WHEN ARE TOOL CALLS NECESSARY? COUNTERFACTUAL EVALUATION AND PRE-GENERATION EPISTEMIC GATING IN TOOL-AUGMENTED LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

When a tool-augmented language model outperforms its closed-book counterpart, the improvement is routinely credited to external computation or retrieval. We show that standard evaluations conflate two effects: the information a tool provides (Non-Parametric Information Gain, NPIG) and the effect of switching to a protocol in which the model must format a tool call before answering (Structural Scaffolding Effect, SSE). To separate them, we introduce an 8-Arm Counterfactual Matrix that holds Turn-1 tool calls byte-identical across real execution, self-simulation, schema-echo, empty return, and adversarial conflict. On solvable parametric tasks, action-first JSON calling lowers accuracy below closed-book answering. The largest group of lost items has a valid call but no checking after the tool returns (Stage-2 Post-Call Deliberation Suppression); smaller groups have call arguments corrupted before reasoning (Stage-1 Premature Schema Compression) or no call at all. Control experiments show that the loss is not caused by showing tool schemas or by our echo wording: it appears once a call is required, persists with neutral echo text, native tool templates, and grammar-constrained JSON, and mostly disappears when the model is asked to verify its answer after the tool returns. In a 40-item-per-benchmark study across 12 SOTA datasets with a local lookup/solver tool, an oracle that returns the gold answer on every call gives 29.9 percentage points higher accuracy than real execution; on the calls that return the gold answer, text LLMs jump by 55.7 percentage points over an echo ($k = 273$), whereas Qwen2.5-VL ($k = 24$) already reads the answer from the image closed-book and barely changes across real, echo, or empty returns. Finally, our truncation-corrected entropy (H_{corr}) predicts closed-book errors on lookup tasks, but requires per-task calibration and does not beat simpler prefix scores on actual tool benefit; when tools are fallible, prefix uncertainty is near chance and our passive 90-feature Logit-Lens probes fall to chance on unseen models and benchmarks.

1 INTRODUCTION

Modern language models (LLMs) and vision-language models (VLMs) increasingly solve complex tasks by interleaving text generation with external tool calls—querying SQL databases, executing Python code, and invoking web APIs (Schick et al., 2023; Patil et al., 2024; Qin et al., 2024; Yao et al., 2023). Yet existing benchmarks evaluate tool use in one of two incomplete ways. Single-turn benchmarks check only whether the first-turn tool call matches a reference syntax tree (Patil et al., 2025; Guo et al., 2024). End-to-end benchmarks simply compare a two-turn tool-augmented model against a single-turn closed-book baseline (Zhuang et al., 2023), measuring total accuracy gain without isolating intermediate causes. Neither protocol includes the counterfactual controls needed to answer a basic question: did the tool's returned output actually turn a wrong answer into a right one?

Moving from single-turn generation to a two-turn tool loop changes two factors (§3): **Structural Scaffolding Effect** (SSE), the effect of switching to a two-turn protocol that formats a Turn-1 action before answering (SSE_{pure} after matching the two-turn horizon via Arm 4b), and **Non-Parametric Information Gain** (NPIG), the gain from external execution results. Our primary contribution is an **8-Arm Counterfactual Evaluation Matrix** holding the Turn-1 call c_1 byte-identical across five Turn-2 observations (real execution, self-simulation, empty return, content-free echo, and adversarial

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

perturbation), plus single-turn (Arm 4) and two-turn reflection (Arm 4b) baselines. We apply it across 77,880 trajectories in **Suite I** (56,000 8-arm trajectories across 10 models from 3B to 72B plus 13,480 generalization rollouts; §4) and **Suite II** (1,920 trajectories on 40 fixed-seed items per benchmark across 12 SOTA LLM and VLM datasets with a local tool, 6,480 routing evaluations on MetaTool and SMART-ER, and a six-harness audit; §5):

