Correction and Corruption: A Two-Rate View of Error Flow in LLM Protocols

Anonymous authors
Paper under double-blind review

Abstract

Large language models are increasingly deployed as *protocols*: structured multi-call procedures that spend additional computation to transform a baseline answer into a final one. These protocols are usually evaluated only by end-to-end accuracy, which reveals whether they deliver gains on average but gives limited insight into when they help, when they hurt, and whether their behavior transfers under distribution shift or composition. We propose a *paired-outcome measurement interface* for auditing a single protocol step on exact-match tasks. For each instance, the interface records a baseline correctness bit $E_0 \in \{0, 1\}$ and a post-step correctness bit $E_1 \in \{0, 1\}$, with accuracies $p_t := \Pr(E_t = 1)$. This separates *correction*, $E_0{=}0 \to E_1{=}1$, from *corruption*, $E_0{=}1 \to E_1{=}0$, through two conditional rates: the correction rate $c = \Pr(E_1{=}1 \mid E_0{=}0)$ and the corruption rate $\gamma = \Pr(E_1{=}0 \mid E_0{=}1)$. These two rates are sufficient to predict accuracy changes and determine whether a step helps at a given baseline. They also define a reusable empirical interface whose transfer can be tested across seeds, difficulty mixtures, and composed pipelines. We identify three mechanisms by which this interface can fail to transfer. Under **mixture shift**, estimates of (c, γ) pooled across difficulty regimes become biased when calibration and deployment mixtures differ; conditioning on depth identifies a regime variable under which the interface becomes stable and enables predictive transfer, substantially reducing this bias without additional model calls. Under **presentation contamination**, selection protocols can change the measured interface through stable presentation artifacts even when candidate content is fixed. Finally, under **state insufficiency**, the correctness bit alone may not carry enough history for multi-step pipelines to compose predictably; a testable Markov factorization characterizes when composition is valid and identifies where additional state is needed when it is not. When a protocol step passes these diagnostics, it becomes an auditable module: it can be gated by estimated gain, conditioned on difficulty proxies to correct mixture bias, and composed into multi-step pipelines with predictable accuracy. We demonstrate these ideas on synthetic mathematical tasks with controlled difficulty and on GSM8K using observable complexity proxies, where the calibrated interface correctly predicts when protocol steps should be activated or suppressed. A targeted exploratory check on executable Python repair tasks, with unit-test correctness, shows that the same paired audit can be instantiated outside exact-match mathematics and localizes a helper-context corruption channel that an aggregate pass-rate comparison would not identify.

1 Introduction

LLMs are often deployed not as single-shot predictors but as *protocols*: short, structured multi-call interactions that allocate additional compute to transform an initial attempt into a final answer. This shift is pragmatic. Contemporary frontier systems (e.g., GPT-4 and GPT-4o) illustrate both the prevalence of system-level scaffolding and the corresponding evaluation challenge (OpenAI et al., 2024a;b). A single call can fail in ways that are hard to anticipate, and in many applications an uncorrected error is far more costly than an additional interaction step. As a result, practitioners routinely wrap solvers in patterns that condition on auxiliary context, critique and revise, verify and repair, or generate multiple candidates and

select among them. Here we propose a deliberately minimal interface for auditing a single protocol step that supports mechanism-level analysis beyond aggregate accuracy gains.

We use *protocol* in this operational sense: a structured sequence of model calls and, when present, external operations or checks such as tool use, tests, retrieval, verification, repair, or selection. Our unit of analysis is a particular transition within such a protocol, not the agent or system as a whole.

The relevant protocol landscape is broad. Some methods change the *within-model* inference process, for example through chain-of-thought-style elicitation and self-consistency (Wei et al., 2022; Wang et al., 2023). Others introduce explicit multi-call structure: self-feedback and refinement (Madaan et al., 2023), reflection-style loops (Shinn et al., 2023), debate (Du et al., 2024), tool- or trajectory-guided search (Yao et al., 2023b;a), verifier-based selection (Cobbe et al., 2021), and judge-based evaluation and selection (Zheng et al., 2023). These protocols differ operationally, but they share an observable evaluation structure whenever the same item can be scored before and after a step.

The key ambiguity is directional. An interaction step may improve final accuracy because it repairs initially wrong answers, or it may lose accuracy because it corrupts answers that were already correct. Intrinsic self-correction illustrates the harmful side of this tradeoff: Huang et al. (2024) show that asking a model to revise its own reasoning without reliable external grounding can substantially degrade accuracy. By contrast, protocols with structured feedback, verifiers, judges, tools, unit tests, or curated signals can improve performance by increasing repair, reducing corruption, or both (Madaan et al., 2023; Cobbe et al., 2021; Lightman et al., 2024). This motivates a measurement interface rather than a new protocol: given paired pre/post outcomes, measure how probability mass moves between correctness states and ask whether that movement transfers out of sample.

Broad surveys of self-correction and automated correction provide the wider background for these phenomena (Kamoi et al., 2024; Pan et al., 2024). Recent formal work has independently identified closely related transition quantities in self-correction and refinement settings. Yang et al. (2025a) provide the closest prior one-step confidence–critique decomposition: in our notation, retention of initially correct answers corresponds to $1-\gamma$, while repair of initially incorrect answers corresponds to c. Yang et al. (2025b) extend this line to multi-round accuracy curves, and Liu & Meng (2026) formulate repeated self-correction in terms of error-correction and error-introduction rates. We treat these settings as important special cases of a more general transition-level abstraction: the transition need not be self-critique or refinement, but may be verification, repair, selection, tool use, retrieval, or any other protocol step for which paired pre/post correctness labels can be logged. The abstraction separates beneficial error correction from harmful corruption at a specified protocol transition. That separation supports the audit questions studied here: whether the measured transition is stable across seeds, slices, mixtures, and presentations, and when the corresponding protocol step should be used in deployment. For stacked protocols, Section 7 asks when the binary correctness state is sufficient for composing protocol steps.

Formally, let $\mathcal{D}$ be a distribution over tasks with ground-truth answers, and let $E_0, E_1 \in \{0,1\}$ denote correctness before and after a protocol step. A step can change correctness in only two ways: it can *correct* a baseline failure ($E_0{=}0 \to E_1{=}1$) or *corrupt* a baseline success ($E_0{=}1 \to E_1{=}0$). We refer to these two transitions as *correction* and *corruption*, respectively, and summarize them through the *correctness channel*,

$$(c,\gamma) := \big(\Pr_{x\sim\mathcal{D}}(E_1{=}1 \mid E_0{=}0), \Pr_{x\sim\mathcal{D}}(E_1{=}0 \mid E_0{=}1)\big).$$

As we show, these two rates are sufficient to predict accuracy changes, characterize fragility at high baselines, and test whether a step's behavior transfers across seeds, difficulty mixtures, and composed pipelines. Definition 1 later formalizes the sense in which this channel can be treated as a stable property of a protocol step under a class of evaluation distributions.

Writing $p_t := \Pr(E_t = 1)$, conditioning on E_0 yields

$$p_1 \;=\; p_0 + (1-p_0)\,c - p_0\,\gamma, \tag{1}$$

which decomposes any accuracy change into rescued mass $(1-p_0)c$ and destroyed mass $p_0\gamma$. The identity holds on any fixed paired evaluation set by the law of total probability; its empirical content lies entirely in whether the estimates $(\hat{c}, \hat{\gamma})$ of this channel transfer out of sample—and, when transfer fails, in why.

A rearrangement of Eq. (1) makes fragility explicit:

$$p_1 > p_0 \quad \Longleftrightarrow \quad c \; > \; \frac{p_0}{1-p_0}\,\gamma, \tag{2}$$

which amplifies corruption by the factor $\frac{p_0}{1-p_0}$. When baseline accuracy is high, even a modest corruption rate can dominate the available correction mass. This offers a direct mechanism-level reading of how intrinsic self-correction degrades performance at strong baselines (Huang et al., 2024): without reliable feedback, revision induces nontrivial corruption and the break-even requirement becomes stringent as p_0 rises. Conversely, protocols incorporating reliable external signals (verifiers, tools, unit tests, or curated feedback) succeed by shifting the interface toward larger c and/or smaller γ (Cobbe et al., 2021; Lightman et al., 2024). More generally, model-mediated oversight can be treated as an external signal shaping the effective channel (Bai et al., 2022; Perez et al., 2022). This correction–corruption framing provides a unified reading of two canonical refinement regimes: the fragility documented by Huang et al. (2024) and the modest gains reported by Madaan et al. (2023) are operating points of the same tradeoff, as we show in a phase-plane diagram in Section 6.3.

The interface is intentionally agnostic to protocol internals, which lets us compare operationally different regimes through the same channel. In alternation (ALT), a helper transcript conditions a second solve; in verification (VER, VER-FIX), the second model judges and optionally repairs; in best-of-K judging (JUDGE-K, Cobbe et al., 2021), a judge selects among multiple candidates. Across these regimes the practical question is identical: when does interaction buy accuracy, and when does it quietly pay for it by corrupting correct answers?

In practice, heterogeneity across difficulty regimes means pooled estimates can fail to transfer even when slice-conditioned kernels are stable—a failure mode we make explicit with controlled mixture-shift stress tests (C2 below). Selection protocols face a further difficulty: the measured channel can change for reasons unrelated to candidate content, such as presentation artifacts in best-of-K protocols. And in multi-step pipelines, the correctness bit E_t may not capture enough history for steps to compose predictably. We treat each of these as a distinct mechanism by which transfer can fail, and design a diagnostic for each.

The correctness-kernel view connects naturally to several existing measurement frameworks. First, it is structurally analogous to policy evaluation in finite Markov decision processes, where a one-step transition matrix is the fundamental object (Sutton & Barto, 2018); in this language, our composition gap tests whether a two-step protocol is adequately described by a Markov transition on the correctness bit alone. Second, the break-even condition $c > \frac{p_0}{1-p_0}\gamma$ has the same form as the abstaining rule in selective prediction (Geifman & El-Yaniv, 2017): invoke the protocol step only when the expected gain $(1-p_0)c - p_0\gamma$ exceeds a threshold. Third, the slice-conditioned estimation used in Section 5 is closely related to importance-weighted subpopulation calibration under covariate shift (Quiñonero-Candela et al., 2009); our contribution is to show that difficulty-mixture shifts induce this failure mode naturally in protocol evaluation and that it can be diagnosed directly by controlled depth reweighting. Finally, positional bias in LLM-based judges has been documented previously (Zheng et al., 2023; Wang et al., 2024); our contribution is to decompose that behavior by candidate-set structure, separating anchor-preserving bias from genuine rescue suppression.

The paper's novelty lies in the measurement discipline and diagnostics that make $(\hat{c}, \hat{\gamma})$ a reusable audit object; the one-step law in Eq. (1) supplies the accounting identity. Concretely, we make five contributions:

C1. Uniform paired-outcome logging contract (Section 4). A shared scoring interface makes $(\hat{c}, \hat{\gamma})$ comparable across ALT, VER, VER-FIX, and JUDGE-K, removing a systematic evaluation incompatibility.

C2. Mixture-shift stress tests (Section 5). Using depth as an explicit regime label and counterfactual mixture reweighting as the intervention, we show that pooled calibration exhibits systematic signed bias while slice-conditioned calibration remains stable—a concrete covariate-shift failure with a direct fix.

C3. GSM8K audit under JUDGE-K (Section 6). On a natural benchmark without an explicit regime label, we analyze error flow under JUDGE-K, develop a stratified presentation-invariance audit that

decomposes position bias by candidate-set structure, and use a phase-plane view to interpret reported refinement gains and failures.

C4. Composition-gap diagnostic (Section 7). Representing each step as a 2×2 correctness kernel, departures from a testable Markov condition in two-step stacks are quantified as a composition gap, turning state-sufficiency into a measurable, falsifiable property.

C5. Deployment decisions from the measured interface (Section 8). A leave-one-seed-out evaluation on the synthetic suite shows that the gain rule $(1 - \hat{p}_0)\hat{c} - \hat{p}_0\hat{\gamma} > 0$ supports concrete deployment decisions: it can distinguish between harmful and beneficial directed interaction steps using calibration-set statistics alone. A proxy analysis on GSM8K illustrates the same decision logic on a natural benchmark and shows how its practical value depends on the strength of the available observable proxy.

The interface tracks redistribution of probability mass between correctness states. Its purpose is to test whether a measured protocol step transfers and, when it does not, to localize the failure mode. When composition diagnostics indicate that the correctness bit is not a sufficient intermediate state, the natural extension is to augment the state with richer intermediate features, such as confidence, error type, or transcript signatures (Section 8).

The remainder of the paper develops and tests the interface across progressively richer settings, corresponding to the five contributions above. Section 2 formalizes the correction–corruption interface: the one-step identity, the break-even boundary, and the Jeffreys-smoothed estimators used throughout. Section 3 defines the transfer tests that give the interface its empirical content. Section 4 specifies the logging and scoring contract and instantiates the interface for ALT, VER, VER-FIX, and JUDGE-K (C1). Section 5 demonstrates the mixture-shift stress test on depth-labeled synthetic tasks (C2). Section 6 extends the audit logic to GSM8K under JUDGE-K, including the stratified presentation-invariance audit (C3) and a phase-plane interpretation of refinement gains and failures. Section 7 develops the composition-gap diagnostic for two-step stacks (C4). Section 8 discusses deployment decisions from the measured interface (C5), the interface as a design tool, and the scope of the correctness-bit abstraction.

Scope. The main controlled experiments use *exact-match binary scoring* on mathematical tasks: synthetic arithmetic stratified by expression depth, i.e. by the number of composed operations, 2×2 linear systems, and GSM8K. The correctness-bit abstraction is well-defined in these settings; extension to tasks with graded correctness or model-based evaluation is discussed in Section 8.

The same interface applies outside these mathematical evaluations when a protocol step has an auditable pre/post outcome, i.e., when correctness before and after the step can be assigned by an external criterion. In coding, E_t may denote passing an executable test suite before and after repair; in query generation, successful execution or denotation/answer equivalence; in extraction or classification, whether the pre/post output matches a held-out label under the task's scoring rule; and in factual QA, whether the answer is supported by retrieved evidence or by a human-audited reference. In each case, the external criterion induces the binary audit labels E_0 and E_1, analogous to validation labels or executable test outcomes. In such domains the measured channel is not general model competence, but the specified step under a task distribution, baseline solver, interface, downstream mechanism, and correctness criterion. Section 8.5 gives a targeted exploratory coding check of this form using executable unit tests.

Model comparisons in our main controlled experiments use open-weight checkpoints from two architecture families (Mistral/Llama and Qwen) spanning 7B–32B parameters; applying the full audit beyond this controlled model range remains future work.

2 The Correction–Corruption Interface

We study protocol behavior through the simplest observable interface: whether an item is correct before an interaction step and whether it is correct after. Models and prompts are treated as black boxes. The analysis uses only paired 0/1 correctness indicators extracted from final parsed answers on the *same* items. This deliberately coarse view cannot explain *why* a protocol works, but it is sufficient to separate the two

mechanism-level effects that determine whether a step is usable in practice: how often it fixes baseline errors and how often it breaks baseline successes.

After a verification, repair, or selection step, there are four possible correctness transitions: wrong-to-correct, correct-to-correct, correct-to-wrong, and wrong-to-wrong. The two-rate interface focuses on the beneficial wrong-to-correct transition, which we call correction, and the harmful correct-to-wrong transition, which we call corruption.

Let $\mathcal{X}$ denote the space of evaluation items, and let $X \sim \mathcal{D}$ be a random variable taking values in $\mathcal{X}$, where $\mathcal{D}$ is the evaluation distribution. Each realized item $x \in \mathcal{X}$ has a ground-truth answer $y(x)$. Fix a baseline solver $\hat{y}_0$ and a post-step solver $\hat{y}_1$ produced by a single interaction mechanism $\mathcal{A}$.

For each realized item $x \in \mathcal{X}$, define the item-level correctness indicators

$$e_0(x) := \mathbf{1}\{\hat{y}_0(x) = y(x)\} \in \{0,1\}, \qquad e_1(x) := \mathbf{1}\{\hat{y}_1(x) = y(x)\} \in \{0,1\}.$$

These induce Bernoulli random variables

$$E_0 := e_0(X), \qquad E_1 := e_1(X),$$

with accuracies

$$p_0 := \Pr(E_0 = 1), \qquad p_1 := \Pr(E_1 = 1).$$

A single step can change correctness in only two ways: it can correct a baseline failure or corrupt a baseline success. We define the corresponding conditional rates

$$c := \Pr(E_1 = 1 \mid E_0 = 0), \qquad \gamma := \Pr(E_1 = 0 \mid E_0 = 1). \tag{3}$$

The pair (c, γ) is the *correctness channel* of $\mathcal{A}$ under $\mathcal{D}$. At the level of realized items, the indicators $e_0(x)$ and $e_1(x)$ may of course depend on x; (c, γ) summarize the resulting transition behavior after averaging over $X \sim \mathcal{D}$.

Empirically, let

$$S_n = (x_1, \ldots, x_n) \in \mathcal{X}^n$$

be a finite evaluation sample, typically realized as i.i.d. draws from the chosen evaluation distribution $\mathcal{D}$ (often uniform over a benchmark split, slice, or specified mixture). On this sample we observe the realized correctness pairs

$$\big(e_0(x_k), e_1(x_k)\big), \qquad k = 1, \ldots, n,$$

and form the paired counts

$$n_{ij} := \#\{k : \ e_0(x_k) = i, \ e_1(x_k) = j\}, \qquad i, j \in \{0,1\}. \tag{4}$$

We also write

$$n_0 := n_{00} + n_{01}, \qquad n_1 := n_{10} + n_{11},$$

for the conditional support sizes underlying estimation of c and γ, respectively. From these counts we estimate (c, γ) using Jeffreys-smoothed conditional estimators (Appendix A).

Definition 1 (Stable correctness channel)**.** *For any distribution $\mathcal{D}$ on $\mathcal{X}$, let*

$$c_{\mathcal{D}} := \Pr_{\mathcal{D}}(E_1 = 1 \mid E_0 = 0), \qquad \gamma_{\mathcal{D}} := \Pr_{\mathcal{D}}(E_1 = 0 \mid E_0 = 1),$$

where the probabilities are taken under $X \sim \mathcal{D}$ and the protocol step $\mathcal{A}$. We say that the correctness channel of $\mathcal{A}$ is stable across *a family of distributions $\mathcal{F}$ if there exists a pair $(c^\star, \gamma^\star)$ such that*

$$(c_{\mathcal{D}}, \gamma_{\mathcal{D}}) = (c^\star, \gamma^\star) \qquad \textit{for every } \mathcal{D} \in \mathcal{F}.$$

Equivalently, for any $\mathcal{D}, \mathcal{D}' \in \mathcal{F}$,

$$(c_{\mathcal{D}}, \gamma_{\mathcal{D}}) = (c_{\mathcal{D}'}, \gamma_{\mathcal{D}'}).$$

The first transfer question is the simplest one: if two independent samples are drawn from the same evaluation distribution $\mathcal{D}$, does a channel estimated on one sample agree, up to finite-sample error, with the channel estimated on the other? In large samples this reduces to ordinary consistency of $(\hat{c}, \hat{\gamma})$ for the population channel $(c_{\mathcal{D}}, \gamma_{\mathcal{D}})$.

