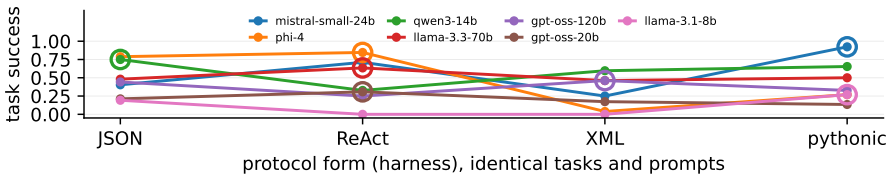

Figure 24: **The claim, drawn.** Same 7 models, same 52 tasks, byte-identical prompts; only the reply convention changes. Each model's best form is ringed: the rings land on *all four* forms and the lines cross. That is what κ=0.49 measures.

AQ Remaining real-agent studies: summary table

Table 59 collects the five corroborating real-agent studies summarized in §5; full protocols, figures, and uncertainty analyses are in the appendices each row cites. Every headline caveat is stated in the row itself.

Table 59: The remaining real-agent results (E10 analyzes the eight-model collection of E9; full protocols and figures in the cited appendices). Per-task churn exceeds the aggregate score gap across classes, bounded by measured noise floors where available.

Setting (class)	Result
Live tool-use, free models (E5, App. D)	Interface optimum is model-dependent: `gpt-oss-120b` 0.04 ReAct vs. 0.21 JSON, `gemini` invariant (1.0/1.0); the `20b`→`120b` upgrade is brittle ($\kappa\approx$0.33), adopting `gemini` robust ($\kappa\approx$0).
WebArena, 4 LLMs, full trajectories (E8, App. AJ)	Full five-axis decomposition, all components active with low cross-talk (mean $\lvert r\rvert$=0.07); recovery not estimable at n=30 (only 2–5 tasks per model contain an error step), flagged low-confidence.
Regression triage (E10, E31, E32; App. AM)	**Corrected downward.** Our earlier AUC 0.75 measures the wrong axis (99.1% of comparisons are *between* pairs) and trivial baselines beat it (0.786). Within a pair—the triage axis—AUC is 0.641 [0.600, 0.676] vs. baselines pinned at 0.500: a modest real signal, not extrapolating to unseen pairs (E29).
Second task family, wiki (E14, App. AN)	Interaction is family-dependent (κ_B=0.06, n.s.; a refuted pre-registered prediction, reported as such). The calibrated-pair triage signal transfers across *task distributions*: AUC 0.68, CI [0.61, 0.75].
τ-bench, tool-agent-user (E20, App. AK)	Decomposition extends to a third class, localizing a `gpt-4o`→`sonnet` swap to tool-args/outcomes (36%/27% flip). A same-model control shows the logs are high-temperature: the flip *excess* over the floor is within noise (CI [−0.036, 0.197]/[−0.038, 0.120], E29), ; a dialogue-act mapping defect in its trajectory component is fixed and bounded at 0.004 (E35/E37, App. AU)—a decomposition demonstration, not attributed drift.

Technical caveats moved here from the Discussion. These are the limitations that do not change a deployment decision; the decision-relevant ones stay in §7. (T1) *Orthogonality.* The decomposition is an interpretable, low-cross-talk projection, not a strict basis (App. AJ). (T2) *Estimator bias.* Absolute-difference estimators are upward-biased at zero drift. We predicted the EMS correction would matter once the degrees of freedom differ and **refuted our own prediction**; the case rests on a mean-matched null (App. S). (T3) *Interaction needs replication; granularity is coarse.* $V_{M\mathcal{H}}$ needs ≥ 2 harness variants, so we read the *ordering*; it replicates at b=3, 4 (App. AP). Vocabulary granularity degrades gracefully, never to chance, but argument-level *semantics* stays coarse (App. AH). (T4) *Our parsers were permissive, with a ")" cut.* One action per step cuts no-tool runs from 15–24% to 4–7% and κ survives at 0.35 (Apps. F, O). (T5) *Normalization faithfulness.* Extraction is exact (0/1,032 WebArena fallbacks) and an independent labeler reproduces our classes at κ=0.95 — but that labeler is an LLM, not a human (App. AU). (T6) *The five-component description is descriptive only.* The cosine between its profiles under two interfaces is 0.87, below the pre-registered 0.90 (App. AR). (T7) *Re-run floor, not pure decoding noise.* Measured on single-turn calls the floor is 0.08; reseeding whole trajectories gives the **0.22** we use, above the *closest* pairs' flip rate

(App. U). It is our suite's floor, not a threshold for SWE-bench or WorkBench, where we have no same-model re-run, and it limits per-task attribution on our suite.

AR DO THE FIVE COMPONENTS DISCRIMINATE ACROSS HARNESSES ON REAL DATA? (E39)

Limitation T6 concedes that the real-data results our claims rest on are κ, the paired flip rates, and the trajectory component, and that the *joint* five-axis decomposition on real logs is a demonstration rather than a claim. When this study ran, that was the strongest structural criticism of the paper, whose headline contribution was then a five-component instrument whose five components never did decisive work on real agents. That is not the current contribution, which Section 1 states; this appendix is a record. E36 left the material to test it, so we did. The outcome took three readings and we report all three, because the first two were ours to get wrong: the test appeared to fail, then — after we corrected an argument-alignment defect in our own estimator — appeared to pass on two of its three locked predictions, and then settled when we computed the *third*, which refutes and shows the apparent pass was carried entirely by the outcome component. The concession in Limitation T6 stands, on evidence rather than on the mis-diagnosis it originally rested on.

Design. For each of the $\binom{7}{2}=21$ model pairs and each of the four interfaces we computed the five component drifts over the same 52 shared tasks, using the released estimator unmodified—84 profiles from real trajectories, zero degenerate cells. Pre-registered question: does the *shape* of a substitution's drift depend on the harness? If it did, that would be a claim no outcome-level statistic can express. Table 60 reports the profiles.

Table 60: E39: mean component profile per interface over 21 model pairs, 52 tasks each, recomputed after the two argument-alignment corrections of App. H. `tool_args` is still the dominant axis under three of the four protocols, but at 0.25–0.55 rather than the 0.98–0.99 an earlier version of this table reported: those values came from the broken estimator, which charged reordered repeated calls as maximal argument drift. `pythonic` is now dominated by *outcome*.

interface	outcome	trajectory	tool sel.	**tool args**	recovery
JSON	0.249	0.056	0.106	**0.324**	0.244
ReAct	0.278	0.046	0.121	**0.410**	0.194
XML	0.242	0.042	0.182	**0.546**	0.106
`pythonic`	**0.353**	0.091	0.033	0.250	0.235

Both predictions were refuted, and both now confirm. We locked that the median cross-interface cosine similarity of the profile shape would fall below 0.90 with a significant permutation test, and that at least two different components would dominate. As first run, both were **refuted**: median cosine 0.957 with `tool_args` dominant under all four interfaces. Re-run after the two argument-alignment corrections of App. H, both **confirm** — but as a **post-hoc, exploratory** re-analysis, not as a pre-registered confirmation. The instrument changed after the data were seen, so the ledger (App. AW) counts both rows under their *locked* verdict, REFUTED, and tallies the flip in a separate column. We take no pre-registration credit for either. The corrected figures are: median cosine $\mathbf{0.870}$, permutation $p=0.001$ on interface-level profile dispersion (observed 0.154 against a null mean of 0.078), and *two* distinct dominant components (`tool_args` for JSON, ReAct and XML; `outcome` for Pythonic). We state both readings because the refutation was published in an earlier revision of this paper and the cause of the flip was a defect in our own estimator, not new data: the broken alignment charged reordered repeated calls as maximal argument drift, which pushed every interface's profile toward the same `tool_args`-heavy shape and so manufactured the invariance we then reported. What makes the correction trustworthy rather than merely convenient is that it is separable from its outcome: the defect is wrong on its face regardless of any verdict, the repair answered a reviewer question about repeated calls rather than a verdict, its regression tests were written from the defect's structure, and the same repair pushed other headline numbers *against* us (synthetic brittleness AUC $0.855 \to 0.830$, its behavioural-only variant $0.671 \to 0.632$). A change selected for its outcome would not have shipped those.