- **1. Counterfactual Evaluation of Action-First Tool Use (§4–5):** Holding the Turn-1 call byte-identical across real execution and a content-free echo splits total tool gain into structural scaffolding (SSE) and tool information (NPIG; Definition 1). On parametric text math, requiring an action-first JSON call lowers accuracy below closed-book answering, by 12.0 pp on `calc-reach` and 6.6 pp on `gsm8k-easy` pooled over 10 models and by up to 30.0 pp per model (10 of 70 cells significant after Holm correction, none positive; Table 2). The 462 items lost under the echo split into: (i) Stage-1 Premature Schema Compression (29.9%), where writing the call before working out intermediate values corrupts arguments; (ii) Stage-2 Post-Call Deliberation Suppression (43.9%), where the call is valid but the model stops checking once a result returns; and (iii) Turn-1 Non-Invocation (26.2%; 53 of 121 from the two R1-Distill models run without thinking). Controls on five models show the loss is not from prompt schemas or echo wording and arises mostly after the call: a Turn-2 verification prompt removes about three quarters of the neutral-echo loss, vs. about a fifth from reasoning before the call. In Suite II with a non-visual tool (`solve_or_lookup`), Qwen2.5-VL also gets no gain from a content-free call (SSE $= -1.7$ pp over 480 items; Table 4), and on calls returning the gold answer, text LLMs jump from 22.7% under the echo to 78.4% with the real output (NPIG $= +55.7$ pp; $k = 273$), whereas Qwen2.5-VL ($k = 24$) solves 91.7% closed-book from the image and stays near that level under real (91.7%), echo (87.5%), and empty returns (83.3%; NPIG $= +4.2$ pp).
- **2. Call-Only Grading vs. Real Tool Execution (§5):** Auditing six open-source evaluation harnesses (`Gorilla/BFCL`, Patil et al., 2025; API-Bank, Li et al., 2023; MetaTool, Huang et al., 2024; `Open-SMARTAgent`; ToolBench, Qin et al., 2024; and τ-bench, Yao et al., 2025) shows that single-turn tracks check the Turn-1 call against a reference rather than grading a final answer after real execution, and none includes a Turn-1-matched echo control. In our 40-item-per-benchmark Suite II setup across 12 datasets and eight models with `solve_or_lookup`, an oracle that returns the gold answer on every call gives 29.9 pp higher Turn-2 accuracy than real execution (31.5 pp on the seven models with $\geq 10\%$ valid calls: 69.6% vs. 38.1%), and closed-book failure mispredicts actual tool benefit on 35%–87% of items per model.
- **3. Diagnostic Limits of Pre-Generation Tool Routing (§3–5):** Using our counterfactual labels, we test when a pre-generation gate can decide whether to call a tool. **(a) Attempt 1 (closed-book entropy):** Because APIs return only top-20 log-probabilities, single-bucket entropy drops when probability spreads into the tail. Using Shannon's grouping identity (Shannon, 1948), we derive a monotone **Grouping-Corrected Upper Bound** (H_{corr}; Proposition 1) that ranks closed-book errors well on lookups (AUROC 0.76–0.96), though it does not beat simpler prefix scores on actual tool benefit and requires per-task cross-validation for its +0.7 pp accuracy gain (at 29.6% fewer calls). **(b) Attempt 2 (uncertainty under fallible tools)** fails because tool benefit factors into closed-book failure times conditional tool recovery (Eq. 4): when our Suite II tool fixes only 18.5% of closed-book errors (8.7%–38.7% per model), closed-book uncertainty cannot predict tool success. **(c) Attempt 3 (passive Logit-Lens probes)** trains classifiers on 90 scalar features across 10 layers; their within-benchmark AUROC (0.66–0.88) reflects dataset and topic shortcuts (one-hot dataset name alone: 0.7317) and falls to chance (0.5006) under zero-shot transfer to unseen models and benchmarks.

2 Related Work

Tool-Augmented Language Models & Evaluation Harnesses. Equipping language models with external tools has progressed from self-supervised bootstrapping (Nakano et al., 2021; Parisi et al., 2022; Schick et al., 2023) to API instruction tuning (Patil et al., 2024; Qin et al., 2024; Song et al., 2023; Tang et al., 2023; Chen et al., 2023a; Liu et al., 2024) and multimodal program execution (Surís et al., 2023; Hu et al., 2024). While ToolQA (Zhuang et al., 2023), StableToolBench (Guo

et al., 2024), and BFCL (Patil et al., 2025) evaluate execution or AST call accuracy, recent work documents widespread over-invocation (Huang et al., 2024; Yakovlev et al., 2024), vision-language "pseudo-thinking" where VLMs ignore intermediate tool outputs (Yang et al., 2026; Wang et al., 2026; Ma et al., 2026), and sycophancy under tool-memory conflict (Cheng et al., 2026; Lyu et al., 2026). Our byte-identical counterfactual framework supplies an exact additive decomposition that separates these effects across text, code, tabular, and visual domains.