A stronger question is whether the channel remains stable under structured distribution shift. Let $s : \mathcal{X} \to \mathcal{S}$ be a slice map, where $\mathcal{S}$ is a finite set of slice labels. In our synthetic suite, $s(x)$ is the item depth. For each slice value $\sigma \in \mathcal{S}$, define the slice-conditioned channel under $\mathcal{D}$ by

$$c_{\mathcal{D}}(\sigma) := \Pr_{\mathcal{D}}(E_1 = 1 \mid E_0 = 0,\ s(X) = \sigma), \qquad \gamma_{\mathcal{D}}(\sigma) := \Pr_{\mathcal{D}}(E_1 = 0 \mid E_0 = 1,\ s(X) = \sigma).$$

If two distributions $\mathcal{D}$ and $\mathcal{D}'$ differ only in their mixture weights over slices, then the pooled channel may change:

$$(c_{\mathcal{D}}, \gamma_{\mathcal{D}}) \neq (c_{\mathcal{D}'}, \gamma_{\mathcal{D}'}).$$

The relevant hypothesis in that setting is instead *slice-level stability*: for every $\sigma \in \mathcal{S}$,

$$(c_{\mathcal{D}}(\sigma), \gamma_{\mathcal{D}}(\sigma)) = (c_{\mathcal{D}'}(\sigma), \gamma_{\mathcal{D}'}(\sigma)).$$

Under this hypothesis, a channel estimated on one sample from slice σ should remain valid on an independent sample from the same slice. Section 3 turns this reuse requirement into an explicit operational test.

For a fixed evaluation distribution $\mathcal{D}$, the corresponding unconditional accounting identity is the one-step interaction law. Writing

$$p_{0,\mathcal{D}} := \Pr_{\mathcal{D}}(E_0 = 1), \qquad p_{1,\mathcal{D}} := \Pr_{\mathcal{D}}(E_1 = 1),$$

conditioning on E_0 yields

$$p_{1,\mathcal{D}} = p_{0,\mathcal{D}} + \big(1 - p_{0,\mathcal{D}}\big) c_{\mathcal{D}} - p_{0,\mathcal{D}} \gamma_{\mathcal{D}}. \tag{5}$$

Equivalently,

$$\Delta p_{\mathcal{D}} := p_{1,\mathcal{D}} - p_{0,\mathcal{D}} = \big(1 - p_{0,\mathcal{D}}\big) c_{\mathcal{D}} - p_{0,\mathcal{D}} \gamma_{\mathcal{D}}. \tag{6}$$

Indeed,

$$\Pr_{\mathcal{D}}(E_1 = 1) = \Pr_{\mathcal{D}}(E_1 = 1 \mid E_0 = 0) \Pr_{\mathcal{D}}(E_0 = 0) + \Pr_{\mathcal{D}}(E_1 = 1 \mid E_0 = 1) \Pr_{\mathcal{D}}(E_0 = 1),$$

so substituting $\Pr_{\mathcal{D}}(E_1 = 1 \mid E_0 = 0) = c_{\mathcal{D}}$, $\Pr_{\mathcal{D}}(E_1 = 1 \mid E_0 = 1) = 1 - \gamma_{\mathcal{D}}$, and $\Pr_{\mathcal{D}}(E_0 = 1) = p_{0,\mathcal{D}}$ gives (5); subtracting $p_{0,\mathcal{D}}$ gives (6).

Equation (5) is an exact accounting identity under the fixed evaluation distribution $\mathcal{D}$. It separates protocol impact into corrected mass $\big(1 - p_{0,\mathcal{D}}\big) c_{\mathcal{D}}$ and corrupted mass $p_{0,\mathcal{D}} \gamma_{\mathcal{D}}$, making explicit that a protocol can fail by introducing even a modest corruption rate when $p_{0,\mathcal{D}}$ is already high. The empirical content of the interface begins not with the identity itself, but when an estimated channel is *reused* out of sample.

A simple but practically important consequence is the break-even boundary:

$$p_{1,\mathcal{D}} > p_{0,\mathcal{D}} \quad \Longleftrightarrow \quad c_{\mathcal{D}} \ > \ \frac{p_{0,\mathcal{D}}}{1 - p_{0,\mathcal{D}}} \gamma_{\mathcal{D}}, \tag{7}$$

which amplifies corruption by the factor $\frac{p_{0,\mathcal{D}}}{1 - p_{0,\mathcal{D}}}$ as baseline accuracy increases. This boundary will later provide a useful phase-plane interpretation of why some refinement regimes help while others degrade performance.

For compactness we package the channel under the fixed distribution $\mathcal{D}$ as the *correctness kernel*, by which we mean the 2×2 conditional transition matrix over the binary correctness state (rows and columns ordered $0 =$ incorrect, $1 =$ correct)

$$T_{\mathcal{D}} = \begin{pmatrix} 1 - c_{\mathcal{D}} & c_{\mathcal{D}} \\ \gamma_{\mathcal{D}} & 1 - \gamma_{\mathcal{D}} \end{pmatrix}, \tag{8}$$

so that the marginal correctness distribution propagates by right-multiplication:

$$\pi_{1,\mathcal{D}} \ = \ \pi_{0,\mathcal{D}} T_{\mathcal{D}}, \qquad \pi_{t,\mathcal{D}} = [\Pr_{\mathcal{D}}(E_t = 0),\ \Pr_{\mathcal{D}}(E_t = 1)].$$

Likewise, each slice value $\sigma \in \mathcal{S}$ induces a slice-conditioned kernel

$$T_{\mathcal{D}}^{(\sigma)} = \begin{pmatrix} 1 - c_{\mathcal{D}}(\sigma) & c_{\mathcal{D}}(\sigma) \\ \gamma_{\mathcal{D}}(\sigma) & 1 - \gamma_{\mathcal{D}}(\sigma) \end{pmatrix}.$$

When the underlying distribution or slice is clear from context, we suppress the corresponding subscripts and write simply p_0, p_1, c, γ, and T. These kernel representations will be useful both for conditioning on regimes (to address heterogeneity) and for composing multiple steps in stacks. Section 3 next turns this formal interface into an operational transfer test.

3 Transfer as a Diagnostic and Design Contract

The one-step law (5) is an accounting identity on any fixed paired evaluation set. Section 2 defined the corresponding population quantities $(p_{0,\mathcal{D}}, p_{1,\mathcal{D}}, c_{\mathcal{D}}, \gamma_{\mathcal{D}})$ under an evaluation distribution $\mathcal{D}$. The empirical content of the interface begins only when we *reuse* sample-based estimates of these quantities out of sample to forecast post-step behavior under controlled changes in seed, slice, mixture, or protocol presentation. Throughout the paper we treat this reuse requirement not merely as a way to invalidate unstable channels, but as a constructive contract: interfaces that remain stable under the shifts we care about can be used as modular components in protocol design.

Operationally, let $A, B \subset \mathcal{X}$ denote two finite evaluation samples of the form introduced in Section 2, playing the roles of calibration and target streams. If A is drawn under some evaluation distribution $\mathcal{D}_A$ and B under $\mathcal{D}_B$, then the fitted quantities $(\hat{c}_A, \hat{\gamma}_A)$ estimate the population channel $(c_{\mathcal{D}_A}, \gamma_{\mathcal{D}_A})$, while $\hat{p}_{0,B}$ and $\hat{p}_{1,B}^{\text{emp}}$ estimate $p_{0,\mathcal{D}_B}$ and $p_{1,\mathcal{D}_B}$. Given A and B (for example, corresponding to different seeds, slices, or mixtures), we fit $(\hat{c}_A, \hat{\gamma}_A)$ on A and *predict* post-step accuracy on B from the baseline accuracy measured on B via

$$\hat{p}_{1,B}^{\text{pred}} \;=\; \hat{p}_{0,B} + (1 - \hat{p}_{0,B})\hat{c}_A - \hat{p}_{0,B}\hat{\gamma}_A.$$

Letting

$$\hat{p}_{1,B}^{\text{emp}} = \frac{n_{01}^B + n_{11}^B}{n_{00}^B + n_{10}^B + n_{01}^B + n_{11}^B},$$

where n_{ij}^B are the counts in (4) for stream B, we evaluate the residual

$$\hat{p}_{1,B}^{\text{emp}} - \hat{p}_{1,B}^{\text{pred}}.$$

We interpret this residual relative to the finite-sample uncertainty implied by the paired counts, especially when the conditional supports used to estimate $(\hat{c}, \hat{\gamma})$ are thin. In the language of Section 2, the question is whether the channel estimated from A behaves as if it were a valid approximation to the target channel for B. Equivalently, the empirical transfer test asks whether the stability condition of Definition 1 holds to a useful approximation for the calibration and target distributions under the perturbation being studied. Supporting subsampling and convergence diagnostics are provided in the Supplementary Material. Residual structure that persists beyond this scale indicates that the correctness bit is not a sufficient state summary for stable reuse under the tested perturbation.

The main empirical findings then organize into four body-level claims; the remaining analyses and appendices support and refine these themes.

1. **Transfer can be measured directly at the channel level.** For several protocol families, the fitted $(\hat{c}, \hat{\gamma})$ predicts held-out post-step behavior across seeds within sampling error, while other channels exhibit systematic drift. We report both point estimates and residual diagnostics, making stability (or instability) explicit rather than implicit in end-to-end deltas (Section 5).

2. **Mixture shift is a first-order failure mode of pooled interfaces, and conditioning restores stability.** When behavior differs across latent regimes, pooled $(\hat{c}, \hat{\gamma})$ becomes mixture-dependent even if regime-conditioned kernels are stable. On depth-labeled synthetic tasks, depth-conditioned

interfaces support accurate mixture predictions under controlled mixture shifts without additional model calls (Sections 5 and 6).

When regime labels like depth are unavailable, we recommend selecting the *coarsest partition that stabilizes the channel* under the stress tests of this paper: start pooled; if mixture-stress residuals show systematic drift, introduce a small number of bins using an available difficulty proxy (e.g., input length, model self-reported confidence, judge margin, or a learned difficulty score) and refine only until bias under plausible mixture shift is reduced while conditional supports n_0 and n_1 remain large enough for stable estimation. This makes conditioning a controlled bias–variance trade rather than an ad hoc choice.

3. **Interface contamination is real and must be audited.** Selection protocols—such as verifier-based selection (Cobbe et al., 2021) or judge-style evaluation frameworks (Zheng et al., 2023)—can induce presentation artifacts that change the measured channel without changing the underlying candidate set. On GSM8K we include an invariance audit for JUDGE-K at K=4 by deterministically permuting candidate order per item and mapping decisions back to original indices; removing a strong first-position prior shifts inferred $(\hat{c}, \hat{\gamma})$ and reduces apparent gains, illustrating why selection mechanisms benefit from explicit invariance checks (Section 6).

4. **Multi-step stacks admit a composition audit.** Representing each step as a correctness kernel yields exact *marginal* propagation by kernel products. A stronger statement, namely, that the collapsed *conditional* two-step kernel matches the product of adjacent one-step kernels, requires a correctness-level state-sufficiency condition. We quantify violations via a *composition gap*, turning multi-step behavior into a diagnostic of step interaction and insufficient state summaries (Section 7.1; diagnostics in Section 7.2). Support-sensitivity and pooled out-of-sample composition-gap summaries are reported in the Supplementary Material, Sec. S5. When composition fails in well-supported slices, the correctness bit is not a sufficient Markov state for the stack; practical enrichments include verifier/judge confidence or margin, candidate-set disagreement features (e.g., top-2 gap), structured error tags extracted from critiques, and simple transcript signatures (length, presence of explicit checks).

Used constructively, the same view supports protocol design. The interface makes the correction–corruption tradeoff explicit, so steps can be compared by their channel parameters rather than only by end-to-end gains; conditioning choices can be evaluated by whether they stabilize kernels under mixture changes; invariance audits separate genuine improvements from presentation artifacts; and composition-gap tests determine when multi-step stacks can be engineered modularly.

4 Experimental Protocol

Our objective is to isolate the effect of a minimal *interaction pattern* from confounders such as stochastic decoding, elaborate prompt engineering, or changes to model weights. Accordingly, we use fixed prompt templates and a uniform answer-extraction pipeline across all regimes. Where possible we use deterministic decoding so that differences in measured transition behavior arise from the *conditioning and instruction structure*—that is, what information is provided to which model and what the model is asked to do—rather than from sampling variance.

4.1 One-step protocols

Because interaction effects are defined *by their conditioning context*, we treat prompt text as part of the experimental specification. This section specifies each regime at the level of inputs, outputs, and behavioral contracts; the exact system/user templates, formatter contract, stable prompt identifiers, and logged JSONL artifacts are provided in the Supplementary Material, Section S8, and in the accompanying code release.

Models, decoding, and per-task logging. We evaluate open-weight LLMs served locally via `ollama` to keep the serving stack and decoding settings controlled. Model selection is not part of the theory;

we use representative open-weight checkpoints spanning families and scales, guided by public comparisons such as the Open LLM Leaderboard (Fourrier et al., 2024); see Table 1. The reported results show the same qualitative phenomena across two architecture families (Mistral/Llama and Qwen) spanning 7B–32B parameters within the open-weight range we study, but do not address substantially different capability levels.

For INDEP, ALT, VER, and VER-FIX, we use deterministic decoding (temperature 0) so that estimated transition rates reflect mechanism differences rather than sampling noise. For JUDGE-K, candidate *generation* uses stochastic decoding (temperature 0.7) to produce a meaningful candidate set, while the *judge* itself is run deterministically (temperature 0) so that selection behavior is not additionally confounded by judge sampling. This separation makes randomness an explicit part of the generate–select regime, rather than an uncontrolled nuisance across all protocols.

For protocols with temperature > 0, the estimated $(\hat{c}, \hat{\gamma})$ averages over the randomness of the specific candidate sets realized during evaluation; it is therefore a property of the *protocol setting* (model, temperature, prompt), not of a single call. This is the correct object for deployment decisions: a practitioner who fixes temperature and prompt encounters the same distribution of corrections and corruptions in repeated use. In our experiments, temperature is treated as a fixed protocol parameter held constant within each regime, so it does not confound cross-regime differences in the measured channel.

Model	**Family**	**Scale**	**Instruction-tuned**	**Primary roles**
`mistral`	Mistral	∼7B class	yes	solver, helper, verifier
`llama3.2`	Llama	∼8B class	yes	solver, helper
`qwen2.5`	Qwen	∼14B class	yes	verifier, judge
`qwen-32b`	Qwen	∼32B class	yes	judge, stress tests

Table 1: Representative open-weight checkpoints spanning multiple families and scales. The specific identities are not parameters of the theory; they serve to demonstrate that the measured channel phenomena persist across distinct architectures and capability ranges.

All runs are logged at the per-task level with: model identifiers; regime label (INDEP/ALT/VER/VER-FIX/JUDGE-K); raw model text; formatter output (single-line contract answer); parsed normalized answer; and binary correctness indicators. These logs support reconstruction of the paired counts n_{ij} in Eq. (4) and enable strict separation of fitting from evaluation in hold-out transfer tests (Sections 5–7). The Jeffreys-smoothed estimators $(\hat{c}, \hat{\gamma})$ are described in Appendix A.

Tasks, slice labels, and evaluation streams. We evaluate on reproducible task generators with known ground truth, designed to control task structure while keeping evaluation exact. The paper uses two synthetic families and one natural benchmark:

- **Depth-stratified arithmetic** with discrete depth $d \in \{1, 2, 3, 4, 5\}$. Here d is the generator-defined structural depth (number of composed operations / nesting steps as defined by the task family), and it provides an explicit slice label for regime heterogeneity studies.
- 2×2 **linear systems** with integer or rational solutions, providing a second controlled family with different surface form and error structure.
- **GSM8K** as a natural-data setting where no ground-truth regime label exists; we therefore emphasize held-out stability, residual diagnostics, and audit procedures based on observable complexity proxies rather than generator-defined depth conditioning (Section 6).

Unless otherwise stated, synthetic evaluations use three root seeds (123/124/125). Each synthetic task instance x is deterministically generated from a task ID and a root seed; the generator, parameter ranges, and exact formatting are released with the code. We report three types of summaries: within-seed depth-stratified summaries at the bin level; pooled summaries that aggregate bins within a seed when a single

pooled channel is being evaluated; and cross-seed summaries, reported both in pooled form and by depth, that aggregate or average the corresponding quantities across the three root seeds. Correspondingly, we fit both a pooled transition model (c, γ) and a depth-conditioned model $\big(c(d), \gamma(d)\big)$. A central empirical finding is that depth conditioning is not merely descriptive: when calibration and test mixtures differ, depth-conditioned interfaces enable accurate *mixture prediction* by recombining slice-wise forecasts, while pooled interfaces exhibit systematic bias under mixture shift (Section 5).

Interaction regimes as pre→post channels. All regimes in this paper are operational instantiations of the same pre→post channel defined in Section 2. For each regime we specify the concrete inputs and outputs—what is shown to which model, what instruction is issued, and what binary post decision is scored. For solve-type regimes (INDEP, ALT, VER-FIX, and JUDGE-K), the post decision is correctness of a final task answer; for verdict-only verification (VER), the post decision is correctness of the verifier's binary verdict about a candidate answer. The substantive distinction between regimes is therefore how the post decision is produced. In ALT the measured transition is induced by transcript-conditioned re-solving; in VER it is induced by changing the instruction from *solve* to *verify*; in VER-FIX by verify-then-repair; and in JUDGE-K by generate–select with an LLM judge. These different post-decision mechanisms can materially alter (c, γ).

- **Independent baseline (INDEP).** For a model M, the prompt contains only the task statement. The model produces a solve transcript, which is converted to a contract answer by the formatter and parsed into canonical form, yielding a binary correctness indicator. This regime defines the independent baseline used as E_0 in the paired evaluations below.

- **Alternation (ALT): transcript-conditioning with a *solve* instruction.** In alternation direction $A \to B$, model A is the *helper* that provides conditioning context, and model B is the *target* whose pre→post transition is being measured. The helper model A first produces a solve transcript $r_A(x)$. We then query the target model B on the same task while conditioning on $r_A(x)$, inserted verbatim as a delimited context block. Crucially, B is not asked to critique, verify, or evaluate A; it is instructed simply to solve the task and provide an answer. Here E_0 is correctness of the target model's independent baseline answer, and E_1 is correctness of that same target model's answer when conditioned on the helper transcript. Conditioning on the helper's full transcript preserves intermediate structure that can enable both correction (useful scaffold) and corruption (induced consistent mistake).

- **Verdict-only verification (VER): judge a single candidate.** In verification direction $A \to B$, model A is the *solver* that produces the candidate being judged, and model B is the *verifier* whose post decision is measured. The solver A first produces a single candidate solution (transcript and final answer). A verifier model B is then instructed only to assess whether that candidate completion is correct, returning a binary verdict (`correct/incorrect`). Here E_0 is correctness of the solver's candidate answer itself, while E_1 is correctness of the verifier's verdict relative to that candidate's ground-truth status. This defines a verification channel in which the post step is instructed to *verify* rather than *solve*, under a fixed task instance and candidate.