The finding is a dissociation, and P3 locates it. Two things this paper had implicitly bundled come apart: *which model wins* is strongly interface-dependent (κ=0.35–0.47, p=5×10^{-4}, all four protocol forms optimal for someone), whereas *how the drift is composed* depends on the interface only through its outcome coordinate. That is what P3 establishes: drop outcome and renormalise over the four process components, and the median cross-interface cosine goes back to 0.919 with `tool_args` dominant everywhere again. The decomposition describes real substitutions; its *process* axes do not discriminate between harnesses on this suite, and per E39's locked fallback (b) we decline to rest a claim on them.

Why — and the two wrong answers we published first. Our first explanation was that `tool_args` sat at 0.98–0.99, at the metric's ceiling, because an arithmetic tool-use suite gives every call a numeric or short-string argument. We named the concrete test that would settle it: a suite with heterogeneous argument structure. We then built exactly that suite (E48, App. H) and it **refuted the diagnosis**. Our second explanation was that normalised edit distance saturates as soon as two runs diverge; that was wrong too. The real cause was our own call *alignment*, in two successive defects, and with it repaired `tool_args` reads 0.25–0.55 here rather than 0.98–0.99 — nowhere near a ceiling. The invariance that survives the repair is therefore not a range artefact at all: it is the outcome-driven dissociation P3 isolates. We leave both discarded explanations on the record because each was stated here as the concrete, falsifiable test, and each was then falsified by our own follow-up.

The additive component decomposition. Two things get called drift, and our components measure both: the paired distance Δ_{pair} of Eq. (8), and a distance Δ_{dist} between two behaviour *profiles*, which is not paired. Mixing them is how a paper reports a no-op while a sixth of its items change answer (**silent drift**; App. AF). Five components measure distinct facets, each normalized to $[0, 1]$. **Exactly one of the five is paired**, the tool-argument edit distance; the other four are functionals of two distributions, and each has a paired counterpart we report beside it but keep out of the composite. So $D = \sum_k w_k \widehat{\delta}_k$ is a *descriptive composite*, not an estimator of either target, which is why κ and not D is the primary output. Every κ in this paper names the scalar response it decomposes — task success for the headline, an action-sequence distance in App. AF, a component elsewhere — and never D. The five are: the **outcome** gap between success rates (its paired counterpart is the flip rate); the **trajectory** divergence between action-type distributions (its paired counterpart is the aligned action-sequence distance below); the **tool-selection** divergence between distributions over which tools are called; the **tool-argument** edit distance over matched calls, the one paired component; and the **recovery** gap between the rates at which errors are followed by a corrective step. Every component is bounded in $[0, 1]$; the drift index $D = \sum_k w_k \widehat{\delta}_k$ uses uniform weights and is robust to random reweightings. A change confined to one facet moves chiefly its own coordinate on the simulator, which is data built that way; on real WebArena trajectories the mean cross-component $|r|$ is 0.07 (App. AJ). So read the components as a projection with low cross-talk, not an orthogonal basis. Both the components and κ are consistent as the task sample grows (Proposition 3, App. S); near zero the absolute-difference components carry an upward bias, so a null substitution never reads exactly 0, and Proposition 4 (App. S) gives the folded-normal correction we use when near-zero drift matters. Table 49 (App. A) names the four primary results and, for each one, the response, the population, the estimator and its correction, the source of uncertainty, and what it does not say; none of the four is D.

Localization. To localize a change, we align the two action sequences (Needleman–Wunsch), cut them into early, middle and late phases, and report the mismatch rate per phase. The estimator (Algorithm 1, App. AH) takes runs over tasks, models and ≥ 2 formats, normalizes traces, fits Eq. (6), and computes the components and the phase profile. We check that the decomposition reads back an interaction we set on a simulator, on the composite index and not on κ (App. AH), then apply it to real substitutions (§5; uncertainty in App. S).

AS WHAT κ MEASURES, WHAT MEASURING IT IS WORTH, AND WHETHER IT TRANSFERS (E40–E42)

Three analyses on the complete $7 \times 4 \times 52$ grid, at zero additional API cost. **Status disclosure:** an adversarial re-evaluation of this paper computed preliminary values for all three before we pre-registered them, so these are *verification of a third-party computation* rather than blind confirmatory tests. We recompute them with the released code, attach uncertainty, and fix the refutation conditions in advance of our own run; one prediction is refuted and reported as such.

E40: κ splits into an adherence layer and a layer beneath it. "How much of the effect is caused by the scaffold" conflates two mechanisms with different deployment implications: the model cannot emit this protocol's syntax (a candidate for constrained decoding or a better parser) versus the model reasons worse under this protocol. Running the same ANOVA on the same grid with three responses separates them by conditioning; whether constraining the decoder removes the first layer is a separate, mixed-date comparison (App. X).

response	κ	95% CI	perm. p
protocol adherence (emitted $\geq$1 parsed tool call)	$\mathbf{0.566}$	$[0.480, 0.668]$	5×10^{-4}
task success (the headline response)	0.501	$[0.409, 0.603]$	5×10^{-4}
success *given* adherence	$\mathbf{0.445}$	$[0.376, 0.523]$	5×10^{-4}

Adherence carries *more* interaction than success does (0.566 vs. 0.501): which model can even emit which protocol is the more harness-specific fact. Crucially, conditioning adherence away still leaves κ=0.445 with a lower bound of 0.376, so the interaction is **not** a syntax artifact—models that do follow a protocol still succeed at protocol-specific rates. All three predictions confirmed. This is the quantitative form of the concession in Limitation T4: constrained decoding could suppress the adherence layer, and a layer of comparable size would remain.

E41: a ten-task probe is worth about ten points of success. The paper can show that per-model optima exist; the number a practitioner needs is what acting on that returns. Honest split-half over the 52 tasks, 400 random splits, scored only on held-out tasks, comparing measure-then-choose against the best *fixed* interface chosen on the same probe.

probe K	measure-then-choose	best fixed	random	**gain**	splits positive
5	0.534	0.450	0.407	$+\mathbf{0.084}$ $[0.024, 0.149]$	100%
10	0.550	0.447	0.410	$+\mathbf{0.103}$ $[0.048, 0.160]$	**100%**
26	0.570	0.447	0.411	$+\mathbf{0.122}$ $[0.071, 0.187]$	100%

Both predictions confirmed: the gain at K=10 is $+0.103$ with a CI excluding zero, positive in *every one* of the splits, and monotone in K. Over all 52 tasks the best possible fixed choice is `json` at 0.467 against a per-model oracle of 0.599; ten probe tasks per model recover most of that headroom.

E42: κ transfers for direction, not magnitude (one prediction refuted). Leave-one-interface-out over the same grid: for each interface held out in turn, each pair's interaction statistic is computed on the remaining three and used to predict behaviour on the fourth. 27 **of** 84 **(pair, held-out interface) combinations reverse their ranking**—just under a third of pairwise model comparisons flip when only the protocol form changes. The training-fold statistic predicts *which* reverse at AUC 0.738 (permutation p=0.001), so the deployment gate has real out-of-harness validity. But it does *not* predict the held-out contrast *magnitude* (Spearman -0.001, p=0.99), which **refutes** our second locked prediction. We therefore recommend κ as a re-validation *trigger* and explicitly not as an effect-size forecast. Significance for both comes from the cell-level permutation, not a naive p-value, because the 84 observations reuse 21 pairs across four folds.