Structured Generation Constraints vs. Expressive Action Spaces. Production agent frameworks enforce strict JSON/XML schemas via constrained decoding (Willard & Louf, 2023; Zheng et al., 2024). However, Tam et al. (2024) showed that format constraints degrade mathematical reasoning relative to free-form text, while Wang et al. (2024a), Gao et al. (2023), and Chen et al. (2023b) showed that programmatic Python actions (CodeAct, PAL, PoT) or interleaved thoughts (ReAct, Yao et al., 2023) preserve intermediate variable bindings better than rigid JSON schemas. We separate pre-call argument errors from post-call deliberation suppression, and show that under action-first JSON calling text LLMs lose accuracy and VLMs show no gain under a content-free echo relative to closed-book answering.

Epistemic Uncertainty, Adaptive Retrieval & Tool Routing. Pre-generation gating builds on three distinct lineages: (i) supervised and heuristic tool triggering, including intent classifiers on human topic labels (MetaTool, Huang et al., 2024), reranked rejection tokens (`Toolken+`, Yakovlev et al., 2024), and dual-mode visual switches (`AutoTool`, Ma et al., 2026; `Beacon`, Wang et al., 2026); (ii) output-logit uncertainty and adaptive retrieval, such as entity popularity heuristics (Mallen et al., 2023), generation-time confidence thresholds (FLARE, Jiang et al., 2023), reflection tokens (Self-RAG, Asai et al., 2024), question complexity classifiers (Adaptive-RAG, Jeong et al., 2024), and semantic clustering (Kadavath et al., 2022; Kuhn et al., 2023; Fadeeva et al., 2023); and (iii) internal representation probing and steering, which inspect intermediate hidden states via the Logit Lens (nostalgebraist, 2020), Tuned Lens (Belrose et al., 2023), contrastive decoding (DoLa, Chuang et al., 2024), linear semantic probes (SEP, Kossen et al., 2024), training-free metacognitive triggers (`MeCo`, Li et al., 2025), or activation routers and steering (Sun et al., 2026).

Contributions Beyond Prior Tool Routers. Our work advances this literature in three ways. First, while prior tool routers train and evaluate on static topic labels or single-turn syntax matching, we evaluate routers against counterfactual tool benefit and find that MetaTool's annotated need labels are uncorrelated with the model's own closed-book failure ($\phi = -0.0414$). Second, whereas prior logit gates assume full-vocabulary softmax access or become non-monotone under production top-k APIs, we derive the monotone Grouping-Corrected Bound (H_{corr}) from Shannon's grouping identity (Shannon, 1948), which improves closed-book-error AUROC over the single-bucket estimator by $+0.015$ on average and by up to $+0.16$ on diffuse-tail cases, without retraining. Third, through seven diagnostic experiments across nine models, we use the benefit factorization (Eq. 4) to explain why closed-book uncertainty predicts tool benefit poorly when tools are fallible, and show that hidden-state probes fall to chance (0.5006) on unseen models and benchmarks.

3 Counterfactual Decomposition Framework

3.1 Observation Interventions

To separate what an external tool tells a model from how the act of formatting a tool call alters its reasoning, we intervene directly on the tool observation while holding the model's first-turn action constant (dependency graph $X \to (S, H) \to C_1 \to O \to Y$ in Appendix C). Consider an autoregressive language model $\mathcal{M}_\theta$ processing query $x \in \mathcal{X}$. In single-turn closed-book generation (Arm 4), the model generates free-form reasoning and final answer $y_0 \sim \mathcal{M}_\theta(\cdot \mid x, \mathcal{P}_{\text{closed}})$. In action-first real tool execution (Arm 1), the model first serializes a structured Turn-1 tool call $c_1 \sim \mathcal{M}_\theta(\cdot \mid x, \mathcal{P}_{\text{tool}})$, receives execution output $o_{\text{real}} = \mathcal{E}(c_1)$, and generates its Turn-2 answer $y_1 \sim \mathcal{M}_\theta(\cdot \mid x, c_1, o_{\text{real}})$. Our 8-Arm Counterfactual Matrix holds the query and action trace (x, c_1) byte-identical across five observation interventions (Arms 1, 2a, 2b, 2c, 3), varying only observation O, paired with natural-language baselines (Arms 4 and 4b) and pre-generation gating (Arm 5):

- **Arm 1 (Real Execution)**: Injects ground-truth execution output $o_{\text{real}} = \mathcal{E}(c_1)$ from deterministic Python, SQL, or lookup engines and grades the Turn-2 answer against gold y^*.