- **Verification-and-fix (VER-FIX): judge then solve if needed.** In verification-and-fix direction $A \to B$, model A is the *solver* that produces the candidate being checked, and model B is the *verifier* that produces the final post-step answer. The solver A first produces a single candidate transcript. The verifier B is instructed to (i) decide whether the candidate is correct and (ii) if incorrect, solve the task correctly and output a final answer under the same extraction contract as the baseline solver. Here E_0 is correctness of the solver's candidate answer, and E_1 is correctness of the verifier's final output after the verify-and-fix procedure. Its $(\hat{c}, \hat{\gamma})$ are therefore comparable to those of other regimes at the channel level, while representing a different mechanism from transcript-conditioned re-solving in ALT.

- **Best-of-K judging (JUDGE-K): generate–select with an LLM judge.** In JUDGE-K, the *solver* generates the candidate set and the *judge* produces the post decision by selecting one candidate. A solver generates K candidate solutions for each task, and a judge model selects a single candidate

to output (cf. Cobbe et al. (2021)). We define E_0 as correctness of a designated anchor candidate, and that same anchor candidate is included among the K judged candidates; E_1 is correctness of the judge-selected output among the full candidate set. When $K > 1$, we additionally log an oracle indicator E_{oracle} for whether any of the K candidates is correct, yielding an empirical oracle rate p_{oracle} computed directly from the sampled candidate sets. For intuition, p_{oracle} is upper-bounded by $1 - (1 - p)^K$ under independent candidates with per-candidate correctness p, but we do not assume independence for estimation or claims; we report oracle headroom empirically from the observed sets.

Across regimes, the pre-step object is chosen in the natural operational way for that protocol: the model's independent solve in INDEP, the target model's independent solve in ALT, the candidate being judged in VER and VER-FIX, and the designated anchor candidate in JUDGE-K. The common abstraction is therefore a paired pre→post correctness transition on the same task, rather than an identical notion of "baseline" across all interaction mechanisms. These regimes provide distinct interaction mechanisms under a common paired-outcome scoring contract, allowing the interface to be tested across re-solving, verification, repair, and selection.

Answer extraction and correctness criteria. To avoid regime-specific parsing artifacts, every run (INDEP, ALT, VER, VER-FIX, JUDGE-K) uses the same deterministic two-stage extraction *pattern*: first, the queried model produces free-form text; second, a fixed-template *formatter* prompt converts that text into a single-line output under an explicit contract (Appendix B). The formatter has no access to ground truth; it is a shared normalization step whose purpose is to reduce formatting-induced variance across regimes. Contract compliance determines whether an output can be parsed under the regime-appropriate extraction contract, whereas the paper's reported correctness indicators are based on mathematical correctness after deterministic normalization (or, for verdict-only verification, correctness of the parsed verdict relative to the candidate's ground-truth status); formatter success, parse category, and correctness are logged separately so these failure modes can be audited.

For solve-type regimes (INDEP, ALT, VER-FIX, JUDGE-K), the contract constrains *how* task answers are expressed so they can be parsed uniformly; correctness is then *mathematical*: after deterministic normalization, the parsed output is compared to the ground-truth answer and scored correct iff they match exactly (for vector outputs, all coordinates must match). For scalar tasks the contract takes the form `Answer = <integer or reduced rational p/q>`. For 2×2 systems it is `Answer = (<x>,<y>)` with each coordinate an integer or reduced rational. The parser deterministically normalizes integers, rationals a/b, and (when enabled for a family) terminating decimals mapped to rationals when unambiguous.

For verdict-only verification (VER), the formatter instead maps the verifier transcript to a binary contract (`correct/incorrect`), and E_1 is scored by comparing that parsed verdict to the true correctness status of the candidate being judged. Thus the extraction architecture is shared across regimes, but the output contract matches the type of post decision being evaluated. This removes a common failure mode where different regimes induce different formatting compliance and thereby change measured accuracy for reasons unrelated to the underlying interaction mechanism.

Holdouts, mixture shift, and transfer evaluation. Section 3 frames transfer operationally in terms of a calibration sample A and a target sample B. The present section specifies how those samples are instantiated in our experiments. When $\big(\hat{p}_0, \hat{p}_1, \hat{c}, \hat{\gamma}\big)$ are computed on the same set of tasks, the one-step law holds by construction (Equation (5)). The nontrivial question is whether (c, γ) behave as stable characteristics of a regime that generalize out of sample. We therefore evaluate reuse through two complementary holdout notions.

First, we run multiple independent experimental roots (distinct random seeds that select and/or order task IDs and control any protocol-internal randomness). For each directed pair $A \to B$ and each slice definition (e.g., depth d), we use training roots as calibration streams and held-out roots as target streams: we fit $\big(\hat{c}, \hat{\gamma}\big)$ (or $\big(\hat{c}(d), \hat{\gamma}(d)\big)$) on the calibration roots and predict held-out $\hat{p}_1$ on the target roots using only held-out $\hat{p}_0$ and the fitted transition rates, then swap train/test roles.

Second, holding the experimental root fixed, we perform paired within-root task-ID splits to isolate instance-level generalization within a regime. Concretely, we partition task IDs into disjoint train/test sets and exclude the same held-out IDs from both the INDEP logs (which define E_0) and the corresponding post-regime logs (which define E_1). This yields within-root calibration and target samples that test whether the estimated within-slice transition mechanism transfers from one subset of instances to new instances at the same depth, without conflating the effect with regeneration under a new root.

To stress-test stability under changing slice distributions, we additionally evaluate explicit mixture shift: we fit a pooled $(\hat{c}, \hat{\gamma})$ on a calibration depth mixture (e.g., uniform over d) and predict performance on a target set concentrated on a single depth (or on a different predefined mixture). We compare pooled predictions to depth-conditioned predictions using $\big(\hat{c}(d), \hat{\gamma}(d)\big)$, which avoid mixture-induced bias when predicting within each depth.

Because transfer is evaluated from finite paired counts, residuals must be interpreted relative to sample scale. In our synthetic evaluations, pooled summaries aggregate over substantially more items than individual depth slices, so residuals of a few percentage points at the slice level are compatible with ordinary finite-sample variation, especially when one conditional denominator is thin. We therefore explicitly check this uncertainty (Supplemental Material S4) and read persistent structure across seeds or mixtures as evidence against sufficiency, but isolated slice-level deviations of that scale as expected sampling noise rather than as theory failure. When discussing conditional support, we refer explicitly to the denominators $n_0 = n_{00} + n_{01}$ for $\hat{c}$ and $n_1 = n_{10} + n_{11}$ for $\hat{\gamma}$.

4.2 Stacked protocols for composition validation

To test whether one-step summaries behave predictively under protocol stacks (Section 7), we include composed regimes in which each task produces a chain of correctness indicators (E_0, E_1, E_2).

Fix two interaction steps: a first step that maps a baseline solve to an intermediate outcome (e.g., ALT in direction $A \to B$, where A provides the conditioning transcript and B produces the post-step answer), and a second step that maps the intermediate outcome to a final decision (e.g., JUDGE-K selection with an LLM judge, VER-FIX repair, or a second ALT step with a different helper). For each task x, we log the item-level indicators

$$e_0(x) \in \{0,1\}, \qquad e_1(x) \in \{0,1\}, \qquad e_2(x) \in \{0,1\},$$

where $e_0(x)$ is baseline correctness, $e_1(x)$ is correctness after step 1, and $e_2(x)$ is correctness after the full stack. At the distribution level these induce Bernoulli random variables

$$E_0 := e_0(X), \qquad E_1 := e_1(X), \qquad E_2 := e_2(X),$$

in the same notation as Section 2.

On a calibration split, we estimate adjacent-step conditional kernels $\hat{T}_{01}$ from paired counts of (e_0, e_1) and $\hat{T}_{12}$ from paired counts of (e_1, e_2), each with the two-rate form

$$\hat{T}_{01} = \begin{pmatrix} 1-\hat{c}_1 & \hat{c}_1 \\ \hat{\gamma}_1 & 1-\hat{\gamma}_1 \end{pmatrix}, \qquad \hat{T}_{12} = \begin{pmatrix} 1-\hat{c}_2 & \hat{c}_2 \\ \hat{\gamma}_2 & 1-\hat{\gamma}_2 \end{pmatrix}.$$

In the terminology of Section 3, the calibration split supplies the fitted kernels and the disjoint test split plays the role of the target sample. Let

$$\hat{\pi}_0 := [\hat{p}^{\text{emp}}_{0,\text{test}}(0),\ \hat{p}^{\text{emp}}_{0,\text{test}}(1)]$$

denote the empirical baseline correctness distribution on the test split, where

$$\hat{p}^{\text{emp}}_{0,\text{test}}(a) := \frac{\#\{x \in \text{test} : e_0(x) = a\}}{|\text{test}|}, \qquad a \in \{0,1\}.$$

Using $\hat{T}_{01}$ and $\hat{T}_{12}$ fit on calibration, marginal propagation predicts

$$\hat{\pi}_2 \ = \ \hat{\pi}_0 \hat{T}_{01} \hat{T}_{12}, \qquad \hat{p}^{\text{pred}}_2 \ := \ \hat{\pi}_2(1),$$

which we compare to the empirical $\hat{p}_2^{\text{emp}}$ on the test split. This mirrors the one-step transfer test at the level of final accuracy through the comparison of $\hat{p}_2^{\text{pred}}$ and $\hat{p}_2^{\text{emp}}$. Composition adds a stronger diagnostic: on the same split we also compare the directly estimated collapsed kernel $\hat{T}_{02}$ to the product $\hat{T}_{01}\hat{T}_{12}$. Discrepancies are summarized as a *composition gap* between the empirically collapsed kernel and the product prediction. We report two complementary summaries: the entrywise maximum deviation

$$\Delta_{\text{comp}}^{\max} := \left\|\hat{T}_{02} - \hat{T}_{01}\hat{T}_{12}\right\|_\infty = \max_{a,b\in\{0,1\}} \left|\hat{T}_{02}(a,b) - \big(\hat{T}_{01}\hat{T}_{12}\big)(a,b)\right|, \tag{9}$$

and a row-averaged ℓ_1 discrepancy,

$$\Delta_{\text{comp}}^{\text{mean}} := \frac{1}{2} \sum_{a\in\{0,1\}} \sum_{b\in\{0,1\}} \left|\hat{T}_{02}(a,b) - \big(\hat{T}_{01}\hat{T}_{12}\big)(a,b)\right|. \tag{10}$$

$\Delta_{\text{comp}}^{\max}$ is directly interpretable as the largest conditional probability mismatch induced by assuming the Markov factorization, while $\Delta_{\text{comp}}^{\text{mean}}$ measures typical deviation and is more stable under sampling noise. Both quantities diagnose kernel-level composition failure, whereas the gap between $\hat{p}_2^{\text{pred}}$ and $\hat{p}_2^{\text{emp}}$ is its corresponding marginal manifestation at the level of final accuracy. Section 7 reports both metrics, relates them to stacked prediction error, and distinguishes structural gap from finite-support effects.

5 Controlled Validation on Depth-Labeled Synthetic Tasks

This section is the paper's controlled validation core. On synthetic tasks we have three advantages that do not exist on natural benchmarks: (i) exact ground truth, (ii) reproducible instance generation, and (iii) an explicit regime label—*structural depth*—that induces systematic heterogeneity in how interaction changes correctness. The goal here is not to show that any protocol "creates capability," but to show that the *pre→post correctness interface* behaves like a transferable mechanism *when conditioned on the right regime variable*, and to make mixture dependence visible and testable when it does not.

We evaluate two synthetic families with exact scoring:

1. **Depth-stratified arithmetic** with discrete depth $d \in \{1, 2, 3, 4, 5\}$. Here d is the generator-defined structural depth (number of composed operations / nesting steps as defined by the task family), and it provides an explicit slice label for regime heterogeneity studies. Instances are generated by sampling integer operands and operators from $\{+, -, \times, \div\}$ under fixed constraints (bounded operand range, well-formedness, and avoidance of degenerate cases such as division by zero), then composing these primitives recursively to achieve the desired depth. The generation procedure is deterministic given the seed and task ID, so the same instance set can be reproduced exactly.
2. **Small linear systems** (e.g., 2×2) with integer coefficients and integer or rational solutions.

Each instance is deterministically generated from a root seed and a task ID. In the standard synthetic design, each seed contributes $n = 600$ items in total: 100 for each depth bin $d \in \{1, \ldots, 5\}$ and 100 for the 2×2 family. The depth label d is *not inferred*; it is part of the generator and therefore a clean regime index. In this synthetic setting, depth functions as a controlled difficulty proxy, which is exactly what makes it useful for testing whether conditioning on the right regime variable stabilizes the interface.

Across all regimes we score *mathematical correctness* after deterministic normalization: the model produces a free-form transcript, a shared formatter converts it into a single-line contract output, and a parser reduces that output to a canonical representation (integers, reduced rationals, and, where enabled, terminating decimals mapped to rationals when unambiguous). Correctness is exact match to the generator's ground truth after normalization. The contract is an auditable interface; it is not itself the notion of correctness.

5.1 ALT as a controlled probe: depth-heterogeneous correction and corruption

We begin with alternation (ALT) because it is the minimal interaction pattern: a helper model A produces a solve transcript $r_A(x)$, and the target model B is asked to solve the *same* instance again while conditioning

on that transcript, without being instructed to critique or verify. This makes ALT a controlled probe of how conditioning context moves probability mass between correctness states while holding the post-step action fixed ("solve").

For a fixed ordered pair $A \to B$, let $e_0(x)$ denote correctness of the target model B's independent baseline answer on instance x, and let $e_1(x)$ denote correctness of B's alternated answer on the same instance when conditioned on the helper transcript $r_A(x)$. On the depth-stratified arithmetic family, we study the resulting depth-conditioned interface

$$c(d) = \Pr(E_1 = 1 \mid E_0 = 0,\ s(X) = d), \qquad \gamma(d) = \Pr(E_1 = 0 \mid E_0 = 1,\ s(X) = d),$$

estimated with Jeffreys smoothing as throughout the paper. The one-step law still holds identically within any fixed evaluated set; the empirical question here is whether this depth-conditioned interface transfers out of sample. In the language of Definition 1, the seed-holdout test corresponds to the same-distribution case $\mathcal{D}' = \mathcal{D}$: by construction, different seeds provide repeated independent finite samples from the same controlled evaluation distribution. This is purposeful, so that ordinary seed-holdout tests same-distribution stability, while mixture shift is introduced separately and explicitly in Section 5.2.

Empirically, ALT behaves as a family of regime-dependent transformations: the correction and corruption tendencies can change materially across synthetic slices even for a fixed ordered pair $A \to B$. Figure 1 makes this heterogeneity explicit by plotting the estimated interface across the synthetic slices in the $(\hat{\gamma}, \hat{c})$ plane.

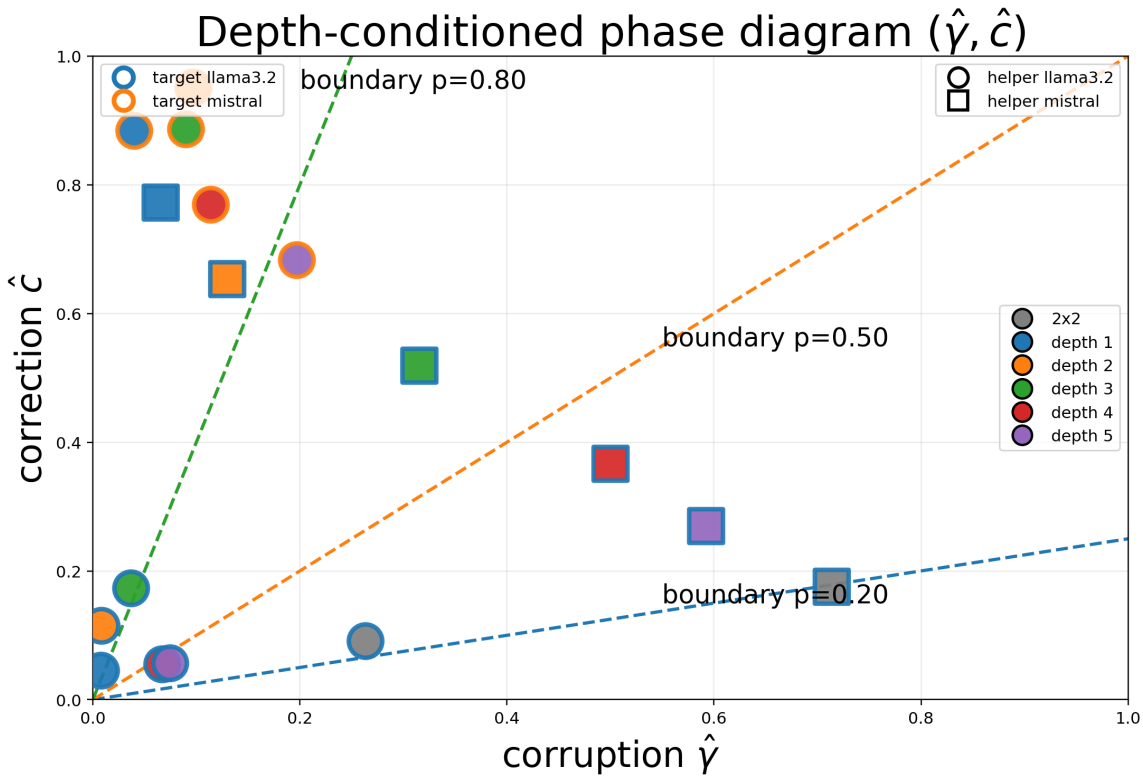


Figure 1: Phase diagram over synthetic slices for ALT (Jeffreys-smoothed transition-rate estimates). Each point is one (helper $\to$ target, slice) estimate plotted at $(\hat{\gamma}, \hat{c})$, where $\hat{c} = \Pr(E_1 = 1 \mid E_0 = 0, \text{slice})$ and $\hat{\gamma} = \Pr(E_1 = 0 \mid E_0 = 1, \text{slice})$ are estimated from paired outcomes with Jeffreys smoothing. The slices comprise the arithmetic depth bins together with the 2×2 synthetic family. Estimates are pooled over seeds and filtered to remove low-support slices before plotting. Overlaid lines show the phase boundary $c = \frac{p}{1-p}\gamma$ for representative baseline accuracies p.

We next test predictive transfer by seed-holdout. The key comparison is between a single pooled fit $(\hat{c}, \hat{\gamma})$ and the depth-conditioned fit $\big(\hat{c}(d), \hat{\gamma}(d)\big)$, both evaluated by predicting held-out post accuracy from held-out

baseline accuracy via the one-step map. At the depth-bin level, the predicted quantity is $\hat{p}_1(d)$ computed from held-out $\hat{p}_0(d)$.

Tables 2 and 3 report the per-seed pooled summaries and the seed-holdout prediction errors. The key point is not that every residual is uniformly small at every depth: depth-binned errors can be visibly affected by finite sampling in the conditional rates, because $c(d)$ is estimated only on the error set $\#\{E_0 = 0\}$ and $\gamma(d)$ only on the success set $\#\{E_0 = 1\}$, which can be thin in very easy or very hard slices even when the total per-depth support is moderate. Empirically, for slices with adequate conditional support, residuals on the order of a few percentage points are consistent with the finite-sample diagnostics reported in the Supplementary Material. What matters for the mechanism view is the structure of the out-of-sample deviations: when depth induces real variation in $(c(d), \gamma(d))$, pooled calibration can drift systematically as a function of depth mixture, while depth-conditioned calibration tracks held-out depth profiles more closely (Figure 2). We quantify the finite-sample and low-support contribution to these deviations explicitly in the convergence and support diagnostics reported in the Supplementary Material, so that mixture effects are not conflated with thin-denominator noise.