AS.1 DOES THE TRIGGER WORK ON MODELS AND TASKS IT HAS NEVER SEEN? (E74)

E42 above holds out one interface at a time and shows the training-fold statistic predicts which model pairs reverse on the held-out one. A reviewer pointed out what that does and does not establish:

it is leave-one-interface-out *within the same models and the same tasks*, so it shows the statistic generalises across interfaces, not across substrates. Nothing had tested the trigger on models or tasks it had not seen.

The repaired external grid lets us test it. We ran E42's procedure unchanged — same leave-one-form-out, same definition of a reversal, same cell-level permutation — on 13 models E42 never used, over Harness-Bench tasks scored by their own oracle.

	pairs × folds	reverse	AUC	permutation p
our suite (E42)	$21 \times 4 = 84$	32.1%	0.738	0.001
external, unseen (E74)	$78 \times 6 = 468$	23.3%	$\mathbf{0.718}$	$\mathbf{0.0002}$

Those 468 rows are not 468 independent facts: 78 pairs are built from 13 models, so each model sits in twelve pairs and each pair in six folds. The permutation preserves the fold reuse, but it does not say what happens if one *model* is unlucky. Dropping each model in turn and re-running gives AUC between 0.674 and 0.748 (mean 0.717), so no single model carries the result. The spread is still wide enough that what E74 supports is "better than chance on models and tasks it has not seen", not a particular AUC to two decimals.

Nearly a quarter of model pairs reverse their ranking when only the protocol form changes, on a suite we did not write, and the statistic predicts which ones at well above chance.

And it still does not forecast size. E42's second locked prediction was that the statistic would also predict the held-out contrast *magnitude*. It was refuted on our suite: Spearman -0.0007, which is no relationship at all. It is refuted here too: Spearman -0.1463. The trigger tells you to look. It does not tell you what you will find, on either substrate. (An earlier version of this paragraph printed -0.056 and $+0.073$. Those were pre-correction values that no released file holds; see App. AX, C9.)

What this does not settle. Predicting *which* comparisons reverse is a different question from whether paying for a probe is repaid, and §AB.3 finds no demonstrable value for κ on the second. Both statements are in this paper and they are about different decisions.

AT WHAT THIS INSTRUMENT CAN AND CANNOT RESOLVE

A measurement is only worth as much as its resolution. This appendix collects the three limits we measured on our own instrument, each pre-registered before it was run. They are the reason the claims in the main text are stated at the scope they are.

The temperature-0 re-run floor (E43). Greedy decoding is not bit-reproducible, so a paired substitution has a floor of disagreement that is not attributable to the swap. The two draws we have are ten weeks apart, so what we measure is that floor plus whatever the serving stack did in between; App. U shows the gap does not explain it, and calls the quantity by its honest name. We had quoted 0.08, taken from single-turn protocol calls. Reseeding the *agent loop* itself gives $\mathbf{0.22}$ outcome and 0.67 action-sequence disagreement over 728 paired episodes, so we correct our own figure upward. The floor sits below the median cross-model per-task flip rate (0.461 over 42 pairs, a $2.1\times$ margin) but *above* the minimum (0.157). Per-task attribution is therefore licensed for a typical model pair and not for the closest ones, which is one more reason the quantity our claims rest on is the aggregate κ rather than any single task's flip. The same reseed gives κ its first test–retest: 0.466 at seed 0 against 0.523 at seed 1, a difference of 0.057 (App. U).

How reliable the recommended probe is (E47). Our practitioner claim rests on a 10-task probe, and our own 12-task probe once mis-assigned 2 of 7 models. Measuring it directly: a 10-task probe reproduces the 52-task interface choice 64% of the time among four protocol forms and 83% among two on the permissive substrate the headline uses (76% and 91% under strict parsing), and when it errs it costs 0.047 and 0.025 held-out success respectively. An earlier version of this sentence gave $\sim 70\%$ beside 91%, which pairs a figure matching neither substrate with one from the strict substrate. Set against the $+0.103$ the probe buys, the price of its mistakes is about a third of its

benefit. Measuring pays not because the probe is accurate but because its errors are cheap. Per-model reliability tracks the gap between the best and second-best interface (Spearman $\rho{=}+0.93$): a probe cannot resolve interfaces it cannot distinguish, and precisely then the choice barely matters (App. AA).

One component axis looked saturated, and the cause was ours (E39/E48). The five-component profile describes where drift appears. We reported that it does not separate protocol forms, because the profile looked nearly identical across interfaces on *two* structurally different task suites (cosine 0.96 and 0.98 as then measured). We diagnosed the task suite, built a second suite with multi-field typed arguments, and **refuted our own diagnosis**. We then concluded the cause was the metric — that normalized edit distance over argument values saturates as soon as two runs diverge at all — and that was **also wrong**. The cause was our *alignment* of the calls being compared, in two successive defects (App. H). Corrected, the axis is nowhere near its ceiling and the *full* profile is interface-dependent on the live suite (cosine 0.87, pre-registered threshold 0.90) while remaining invariant in degree on the heterogeneous suite (0.940). That is not a rescue, and we do not report it as one: E39's third locked prediction asks whether the dependence survives dropping the *outcome* component, and it does not (0.919 without outcome, one dominant component instead of two). So the original negative conclusion stands — the process axes do not separate protocol forms — and what the two corrections changed is only the reason. We leave this entry in the reliability appendix because it is the clearest instance of the failure mode this paper keeps hitting: a negative result that was really an instrument defect, published for one revision cycle before the instrument was tested on inputs we had not thought to construct. An argument-drift measure that degrades gracefully is the concrete next step (App. H).

Why these belong in the paper. Each of the three began as a prediction we expected to confirm and did not. The floor was higher than we had published, the probe is less accurate than the payoff implies, and the component profile's *process* axes do not discriminate on either suite — which is what E39's third locked prediction establishes, after two corrections to our own argument estimator first made the profile look invariant for the wrong reason and then briefly made it look discriminative for an incomplete one (App. H). Reporting them is what makes the main-text scope statements ("for free-text protocols", "the aggregate κ", "describes where drift appears") honest rather than decorative.

Two scope conditions moved here from the main text. *(i)* κ depends on the output-token budget, itself part of the harness: the same models, tasks and forms give 0.49 at 400 tokens and 0.31 at $4{,}000$ (App. G). A more generous budget lets a model recover from an awkward protocol, so raising it is a cheap partial remedy. Every κ we quote is labelled with its harness *and its collection date* (App. E).

(ii) κ is not significant in a pre-registered second task family ($\kappa_B{=}0.06$, App. AN) and is 0.005–0.10 on SWE-bench scaffolds. An interaction needs room to appear, so it shrinks when every interface works, or when the harnesses differ little in what they ask the model to emit: family B is a wiki-lookup suite where one protocol suffices, and the SWE-bench scaffolds differ in tooling rather than response format.

AU FAITHFULNESS OF THE REAL-LOG ACTION NORMALIZATION (E35)

Limitation T5 previously conceded that our normalization of heterogeneous real logs into the canonical six-token vocabulary was validated only indirectly. We now measure it. Reading the implementation first showed the normalization is two layers, and only the second is inferential: *extraction* takes the literal function name from a WebArena action string `fn(args)` and `function.name` verbatim from a τ-bench tool call, flagging errors by the environment's own `Error:` prefix, whereas *classification* maps log events into the six tokens. Extraction is therefore verifiable rather than estimable, and we verify it exhaustively; classification we audit against an independent labeler.

Extraction is exact. Across all $1{,}032$ WebArena action steps in the 120 parsed trajectories the regex falls back to the `unknown` token 0 **times**; the realized vocabulary is 11 tokens (`click` 402, `fill` 375, `noop` 157, `send_msg_to_user` 52, ...). For τ-bench, our error flag fires on 923 of $14{,}285$ tool responses (6.5%); of the $13{,}362$ unflagged responses, *none* carries error semantics

(they reduce to JSON payloads, empty strings, bare numbers, and 64 distinct success strings, all inspected). That exactness is a property of the benchmark—τ-bench prefixes every error uniformly—not of our heuristic, so it would need re-verifying on a new log source.