- **Arm 2a (Self-Simulation)**: Prompts an isolated model instance to predict tool return $o_{\text{sim}} \sim \mathcal{M}_\theta(\cdot \mid x, c_1, \mathcal{P}_{\text{sim}})$ from parametric memory, testing whether self-generated observations match real execution (conditioning on c_1 alone changes accuracy by only 0.15 pp; Appendix H).
- **Arm 2c (Schema-Echo)**: Returns a content-free echo o_{echo} repeating call arguments $c_1.\text{args}$ without execution (Appendix H compares four echo wordings). Adding an explicit Turn-2 verification instruction to the echo (Arm 2c-DirectCoT) tests how much of the loss comes from missing verification after the tool returns (Stage 2) rather than from a flawed call (Stage 1).
- **Arm 2b (Sensory-Null)**: Injects an empty payload $o_{\text{null}} = \{\}$ with no observation content.
- **Arm 3 (Adversarial Conflict)**: Injects a perturbed return $o_{\text{adv}} \neq o_{\text{real}}$ ($+15\%$ numerical shift, boolean flip, or distractor entity) to test sycophancy vs. parametric resistance.
- **Arm 4 (Closed-Book)** & **Arm 4b (Two-Turn Reflection)**: Single-turn free-form generation ($\mathcal{P}_{\text{closed}}$) and two-turn natural-language decomposition/verification (matching the two-turn horizon without JSON syntax constraints).
- **Arm 5 (Pre-Generation Epistemic Gate)**: Probes closed-book prefix uncertainty $H(x)$ before invocation; confident queries answer directly via Arm 4, while uncertain queries route to Arm 1.

What the Design Identifies. Because the Turn-1 call c_1 is identical across Arms 1, 2a, 2b, 2c, and 3, differences among these arms are controlled interventions on the observation alone: they measure the value of what the tool returns, given the call the model actually made. Contrasts with the natural-language Arms 4 and 4b are different in kind, because they change the whole protocol (instructions, tool definitions, output format, and token use). We therefore treat SSE as a protocol contrast rather than a pure schema effect, and test its likely cause with separate control ablations (Section 4). Telescoping these three quantities splits total tool gain into two additive experimental contrasts:

Definition 1 (Exact 3-, 4-, and 5-Way Additive Decompositions)**.** Let $\text{Acc}(\text{Arm } m) = \mathbb{E}_{x\sim\mathcal{D}}[\mathbb{I}(\hat{y}^{(m)}(x) = y^*(x))]$ denote expected task accuracy under counterfactual arm m. The Total Tool Gain $\Delta_{\text{total}} \equiv \text{Acc}(\text{Arm } 1) - \text{Acc}(\text{Arm } 4)$ decomposes into:

$$\Delta_{\text{total}} = \underbrace{\big[\text{Acc}(\text{Arm 2c}) - \text{Acc}(\text{Arm 4})\big]}_{\text{SSE (Composite Structural Scaffolding Effect)}} + \underbrace{\big[\text{Acc}(\text{Arm 1}) - \text{Acc}(\text{Arm 2c})\big]}_{\text{NPIG (Non-Parametric Information Gain)}}. \tag{1}$$

Comparing against Two-Turn Reflection (Arm 4b) separates the schema-first contrast $\text{SSE}_{\text{pure}} \equiv \text{Acc}(\text{Arm 2c}) - \text{Acc}(\text{Arm 4b})$ from horizon extension $\Delta_{\text{horizon}} \equiv \text{Acc}(\text{Arm 4b}) - \text{Acc}(\text{Arm 4})$, so that composite $\text{SSE} = \Delta_{\text{horizon}} + \text{SSE}_{\text{pure}}$. Intervening with Arm 2a further partitions information gain into $\text{NPIG} = \text{Hallucination Effect} + \text{NPIG}_{\text{cond}}$, where $\text{Hallucination Effect} = \text{Acc}(2a) - \text{Acc}(2c)$ measures the effect of self-generated observations, and $\text{NPIG}_{\text{cond}} = \text{Acc}(1) - \text{Acc}(2a)$ measures the gain from the real return over a self-simulated one, given the same call. Comparing against Arm 2c-DirectCoT further splits SSE_{pure} into a residual protocol contrast (Arm 2c-DirectCoT minus Arm 4b) and the effect of a Turn-2 verification instruction given the same call and echo; see Appendix C for complete derivations.