Table 2: Per-seed pooled calibration summaries for ALT. We report the target's baseline accuracy p_0 (independent solve), the alternated accuracy p_1, and the pooled Jeffreys-smoothed transition rates $(\hat{c}, \hat{\gamma})$ inferred from paired outcomes. The (one-step) in-sample prediction is omitted because it coincides with p_1 by identity up to rounding.

seed	pair (helper→target)	p_0	p_1	$\hat{c}$	$\hat{\gamma}$
123	llama3.2→llama3.2	0.8373	0.8214	0.1037	0.0393
123	llama3.2→mistral	0.3463	0.8513	0.8034	0.0603
123	mistral→llama3.2	0.8373	0.6737	0.3963	0.2726
124	llama3.2→llama3.2	0.8593	0.8393	0.0352	0.0290
124	llama3.2→mistral	0.3443	0.8613	0.8313	0.0838
124	mistral→llama3.2	0.8593	0.6417	0.4155	0.3213
125	llama3.2→llama3.2	0.8493	0.8234	0.0987	0.0481
125	llama3.2→mistral	0.3024	0.8593	0.8243	0.0625
125	mistral→llama3.2	0.8493	0.6297	0.2829	0.3087

Table 3: Seed-holdout error for one-step prediction of alternated accuracy in ALT. For each held-out seed and ordered pair, we fit $(\hat{c}(d), \hat{\gamma}(d))$ on the other two seeds and predict $p_1(d)$ from held-out $p_0(d)$ across the five depths; MAE and bias aggregate over depths.

held-out seed	pair (helper→target)	mae	bias	ae95	max abs err
123	llama3.2→llama3.2	0.0100	0.0067	0.0252	0.0293
123	llama3.2→mistral	0.0511	-0.0106	0.1261	0.1443
123	mistral→llama3.2	0.0467	0.0391	0.1097	0.1277
124	llama3.2→llama3.2	0.0149	0.0025	0.0284	0.0310
124	llama3.2→mistral	0.0285	0.0064	0.0546	0.0558
124	mistral→llama3.2	0.0265	-0.0141	0.0575	0.0620
125	llama3.2→llama3.2	0.0087	-0.0083	0.0160	0.0172
125	llama3.2→mistral	0.0325	0.0040	0.0797	0.0912
125	mistral→llama3.2	0.0352	-0.0274	0.0638	0.0672

Table 2 reports pooled seed-level summaries, whereas Figure 1 plots depth-sliced estimates; they support the same mechanism from different aggregation levels and should not be read as representing the same points.

5.2 The central stress test: pooled calibration fails under mixture shift, conditioning fixes it

Depth gives us a controlled way to separate two effects that are usually conflated in protocol evaluations. First, the interaction mechanism itself can differ across depths (true regime heterogeneity). Second, even when the depth-conditioned channel is stable across seeds, a pooled estimate can fail to transfer when the

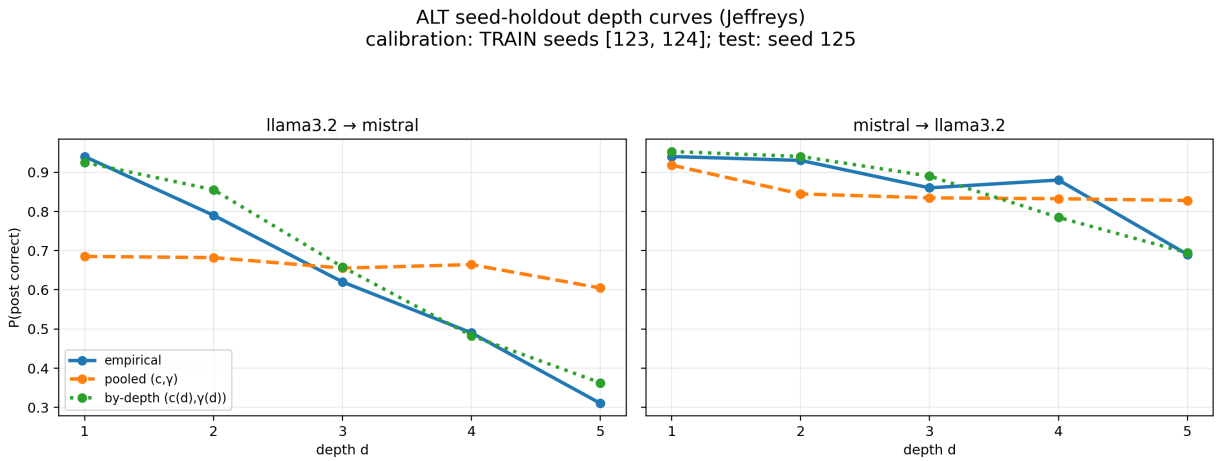


Figure 2: ALT seed-holdout depth profiles under Jeffreys smoothing (train seeds 123/124; test seed 125). Left: `llama3.2`→`mistral`. Right: `mistral`→`llama3.2`. For each depth bin d, the solid curve shows empirical post accuracy $\hat{p}_1(d)$ on the held-out seed. The dashed curve plots one-step predictions using a pooled $(\hat{c}, \hat{\gamma})$ fit on the training seeds, evaluated at the held-out $\hat{p}_0(d)$. The dotted curve plots predictions using depth-conditioned $(\hat{c}(d), \hat{\gamma}(d))$ fit on the training seeds. Systematic deviations of the pooled curve across d illustrate mixture dependence when (c, γ) vary by depth, while the depth-conditioned predictions track the held-out profile up to finite-sample error.

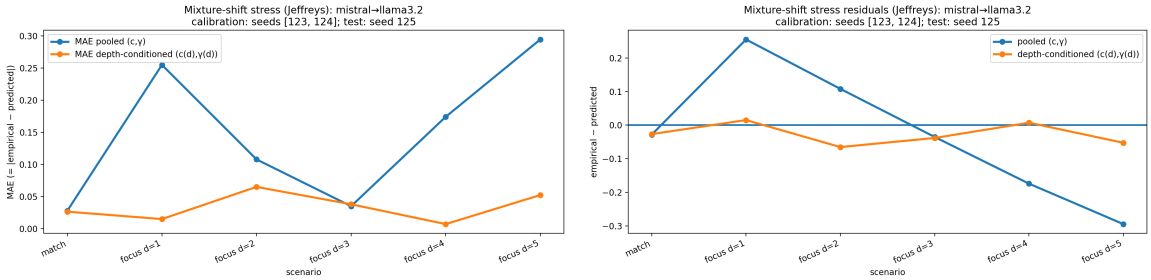
evaluation set reweighs depths, because it implicitly averages over heterogeneous slices. The consequence is practical rather than semantic: a pooled calibration can be accurate on a matched mixture yet exhibit signed, systematic out-of-sample bias under a shifted mixture, while a depth-conditioned calibration remains stable under the same counterfactual reweighings. The stress test below isolates this effect by changing only mixture weights over d, without making any additional model calls. What is notable is not merely that depth correlates with performance, but that conditioning on depth specifically identifies a regime variable under which the interface becomes stable and enables predictive transfer.

Specifically, the holdout test consists of keeping the trained mechanism fixed (fit on training seeds), and evaluating predictions on a held-out seed under counterfactual depth mixtures $w(d)$ that change only the evaluation weights, not the underlying per-depth accuracies. The test has a simple interpretation:

- **Matched mixture:** pooled and depth-conditioned predictors can agree.
- **Shifted mixture:** pooled predictors can drift with signed bias; depth-conditioned prediction is designed to remain stable.

Figure 3 and Tables 4–5 foreground the out-of-sample signature of mixture sensitivity under a controlled intervention, but they play different roles here. Figure 3 shows one representative seed-holdout fold for the directed pair mistral→llama3.2 (calibration: seeds 123/124; test: seed 125). Table 4 reports fold-averaged summaries for the directed pair llama3.2→mistral, and Table 5 reports the per-fold signed residuals $\hat{p}_1^{\text{emp}} - \hat{p}_1^{\text{pred}}$ for that same direction. Under matched mixtures, the pooled predictor can be highly accurate. Under mixture shift (counterfactual focus on a single depth at evaluation time, still on the held-out seed), the pooled predictor exhibits large, systematically signed errors, while the depth-conditioned predictor remains substantially more stable in both average and worst-case error. The reverse-direction summaries are reported in the Supplementary Material.

We emphasize one practical consequence: conditioning is not free. It trades mixture-induced bias for increased variance when conditional support is thin, because $c(d)$ is estimated on the error set and $\gamma(d)$ on the success set. This variance cost is a finite-sample issue rather than a failure of the interface itself: as sample size grows, the conditional denominators within each slice thicken and the depth-conditioned estimates stabilize; see Supplementary Figures S10 and S11.



(a) Mean absolute error (MAE) of one-step predictions under controlled mixture shift.

(b) Signed bias (mean residual) under the same scenarios.

Figure 3: Mixture-shift stress test for ALT (Jeffreys), shown for one representative seed-holdout fold of the directed pair mistral→llama3.2 (calibration: seeds 123/124; test: seed 125). "Pooled" uses a single $(\hat{c}, \hat{\gamma})$ estimated on the calibration mixture; "depth-conditioned" estimates $(\hat{c}(d), \hat{\gamma}(d))$ within each depth and aggregates predictions under the test mixture. Left: mean absolute error (MAE). Right: mean signed residual $\hat{p}_1^{\text{emp}} - \hat{p}_1^{\text{pred}}$.

scenario	MAE pooled (c, γ)	MAE by-depth $(c(d), \gamma(d))$
match (unif.)	0.0078	0.0084
focus avg (mean over $d = 1..5$)	0.0711	0.0375
focus worst (max over $d = 1..5$)	0.1408	0.0990

Table 4: Fold-averaged mixture-shift summaries for ALT in the directed pair llama3.2→mistral. "Match" denotes evaluation on the original uniform depth mixture; "focus d" denotes counterfactual reweighting to a delta mass on depth d. Reported values summarize prediction error over the seed-holdout folds.

test seed	scenario	resid pooled	resid by-depth
123	match (unif.)	-0.0105	-0.0071
123	focus d=1	+0.0385	+0.0136
123	focus d=2	+0.0730	-0.0058
123	focus d=3	+0.0620	+0.0425
123	focus d=4	-0.1211	-0.1420
123	focus d=5	-0.1051	+0.0564
124	match (unif.)	+0.0042	+0.0104
124	focus d=1	+0.0275	+0.0151
124	focus d=2	+0.0890	+0.0255
124	focus d=3	+0.0428	-0.0013
124	focus d=4	+0.0253	+0.0598
124	focus d=5	-0.1634	-0.0472
125	match (unif.)	+0.0085	+0.0076
125	focus d=1	+0.0219	-0.0127
125	focus d=2	+0.0854	-0.0101
125	focus d=3	+0.0255	-0.0301
125	focus d=4	+0.0477	+0.0954
125	focus d=5	-0.1379	-0.0042

Table 5: Per-fold signed residuals $\hat{p}_1^{\text{emp}} - \hat{p}_1^{\text{pred}}$ for the mixture-shift stress test in ALT, for the directed pair llama3.2→mistral. Rows correspond to evaluation mixtures and columns to held-out folds.

5.3 Verifier-style regimes: judging versus repairing are different channels

After ALT, we switch from transcript-conditioning under a solve instruction to protocols where the second model is instructed to *judge* a candidate. This instruction change is exactly the kind of intervention the two-rate abstraction is meant to isolate: the post decision is produced by a different behavioral contract, even when prompts and extraction are held fixed.

We study two closely related verifier regimes:

- VER (*verdict-only*): the verifier outputs only a binary verdict (`correct`/`incorrect`) on a single candidate.
- VER-FIX (*judge then repair*): if the candidate is judged incorrect, the verifier must solve and output a corrected final answer under the same formatting contract.

Both are logged per item with paired outcomes so they fit the same interface accounting.

Importantly, VER is a *labeling channel* if we interpret its verdict as the "post" bit, while VER-FIX is an *end-to-end correction channel*. This is not semantics: a model can be a good judge without being a reliable repairer, and vice versa. The point of this section is comparative rather than to introduce a new transfer stress test. We use the same paired-outcome interface to show that these verification-style regimes can be analyzed within the same channel framework while still occupying different locations in channel space. Figure 4 summarizes the verifier's accept/reject behavior directly: in the VER interpretation, the false positive and false negative rates are the same paired-outcome quantities viewed as a labeling channel, namely $\hat{c} = \Pr(E_1 = 1 \mid E_0 = 0)$ and $\hat{\gamma} = \Pr(E_1 = 0 \mid E_0 = 1)$ under the verdict-as-post-bit convention. Figure 5 and Table 6 then place VER and VER-FIX in the same channel comparison framework as ALT. Together they make the contrast concrete.

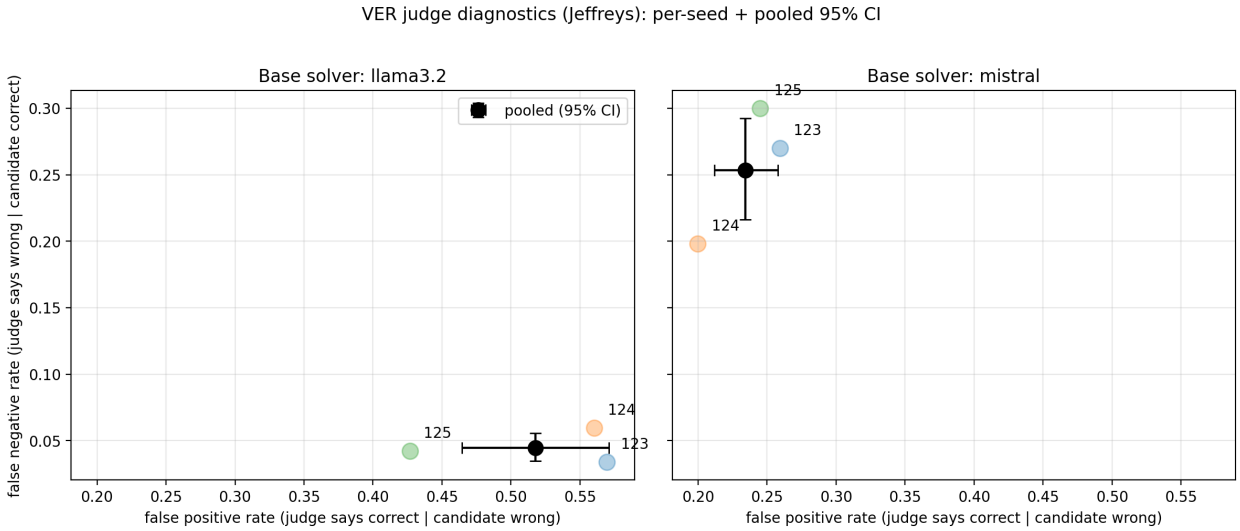


Figure 4: VER judge diagnostics (Jeffreys). Each point corresponds to one seed; axes are $\widehat{\mathrm{FPR}} = \Pr(\text{judge says correct} \mid \text{candidate wrong})$ and $\widehat{\mathrm{FNR}} = \Pr(\text{judge says wrong} \mid \text{candidate correct})$. Together they summarize the verifier's accept/reject behavior on a fixed base solver's candidates.

The comparison admits a straightforward interpretation under the correction–corruption decomposition. When the base solver is stronger, there is relatively little error mass left to recover, so even moderate corruption is heavily penalized; useful post-step protocols must therefore keep $\hat{\gamma}$ very low. When the base solver is weaker, there is much more error mass available for recovery, so large $\hat{c}$ can dominate even if corruption is not negligible. In this light, VER-FIX is notable because across both bases it combines high correction with very low corruption, whereas VER remains a labeling channel and does not reach the same end-to-end region of channel space. The same logic also recovers the earlier ALT pattern: alternation can

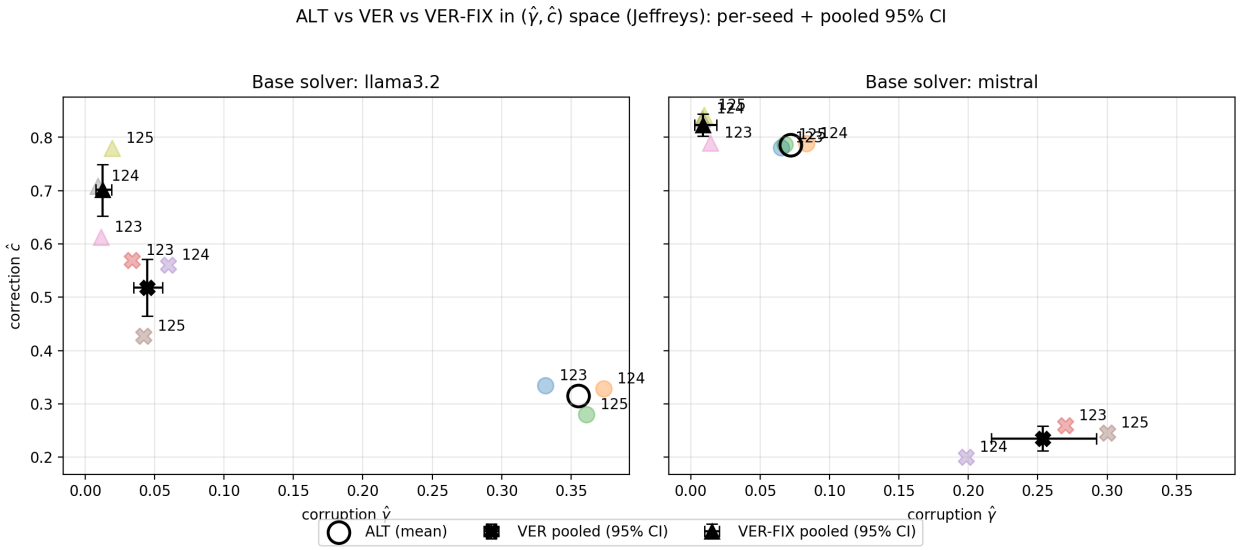


Figure 5: Interface comparison across regimes (Jeffreys), shown in $(\hat{\gamma}, \hat{c})$ space using the same base candidates per seed. Colored markers show pooled per-seed estimates for ALT, VER, and VER-FIX. Black markers show pooled regime summaries across seeds, with 95% confidence intervals for VER and VER-FIX; the open circle marks the mean ALT location across seeds. For VER, $\hat{c}$ and $\hat{\gamma}$ summarize the verifier's labeling behavior when the verdict is interpreted as a correctness bit (false-positive and false-negative rates). For VER-FIX, $(\hat{c}, \hat{\gamma})$ summarize end-to-end correctness after the fix attempt.

base	seed	regime	other	p_0	p_1	$\hat{c}$	$\hat{\gamma}$
llama3.2	123	ALT	mistral	0.8100	0.6050	0.3348	0.3316
llama3.2	123	VER	qwen2.5	0.8100	0.8917	0.5696	0.0339
llama3.2	123	VER-FIX	qwen2.5	0.8100	0.9183	0.6130	0.0113
llama3.2	124	ALT	mistral	0.8217	0.5733	0.3287	0.3735
llama3.2	124	VER	qwen2.5	0.8217	0.8733	0.5602	0.0597
llama3.2	124	VER-FIX	qwen2.5	0.8217	0.9417	0.7083	0.0091
llama3.2	125	ALT	mistral	0.8083	0.5700	0.2802	0.3611
llama3.2	125	VER	qwen2.5	0.8083	0.8567	0.4267	0.0422
llama3.2	125	VER-FIX	qwen2.5	0.8083	0.9433	0.7802	0.0195
mistral	123	ALT	llama3.2	0.2917	0.8267	0.7805	0.0653
mistral	123	VER	qwen2.5	0.2917	0.3967	0.2594	0.2699
mistral	123	VER-FIX	qwen2.5	0.2917	0.8483	0.7899	0.0142
mistral	124	ALT	llama3.2	0.2883	0.8267	0.7886	0.0833
mistral	124	VER	qwen2.5	0.2883	0.3733	0.1998	0.1983
mistral	124	VER-FIX	qwen2.5	0.2883	0.8817	0.8353	0.0086
mistral	125	ALT	llama3.2	0.2567	0.8250	0.7864	0.0677
mistral	125	VER	qwen2.5	0.2567	0.3617	0.2450	0.3000
mistral	125	VER-FIX	qwen2.5	0.2567	0.8817	0.8423	0.0097

Table 6: Cross-protocol comparison (Jeffreys), using the same base candidates per seed. For VER, $(\hat{c}, \hat{\gamma})$ summarize the verifier's labeling behavior when the verdict is interpreted as a correctness bit (false-positive and false-negative rates). For VER-FIX, $(\hat{c}, \hat{\gamma})$ summarize end-to-end correctness after the fix attempt.

be strongly helpful for a weak base solver yet damaging for a strong one when its corruption rate is not sufficiently controlled.