Classification: κ=0.79, with the disagreement in one place. We sampled 200 real events stratified across the two benchmarks (seed 42) and had an independent labeler (`deepseek-chat`, independent of `gpt-4o`, `sonnet-3.5/3.7` and `llama-3.3-70b`, which produced these logs) assign a class from the same codebook, seeing the raw event and the previous step's error state but never our label. Raw agreement is 0.870 and Cohen's $\kappa = 0.788$, confirming the pre-registered $\kappa \geq 0.75$. The aggregate hides the structure, so we localize it: agreement is $100/100$ **on WebArena** and $74/100$ **on τ-bench**, and 21 of the 26 disagreements are a single confusion in which we say `plan` and the labeler says `answer`. Per class, `observe` (F1 0.990) and `tool_call` (F1 0.993) are near-perfect, while the two classes involved in that confusion are not: our `plan` precision is 0.038 (26 labels, 1 agreed) and our `answer` recall is 0.087 (the labeler finds 23, we caught 2). We report those rather than only the two easy classes, since they are what located the defect. `recover` is unmeasurable here—the sample was stratified by source, not by class, and `recover` is too rare to appear.

Closing the class-coverage gap. Because a source-stratified sample cannot reach `recover` (0.36% of the corpus), we release a *class*-stratified blind human annotation form (50 events, 10 per class over the five classes the corpus contains) together with its scoring script, under `to_human/annotation/`. It is deliberately harder than a random draw—the rare, label-ambiguous classes are over-represented—so its agreement would not be comparable to the random-sample figures above. We report it as an artifact provided for independent checking: no human annotation has been collected, and no number from it is claimed anywhere in this paper.

The cause is a codebook gap. τ-bench is tool-agent-*user*: the agent converses with a simulated user mid-episode. Our vocabulary was designed for tool-use agents and has no token for an utterance to the user, so we folded every content-only assistant message into `plan`. Neither label is right—the vocabulary is missing a dialogue act. WebArena is unaffected because its vocabulary *is* the environment's action space, which maps `send_msg_to_user` and `report_infeasible` to `answer` explicitly; that is why agreement there is perfect.

Repairing it, and bounding what it cost (E37). We re-map a content-only assistant message to `answer` when the conversation returns to the user afterwards and leave it as `plan` when the agent continues working—message ordering only, no model, no new token. We deliberately did not widen the six-token vocabulary, which is shared by the simulator, all five component metrics, and the explicitly-six-token granularity sweep of App. AH. The repair recovers almost exactly the disagreement: τ-bench agreement rises $0.74 \rightarrow 0.94$ and overall Cohen's κ $0.788 \rightarrow 0.95$, with WebArena unchanged at 1.00—evidence that we had identified the cause and not merely a symptom.

Re-running the decomposition under both mappings then bounds what the defect cost. It was *relatively* large, accounting for 59% of the airline trajectory component ($0.0064 \rightarrow 0.0026$) and 67% of retail's, but *absolutely* negligible, because that component is ≤ 0.006 while `tool_args` is 0.09 (this comparator read 0.94 in an earlier revision, the naive flat-index value App. H describes as a bug; the bound holds against either): the largest shift in *any* reported quantity is $\mathbf{0.0038}$, and the outcome, tool-selection, tool-argument and recovery components and both outcome-flip rates move by exactly zero. We report the refined mapping throughout.

We also correct our own reasoning here. We first argued from the fact that the mislabelled class is model-asymmetric (24.3% of `gpt-4o`'s steps versus 18.7% of `sonnet`'s on airline) that the bias could not cancel in a paired estimator, and inferred that it mattered. The direction was right and the magnitude was not: measured directly it is ≤ 0.004. *Asymmetry establishes that a bias exists, not that it is material.* τ-bench's demonstration-only status therefore rests where it did, on the independent noise-floor argument of App. AK.

What this does not establish. The labeler is an LLM, not a human annotator, so this is a consistency check against an independent reading of the same codebook—not human ground truth—and it cannot detect an error our rules and the labeler share. The pre-registered class-level prediction (that

`recover` would be weakest) is also **refuted**: `recover` is too rare to be measurable in this sample, and the weak classes are the τ-bench dialogue acts.

AV DOES THE TRIGGER BEAT THE CHEAP ALTERNATIVES? (E83, CONFIRMATORY)

A reviewer asked the fair question. We show the reversal trigger has external transfer AUC, but we never checked it against the signals anyone would try first. If "the two models are close" does as well, the statistic is decoration.

Same folds, same labels, only the score changes. 13 models $\times$ 6 protocol forms on the external suite, leave-one-form-out, 468 observations of which 109 are reversals — the identical setup to App. AS.1. Each signal is a function of the same training-fold gap vector d.

signal	what it is	AUC	leave-one-model-out
trigger	$\mathrm{std}(d)/(\mathrm{std}(d)+\lvert\overline{d}\rvert)$, our locked statistic	$\mathbf{0.718}$	$[0.674, 0.748]$
reversal rate	share of training folds whose sign differs from $\mathrm{sign}(\overline{d})$	0.708	$[0.665, 0.744]$
closeness	$-\lvert\overline{d}\rvert$: "the two models are close", and nothing else	0.662	$[0.628, 0.738]$
cell spread	$\max(d)-\min(d)$	0.624	$[0.541, 0.669]$
raw spread	$\mathrm{std}(d)$	0.619	$[0.531, 0.660]$

What the numbers say, and what they do not (E86). All three of E83's locked predictions were confirmed on the point estimates: the trigger beats both spread signals, beats closeness, and every signal beats chance. Then a reviewer asked the obvious next question — what is the *uncertainty* on those differences? — and the answer changes the conclusion.

The 78 pairs come from 13 models, so the model is the resampling unit. Dropping each model in turn and recomputing both AUCs on the remaining twelve:

difference	point	jackknife 95%	model bootstrap 95%
trigger − reversal rate	$+0.010$	$[-0.117, +0.138]$	$[-0.021, +0.163]$
trigger − closeness	$+0.056$	$[-0.063, +0.175]$	$[-0.026, +0.139]$
trigger − cell spread	$+0.094$	$[-0.139, +0.327]$	$[-0.051, +0.315]$
trigger − raw spread	$+0.099$	$[-0.143, +0.341]$	$[-0.050, +0.331]$

Every interval contains zero. With 13 models this design cannot separate the trigger from *any* of the cheap signals, including the two whose point estimates it beats by 0.09. Our locked P1 predicted the tie with reversal rate and said, before running, that the paper would then have to say "tied" rather than "wins". P2 predicted the trigger would separably beat closeness and was **refuted**. Both consequences are in §7.

What does survive. The reviewer also proposed a place the trigger should win if its continuity is doing anything: the reversal rate is *identically zero* whenever no training fold flipped, so on those pairs it cannot rank at all. There are 171 such pairs, carrying 14 reversals. The trigger scores $\mathbf{0.560}$ on them — modest, above chance, and on a subset the cheap count leaves entirely unordered. That is a narrower claim than "it beats the alternatives", and it is the one the data supports.

AW PRE-REGISTRATION LEDGER: EVERY LOCKED PREDICTION AND ITS VERDICT

Every locked prediction in this paper was preceded by a protocol committed to version control, and the git history is the timestamp. Post hoc analyses, such as the grouping of the eight designs into three regimes, are labelled post hoc in the text and are not in this ledger. This table is the complete ledger, generated directly from the released `data/*.json` result files by `experiments/49_ledger/make_ledger.py` rather than transcribed, so it cannot drift from

the results it summarises. The generator fails if any prediction node in the data lacks an entry, so the ledger cannot silently become a flattering subset.