3.2 Grouping-Corrected Entropy Gating

Entropy as an Uncertainty Meter. Shannon entropy $H(p) = -\sum_i p_i \ln p_i$ of the next-token distribution $\mathbf{p} \in \Delta^{V-1}$ measures model uncertainty: a memorized fact ("The capital of France is" $\rightarrow$ "Paris") puts almost all probability on one token ($H \approx 0$), while an unmemorized lookup (a flight time or an obscure date) spreads probability across many candidates ($H \gg 0$).

Why Pre-Call Entropy Gating Is Needed. Because a required tool call can lower accuracy on solvable parametric tasks (Section 4), a deployed agent should decide before calling whether it needs the tool. Instead of generating an entire response closed-book, we generate just the first $K = 3$ prefix tokens following a direct answer trigger (e.g., `<answer>`) on the closed-book prompt $\mathcal{P}_{\text{closed}}$: if prefix entropy is low ($H < \tau$), the model answers directly without tools; if prefix entropy is high ($H \geq \tau$), the model is guessing, and execution immediately branches to an external tool. This check

is training-free but costs one extra closed-book request: 0.17 s per item end to end, or 2.4% of an ungated tool-calling episode (Section 5).

The Top-k Truncation Bug in Production APIs. In production serving engines (vLLM, SGLang), APIs expose only the top-k ($k = 20$) log-probabilities $\{\ell_{t,1}, \ldots, \ell_{t,k}\}$ out of the full vocabulary $V \in [32{,}768, 262{,}144]$. Standard uncertainty probes renormalize top-k probabilities ($\tilde{p}_{t,i} = p_{t,i} / \sum_{j=1}^{k} p_{t,j}$) to compute $H_{\text{top-norm}}(t)$, or lump the unobserved tail mass $P_{\text{tail}}(t) = 1 - \sum_{i=1}^{k} p_{t,i}$ into a single residual bucket, defining the single-bucket estimator $H_{\text{exact}}(t) = (1 - P_{\text{tail}}(t))H_{\text{top-norm}}(t) + H_{\text{binary}}(P_{\text{tail}}(t))$. Single-bucket tail entropy fails on flat distributions because it treats all probability mass outside the top k as if it landed on one token. When a model is guessing and probability spreads across thousands of tail tokens, H_{exact} perversely drops toward zero and signals high confidence.

The Shannon Grouping Correction. By the Shannon grouping axiom (Shannon, 1948), true full-vocabulary entropy equals observed top-k entropy plus tail uncertainty: $H(p_t) = H_{\text{exact}}(t) + P_{\text{tail}}(t)\, H(p_{\text{tail}})$. Because the API hides the true tail distribution p_{tail}, we upper-bound its uncertainty using the maximum-entropy uniform distribution across the remaining $(V - k)$ vocabulary tokens ($H(p_{\text{tail}}) \le \ln(V - k)$), adding $P_{\text{tail}}(t)\, \ln(V - k)$ to observed top-k entropy $H_{\text{exact}}(t)$ to obtain our **Grouping-Corrected Upper-Bound Entropy**:

$$H_{\text{corr}}(t) = H_{\text{exact}}(t) + P_{\text{tail}}(t)\, \ln(V - k), \qquad H_{\text{corr}}(x) = \frac{1}{K} \sum_{t=1}^{K} H_{\text{corr}}(t). \tag{2}$$

Proposition 1 (API Truncation Bias and Corrected Monotonicity)**.** Collapsing unobserved tokens into a single tail bucket (H_{exact}) causes measured entropy to paradoxically decrease whenever tail leakage exceeds

$$P_{\text{tail}}(t) > \frac{1}{1 + e^{H_{\text{top-norm}}(t)}} \;\ge\; \frac{1}{k+1} \;\approx\; 4.8\% \quad (\text{for } k = 20), \tag{3}$$

falsely signaling high confidence when model uncertainty is maximal. In contrast, the grouping-corrected bound $H_{\text{corr}}(t)$ (Equation 2) is strictly increasing in $P_{\text{tail}}(t)$ at fixed $H_{\text{top-norm}}(t)$ across all non-degenerate tail regimes ($P_{\text{tail}}(t) < 1 - k/V$), guaranteeing that measured uncertainty rises as probability mass escapes into the tail (proof in Appendix C).