5.4 Judge-K on synthetic tasks: selection requires stochastic candidates

Selection protocols add a second, orthogonal bottleneck: even with negligible corruption, a judge can leave accuracy on the table by failing to select an available correct candidate. For this reason we treat JUDGE-K as a one-step channel *plus* an explicit oracle ceiling.

Operationally, a solver produces K candidate transcripts for the same instance and a judge selects one to output. In our runs, the judge itself is deterministic (temperature 0) so that selection noise is not judge noise. The only place we use temperature > 0 is candidate generation, where we sample the additional candidates at temperature 0.7 (otherwise best-of-K degenerates because candidates collapse). We report

$$(p_0,\ p_1,\ p_{\text{oracle}}),$$

where p_0 is single-candidate accuracy (a designated anchor candidate consistent with the unit used elsewhere), p_1 is judge-selected accuracy among K candidates, and p_{oracle} is the empirical best-of-K oracle accuracy on the realized candidate sets. The gap $p_{\text{oracle}} - p_1$ is oracle headroom.

On synthetic tasks, depth again makes heterogeneity visible, so we report both pooled and depth-conditioned fits together with seed-holdout prediction errors. For $K = 2$, however, the main point is not that conditioning uniformly improves transfer, but that the one-step channel remains predictive and that the usefulness of judge selection depends jointly on candidate diversity and judge reliability. The phase diagram in Figure 6 and Tables 7 and 8 make the mechanism explicit: Table 7 shows the available oracle headroom, Figure 6 shows the associated correction–corruption operating point, and Table 8 shows that this channel remains predictive out of sample.

Pair	Bin	N	$p_{m\,\text{pre}}$	$p_{m\,\text{post}}$	$p_{m\,\text{oracle}}$	gap	$\hat{c}$	$\hat{\gamma}$
`llama3.2` → `qwen2.5`	pooled	1800	0.813	0.831	0.883	0.052	0.099	0.0010
	2x2	300	0.633	0.707	0.730	0.023	0.203	0.0026
	depth_1	300	0.947	0.957	0.980	0.023	0.206	0.0018
	depth_2	300	0.937	0.937	0.960	0.023	0.025	0.0018
	depth_3	300	0.863	0.870	0.950	0.080	0.060	0.0019
	depth_4	300	0.810	0.813	0.890	0.077	0.026	0.0020
	depth_5	300	0.690	0.703	0.787	0.083	0.059	0.0072
`mistral` → `qwen2.5`	pooled	1800	0.279	0.289	0.348	0.059	0.017	0.0070
	2x2	300	0.020	0.027	0.030	0.003	0.008	0.0714
	depth_1	300	0.927	0.930	0.957	0.027	0.065	0.0018
	depth_2	300	0.300	0.320	0.403	0.083	0.031	0.0055
	depth_3	300	0.177	0.187	0.273	0.087	0.022	0.0463
	depth_4	300	0.113	0.127	0.197	0.070	0.017	0.0143
	depth_5	300	0.137	0.143	0.230	0.087	0.013	0.0357

Table 7: JUDGE-K pooled estimates for $K = 2$ (seeds 123/124/125 pooled). p_0 is solver accuracy for a single candidate; p_1 is judge-selected accuracy among K candidates; p_{oracle} is best-of-K oracle accuracy; gap $= p_{\text{oracle}} - p_1$ measures oracle headroom. Channel parameters are $\hat{c} = \Pr(E_1 = 1 \mid E_0 = 0)$ and $\hat{\gamma} = \Pr(E_1 = 0 \mid E_0 = 1)$ (Jeffreys-smoothed).

Pair	pooled MAE	pooled bias	by-bin wMAE	by-bin wbias	max-bin
`llama3.2` → `qwen2.5`	0.0050	0.0002	0.0112	0.0010	0.0500
`mistral` → `qwen2.5`	0.0039	-0.0000	0.0129	-0.0001	0.0329

Table 8: Three-fold seed holdout (seeds 123/124/125) for JUDGE-K at $K = 2$. "pooled" fits $(\hat{c}, \hat{\gamma})$ on two seeds and predicts the held-out seed's p_1 from its p_0. "by-bin" fits $(\hat{c}(d), \hat{\gamma}(d))$ per bin on the two training seeds and predicts $p_1(d)$ on the held-out seed; wMAE/wbias are weighted by bin sizes. "max-bin" is the worst absolute per-bin prediction error across all folds.

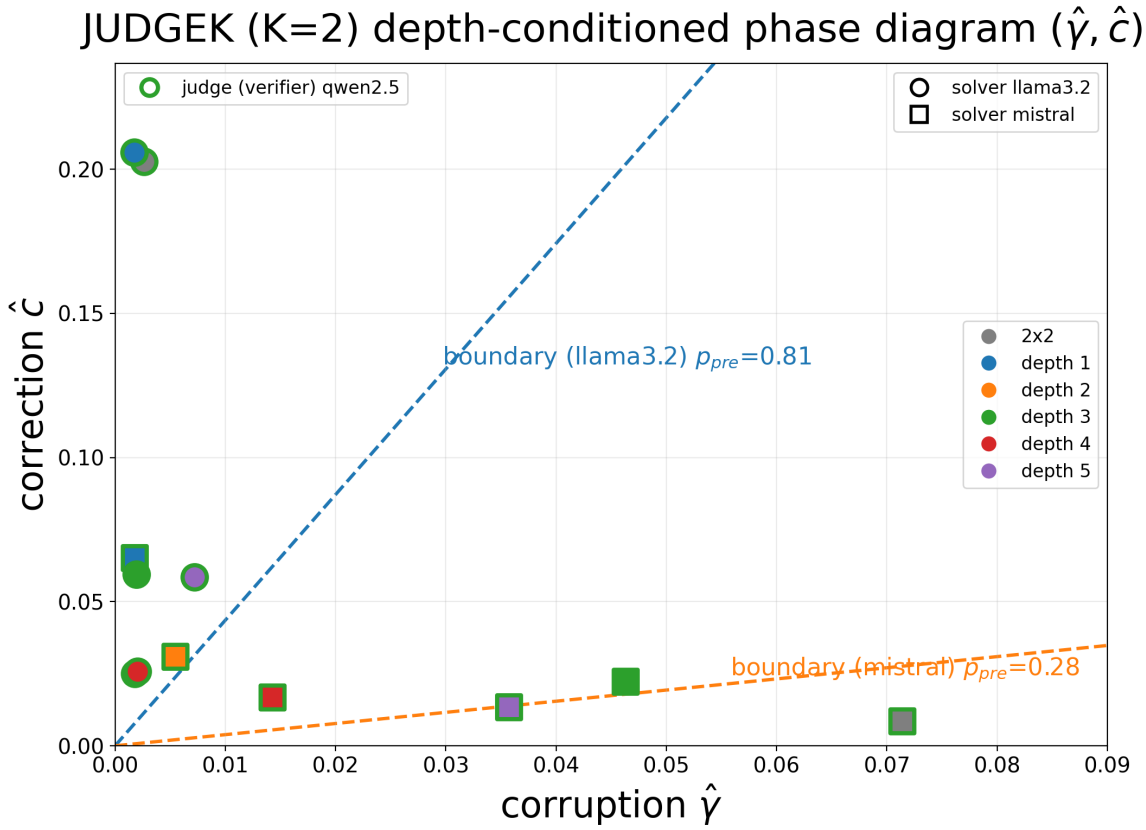


Figure 6: Phase diagram over synthetic slices for JUDGE-K on synthetic tasks ($K = 2$, Jeffreys). Each point is one (solver $\rightarrow$ judge, slice) estimate plotted at $(\hat{\gamma}, \hat{c})$, inferred from paired pre/post outcomes under judge selection. The slices comprise the arithmetic depth bins together with the 2×2 synthetic family. Overlaid lines show the phase boundary $c = \frac{p}{1-p}\gamma$ for representative baseline accuracies p.

To make the residual structure visible rather than only summarized, we report a representative by-bin breakdown in Table 9. This view makes clear where the remaining prediction errors concentrate across slices and shows that the largest deviations occur in the thinner bins.

Bin	mean |err|	mean err	max |err|
2x2	0.0349	0.0015	0.0500
depth_1	0.0079	0.0016	0.0143
depth_2	0.0004	0.0001	0.0009
depth_3	0.0055	0.0005	0.0074
depth_4	0.0037	0.0010	0.0071
depth_5	0.0147	0.0010	0.0236

Table 9: By-bin holdout errors for JUDGE-K with `llama3.2` $\rightarrow$ `qwen2.5` at $K = 2$ (3-fold seed holdout over seeds 123/124/125). Errors are $p_1^{m\,\mathrm{emp}}(d) - p_1^{m\,\mathrm{pred}}(d)$.

These by-bin holdout errors are modest at the scale of the corresponding slice evaluations. The larger values occur in thinner bins, where one conditioning row is relatively sparse, so they should be understood as finite-sample uncertainty rather than as failure of the one-step interface.

6 GSM8K: Evaluation Beyond Controlled Regimes

Synthetic tasks let us condition on a known regime variable (depth). GSM8K (Cobbe et al., 2021) does not. The question is therefore different: not whether the one-step identity holds on the observed sample (it does, by construction), but whether the interface estimates are stable enough to support transfer diagnostics when the regime variable is only partially observed.

We evaluate JUDGE-K on GSM8K with solver `llama3.2` and judge `qwen2.5`. Section 6.1 studies the $K = 2$ setting, reporting the same triple $(p_0, p_1, p_{\text{oracle}})$ and Jeffreys-smoothed $(\hat{c}, \hat{\gamma})$; Section 6.2 audits presentation effects at $K = 4$ on a separate 200-item subset. Candidate generation uses temperature 0.7, while the judge call is deterministic (temperature 0). This keeps the meaning of the fitted interface clear: variability comes from the candidate set, not from judge randomness.

All GSM8K analyses in this section are performed on a fixed judged pool (Table 10). A *split* means a random partition of this pool into a calibration subset and a held-out evaluation subset at a specified train fraction (Table 12 reports n_{tr} and n_{te}). When we refer to a *mixture* on GSM8K, we mean the induced distribution over an observable proxy slice variable, here question-length bins measured in characters. Under random splits the proxy mixture matches in expectation, up to finite-sample fluctuation; under controlled reweighting the mixture is changed by design.

We organize the GSM8K results around four diagnostics aligned with the paper's thesis: (1) pooled channel estimates with a coarse proxy for heterogeneity (question length; Table 10); (2) a controlled proxy mixture-shift reweighting test (length reweighting; Table 11), mirroring the depth-mixture stress tests; (3) order-robust stability under random permutations of the judged pool (Figure 7); and (4) held-out transfer under i.i.d. splits (Table 12 and Figure 8). In addition, we include a targeted audit of positional bias in judge-based protocols (Section 6.2).

6.1 Error flow under JUDGE-K without regime labels

GSM8K does not provide a regime label analogous to depth, so we use question-length bins measured in characters as an observable proxy. The proxy is *not* assumed to be the true regime variable; it is used only to (i) test whether the induced interface parameters vary across observable slices and (ii) construct a controlled mixture-shift stress test by reweighting slices.

Let s denote a length bin. For any population (pooled or within s), we estimate $p_0 = \Pr(E_0{=}1)$ and $p_1 = \Pr(E_1{=}1)$. From paired outcomes (E_0, E_1) we estimate the two-rate channel $(\hat{c}, \hat{\gamma})$ (Jeffreys; Appendix A), and we predict the one-step post accuracy via

$$\hat{p}_1^{\text{pred}}(p_0; \hat{c}, \hat{\gamma}) = p_0 + (1 - p_0)\hat{c} - p_0\hat{\gamma}.$$

Table 10 reports pooled and binwise estimates and, crucially, the bin residuals

$$r(s) \;=\; p_1(s) - \hat{p}_1^{\text{pred}}\big(p_0(s); \hat{c}_{\text{pooled}}, \hat{\gamma}_{\text{pooled}}\big),$$

together with the standardized residual

$$z(s) \;=\; \frac{r(s)}{\text{SE}[p_1(s)]},$$

where $\text{SE}[p_1(s)]$ is the binomial standard error of $p_1(s)$. The binwise rates show that $(p_0, \hat{c}, \hat{\gamma})$ vary across proxy bins, so a pooled fit is, in principle, mixture-dependent. Even without access to the true regime variable, observable proxy heterogeneity is enough to make pooled reuse sensitive to the evaluation mixture. The magnitude of $r(s)$ should therefore be interpreted relative to sampling error: under matched mixtures (random splits), we expect pooled transfer to be accurate up to finite-sample noise, while under deliberate proxy-mixture shift (reweighting) we can induce a systematic bias.

Interpreting magnitudes requires separating *noise-limited* residuals from *bias-limited* mixture effects. In Table 10, $r(s)$ compares an empirical $p_1(s)$ (finite N_s) to a pooled-channel prediction, so even if the pooled mapping were exactly correct for bin s we would still expect $r(s)$ to fluctuate on the scale $\text{SE}[p_1(s)] \asymp$

Table 10: GSM8K pooled and question-length-binned estimates (Jeffreys-smoothed). We report p_0, p_1, p_{oracle}, inferred $(\hat{c}, \hat{\gamma})$, and the pooled-channel prediction $\hat{p}_1^{\text{pred}}(p_0(s); \hat{c}_{\text{pooled}}, \hat{\gamma}_{\text{pooled}})$. The table shows the residual $r(s) = p_1(s) - \hat{p}_1^{\text{pred}}(\cdot)$ together with the standardized residual $z = r(s)/\text{SE}[p_1(s)]$. The final row reports the pooled $K = 4$ standard-ordering audit subset used in the position-bias analysis (Section 6.2), included here to allow direct comparison of the inferred channel parameters across K.

bin	N	p_0	p_1	p_{oracle}	gap	$\hat{c}$	$\hat{\gamma}$	$\hat{p}_1^{\text{pred}}$	$r(s)$	z
pooled	1319	0.7180	0.7400	0.8036	0.0637	0.1193	0.0164	0.7399	–	–
(72.999, 157.0]	269	0.8364	0.8439	0.9108	0.0669	0.1000	0.0111	0.8423	0.0016	0.072
(157.0, 203.0]	263	0.7643	0.7833	0.8441	0.0608	0.1508	0.0223	0.7799	0.0034	0.133
(203.0, 245.0]	262	0.7176	0.7481	0.7977	0.0496	0.1400	0.0132	0.7395	0.0086	0.320
(245.0, 308.4]	261	0.7280	0.7510	0.8276	0.0766	0.1181	0.0131	0.7485	0.0024	0.091
(308.4, 848.0]	264	0.5417	0.5720	0.6364	0.0644	0.1107	0.0382	0.5875	-0.0155	-0.510
$K = 4$ audit subset (standard ordering, 200 items)										
K=4 pooled	200	0.7400	0.7950	0.8800	0.0850	0.2740	0.0230	–	–	–

$\sqrt{p_1(s)(1 - p_1(s))/N_s}$. Thus $|z| < 1$ should be read as *no detectable within-bin deviation at the current N_s*, not as evidence that $(\hat{c}_{\text{pooled}}, \hat{\gamma}_{\text{pooled}})$ holds exactly for every slice.

Table 11 isolates *mixture dependence* of pooling by reweighting bins and using deterministic slice aggregation. In the matched-mixture rows, both p_1^{true} and p_1^{pred} are computed under the *same* length mixture, so the residuals are ≈ 0 up to numerical precision; these 10^{-5} values are not sampling errors. In the cross-mixture rows, the $\approx 0.47\%$ shift reflects a population-level mixture-induced bias of the pooled interface: for fixed slice-specific rates and fixed reweighting, the discrepancy persists as sample size grows, even though the empirical estimates around it become less noisy. At the present per-bin sample sizes ($N_s \approx 260$), the estimated cross-mixture bias is smaller than the binomial noise scale ($\sim 2\%$–3%), so it need not be statistically detectable within bins, even though the reweighting experiment identifies a nonzero population-level discrepancy under the chosen slice aggregation.

Table 11: **Controlled proxy mixture shift on GSM8K (question-length bins).** We fit a pooled channel $(\hat{c}, \hat{\gamma})$ under a calibration length-mixture and evaluate it under a test length-mixture. p_1^{true} is the slice-aggregated truth under the test mixture (length-conditioned aggregation). p_1^{pred} uses the pooled channel fit under the calibration mixture and the test-mixture p_0. Residual is $p_1^{\text{true}} - p_1^{\text{pred}}$ (positive means pooled under-predicts).

scenario	$\hat{c}_{\text{pool}}$	$\hat{\gamma}_{\text{pool}}$	p_0^{test}	p_1^{true}	p_1^{pred}	resid	p_1^{cond}	$\text{resid}^{\text{cond}}$
cal:uniform → test:uniform	0.123342	0.018382	0.717577	0.739235	0.739221	0.000015	0.739235	0.000000
cal:short → test:short	0.126181	0.016227	0.755126	0.773789	0.773771	0.000019	0.773789	0.000000
cal:long → test:long	0.121155	0.020789	0.680028	0.704681	0.704657	0.000024	0.704681	0.000000
cal:short → test:long	0.126181	0.016227	0.680028	0.704681	0.709367	-0.004686	0.704681	0.000000
cal:long → test:short	0.121155	0.020789	0.755126	0.773789	0.769095	0.004695	0.773789	0.000000

Next, independent of any proxy choice, we emphasize stability checks that do not rely on a latent regime assumption. Figure 7 provides an audit: as we accumulate more judged items under random permutations of the evaluation order, the estimated rates and channel parameters settle into a stable range as n grows.

Figure 7 shows that, under random re-orderings of the judged pool, both the accuracy rates and the fitted channel estimates stabilize as n grows, supporting the use of pooled estimates for the subsequent held-out transfer check.

We then test transfer directly by holding out data: fit pooled $(\hat{c}, \hat{\gamma})$ on training splits and predict held-out p_1 from held-out p_0.