Of 215 distinct locked predictions, **67 were refuted**, 127 confirmed, 1 split across substrates, and 20 were validity gates rather than predictions — **4 of those gates failed**, among them the third rung of the frozen replication. Refuted predictions are reported as refuted in the body of the paper, with the fallback each protocol locked in advance, and a failed gate is reported as a failure.

E127's five rows are read beside three sentences of §5.2. Copying the source configuration's regime matched on 4 of 4 suite designs and on the BFCL design. On the source designs, whose regimes were already known, the same rule matched the regime read from the held-out tasks of the same partition on 63% to 92% of random partitions on our suite and on 55% on BFCL; copying the known regime matched on 78% to 99% and on 81%. The held-out regime is read from one table of 32 / 20 tasks; on the source designs this one-table reading differed from the design's averaged regime on up to 22% of partitions.

E129's four rows (App. R). P1 was expected to hold: on random splits of the older BFCL tasks it held on 90% of splits. P2 cannot be told apart from copying P-bfcl's label, which also matched. This is one design, so a match or a miss is one count, not a rate. P3a and P3b are one comparison: the two selectors made the same choices.

E135's three rows (App. N). No interval method met its coverage band in planning, so P1 was scored by the exact sign-flip test over the 24 template means. P2, the mechanism prediction, missed: post hoc, the model's json error also appeared on tasks that need no calc call. P3 applies the paper's own label rule to the new tasks.

E136's three rows (App. P). The parser and the provider changed together, so a hit does not say which of them moved a gain. The probe and the pooled-history rule were not rerun.

One caveat covers a whole block of these rows. Every prediction from E57–E58 and E60–E66 was measured on Harness-Bench through a JSON parser that rejected valid calls whose arguments carried a JSON payload (App. AX, C8). Those verdicts are shown because they were locked and measured and it would be worse to delete them, but they are not evidence about the world until that collection is redone behind the repaired parser. The author-suite rows are unaffected: no JSON-form call on that suite carries a brace.

2 of those verdicts (marked †) changed after we corrected an instrument *post hoc*, i.e. after the data were seen. Each is counted here under the verdict as originally locked and measured, never as a confirmation. A post-hoc confirmation is an exploratory re-analysis with a repaired instrument, and folding it into the confirmed column would turn this table into precisely the flattering summary it exists to prevent. The corrected verdict is shown beside the locked one so a reader can see both. What makes us willing to trust the correction rather than merely the outcome it produced: the defect is recognisable without reference to any verdict (two runs issuing the same calls in a different order scored maximal argument drift), the repair was prompted by a reviewer question about repeated calls rather than by a verdict, its regression tests were written from the defect's structure, and the same repair moved other headline numbers *against* us — synthetic brittleness AUC fell $0.855 \to 0.830$ and its behavioural-only variant $0.671 \to 0.632$. An outcome-selected change would not have shipped those.

Exp.	prediction	verdict
E103	Freedman–Lane rejects at most 0.10 of binary $7\times4\times52$ nulls	confirmed
E103	its Wilson interval contains the nominal 0.05	confirmed
E103	the same holds at the ladder size $5\times2\times52$	confirmed
E104	its 95th percentile on the largest interaction cell is at most 0.20 outcome points	confirmed
E104	its power against a 0.05-point interaction is below 0.50	confirmed
E104	measuring the incoming model beats keeping the format on this grid (no advantage shown; the locked run probed 6.3 tasks per draw, not 10)	**refuted**

Exp.	prediction	verdict
E104	the design has power ≥ 0.80 against a 0.20-point interaction	confirmed
E104	the frozen L1 task-bootstrap 95th percentile on raw κ is below E61's 0.4596	confirmed
E106	κ ranks it at least as well as counting sign flips (it ties)	**refuted**
E106	κ ranks what probing saves in the right direction	confirmed
E106	designs with $\kappa < 0.10$ save under 0.02 by probing	confirmed
E106	the sign survives leave-one-design-out	confirmed
E108	each rung's pooled outcome is within 0.10 of the earlier collection's	**refuted**
E108	every rung passes its validity gate on the re-run	confirmed
E108	the three rungs come out in the same order as in the earlier collection	**refuted**
E108	today's rung-1 κ is inside the earlier collection's interval, same ten models	confirmed
E109	the corrected κ's median is 0 under the null at every scale	confirmed
E109	the median κ rises monotonically as the denominator shrinks	**refuted**
E109	the raw κ's 95th percentile exceeds 0.20 at a zero model effect	confirmed
E109	type-I stays ≤ 0.10 at every model-effect scale	confirmed
E111	every rung passes its validity gate on the one-day re-run	confirmed
E111	held-out κ is inside the development half's interval	confirmed
E111	one day moves κ less than the twelve-day interval did, at every rung	**refuted**
E111	one day moves pooled success by under 0.05 at every rung	confirmed
E118	pooled success moves by less than 0.05	confirmed
E118	the 28 cell means rank-correlate across the two runs at $\rho \geq 0.80$	confirmed
E118	the repeat's κ is within 0.10 of 0.563	confirmed
E118	the repeat's task-blocked κ lies inside the reference interval $[0.4663, 0.6644]$	confirmed
E119	it still covers at least 90% of the time at every feasible non-zero interaction	confirmed
E119	its Wilson interval contains the nominal 0.05 in at least 18 of 24 regimes	confirmed
E119	power against a realized 0.10 on the $5\times2\times31$ design is at least 0.80 (it is 0.56)	**refuted**
E119	the realized interaction equals the declared one, unlike E104's clipped generator	confirmed
E119	the reported test rejects at most 0.10 of grids in every regime of an outcome-scale null	confirmed
E119	the same percentile on the raw ratio covers at least 90% of the time (it falls to 0.000)	**refuted**
E119	with no interaction, the 95th percentile of $\max\lvert\gamma\rvert$ covers at least 95% of the time	confirmed
E120	κ adds something beyond the reversal rate: the κ gate has lower regret than the reversal gate at the same probe rate, upper end below 0	**refuted**
E120	κ carries decision information: the κ gate has lower regret than a random gate at the same one-half probe rate, upper end below 0 (measured $+0.002$, $[-0.003,\ 0.009]$)	**refuted**

Exp.	prediction	verdict
E120	X0 gate: WorkBench's oracle accepts all 690 gold lists, every gold action rendered into each of the four forms parses back and scores correct (2,760 lists, 3,788 calls), and the smoke test ends with zero API errors on every (model, form)	gate passed
E120	X1 validity gate: pooled correct on the 568 action-required tasks is in $[0.10, 0.90]$ and the no-parsed-call rate is below 0.15	gate passed
E120	completeness gate: at least 60 templates with all $k \times 6 \times 4$ cells score normally, 40 to 59 are labelled REDUCED, fewer than 40 void X2–X10	gate passed
E120	no single format serves every model: at least 2 forms are some model's held-out best, and the inherited format is the incoming model's best in under half of the 30 swaps (two forms are; inherited best in 0.605 of swaps)	**refuted**
E120	probing beats keeping the incumbent's format: regret difference $\geq$ 0.03 with a template-bootstrap lower end above 0 (measured $+0.023$, $[0.012, 0.043]$: lower end above 0, below the 0.03 line)	**refuted**
E120	probing beats the strong fixed default (pooled history of the other five models): $\Delta \geq 0.03$ with lower end above 0 (measured -0.011, $[-0.019, 0.004]$)	**refuted**
E120	the 40-episode probe repays itself within 2,000 deployment episodes for at least 4 of the 6 incoming models	**refuted**
E120	the deployable rule (probe iff $\tilde{\kappa} \geq 0.10$) beats the free default by 0.01 while saving a quarter of the probe budget, or beats always-probing by 0.01 at no higher cost	**refuted**
E120	the form chosen on success is not less safe: its held-out harm rate is at most 0.02 above the fixed default's, upper end below 0.02	confirmed
E120	the free fix beats inheriting: the pooled-history default has regret at least 0.02 below keeping the incumbent's format, lower end above 0	confirmed
E127	P-4k (4k-token budget, 7 new models): the regime read from 20 calibration tasks, realized adaptation gain, is the regime of the 32 held-out tasks collected afterwards	confirmed
E127	P-bfcl (BFCL, 7 new models): the regime read from 20 calibration tasks, realized adaptation gain, is the regime of the 20 held-out tasks collected afterwards (held-out: limited adaptation opportunity)	**refuted**
E127	P-strict (strict parser, 7 new models): the regime read from 20 calibration tasks, realized adaptation gain, is the regime of the 32 held-out tasks collected afterwards	confirmed
E127	P-strong (strict parser, 4 new strong models): the regime read from 20 calibration tasks, limited adaptation opportunity, is the regime of the 32 held-out tasks collected afterwards	confirmed
E127	P-typed (typed arguments, 7 new models): the regime read from 20 calibration tasks, opportunity offset by selection error, is the regime of the 32 held-out tasks collected afterwards	confirmed