4 Controlled Evaluation: Failure Taxonomy & Gating

Experimental Suites. We use two non-overlapping suites: one where tools reliably fix closed-book errors and one where they often fail. **Suite I** (Tables 1–2; Appendices A–F) evaluates 10 open-weights models (3B–72B) on seven controlled tasks (100 items per task, 8 arms, 56,000 trajectories), plus 13,480 generalization rollouts on seven 7B–8B models. **Suite II** (Section 5) evaluates eight models on 40 fixed-seed items per benchmark across 12 SOTA datasets ($N = 1{,}920$ two-turn trajectories: six text LLMs on six text benchmarks and two VLMs on six visual benchmarks) and nine models on MetaTool and SMART-ER ($N = 6{,}480$; all sampled items, tool code, prompts, and trajectories are released). All tools are local functions without network calls (2 ms per call): Suite I uses deterministic Python, SQL, and lookup functions over our own tables (the `wiki-mixed` tail facts are synthetic), and Suite II attaches one generic tool, `solve_or_lookup`, whose return contains the gold answer on 19% of text items (273 of 1,440) and 5% of VLM items (24 of 480).

Two-Stage Failure Taxonomy of Action-First Tool Calling. At a matched budget of 512 tokens per turn, an action-first JSON call answered by a content-free echo (Arm 2c) lowers `calc-reach` accuracy below closed-book answering (Arm 4) by 13.0–30.0 pp on four models from 3B to 72B (Tables 1–2 and 6; Figure 1). Doubling the budget to 1,024 tokens leaves closed-book accuracy unchanged (39.5% on average; Appendix H), and two-turn natural-language reflection (Arm 4b) stays within 6 pp of closed-book on 12 of 20 parametric model–task pairs (for Qwen2.5-3B on `calc-reach`, reflection alone costs 17.0 of the 30.0 pp). Across all 10 models, 462 items are right closed-book but wrong under the echo, while 173 move the other way (a net -4.1 pp). Grouping the 462 by Turn-1 call validity gives three failure modes: **(i) Stage-1 Premature Schema Compression (29.9%)**: writing the JSON call before working out intermediate values corrupts its arguments; CodeAct and ReAct avoid this, raising real-tool accuracy by up to 46.0 and 34.0 pp in a strict-parser

Table 1: **Eight-Arm Counterfactual Evaluation Matrix across 10 Open-Weights Models** on the seven-task controlled suite ($N = 700$ items per model; `Acc@lenient (Acc@strict)`, where strict scoring requires a well-formed `<tool_call>` block and a final `<answer>` tag and lenient scoring also recovers calls and answers from free text, e.g., when Coder-32B omits Turn-2 `<answer>` tags; CR is the share of items on which a tool call was executed). On parametric reasoning tasks, an action-first JSON call lowers accuracy below closed-book (Arm 2c $<$ Arm 4); on lookup tasks (`flight-easy`, `table-agg`, `wiki-mixed` tail), real execution yields large NPIG. Arm 5 uses one uncalibrated H_{exact} threshold for all models ($\tau_0 = 0.35$; 64.3% macro-accuracy at a 68.2% call rate); H_{corr} thresholds chosen per task and model by 5-fold cross-validation reach **66.7%** at a **58.2%** call rate (+0.7 pp over Arm 1 with 29.6% fewer calls). CIs and multiple-testing corrections are in Appendix I.