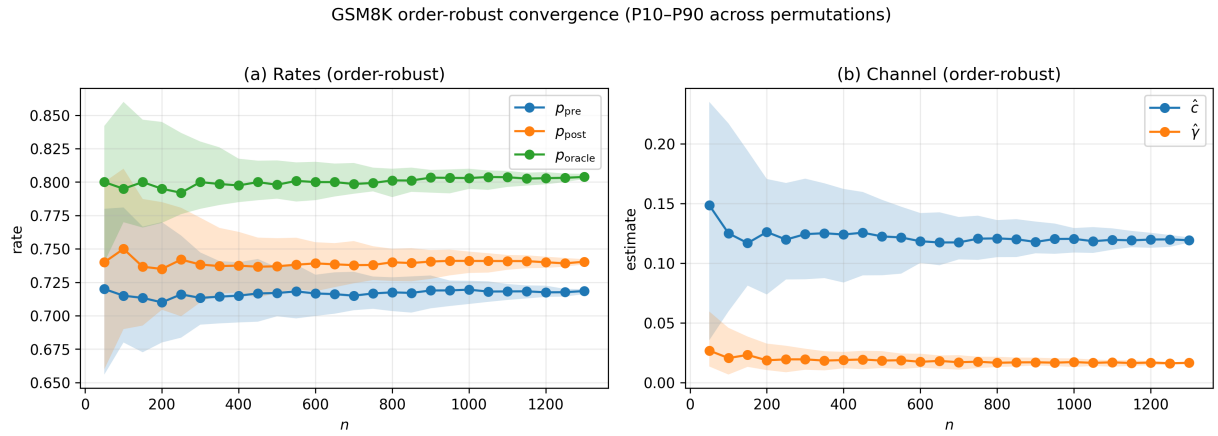


Figure 7: Order-robust convergence on GSM8K under random permutations of the judged set. (a) Rates p_0, p_1, and p_{oracle} versus n. (b) Jeffreys-smoothed channel estimates $(\hat{c}, \hat{\gamma})$ versus n. Lines show medians across a fixed set of random permutations; shaded bands show 10th–90th percentiles at each prefix length n.

Table 12: Held-out transfer on GSM8K: predict held-out $\hat{p}_1^{\text{emp}}$ from held-out $\hat{p}_0^{\text{emp}}$ using pooled $(\hat{c}, \hat{\gamma})$ fit on training splits. Errors are $\hat{p}_1^{\text{emp}} - \hat{p}_1^{\text{pred}}$. (Headers abbreviated for width: frac=train fraction; $n_{\text{tr}}/n_{\text{te}}$ train/test sizes; p10/p50/p90 are error quantiles over repeated random splits.)

setting	frac	n_{tr}	n_{te}	mae	bias	sd	$p10$	$p50$	$p90$
3-fold (equal folds)				0.0128	0.0006				
	0.1000	132.0000	1187.0000	0.0159	-0.0017	0.0192	-0.0238	-0.0029	0.0222
	0.2000	264.0000	1055.0000	0.0114	-0.0016	0.0142	-0.0225	-0.0012	0.0168
	0.3000	396.0000	923.0000	0.0095	0.0003	0.0121	-0.0151	0.0011	0.0148
	0.4000	528.0000	791.0000	0.0093	0.0018	0.0115	-0.0130	0.0022	0.0167
	0.5000	660.0000	659.0000	0.0090	0.0002	0.0112	-0.0125	-0.0008	0.0147
	0.6000	791.0000	528.0000	0.0094	0.0006	0.0116	-0.0142	0.0016	0.0144
	0.7000	923.0000	396.0000	0.0104	0.0000	0.0130	-0.0156	0.0005	0.0157
	0.8000	1055.0000	264.0000	0.0116	-0.0011	0.0140	-0.0190	-0.0024	0.0162
	0.9000	1187.0000	132.0000	0.0136	-0.0002	0.0169	-0.0229	-0.0009	0.0210

Because calibration/evaluation are i.i.d. splits of the same judged pool, this experiment evaluates transfer under matched proxy mixtures (i.e., without deliberate mixture shift). Table 12 and Figure 8(a) summarize errors across train fractions; Figure 8(b) shows residuals by question-length bin with approximate 95% normal-theory error bars, $\pm 1.96\,\text{SE}$, for $\hat{p}_1^{\text{emp}}(s)$.

6.2 Audit of presentation effects at $K = 4$

Finally, we include a targeted audit that is methodologically important: positional bias. With $K = 4$ on a 200-item subset, we compare the standard presentation order (candidate 1 is the baseline attempt) to a deterministic per-item permutation (`posshuffle`), mapping the judge's selected position back to the *original* candidate index. Table 13 and Figure 9 show a strong first-position prior under standard ordering: the judge selects the first presented candidate with probability 0.730, far above the uniform $1/K = 0.25$. Under `posshuffle`, this artifact is substantially reduced: the distribution over *original* chosen candidates moves toward the uniform reference, and the inferred channel $(\hat{c}, \hat{\gamma})$ changes accordingly. The change in p_1 over $N = 200$ items is not the main point; the positional statistics demonstrate that presentation effects can materially alter the measured interface, motivating explicit protocol controls of this type.

Two points are worth separating. First, the judge-selected accuracy changes by -2.5 points ($0.795 \to 0.770$), small relative to sampling variation at $N = 200$. Second, the presentation audit reveals a sharp change in

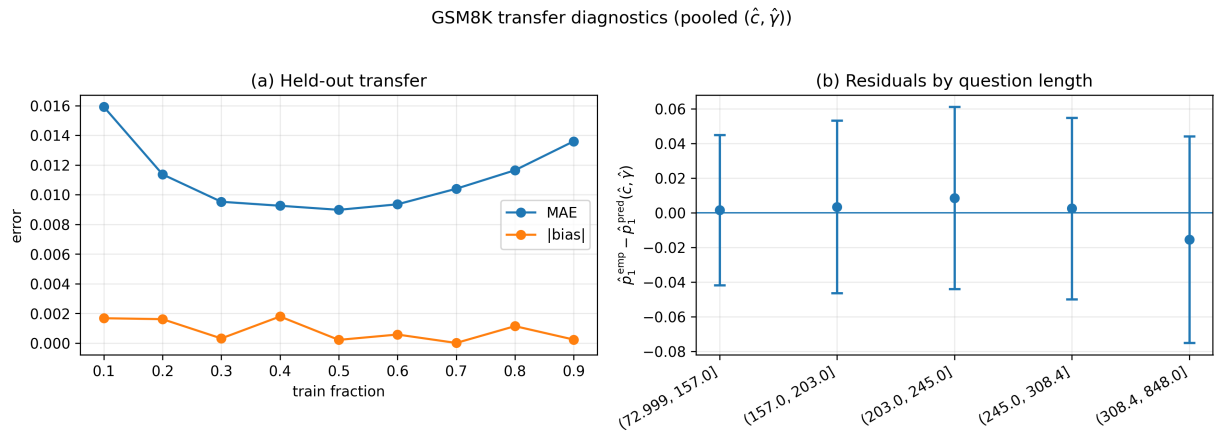


Figure 8: Transfer diagnostics for pooled $(\hat{c}, \hat{\gamma})$ on GSM8K under JUDGE-K. (a) Held-out transfer error versus training fraction, showing MAE and |bias| of $\hat{p}_1^{\text{emp}} - \hat{p}_1^{\text{pred}}$ over repeated random splits. (b) Residuals by question-length bin with error bars $\pm 1.96\,\text{SE}$ for $\hat{p}_1^{\text{emp}}(s)$ (binomial), plotting $\hat{p}_1^{\text{emp}}(s) - \hat{p}_1^{\text{pred}}(s)$.

setting	N	p_0	p_1	p_{oracle}	$\hat{c}$	$\hat{\gamma}$	P(choose presented 1)	P(choose anchor)	P(choose anchor | E_0=1)	P(choose anchor | E_0=0)
baseline	200	0.740	0.795	0.880	0.274	0.023	0.730	0.730	0.824	0.462
posshuffle	200	0.740	0.770	0.880	0.349	0.084	0.675	0.280	0.284	0.269
Δ (posshuffle - baseline)	–	0.000	-0.025	0.000	0.075	0.060	-0.055	-0.450	-0.541	-0.192

Table 13: Position-bias audit for GSM8K under JUDGE-K with $K = 4$. We compare the standard ordering (baseline) to deterministic per-item `posshuffle`. P(choose anchor) refers to the original anchor candidate after shuffled positions are mapped back to original indices.

which *original* candidate is selected: the probability of choosing the first *presented* candidate moves only modestly, from 0.730 to 0.675, whereas the probability of selecting *original* candidate 1 drops from 0.730 to 0.280, much closer to the uniform $1/K = 0.25$ baseline. Moreover, under `posshuffle` the dependence on E_0 is greatly reduced: under standard ordering, $P(\text{orig} = 1 \mid E_0{=}1) \approx 0.824$ and $P(\text{orig} = 1 \mid E_0{=}0) \approx 0.462$, whereas under `posshuffle` these become 0.284 and 0.269, respectively. This indicates that the standard protocol's apparent gain is partly confounded with presentation order. Correspondingly, the measured channel changes as well: under `posshuffle`, the estimated $(\hat{c}, \hat{\gamma})$ shifts together with the judge's selection behavior, showing that the interface as measured by this protocol is sensitive to presentation order.

Candidate-set structure matters for the effect of reshuffling. The aggregate position-bias audit above shows that presentation order changes measured behavior at K=4, but it does not yet distinguish cases in which reshuffling merely perturbs presentation from cases in which it changes access to genuine rescue opportunity. To separate these regimes, we stratify the same 200-item judged pool by candidate-set structure into three mutually exclusive strata: (i) *no diversity*, where all four candidates collapse to the same normalized answer; (ii) *diversity without oracle headroom*, where candidates differ but no alternative improves on the anchor; and (iii) *oracle headroom*, where the anchor is wrong but at least one candidate is correct. Figure 10 and Table 14 show that reshuffling has qualitatively different effects across these strata.

When all candidates collapse to the same answer, reshuffling has no effect on accuracy or on the inferred one-step channel, as expected. When candidates differ but offer no oracle headroom over the anchor, reshuffling lowers judge-selected accuracy from 0.716 to 0.630. In this stratum, the effective correction rate increases from $\hat{c} = 0.024$ to $\hat{c} = 0.119$, but the effective corruption rate increases much more sharply, from $\hat{\gamma} = 0.056$ to $\hat{\gamma} = 0.202$, consistent with loss of anchor-preserving bias rather than recovery of meaningful rescue. By contrast, when true oracle headroom exists, reshuffling improves accuracy from 0.500 to 0.571 by increasing the effective rescue rate from $\hat{c} = 0.500$ to $\hat{c} = 0.569$. In this sense, the effect of ordering in JUDGE-K is not uniformly harmful or uniformly benign: it depends on whether the candidate set contains recoverable headroom.

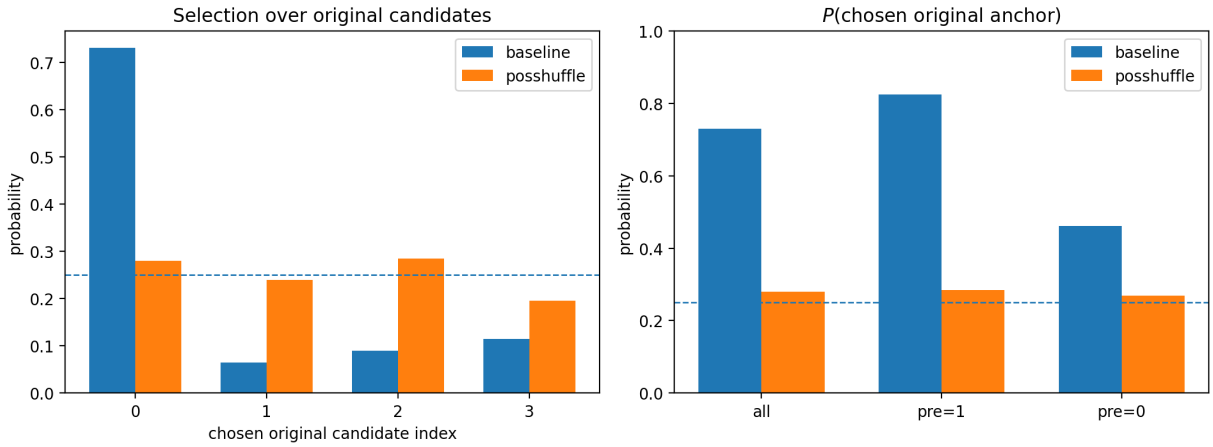


Figure 9: **Position-bias audit on GSM8K under judge-K ($K = 4$, 200 items).** We compare the standard presentation order (candidate 1 is the baseline attempt) to a deterministic per-item *posshuffle* that permutes candidate order before judging and maps selections back to original indices. *Left:* distribution over **original** chosen candidates; standard ordering concentrates strongly on original candidate 1, whereas posshuffle moves much closer to the uniform $1/K$ reference. *Right:* P(choose original candidate 1) overall and conditioned on E_0; under posshuffle this probability is ≈ 0.28, close to $1/K = 0.25$, and much less dependent on E_0, consistent with reduced presentation bias.

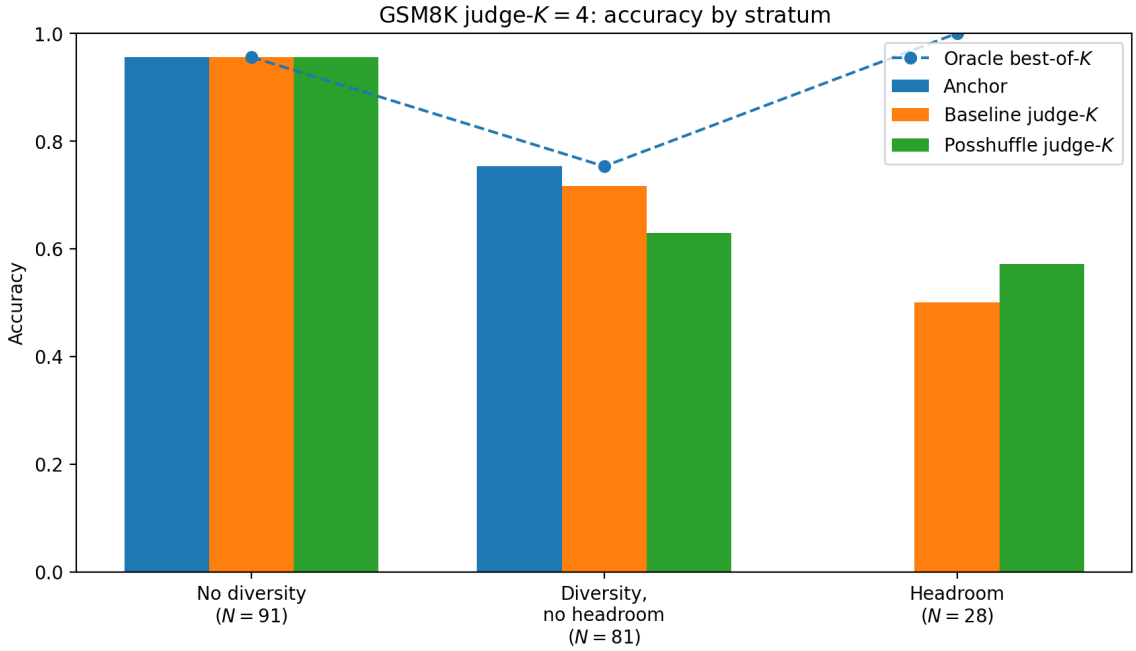


Figure 10: GSM8K judge-$K = 4$: accuracy by candidate-set stratum on the 200-item audit subset. Items are partitioned into no diversity, diversity without oracle headroom, and oracle headroom. "Anchor" is the baseline candidate, "Baseline judge-K" is standard ordering, "Posshuffle judge-K" is deterministic per-item reshuffling before judging, and "Oracle best-of-K" is the empirical best achievable accuracy on the realized candidate sets. Reshuffling has no effect when all candidates collapse to the same answer, hurts when diversity provides no true headroom, and helps when the candidate set contains recoverable oracle headroom.

Table 14: GSM8K judge-$K = 4$ stratified by candidate-set structure on the 200-item audit subset. Items are partitioned into: (i) no diversity, where all candidates collapse to the same normalized answer; (ii) diversity without oracle headroom; and (iii) oracle headroom, where the anchor is wrong but at least one candidate is correct. For each stratum and setting we report anchor accuracy p_0, judge-selected accuracy p_1, oracle best-of-K accuracy p_{oracle}, and the inferred one-step channel $(\hat{c}, \hat{\gamma})$ using Jeffreys smoothing. In the headroom stratum, $\hat{\gamma}$ is undefined because $E_0 = 1$ never occurs.

stratum	setting	N	p_0	p_1	p_{oracle}	$\hat{c}$	$\hat{\gamma}$
no diversity	baseline	91	0.956	0.956	0.956	0.100	0.006
no diversity	posshuffle	91	0.956	0.956	0.956	0.100	0.006
diversity, no headroom	baseline	81	0.753	0.716	0.753	0.024	0.056
diversity, no headroom	posshuffle	81	0.753	0.630	0.753	0.119	0.202
headroom	baseline	28	0.000	0.500	1.000	0.500	–
headroom	posshuffle	28	0.000	0.571	1.000	0.569	–

The reshuffling audit above shows that even within a single protocol family, changes in candidate-set structure and presentation can move the effective interface in qualitatively different directions: in some regimes the main effect is increased corruption, while in others it is increased rescue. This motivates a broader phase-plane reading of refinement behavior. Rather than treating reported gains or failures as isolated phenomena, we can interpret them as different operating points of the same correction–corruption tradeoff.

6.3 Phase-plane interpretation of refinement gains and failures

Recent work reports both gains from iterative refinement and failures of intrinsic self-correction. Huang et al. (2024) show that asking a model to revise its own reasoning *without external grounding* can substantially degrade accuracy (see Huang et al. (2024, Table 3)). In contrast, refinement pipelines that incorporate structured feedback and selection often report gains (e.g., Madaan et al. (2023)). These findings are not contradictory: they correspond to different operating points of the same one-step channel.

By Equation (7), improvement requires the correction rate to exceed the corruption rate scaled by $\frac{p_0}{1-p_0}$. This factor is the core mechanism, because it amplifies corruption. As the baseline accuracy rises, even modest γ requires a rapidly increasing c to offset it. At $p_0 = 0.955$ (GPT-4 on GSM8K under standard prompting in Huang et al. (2024)), $\frac{p_0}{1-p_0} \approx 21.2$, so break-even demands $c > 21.2\,\gamma$.

Equivalently, for any fixed corruption rate $\gamma > 0$, the break-even requirement $c_{\text{be}}(p_0) = \frac{p_0}{1-p_0}\gamma$ rises sharply with baseline accuracy, so high-baseline regimes are intrinsically fragile. This baseline sensitivity is exactly what distinguishes the two conditions Huang et al. (2024) test. Their *oracle* setting (see Huang et al. (2024, Table 2)) supplies reliable external feedback about correctness, which effectively reduces corruption and/or increases targeted correction, shifting the interface into a positive regime. Their intrinsic setting (see Huang et al. (2024, Table 3)) removes that grounding; revision then carries nontrivial corruption risk, and at high baseline the amplified corruption term can dominate.