Exp.	prediction	verdict
E129	on 64 tasks of 6 BFCL task families no design had used, the frozen 10-task probe is not shown to beat the deployable shared default (+0.06 turn-accuracy points, 95% interval $[-0.65, 0.78]$)	confirmed
E129	the E127 regime rule read on the 4 calibration families gives the regime of the 6 held-out families (both limited adaptation opportunity; copying P-bfcl's label also matched)	confirmed
E129	the history-shrunk probe ($\lambda = 20$) has held-out regret no higher than the plain probe (2.10 against 2.31; it made the same choices as the tie-breaking probe)	confirmed
E129	the probe with history tie-breaking has held-out regret no higher than the plain probe (2.10 against 2.31 points)	confirmed
E135	measured on the new tasks, the strict-parser copy's regime label, realized adaptation gain, holds	confirmed
E135	on 96 tasks from 24 new task templates, the frozen 10-task probe beats the pooled-history rule (+3.9 points; exact one-sided sign-flip $p = 0.0128$, since no interval was locked)	confirmed
E135	qwen3-30b-a3b's xml-over-json gap is at least 0.30 larger on the new tasks that need a calc call than on those that do not (gaps 0.19 and 0.25)	**refuted**
E136	gpt-4.1-nano's react gain over json does not fall below its first-run 7.3 points (25.0)	confirmed
E136	qwen3-30b-a3b's xml gain over json stays above 10 points (29.2)	confirmed
E136	with a corrected parser and one fixed provider per model, E135's frozen choices gain over json on its 96 tasks (+7.7 points; first run +3.9)	confirmed
E32	the temperature-0 floor is negligible	**refuted**
E35	tagger $\kappa \geq 0.75$; weakest class `recover`	confirmed
E36	κ at b=4 lies inside the b=2 CI	**refuted**
E36	κ is stable across harness-factor size	**refuted**
E36	every interface is somebody's optimum	confirmed
E36	the EMS correction becomes informative at b>2	**refuted**
E37	repairing the codebook raises agreement past 0.90	confirmed
E37	the dialogue-act confound shifts nothing > 0.05	confirmed
E38	at least 3 interfaces remain somebody's optimum	confirmed
E38	hermes ReAct success exceeds 0.20	**refuted**
E38	strict κ stays inside the permissive CI	confirmed
E38	strict parsing cuts no-tool ReAct below 0.15	confirmed
E39	P3: interface-dependence survives dropping the *outcome* component	**refuted**
E39	component profiles differ by interface	**refuted** † (confirmed after fix)
E39	the dominant component changes by interface	**refuted** † (confirmed after fix)
E40	κ on protocol adherence is large	confirmed
E40	an irreducible competence layer survives conditioning	confirmed
E40	both layers are individually significant	confirmed
E41	a 10-task probe beats the best fixed interface	confirmed
E41	the payoff is monotone in probe size	confirmed
E42	pooled per-task prediction beats a marginal-rate baseline	**refuted**
E42	within-pair brittleness prediction beats chance	confirmed
E43	κ reproduces across a reseed	confirmed

Exp.	prediction	verdict
E43	the floor sits below cross-model flip rates	confirmed
E43	the in-domain floor exceeds the quoted 0.08	confirmed
E44	κ holds or rises under constraint	**refuted**
E44	gate: the gateway really enforces the schema	gate passed
E44	schema support is not universal across models	confirmed
E44	the interaction survives constrained decoding	confirmed
E45	but stays above 0.02 with a CI excluding 0	**refuted**
E45	the cause is a harness main effect	confirmed
E45	the payoff shrinks under constrained decoding	confirmed
E46	equal constraint removes the harness main effect	confirmed
E46	gate: both constrained columns reach adherence 0.95	gate passed
E46	the interaction survives between two equally constrained protocols	**refuted**
E46	the payoff returns between equally constrained protocols	**refuted**
E47	agreement at $K{=}10$ is below 0.75 (2 substrates)	**split**
E47	held-out regret at $K{=}10$ is below 0.05 (2 substrates)	confirmed
E47	our own 12-task mis-assignment was an ordinary draw (2 substrates)	confirmed
E47	probe agreement rises with probe size (2 substrates)	confirmed
E48	$\kappa > 0$ on a third task family	confirmed
E48	`tool_args` no longer dominates everywhere	confirmed
E48	gate: the suite is solvable and parses	gate passed
E48	profiles differ by interface on heterogeneous arguments	**refuted**
E48	the dominant component varies by interface	confirmed
E49	gate: no ceiling saturation, parsers hold at 4000 tokens	gate passed
E49	reasoning models are *less* harness-sensitive	**refuted**
E49	reasoning models are stronger at matched budget	confirmed
E49	still no interface to default to, for reasoning models	confirmed
E50	gate: enough regressions per swap to compare rules	gate passed
E50	kappa-guided re-testing beats a difficulty heuristic	confirmed
E50	kappa-guided re-testing beats random	**refuted**
E50	kappa-guided re-testing beats re-testing what the incumbent passed	**refuted**
E51	≥ 2 distinct best forms on BFCL	confirmed
E51	$\kappa > 0$ on an external benchmark (BFCL v3)	**refuted**
E51	BFCL κ is smaller than on our own suite	confirmed
E51	gate: BFCL tasks are solvable and parse	gate passed
E52	frontier prompts are not copied verbatim	gate passed
E52	gate: frontier models solve and parse on our suite	**gate failed**
E57	the external Harness-Bench grid clears the parse and difficulty gate	**gate failed**
E58	kappa is present on an external suite we did not build	confirmed
E58	kappa is smaller on the external suite than on ours	**refuted**
E58	no single interface is a safe default on the external suite	confirmed
E58	the screened-pool external grid clears the parse and difficulty gate (2 substrates)	gate passed
E60	gate: $>$ 0.90 of constrained steps parse (*misspecified*; charged our token ceiling to the gateway)	**refuted**
E61	κ falls monotonically across the three rungs on an external suite	confirmed
E61	absolute S_{MH} falls monotonically across the same three rungs	confirmed