Model (Scale)	Arm 4: Closed-Book	Arm 4b: Reflect	Arm 2c: Echo	Arm 2a: Self-Sim	Arm 2b: Null	Arm 3: Conflict	Arm 1: Real Tool	Arm 5: Gate ($\tau_0 = 0.35$)
Qwen2.5-3B-Instruct (3B)	28.6% (20.7%)	26.3% (26.0%)	20.6% (9.4%)	18.7% (9.7%)	22.4% (11.1%)	26.9% (11.3%)	55.9% (19.7%)	55.4% (22.3%) @ 73.4% CR
Qwen2.5-7B-Instruct (7B)	33.1% (27.4%)	32.4% (32.3%)	25.4% (11.9%)	22.4% (15.0%)	26.1% (19.4%)	18.3% (6.3%)	83.3% (38.9%)	81.1% (42.7%) @ 79.1% CR
Qwen2.5-Coder-7B-Inst (7B)	34.0% (31.7%)	32.7% (32.6%)	29.3% (28.6%)	27.9% (27.0%)	28.6% (27.6%)	25.7% (20.4%)	60.4% (48.0%)	61.0% (48.1%) @ 47.4% CR
Gemma-3-27B-IT (27B)	38.1% (33.6%)	39.1% (37.6%)	34.6% (33.1%)	33.9% (32.7%)	33.9% (32.7%)	35.4% (32.9%)	92.3% (90.9%)	83.4% (81.0%) @ 52.3% CR
Qwen2.5-Coder-32B-Inst (32B)	38.3% (37.3%)	36.6% (36.4%)	35.1% (2.6%)	28.6% (2.4%)	32.3% (2.6%)	29.6% (2.9%)	86.7% (2.9%)	85.3% (16.9%) @ 77.7% CR
Qwen2.5-72B-Inst-AWQ (72B)	36.0% (33.6%)	36.6% (36.1%)	31.6% (25.3%)	26.6% (25.9%)	30.3% (25.0%)	33.0% (29.9%)	86.9% (83.3%)	83.1% (79.1%) @ 68.7% CR
DeepSeek-R1-Qwen-7B (7B)	34.4% (5.0%)	34.0% (8.4%)	31.0% (1.3%)	28.4% (1.3%)	32.3% (1.3%)	27.4% (1.1%)	39.7% (1.3%)	39.9% (1.7%) @ 57.0% CR
DeepSeek-R1-Llama-8B (8B)	29.3% (2.4%)	27.7% (21.3%)	27.6% (9.4%)	28.3% (10.0%)	27.7% (10.1%)	30.0% (11.3%)	50.1% (31.1%)	51.0% (28.7%) @ 52.1% CR
Llama-3.1-Tulu-3-8B (8B)	27.9% (27.9%)	30.7% (30.6%)	29.4% (28.7%)	25.6% (24.4%)	28.3% (27.6%)	29.1% (28.1%)	67.0% (63.1%)	63.7% (60.4%) @ 79.4% CR
Mistral-7B-Inst-v0.3 (7B)	15.9% (15.9%)	20.0% (19.7%)	9.7% (9.0%)	10.9% (8.6%)	6.4% (5.3%)	6.6% (5.4%)	37.6% (29.3%)	38.6% (31.4%) @ 94.9% CR

ablation (Appendix Tables 9A and 10). **(ii) Stage-2 Post-Call Deliberation Suppression (43.9%)**: the call is valid, but the model stops checking its work once a result returns; an explicit Turn-2 verification instruction raises echo-arm accuracy on `calc-reach` by 6–27 pp across the 10 models (Appendix Table 8). **(iii) Turn-1 Non-Invocation (26.2%)**: 53 of these 121 items come from the two R1-Distill models, run without thinking.

Controls for Prompt and Format Confounds. Arm 2c differs from Arm 4 in more than output format, so we ran control ablations on five models and four tasks (20 cells of 100 items; Appendix H). Adding tool definitions to the closed-book prompt without requiring a call changes accuracy by +2.6 pp on average, whereas requiring a Turn-1 call costs 11.3 pp. Replacing our original echo string with neutral JSON repeating only the call arguments makes the echo loss larger (-8.7 vs. -4.7 pp; -9.5 pp on the 15 cells with $\geq 50\%$ valid calls). An explicit Turn-2 verification instruction shrinks that neutral-echo loss to -2.1 pp (Qwen2.5-7B on `calc-reach`: 13%→47%, vs. 24% under placeholder echo B1), whereas reasoning before the call (ReAct) shrinks it only to -6.9 pp, so most of the loss arises after the tool returns (Stage 2). With real tools on Suite II, however, that verification prompt lowers accuracy by 2.4 pp and doubles Turn-2 time (Appendix J). With raw native chat-template calls, only Qwen2.5-7B emits valid calls reliably, and its echo loss grows (-39 vs. -23 pp with our tags). Grammar-constrained JSON leaves real-tool accuracy nearly unchanged (64.1% vs. 65.6%), so our lenient parser does not inflate Arm 1; however, forcing a bare Turn-1 call lowers `gsm8k-easy` below closed-book for all five models even with the real calculator (Qwen2.5-7B: 71%→43%). On R1-Distill-Llama-8B (not trained for tool calling), enabling native thinking with 2,048 tokens raises `calc-reach` Arm 1 accuracy from 31% to 65%, though it calls the tool on only 11% of items.