Operating points from marginal reports. In many papers only the marginal accuracies (p_0, p_1) are reported, without paired outcomes. In that case (c, γ) are not identifiable, but the one-step law still yields an identifiable scalar summary, the *surplus above break-even*

$$\sigma \;\equiv\; \frac{p_1 - p_0}{1 - p_0} \;=\; c - \frac{p_0}{1 - p_0}\gamma.$$

A gain implies $\sigma > 0$ (operation above break-even) and a degradation implies $\sigma < 0$. Table 15 records representative operating points from prior work, taken directly from the cited tables.

Figure 11 visualizes the same baseline-sensitivity in the (γ, c) phase plane. Because prior work often reports only (p_0, p_1), we do *not* treat those papers as providing point estimates of (c, γ). Instead, we take our measured pooled mechanism $(\hat{c}, \hat{\gamma})$ (GSM8K JUDGE-K, Table 10) as a reference and plot the break-even

Table 15: **Representative operating points from prior work (marginals only).** Reported (p_0, p_1) determine the sign and magnitude of the surplus $\sigma = (p_1 - p_0)/(1 - p_0)$, but do not identify (c, γ) without paired outcomes.

paper & setting	dataset	p_0	p_1	$\Delta = p_1 - p_0$	σ
Huang et al. (2024) intrinsic (GPT-4, round 1)	GSM8K	0.955	0.915	-0.040	-0.889
Huang et al. (2024) oracle (GPT-4)	GSM8K	0.955	0.975	+0.020	+0.444
Madaan et al. (2023) (Table 1, ChatGPT)	GSM8K	0.748	0.750	+0.002†	+0.008†

† A 0.2 pp gain lies within ordinary sampling variability; included to illustrate the surplus formula, not as evidence of above-break-even operation.

requirement $c_{\mathrm{be}}(p) = \frac{p}{1-p}\hat{\gamma}$ as a function of baseline accuracy, along with the implied predicted gain/loss $\Delta_{\mathrm{pred}}(p) = (1-p)\hat{c} - p\hat{\gamma}$. The marked baselines p_0 from Table 15 illustrate how, under the measured GSM8K JUDGE-K interface $(\hat{c}, \hat{\gamma})$, a step that is beneficial at moderate baseline can become harmful at high baseline.

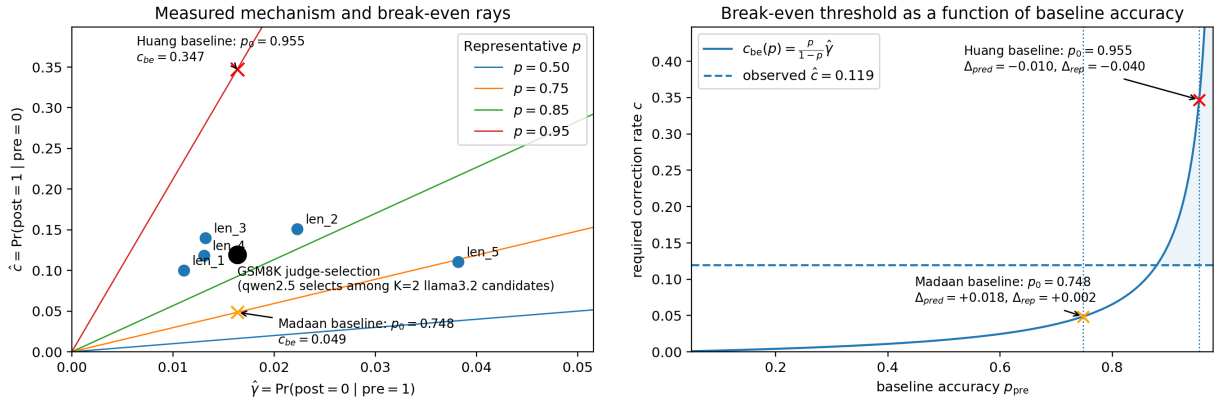


Figure 11: Phase structure of refinement regimes. Left: Empirical $(\hat{\gamma}, \hat{c})$ interface for our GSM8K JUDGE-K step with break-even rays $c = \frac{p}{1-p}\gamma$ for representative baselines p. Right: Required correction rate $c_{\mathrm{be}}(p) = \frac{p}{1-p}\hat{\gamma}$ as a function of baseline accuracy for fixed $(\hat{c}, \hat{\gamma})$. The markers illustrate the change in regime stringency from a moderate baseline (e.g., Madaan et al., Table 1) to a high baseline (e.g., Huang et al. (2024, Table 3)). They indicate the reported baselines p_0 and Δ from Table 15; they are not empirical (c, γ) estimates from those papers.

For Huang's intrinsic GPT-4 GSM8K round 1 (Huang et al. (2024, Table 3)), $p_0 = 0.955$ and $p_1 = 0.915$, so $\sigma \approx -0.889$, a strongly negative surplus consistent with amplified corruption at high baseline. By contrast, Huang's oracle GPT-4 GSM8K condition (Huang et al. (2024, Table 2)), $p_0 = 0.955$ and $p_1 = 0.975$, yields $\sigma \approx 0.444$, placing it decisively above break-even.

This framing also clarifies the connection to the MIT TACL analysis of self-correction (Kamoi et al., 2024). Their core message is that successful self-correction is rarely demonstrated using only feedback produced by prompted LLMs, and that self-correction works best when it relies on reliable external feedback or substantial fine-tuning. In our terms, purely intrinsic feedback typically induces non-negligible corruption γ without a compensating increase in c, and Eq. (7) explains why this failure becomes more likely as baselines rise. External feedback (oracle labels, verifiers, tools, unit tests) improves self-correction precisely by shifting the effective interface toward larger c and/or smaller γ, while fine-tuning can shift the interface itself.

This phase-plane reading also motivates the next step of the paper. A one-step interface can locate a protocol at a particular correction–corruption operating point, but stacked protocols raise a further question: whether these stepwise interfaces compose predictably at the correctness level, or whether downstream behavior depends on upstream history not captured by the intermediate correctness bit alone. Section 7 takes up exactly that question.

7 Beyond Single Steps: Composition and State Sufficiency

We now move from single-step protocols to two-step stacks. We show how adjacent-step correctness kernels compose to predict stacked accuracy, and we introduce a testable state-sufficiency test: whether the correctness bit E_1 is an adequate state variable for predicting the second step. Empirically, we evaluate both accuracy-level transfer (seed-holdout) and kernel-level composition gaps.

7.1 Two-step kernels and state sufficiency

So far we have analyzed a single interaction step as an observable pre→post correctness channel with rates (c, γ) and an exact one-step law linking p_0 to p_1. For composition we log three paired correctness indicators on the *same items*: baseline $E_0 \in \{0, 1\}$, intermediate correctness $E_1 \in \{0, 1\}$ after the first step, and final correctness $E_2 \in \{0, 1\}$ after the second step. For indices $a < b$, let T_{ab} denote the collapsed 2×2 correctness transition kernel with entries

$$T_{ab}(i, k) \;:=\; \Pr(E_b = k \mid E_a = i), \qquad i, k \in \{0, 1\}.$$

We estimate Jeffreys-smoothed kernels for the adjacent steps, T_{01} and T_{12}, and form the analytically composed kernel

$$T_{02}^{\text{comp}} \;=\; T_{01} T_{12}. \tag{11}$$

We view T_{ab} as a row-stochastic kernel acting on row vectors of correctness mass: if $\pi_a = \big(\Pr(E_a = 0), \Pr(E_a = 1)\big)$, then $\pi_b = \pi_a \, T_{ab}$. With this convention, stacking $0 \to 1$ followed by $1 \to 2$ yields $T_{02}^{\text{comp}} = T_{01} T_{12}$, whose implied two-rate parameters $(\hat{c}_{02}, \hat{\gamma}_{02})$ predict p_2 from p_0 via the same one-step law.

In parallel, we can estimate a *direct* collapsed kernel T_{02}^{direct} from paired (E_0, E_2) outcomes. Comparing T_{02}^{direct} to $T_{01} T_{12}$ is not merely a consistency check: it tests whether correctness is an adequate *state variable* for the two-step stack at this granularity.

The composed kernel $T_{01} T_{12}$ is what one would obtain if the two-step pipeline were effectively Markov after collapsing each intermediate answer to the correctness bit E_1 — that is, if the downstream transition depends on earlier history only through whether the intermediate answer was correct:

$$E_2 \;\perp\; E_0 \mid E_1.$$

(This conditional independence is not a consequence of the product rule; it is precisely the condition under which the product rule holds for conditional transitions. We use the two statements interchangeably below.) When this conditional-independence approximation holds, the direct two-step kernel estimated from paired (E_0, E_2) outcomes should match the product kernel up to sampling noise. When it fails, the failure is informative: it indicates that correctness alone is not a sufficient state variable for predicting the downstream step, and that the second step is sensitive to upstream history (e.g., transcript structure or error type) beyond what E_1 captures.

This distinction is also what separates one-step accounting from repeated-loop prediction. Multi-round scaling curves and stationary feedback-control models (Yang et al., 2025b; Liu & Meng, 2026) give useful calculations inside a reduced two-state model of repeated self-correction: each round is summarized by whether the current answer is correct or incorrect, together with correction and error-introduction rates. Such calculations become predictions for a real protocol stack only if the binary correctness state is sufficiently closed for the next transition. In general, however, two items with the same current correctness value may still differ in upstream history, trace quality, latent error type, or other information that affects the next step. The composition gap therefore provides the empirical check that an infinite-loop extrapolation requires but does not by itself supply: it tests whether the two-state transition law is reusable for the composed protocol being audited.

As defined in Section 4.2, we summarize the discrepancy between the direct collapsed kernel $\widehat{T}_{02}$ and the product kernel $\widehat{T}_{01}\widehat{T}_{12}$ using two complementary composition-gap summaries: the row-averaged ℓ_1 gap $\Delta_{\text{comp}}^{\text{mean}}$ and the entrywise maximum gap $\Delta_{\text{comp}}^{\max}$. The mean gap measures typical mismatch, while the maximum gap isolates the largest conditional discrepancy induced by assuming the correctness-level Markov factorization.

Support-sensitive variants and extended out-of-sample diagnostics are reported in Supplementary Material, Sec. S5.

7.2 Empirical composition diagnostics

A full pooled and by-bin kernel/accuracy closure audit (including the in-sample residual) is provided in the Supplementary Material (Table S1). Here we focus on transfer: we fit step kernels on training seeds and predict held-out p_2 from held-out p_0. Figure 12 and Table 16 show that pooled composed calibration can deviate systematically from a held-out depth profile under mixture dependence, while bin-conditioned composition reduces this bias. At shallow depths, even near the accuracy ceiling, the observed deviations remain within finite-sample uncertainty, which further supports the robustness of the measured interface.

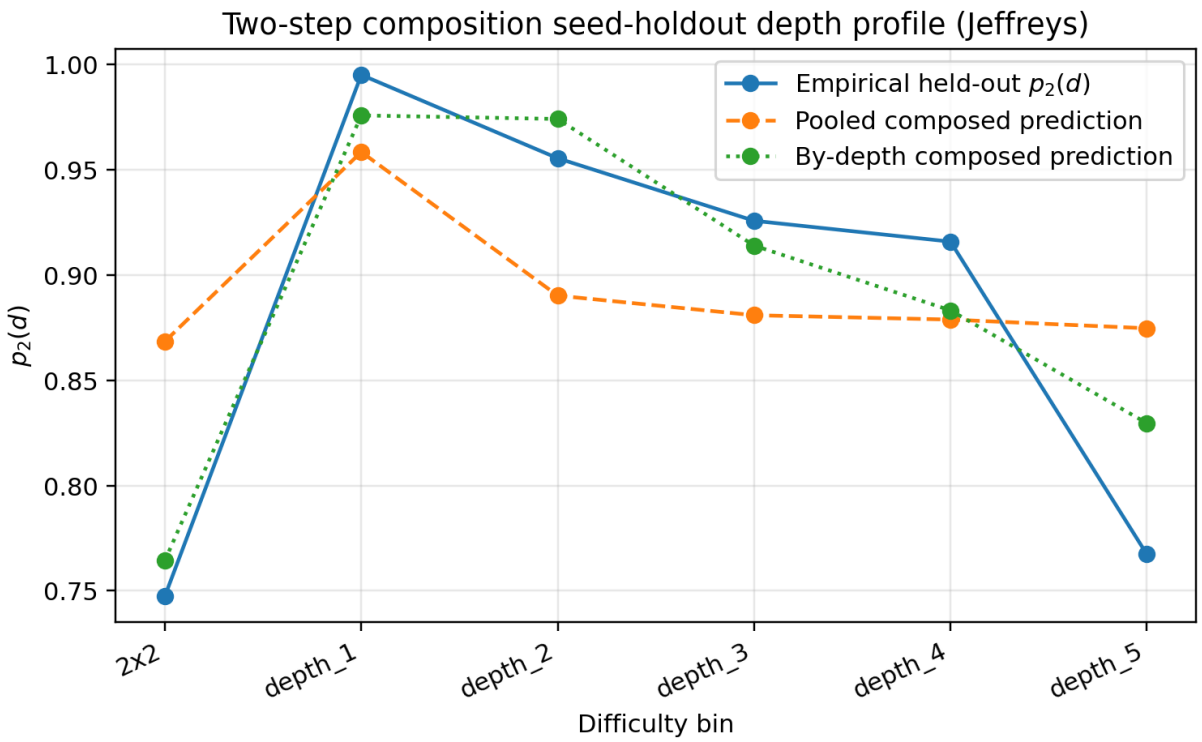


Figure 12: Two-step composition seed-holdout depth profile (Jeffreys). Calibration uses training seeds 123/124 and evaluation uses held-out seed 125. The blue curve is the empirical held-out $p_2(d)$. The orange dashed curve predicts $p_2(d)$ from held-out $p_0(d)$ using a single pooled composed kernel learned on the training seeds. The green dotted curve uses bin-conditioned composed kernels learned on the training seeds. Pooled calibration is mixture-dependent and can deviate systematically from the held-out profile, while bin-conditioning reduces this bias.

Table 17 reports aggregated composition-gap diagnostics by seed. Specifically, it summarizes the by-bin audits by reporting the n-weighted mean of the binwise $\Delta^{\text{mean}}_{\text{comp}}$ values and the maximum, across bins within a seed, of the binwise $\Delta^{\max}_{\text{comp}}$ values. These same-seed summaries should be distinguished from the transfer results above: a small aggregated same-seed residual does not imply that pooled composition transfers correctly across bins or mixtures. The relevant transfer test is the held-out depth-profile prediction in Figure 12 and Table 16, where pooled calibration can deviate systematically from the held-out profile and bin-conditioned composition reduces this bias.

By-bin composition-gap tables and figures are provided in the Supplementary Material (Table S2; Fig. S1). At the reported slice granularity, the observed composition gaps are small enough that correctness appears

bin	N	p_2^{emp}	p_2^{pred} (pooled)	p_2^{pred} (by-depth)	resid (pooled)	resid (by-depth)	MAE (pooled)	MAE (by-depth)
2x2	100	0.7475	0.8685	0.7644	-0.1209	-0.0169	0.1209	0.0169
depth_1	100	0.9950	0.9584	0.9758	0.0366	0.0193	0.0366	0.0193
depth_2	100	0.9554	0.8902	0.9741	0.0653	-0.0187	0.0653	0.0187
depth_3	100	0.9257	0.8809	0.9140	0.0449	0.0118	0.0449	0.0118
depth_4	100	0.9158	0.8788	0.8832	0.0370	0.0327	0.0370	0.0327
depth_5	100	0.7673	0.8747	0.8296	-0.1073	-0.0623	0.1073	0.0623

Table 16: Seed-holdout depth-profile transfer for composition (train seeds 123/124; test seed 125). Predictions use composed kernels fit on the training seeds. Residuals are $p_2^{\text{emp}} - p_2^{\text{pred}}$.

Seed	N_{total}	Weighted mean of binwise $\Delta_{\text{comp}}^{\text{mean}}$	Max binwise $\Delta_{\text{comp}}^{\text{max}}$
123	600	0.023342	0.032810
124	600	0.059199	0.131250
125	600	0.047845	0.095553

Table 17: Aggregated composition-gap audit by seed. The third column is the n-weighted mean of the binwise $\Delta_{\text{comp}}^{\text{mean}}$ values, and the fourth column is the maximum, across bins within a seed, of the binwise $\Delta_{\text{comp}}^{\text{max}}$ values.

to be an adequate approximate Markov state for this two-step stack, although thin-support bins can still produce localized deviations.

Two-step composition therefore yields two complementary empirical checks for stacked protocols. First, composing adjacent-step kernels yields a stacked-accuracy predictor that can be evaluated under seed-holdout; as in the one-step law, pooled calibration is mixture-dependent and can deviate under held-out depth profiles, while bin-conditioning reduces this bias when regime labels are available. Second, the composition gaps $\Delta_{\text{comp}}^{\text{mean}}$ and $\Delta_{\text{comp}}^{\text{max}}$ test whether correctness is a sufficient state variable for downstream behavior at the reported slice granularity. We do not propose a universal threshold for declaring Δ_{comp} "large enough to matter": the appropriate tolerance depends on the precision required for the stacked-accuracy forecast and on the sample size available to estimate $\widehat{T}_{02}$ directly. As a practical guideline, gaps that persist across seeds in the same bin and exceed the finite-sample noise level (quantified by subsampling in Figure S6) indicate that the correctness bit is not a sufficient Markov state; the appropriate response is to enrich the intermediate state (e.g., with judge confidence, candidate disagreement, or transcript signatures) rather than to dismiss the gap as noise. Identifying a principled, data-adaptive threshold remains an open direction. Supporting convergence, pooled out-of-sample and support-sensitivity summaries are reported in the Supplementary Material. In particular, the signed-gap convergence (Figures S8–S9) shows that pooled signed gaps center near zero with narrow bands by $n \approx 100$, supporting the interpretation that the observed composition residuals do not exhibit a systematic directional bias at this sample scale.

8 Discussion: Using the Interface to Design Better Protocols

The experiments support a pragmatic conclusion: paired pre→post outcome logging turns interaction protocols into *testable mechanisms*. The two-rate interface does not explain why a protocol works, and it does not claim that interaction creates new capability. Instead, it provides a compact accounting of how probability mass moves between correctness states and a way to ask whether the measured transition is stable enough to reuse.

The resulting audit is built on the correction–corruption separation: it first logs the transition, then asks how the measured rates transfer across seeds, slices, and mixtures, whether they are sensitive to presentation artifacts, how they behave under composition, and whether the corresponding protocol step should be used in deployment.

A key theme of the paper is that these audits are not only falsifiers. When the interface passes the appropriate transfer and invariance tests, it becomes a *design object*: a step can be summarized by $(\hat{c}, \hat{\gamma})$ (and, where needed, by slice-conditioned $(\hat{c}(s), \hat{\gamma}(s))$), compared against other steps, and composed into pipelines whose expected behavior can be forecast and validated.

8.1 From end-to-end gains to mechanism-level design

Protocol variants are often compared by $\Delta p = p_1 - p_0$ alone. The one-step law decomposes this gain into *rescued* mass $(1 - p_0)c$ and *destroyed* mass $p_0\gamma$. Two constructive consequences follow.