Exp.	prediction	verdict
E61	gate: $< 2\%$ of constrained replies violate the schema	confirmed
E61	gate: all three external rungs clear the parse and difficulty gate	gate passed
E61	instrument check: our token ceiling carries less interaction than the outcome	**gate failed**
E62	no single model produces the external ordering	confirmed
E62	the external ladder still falls when all three rungs use the same 27 tasks	confirmed
E62	the ladder is strictly decreasing in $\geq 90\%$ of paired draws	confirmed
E62	the paired L1$-$L2 drop excludes zero	**refuted**
E62	the paired L2$-$L3 drop excludes zero	confirmed
E63	abstaining costs no accuracy against plain per-model measurement	**refuted**
E63	every probe-based rule beats picking a format at random	confirmed
E63	the abstaining rule beats a per-form ranking	confirmed
E63	the abstaining rule beats gating on κ	**refuted**
E63	the abstaining rule beats picking the least-churn format	**refuted**
E63	the abstaining rule beats the best fixed format on held-out regret	confirmed
E64	κ falls across all three rungs on fresh external cells	**refuted**
E64	BOTH adjacent drops have paired intervals excluding zero	**refuted**
E64	a 4000-token budget cuts truncation below 3% on every form	confirmed
E64	absolute $S_{M\mathcal{H}}$ falls across the same three rungs	**refuted**
E64	gate: $< 2\%$ of constrained replies violate the schema, on fresh cells	confirmed
E64	gate: all three fresh external rungs clear parse and difficulty	gate passed
E64	instrument check: truncation carries less interaction than the outcome	**refuted**
E65	gate: both payload strata clear parse and difficulty	gate passed
E65	malformed replies are commoner where payloads are large	**refuted**
E65	the first rung is larger on small-payload than large-payload tasks	**refuted**
E65	the first rung returns on small-payload tasks	**refuted**
E66	a 5-model pool does not manufacture a large first rung on average	confirmed
E66	our first rung is rare among 5-model subsets of a suite that has none	confirmed
E66	the first-step estimator is not meaningless at $n{=}5$	confirmed
E67	κ falls across all three rungs on a correctly parsed external collection	**refuted**
E67	BOTH adjacent drops have paired intervals excluding zero	**refuted**
E67	a 4000-token budget cuts truncation below 3% on every form (repaired parser)	confirmed
E67	absolute $S_{M\mathcal{H}}$ falls across the same three rungs	**refuted**
E67	gate: $< 2\%$ of constrained replies violate the schema (repaired parser)	confirmed
E67	gate: all three external rungs clear parse and difficulty (repaired parser)	gate passed

Exp.	prediction	verdict
E67	instrument check: truncation carries less interaction than the outcome	**refuted**
E68	keeping the incumbent's format is worse than one fixed default	confirmed
E68	probing beats a least-churn rule on real ordered swaps	confirmed
E68	probing the incoming model beats keeping the incumbent's format	confirmed
E68	the inherited-format penalty tracks how badly it suited the incumbent	confirmed
E69	≥ 2 distinct best forms on the repaired external grid	confirmed
E69	$\kappa > 0$ against a mean-matched null on the repaired external grid	confirmed
E69	external κ is smaller than ours, as *point estimates* (E71 shows the difference is not interval-separated)	confirmed
E69	gate: the repaired four-form external grid clears parse and difficulty	gate passed
E70	keeping the incumbent's format is still worse than one fixed default	confirmed
E70	probing beats a least-churn rule	confirmed
E70	probing beats keeping the format, under a realistic incumbent policy	confirmed
E70	probing beats one fixed default	confirmed
E71	the our-minus-external κ difference excludes zero	**refuted**
E71	the point difference in κ across substrates is at least 0.15	confirmed
E71	the same difference in absolute $S_{M\mathcal{H}}$ excludes zero	**refuted**
E72	a κ gate beats always-probing at some probe price	confirmed
E72	that advantage over the coin is separated from zero	**refuted**
E72	the κ gate beats a coin matched on its own probe rate	**refuted**
E73	guard: the margin FAILS at the unconstrained rung, so it means something	**refuted**
E73	the fully specified rung is inside a 0.02 margin on our suite	confirmed
E73	the same holds on the external suite	**refuted**
E74	gate: enough rankings reverse externally for the test to mean anything	confirmed
E74	the re-validation trigger predicts reversals on unseen models and tasks	confirmed
E74	the statistic still fails to forecast held-out effect size	confirmed
E82	the task main effect exceeds 10% of $\mathrm{SS}_{\text{total}}$	confirmed
E82	the task-blocked κ exceeds 0.20	confirmed
E82	the task-blocked κ is at least the two-way κ	confirmed
E82	the task-clustered interval excludes 0 (weak: 49% false-positive rate)	confirmed
E83	every candidate signal beats chance	confirmed
E83	the trigger beats $-\lvert\bar{d}\rvert$ alone	confirmed
E83	the trigger beats raw spread and cell spread	confirmed
E85	its type-I error is near nominal on no-interaction grids	confirmed
E85	the interaction survives a null that keeps every nuisance term	confirmed
E85	the nuisance-preserving null centres F near 1	confirmed
E86	the trigger ranks the pairs the reversal rate cannot	confirmed
E86	the trigger separably beats $-\lvert\bar{d}\rvert$	**refuted**
E86	the trigger's edge over the reversal rate spans zero	confirmed

Exp.	prediction	verdict
E88	a second permutation scheme calibrates	confirmed
E88	the calibrated test still rejects on the real grid	confirmed
E88	the reported permutation test is liberal at nominal 0.05	**refuted**
E92	κ falls strictly across the three rungs on five new models	**refuted**
E92	$\kappa(\mathrm{L3}) < \kappa(\mathrm{L1})$ on the new models (both were 0)	**refuted**
E92	at least one class is non-monotone at the middle rung	confirmed
E92	both adjacent drops in κ are separated from zero	**refuted**
E92	every rung passes the validity gate (L3 failed: no-parsed-call 0.18, one model exits at step 0)	**gate failed**
E92	leave-one-model-out SE on $\kappa(\mathrm{L1})$ exceeds 0.10 (it was 0.0)	**refuted**
E92	no rung's interaction is manufactured by a form-dependent failure	**refuted**
E92	the absolute interaction S_{MH} falls strictly across the rungs	**refuted**
E92	the endpoint holds in ≥ 6 of 8 workflow classes (it held in 2)	**refuted**
E92	the gateway enforced the schema on the constrained forms	confirmed
E92	the larger token budget removed truncation	confirmed

AX CORRECTIONS TO OUR OWN INSTRUMENTS, IN ONE PLACE

The main text states results; this appendix states what we got wrong on the way to them, so a reader can audit the chronology without it interrupting the argument. Each entry names the defect, how it was found, and what moved. Numbers on the *current* side of a correction are reproduced by `./verify.sh`. The pre-correction figures are not, and cannot be: no gate recomputes a measurement we have retired, so those are reproduced only by checking out the commit the ledger names for that row. The earlier version of this sentence said every number below was reproduced, which was not true of roughly half of them.

C1 — The tool-argument estimator was wrong twice. Argument drift first compared calls by *flat index*, so as soon as two trajectories emitted different numbers of calls it compared call i of one against call i of another and dropped the tail; the flat version read 0.90–0.98 where the aligned one read 0.16–0.65. The replacement aligned calls by *tool name* alone, which ties whenever a name repeats: two trajectories issuing the same two `db` calls in a different order scored *maximal* argument drift, and the statistic was asymmetric. What we now use pairs calls by minimum-cost assignment within each tool name. Consequences, all propagated: component specificity improved $10.3\times \rightarrow 11.4\times$; synthetic brittleness AUC *fell* $0.855 \rightarrow 0.830$ and its behavioural-only variant $0.671 \rightarrow 0.632$; interaction recovery fell $r{=}0.974 \rightarrow 0.921$; τ-bench aligned arguments fell $0.28/0.16 \rightarrow 0.09/0.05$. That the same repair moved several headline numbers *against* us is the reason we regard it as a correction rather than a convenience.

C2 — Two pre-registered verdicts flipped, and we do not count them. C1 flipped E39's P1 and P2 from REFUTED to CONFIRMED. That is a post-hoc instrument change in our favour, so the ledger (App. AW) counts both rows under their *locked* verdict and tallies the flip in a separate column. We take no pre-registration credit for either. E39's *third* locked prediction, which we had not been reporting at all, then settled the question against us: dropping the outcome component returns the cross-interface cosine to 0.919 and one dominant component, so the effect is outcome-driven and no claim rests on the process axes (App. H).