First, deployment decisions based on the measured gain are a principled default. Given a baseline accuracy estimate $\hat{p}_0$ on a target stream and a calibrated interface $(\hat{c}, \hat{\gamma})$, Eq. (6) yields an immediate deployment test: enable the interaction step only when the predicted gain

$$\widehat{\Delta p}(\hat{p}_0) = (1 - \hat{p}_0)\hat{c} - \hat{p}_0\hat{\gamma}$$

exceeds a cost-sensitive threshold. This is not heuristic: it is the unique gain implied by the measured error-flow rates under the correctness-bit abstraction. In regimes where intrinsic revision is known to fail (Huang et al., 2024; Kamoi et al., 2024), the interface typically reveals why: γ is non-negligible while p_0 is high, so the break-even boundary becomes stringent.

Gating in practice. To demonstrate that the interface supports deployment decisions—not only post-hoc diagnosis—we apply the gain rule to the ALT (mistral→llama3.2) regime on the depth-stratified synthetic suite using a leave-one-seed-out protocol. For each depth bin d, we fit $(\hat{c}(d), \hat{\gamma}(d))$ on two training seeds and compute the predicted gain

$$\widehat{\Delta p}(d) \;=\; (1 - \hat{p}_0(d))\,\hat{c}(d) - \hat{p}_0(d)\,\hat{\gamma}(d)$$

on the held-out seed. The resulting decision rule suppresses the ALT step at all five depths. Table 18 reports the corresponding holdout averages: always-on deployment yields mean accuracy 0.6487, well below the baseline 0.8493, while the gating rule recovers the baseline by refusing to invoke the harmful step, for a gain of 0.2007 over always-on. The practical point in this example is not selective depth gating but that the measured interface supports a concrete deployment decision—here, suppression of a harmful step—using only calibration-set statistics and no additional model calls at inference time.

Table 18: Gating experiment for ALT (mistral→llama3.2), leave-one-seed-out (seeds 123/124/125). "Gate?" reports the resulting deployment decision by depth. In this experiment the fitted $(\hat{c}(d), \hat{\gamma}(d))$ from training seeds predicts $\widehat{\Delta p}(d) \leq 0$ for the held-out seed at every depth, so the decision is OFF throughout. Accuracy values are averaged over the three holdout folds.

Depth	Gate?	Always-off	Always-on	Gated	Gain vs. on
1	OFF	0.9467	0.9333	0.9467	0.0133
2	OFF	0.9367	0.8367	0.9367	0.1000
3	OFF	0.8633	0.6500	0.8633	0.2133
4	OFF	0.8100	0.4767	0.8100	0.3333
5	OFF	0.6900	0.3467	0.6900	0.3433
Mean	—	0.8493	0.6487	**0.8493**	**0.2007**

Scope and proxy gating. The gating rule above uses depth as the regime label, which is known in the synthetic setting. In deployment without observed regime labels, gating requires an observable proxy. On GSM8K, a proxy quality analysis over several observable features (question length in characters and tokens, number counts, sentence count, and structural keyword score) shows that proxy strength matters substantially for whether gating is useful in practice.

Among the proxies we examined, sentence count produced the strongest observable heterogeneity. Table 19 reports the bin-level gating decisions. The interface predicts suppression in the highest-sentence-count bin

(6–13 sentences; $\hat{c} = 0.040$, $\hat{\gamma} = 0.075$, predicted gain -0.022), and activation in the three lower bins; all four decisions are consistent with the empirical direction of $p_1 - p_0$. By contrast, question-length in characters—the canonical proxy used throughout the main GSM8K analysis—did not produce a detectable discrimination signal (maximum binwise $|z| = 0.51$, residual RMS $= 0.008$).

Table 19: Proxy gating on GSM8K using sentence count as a heterogeneity proxy (JUDGE-$K = 2$, Jeffreys). For each bin we report baseline accuracy p_0, judge-selected accuracy p_1, bin-conditioned $(\hat{c}, \hat{\gamma})$, the predicted gain $\Delta p = (1-p_0)\hat{c} - p_0\hat{\gamma}$, the gate decision, and whether the empirical direction of $p_1 - p_0$ is consistent with the prediction. The canonical proxy used throughout the main GSM8K analysis (question character length) is unchanged; sentence count is used here because it produces the strongest observable discrimination signal among the proxies examined.

Bin (sent. count)	N	p_0	p_1	$\hat{c}$	$\hat{\gamma}$	Pred. gain	Gate	Consistent?
1–3	682	0.7771	0.8035	0.1405	0.0066	+0.026	ON	✓
4	342	0.7135	0.7310	0.1263	0.0265	+0.017	ON	✓
5	161	0.6273	0.6770	0.1557	0.0147	+0.049	ON	✓
6–13	134	0.5373	0.5149	0.0397	0.0753	−0.022	OFF	✓

The practical lesson is that the same decision logic extends to natural benchmarks, but its value depends on whether an available observable proxy captures genuine variation in the underlying interface. On GSM8K, sentence count provides that discrimination; question length does not.

Second, steps can be compared even when they have different semantics. ALT, VER, and JUDGE are operationally different, but they share the same observable channel. This makes it possible to compare steps by their correction–corruption tradeoff rather than only by end-to-end deltas. For example, verification grounded in explicit checking can move the interface toward high c and low γ (Lightman et al., 2024; Cobbe et al., 2021), while selection protocols may exhibit low γ but still leave substantial oracle headroom (Sections 5.4 and 6.1), implying that selection quality is the bottleneck rather than candidate generation.

8.2 Mixture shift: conditioning as an engineering choice

Pooled $(\hat{c}, \hat{\gamma})$ is a *mixture-specific statistic* when slice behavior differs. The synthetic depth suite makes this visible and testable: even when slice kernels transfer across seeds, pooled calibration can drift systematically under depth-mixture changes (Section 5.4). This is a protocol-level instance of *label-conditional covariate shift* (Quiñonero-Candela et al., 2009): the distribution over difficulty levels s changes between calibration and evaluation, and because $c(s)$ and $\gamma(s)$ both depend on s, pooled $(\hat{c}, \hat{\gamma})$ becomes a mixture-weighted average tied to the calibration weights rather than a stable channel characteristic. The depth-conditioned fix is equivalent to importance-weighted estimation under the evaluation weights $w(d)$ (Sugiyama & Kawanabe, 2012).

Conditioning is therefore best viewed as an engineering knob rather than a theoretical flourish: it trades mixture-induced bias for variance under thin conditional support. In practice, the conservative choice is the coarsest partition that makes mixture-stress tests pass while keeping denominators adequate.

The GSM8K results (Section 6) show that the same logging and auditing discipline carries over to a natural benchmark without a known regime label. In this setting, question length is a much weaker proxy for latent heterogeneity than synthetic depth, so the induced mixture-shift effect is correspondingly modest: the controlled reweighting analysis yields a nonzero population-level bias ($\approx$0.47%, Table 11), but at the present per-bin sample sizes it remains below the sampling-noise floor. This is consistent with the broader lesson of Section 6: on natural data, the interface remains useful for auditing stability, transfer, and proxy dependence even when the available observable proxies are weaker than in the controlled synthetic setting. The GSM8K position-bias results (Section 6.2) also motivate a broader design question for selection protocols: how candidate headroom and presentation invariance jointly shape the measured interface.

8.3 Selection protocols: headroom, invariance, and what to optimize

Best-of-K protocols introduce a second bottleneck beyond (c, γ): *candidate headroom*. Reporting $(p_0, p_1, p_{\text{oracle}})$ separates (i) limits of candidate generation from (ii) limits of selection. This separation is operationally important: if headroom is large, compute is better spent on improving the judge (reducing γ and increasing c) or on better candidate diversity; if headroom is small, improving the judge cannot help much without changing generation.

Selection protocols also require explicit *invariance audits*. The position-bias experiment at $K = 4$ shows that the measured interface can change due to presentation artifacts rather than changes in candidate content, consistent with concerns raised in the broader "LLM-as-a-judge" literature (Zheng et al., 2023) and documented as a first-position and verbosity bias in systematic judge evaluations (Wang et al., 2024). In other words, a protocol can appear to "work" by exploiting a stable shortcut. The interface makes this visible by turning the artifact into a measurable shift in $(\hat{c}, \hat{\gamma})$.

8.4 Stacks and modular pipelines: when kernel composition is a tool

Many deployed systems are *cascades*: they route items through increasingly expensive steps until a stopping condition is met (Dohan et al., 2022). In this setting, the two-rate interface provides two complementary utilities.

First, when composition audits pass within the relevant slices, kernel products become a usable forecasting tool. To within finite-sample uncertainty, one can predict how a proposed stack will trade correction and corruption, and then test that prediction under seed holdout. This enables modular tuning: steps can be adjusted (or swapped) and their impact forecast before committing to a full end-to-end sweep.

Second, when composition audits fail in well-supported slices, the failure is informative: it diagnoses that the correctness bit is not a sufficient intermediate state for the downstream step. In such regimes, one can still use step-wise estimation for local auditing, but forecasting requires richer intermediate features (e.g., confidence, error type, or transcript signatures) or a redesign that reduces history dependence.

8.5 Targeted exploratory coding check

As a targeted exploratory check outside exact-match mathematics, we also apply the paired audit to executable Python repair tasks derived from the Mostly Basic Python Problems (MBPP) benchmark (Austin et al., 2021), where correctness is determined by unit tests rather than numeric answer extraction. The protocol has a simple draft–revision form. Starting from 960 MBPP tasks, Llama-3.2 first produces a candidate solution for each task, and a downstream Llama-3.2 revision call is then asked to return a final solution. The reported paired ablation conditions on the 357 tasks for which the initial Llama-3.2 draft passes the unit tests, and compares two matched downstream interfaces. In the draft-only interface, the revision call receives the task and the initially correct draft. In the helper-augmented interface, the same revision call receives the same task and draft together with a Mistral-generated helper artifact. Corruption rises from $28/357 = 0.078$ to $100/357 = 0.280$ when the same repair call is additionally shown the helper artifact, a paired increase of $+0.202$ (bootstrap 95% CI $[+0.154, +0.249]$).

The example illustrates a design-relevant distinction that a final pass-rate comparison would obscure. A final-score view would show that the helper-augmented interface performs worse on this slice, but not whether the loss comes from fewer repairs of initially wrong drafts, more corruptions of initially correct drafts, or both. Because this slice conditions on initially correct drafts and matches the task, draft, and downstream revision model across interfaces, the paired transition view isolates the second mechanism: the added helper context raises the number of corrupted drafts from 28 to 100 out of 357 (a net increase of 72); at the paired level, 83 initially-correct drafts fail under the helper interface but not under draft-only, against 11 in the reverse direction. Thus the design lesson is not merely that extra context can hurt, but that this interface change creates a corruption channel; a safer protocol would need to gate, constrain, or otherwise protect already-correct drafts rather than simply add the helper artifact. Details, prompt templates, the summary

table, row-level examples, and reproduction code are provided in the supplementary material (Appendix S9) and in the accompanying coding-check notebook.

8.6 Limitations and outlook

The interface is intentionally coarse. It does not capture *how* correctness changes, only *whether* it changes, and therefore cannot by itself yield mechanistic explanations. Most controlled experiments in this paper use exact-match mathematical scoring—depth-stratified integer arithmetic, 2×2 linear systems, and GSM8K—while the targeted coding check uses executable unit tests. Generalization to tasks with graded, subjective, or non-unique correctness remains open. The model can also miss structure when reasoning failures are compositional in a way that a single bit cannot summarize (a limitation closely related to known compositionality failures in transformers (Dziri et al., 2023)). For long-horizon scaffolded pipelines, the same issue appears as a state sufficiency question: repeated application of the one-step law is predictive only when the state used for composition is sufficient for the next transition. If hidden history, accumulated partial work, or prior interventions affect later behavior beyond the current correctness bit, the composition gap diagnoses the resulting failure of Markov-style composition. Extending the audit to such settings therefore means identifying richer states whose transition channels remain stable across steps.

Nevertheless, the paper's results support a simple methodological stance: treat protocol components as *interfaces* that must transfer, not as in-sample hacks. When a step's $(\hat{c}, \hat{\gamma})$ is stable under the shifts and invariances that matter for deployment, it becomes a reusable module that can be gated, conditioned, and composed. When it is not stable, the same diagnostics localize the failure mode (mixture dependence, contamination, or state insufficiency) and thereby suggest what must be changed to build an effective protocol rather than merely report an in-sample gain.

References

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc L, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Yuntao Bai, Saurav Kadavath, et al. Constitutional AI: Harmlessness from AI Feedback. arXiv preprint arXiv:2212.08073, 2022. URL https://arxiv.org/abs/2212.08073.

Karl Cobbe, Vineet Kosaraju, et al. Training Verifiers to Solve Math Word Problems. arXiv preprint arXiv:2110.14168, 2021. URL https://arxiv.org/abs/2110.14168.

David Dohan, Winnie Xu, et al. Language Model Cascades. arXiv preprint arXiv:2207.10342, 2022. URL https://arxiv.org/abs/2207.10342.

Yilun Du, Shuang Li, et al. Improving Factuality and Reasoning in Language Models through Multiagent Debate. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 11733–11763, 2024. URL https://openreview.net/forum?id=8XG3NbMvx1.

Nouha Dziri, Ximing Lu, et al. Faith and Fate: Limits of Transformers on Compositionality. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/deb3c28192f979302c157cb653c15e90-Paper-Conference.pdf.

Clémentine Fourrier, Nathan Habib, et al. Open LLM Leaderboard v2. https://huggingface.co/spaces/open-llm-leaderboard/open_llm_leaderboard, 2024.

Yonatan Geifman and Ran El-Yaniv. Selective Classification for Deep Neural Networks. In I. Guyon, U. Von Luxburg, et al. (eds.), *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, pp. 4878–4887. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/4a8423d5e91fda00bb7e46540e2b0cf1-Paper.pdf.

Jie Huang, Xinyun Chen, et al. Large Language Models Cannot Self-Correct Reasoning Yet. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=IkmD3fKBPQ.

Ryo Kamoi, Yusen Zhang, et al. When Can LLMs Actually Correct Their Own Mistakes? A Critical Survey of Self-Correction of LLMs. *Transactions of the Association for Computational Linguistics*, 12:1417–1440, 2024. doi: 10.1162/tacl_a_00713. URL https://aclanthology.org/2024.tacl-1.78/.

Hunter Lightman, Vineet Kosaraju, et al. Let's Verify Step by Step. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Aofan Liu and Jingxiang Meng. Self-Correction as Feedback Control: Error Dynamics, Stability Thresholds, and Prompt Interventions in LLMs. arXiv preprint arXiv:2604.22273, 2026. URL https://arxiv.org/abs/2604.22273.

Aman Madaan, Niket Tandon, et al. Self-Refine: Iterative Refinement with Self-Feedback. In A. Oh, T. Naumann, et al. (eds.), *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 46534–46594. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/91edff07232fb1b55a505a9e9f6c0ff3-Paper-Conference.pdf.

OpenAI, Josh Achiam, Steven Adler, et al. GPT-4 Technical Report, 2024a. URL https://arxiv.org/abs/2303.08774.

OpenAI, Aaron Hurst, Adam Lerer, et al. GPT-4o System Card, 2024b. URL https://arxiv.org/abs/2410.21276.

Liangming Pan, Michael Saxon, Wenda Xu, Deepak Nathani, Xinyi Wang, and William Yang Wang. Automatically Correcting Large Language Models: Surveying the Landscape of Diverse Automated Correction Strategies. *Transactions of the Association for Computational Linguistics*, 12:484–506, 2024. doi: 10.1162/tacl_a_00660. URL https://aclanthology.org/2024.tacl-1.27/.

Ethan Perez, Saffron Huang, et al. Red Teaming Language Models with Language Models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 3419–3448. Association for Computational Linguistics, 2022. URL https://aclanthology.org/2022.emnlp-main.225.pdf.

Joaquin Quiñonero-Candela, Masashi Sugiyama, et al. (eds.). *Dataset Shift in Machine Learning*. MIT Press, Cambridge, MA, 2009.

Noah Shinn, Federico Cassano, et al. Reflexion: Language Agents with Verbal Reinforcement Learning. In A. Oh, T. Naumann, et al. (eds.), *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 8634–8652. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/1b44b878bb782e6954cd888628510e90-Paper-Conference.pdf.

Masashi Sugiyama and Motoaki Kawanabe. *Machine Learning in Non-Stationary Environments: Introduction to Covariate Shift Adaptation*. MIT Press, Cambridge, MA, 2012.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 2nd edition, 2018.

Peiyi Wang, Lei Li, et al. Large Language Models are not Fair Evaluators. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9440–9450, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.511. URL https://aclanthology.org/2024.acl-long.511/.

Xuezhi Wang, Jason Wei, et al. Self-Consistency Improves Chain of Thought Reasoning in Language Models. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

Jason Wei, Xuezhi Wang, et al. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In S. Koyejo, S. Mohamed, et al. (eds.), *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 24824–24837. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/9d5609613524ecf4f15af0f7b31abca4-Paper-Conference.pdf.

Zhe Yang, Yichang Zhang, Yudong Wang, Ziyao Xu, Junyang Lin, and Zhifang Sui. Confidence v.s. Critique: A Decomposition of Self-Correction Capability for LLMs. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3998–4014, Vienna, Austria, 2025a. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.203. URL https://aclanthology.org/2025.acl-long.203/.

Zhe Yang, Yichang Zhang, Yudong Wang, Ziyao Xu, Junyang Lin, and Zhifang Sui. A Probabilistic Inference Scaling Theory for LLM Self-Correction. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 13573–13587, Suzhou, China, 2025b. Association for Computational Linguistics. doi: 10.18653/v1/2025.emnlp-main.685. URL https://aclanthology.org/2025.emnlp-main.685/.

Shunyu Yao, Dian Yu, et al. Tree of Thoughts: Deliberate Problem Solving with Large Language Models. In A. Oh, T. Naumann, et al. (eds.), *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 11809–11822. Curran Associates, Inc., 2023a. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/271db9922b8d1f4dd7aaef84ed5ac703-Paper-Conference.pdf.

Shunyu Yao, Jeffrey Zhao, et al. ReAct: Synergizing Reasoning and Acting in Language Models. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023b. URL https://openreview.net/pdf?id=WE_vluYUL-X.

Lianmin Zheng, Wei-Lin Chiang, et al. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 46595–46623. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/91f18a1287b398d378ef22505bf41832-Paper-Datasets_and_Benchmarks.pdf.

A Estimation details, Jeffreys smoothing, and prediction

Jeffreys-smoothed estimators for (c, γ), posterior credible intervals, one-step and mixture predictions, oracle headroom for JUDGE-K, and uncertainty propagation are specified in full in the Supplementary Material, Sections S3–S4. The estimators used throughout the paper are:

$$\hat{c} \;=\; \frac{n_{01} + \frac{1}{2}}{n_0 + 1}, \qquad \hat{\gamma} \;=\; \frac{n_{10} + \frac{1}{2}}{n_1 + 1}, \tag{12}$$

where $n_{ij} = \#\{x : E_0(x) = i, E_1(x) = j\}$, $n_0 = n_{00} + n_{01}$, and $n_1 = n_{10} + n_{11}$. Jeffreys smoothing prevents degenerate estimates when the conditioning event is rare and converges to the MLE as $n_0, n_1 \to \infty$.

B Scoring contract and parsing

The shared scoring interface is specified in full in the Supplementary Material, Section S2. All results derive from this contract; given the per-item JSONL logs, every table and figure can be recomputed deterministically.