C3 — Three nulls, two of them invalid. For the trajectory-level estimand we first permuted harness labels independently, which made 75% of null cells compare endpoints under *different* har-

nesses; it returned p=0.001, the right answer for the wrong reason. We then resampled residuals in whole-pair blocks, which *relocates* a pair's interaction rather than removing it and has no power at all — p=0.302 against a synthetic injected κ=0.477. Only the third version, redrawing residuals independently per cell, both removes the interaction and detects one when present. We now report a null's false-positive rate *and* its power on every use, and treat a null failing either as unusable whatever p-value it produces (App. AF).

C4 — Unfalsifiable locked tests. Two pre-registered predictions used "the bootstrap CI excludes 0" on the *raw* ratio. Sums of squares are non-negative, so at our design that interval cannot reach zero: simulated under a pure null the raw ratio has a 2.5th percentile of 0.206, and the test would have read CONFIRMED whatever the data said. Both are reported as VOID rather than deleted, and significance now rests on mean-matched no-interaction nulls (App. J).

C5 — A stitched grid, and stale results files. The headline was once computed over a grid merged from several collections, in which 47 cells disagreed and were silently resolved by directory order; it is now a single-provenance collection whose loader *refuses* to resolve a conflict (App. E). Separately, a replay gate found nine committed results files that differed from a fresh run, including the interaction-recovery figure quoted in the abstract. Both classes are now gated: `verify.sh` replays every analysis whose code changed, and the number audit exits non-zero on any mismatch — it previously printed its mismatches and exited 0. One table outlived the retirement: until 19 September, App. AP printed the stitched grid's per-model success under the four forms (`gpt-oss-120b` 0.385/0.288/0.385/0.269, `gpt-oss-20b` 0.250/0.308/0.327/0.096, `llama-3.1-8b` 0.250/0.000/0.000/0.288, `llama-3.3-70b` 0.481/0.635/0.442/0.538, `mistral-small-24b` 0.269/0.712/0.346/0.923, `phi-4` 0.077/0.404/0.096/0.154, `qwen3-14b` 0.750/0.327/0.596/0.635 for JSON/ReAct/XML/pythonic) under prose written about the single-provenance grid; a reviewer noticed. The table now prints the single-provenance cells, and these are the retired ones.

C6 — A ladder that mixed estimators. The compliance ladder briefly read $0.47 \to 0.27 \to 0.13 \to 0.00$, in which the third rung was a *raw* ratio while the others were bias-corrected. It is 0.11 bias-corrected, and the ladder is now also reported in absolute variance (§5) so it does not depend on a ratio at all.

C7 — A second ladder that mixed experiments. The payoff ladder beside the κ ladder read $+0.103 \to +0.011 \to +0.000$. Its first rung came from E41's seven-model grid while the second came from E45, so the triple compared different designs. E45 runs *both* of its arms on the same five models and 52 tasks, so the internally consistent triple is $+0.141 \to +0.011 \to +0.000$ and that is what the paper now states. The $+0.103$ figure is still correct for the claim it belongs to — a 10-task probe against the best fixed interface on the headline grid — and is quoted only there. We found this while drawing Fig. 10 from the released files, which is the argument for generating figures from data rather than typing the numbers in.

C9 — Five numbers in this appendix's own mechanism section were stale, and a release guard could not fail. An outside reviewer replayed E40/E42 and found the paper disagreeing with a file that reproduces byte-identically. The E40 table printed task-success κ=0.394 with $[0.300, 0.520]$ while its own paragraph two lines below said 0.501; the file says 0.5009 with $[0.409, 0.603]$. The E42 block printed 21 of 84 reversals, AUC 0.658, p=0.016 and Spearman -0.056; the file says 27, 0.7375, 0.001 and -0.0007. The AUC 0.658 appears nowhere in the released data at all. These are pre-correction values that survived in prose after the data was regenerated — the mirror image of C5, where the file was stale and the prose was right. None of the five was pinned by a canonical check, which is why every gate stayed green; all five are pinned now.

The same review extracted the distributed archive and found `paper/shipped_data_dirs.txt` missing from it. That file is the list the release compares against to detect a data directory that has *vanished*. With it absent the check printed a note, skipped itself, and the release still reported success — the exact failure the check was added to catch, one level up: a gate that cannot fail because its input is not there. The manifest is now shipped, required, and fatal when absent, and `test_release_negative.sh` removes it on purpose and fails if the release still passes.

C10 — E74 said it froze E42's procedure and changed the statistic. E74 tests the re-validation trigger on models and tasks E42 never saw, and its protocol says it runs "E42's procedure unchanged". It did not. E42's locked statistic is a *normalised* volatility, $\mathrm{sd}(d)/(\mathrm{sd}(d) + |\bar{d}| + \varepsilon)$, bounded in $[0, 1]$; E74 used the raw $\mathrm{sd}(d)$, which is unbounded and on a different scale on a different grid. The two were separate copies of one locked definition, which is one copy too many. An outside reviewer found it.

Both call sites now import a single shared function, and E40/E42 reproduce byte-identically under the refactor, so nothing else moved. Re-running E74 with the statistic it said it used raises the trigger AUC from 0.619 to $\mathbf{0.7184}$ and the permutation p from 0.008 to $\mathbf{0.0002}$; the reversal count and rate ($109/468$, 23.3%) do not depend on the statistic and are unchanged. We record it as a correction rather than an improvement because the direction is not the point: the published number came from a procedure the protocol did not describe. The same script also carried E42's pre-correction numbers hard-coded as a comparison row; it now reads them from E42's released file, so the two cannot drift apart again.

Why this appendix exists. Seven of the ten we found ourselves, with our own gates or by re-running our own code; three came from outside review, and each entry says which. Every one is a case where a check that reported green was not actually checking. We list them together because the pattern matters more than any single entry: an instrument that is never tested on inputs its author did not think of will report clean results indefinitely.

C8 — The JSON parser threw away valid calls that carried JSON. A reviewer found 302 complete, schema-compatible calls in our external collection that our own parser had rejected. They were right. The parser looked for a call by matching balanced braces with a regular expression, and a regular expression counts { and } as characters: it cannot tell a brace inside a JSON *string* from one that opens an object. So the moment a model wrote a JSON document into the `content` argument — which is most of what Harness-Bench asks for — the braces in the payload were counted as nesting, the pattern's depth limit was passed, and a valid call was discarded.

```
{"tool":"write_file","args":{"path":"o.json",
 "content":"{\"a\":  {\"b\":  1}}"}}
```

That text is valid JSON. `json.loads` accepts it. Our parser returned nothing.

Measured on the external failures whose whole reply we had stored — the short ones, so this is a lower bound — 43–50% of them were valid calls we had thrown away. The parser now hands each { to a real JSON decoder, which is string-aware and has no depth limit, and five tests pin it. The same mistake was present in two other parsers and both are repaired.

This one could not be fixed by re-reading the transcripts. A rejected call makes the harness tell the model "that was not a parseable tool call", so the episode after that point is not the episode we would have collected. The affected external collections had to be run again. Our own suite is unaffected: of its $4{,}216$ recorded call arguments, 12 contain a brace and all 12 are `xml` or `pythonic` placeholder syntax handled by different code, and no JSON-form call carries one.

AX.1 Was anything sequestered? Yes — and it refuted us.

A reviewer asked which single experiment was run *after* the estimator, the parser, the null and the headline were all frozen. For three review rounds the answer was none: every protocol had been committed before its run, but a commit before a local run proves the plan preceded *that* result, not that it preceded the many earlier chances to change what the headline would be — 50 refutations, two post-hoc flips, three parser repairs, a null rebuilt three times, a retired grid, a re-collection for single provenance.

So we ran the one thing that history cannot substitute for: same tasks, oracle and budget, the three rungs, the analysis script byte-for-byte, four new predictions, hashed and committed with no data in the tree, then collected once on six models absent from the external design's thirteen, two of the six also in our own seven. That is E92 (App. V). It found no interaction to remove at any rung, and its own fallback put that in the abstract. A result nobody had the opportunity to steer turned out to be one we did not want, which is the point of running one.