Closed-Book Uncertainty Gating When Tools Are Reliable (Gating Attempt 1). Because required calls can hurt solvable tasks, our first gate uses closed-book prefix uncertainty. On the lookup tasks `wiki-mixed`, `flight-easy`, and `hop-necessity`, the training-free H_{corr} gate ($K = 3$, $k = 20$) predicts closed-book errors with per-model AUROC 0.76–0.96 (0.73–0.87 over all seven tasks; Appendix I), partly because `wiki-mixed` pairs well-known facts with synthetic ones. The truncated estimators $H_{\text{top-norm}}$ and H_{exact} drop when tail mass is large (Proposition 1); the grouping correction improves closed-book error AUROC over $H_{\text{top-norm}}$ by +0.147 on R1-Llama-8B and over H_{exact} by +0.157 on diffuse-tail Tulu-3-8B `calc-reach` (0.824 vs. 0.667). At matched K and k, H_{corr} ranks closed-book errors better than top-1 confidence (0.794 vs. 0.764, $p < 0.001$), but does not beat $1 - p_1$ or $H_{\text{top-norm}}$ on actual tool benefit (0.726 vs. 0.751/0.758 on lookups). Moreover, its accuracy gain requires task-level calibration: 5-fold cross-validation per task and model reaches 66.7% at a 58.2% call rate vs. 66.0% at 82.6% for Arm 1 (+0.7 pp, 95% CI [+0.2, +1.3]; 29.6% fewer calls), whereas one threshold per model across tasks gives 65.8% (15.0% fewer calls), cross-model transfer gives -0.8 pp, and the uncalibrated gate of Table 1 (H_{exact}, $\tau_0 = 0.35$) gives 64.3% at 68.2%.

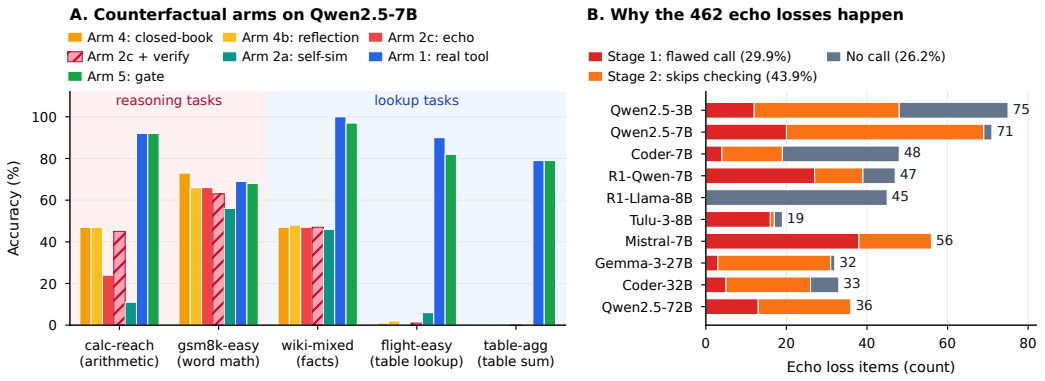

Figure 1: **Separating Structural Scaffolding (SSE) from Tool Information (NPIG) on the Controlled Suite. (A)** All arms for `Qwen2.5-7B-Instruct` on five of the seven tasks (all tasks in Table 6); the hatched bar adds a Turn-2 verification instruction to the echo arm. **(B)** The 462 echo loss items (correct closed-book, wrong under the echo) across 10 models (Table 8): Stage-1 Premature Schema Compression (flawed call, 29.9%), Stage-2 Post-Call Deliberation Suppression (valid call but no verification, 43.9%), and Turn-1 Non-Invocation (26.2%).

Table 2: **Representative Three-Way Decompositions on the Controlled Suite** including Two-Turn Natural Language Reflection (Arm 4b). Values report `Acc@lenient` [95% Wilson CI] (`Acc@strict`) and exact paired McNemar p-values for structural scaffolding (SSE $=$ Acc(2c) $-$ Acc(4)), hallucination effect (Acc(2a) $-$ Acc(2c)), and conditional observation gain (NPIG$_{\text{cond}}$ $=$ Acc(1) $-$ Acc(2a)), where direct NPIG $=$ Acc(1) $-$ Acc(2c) follows Equation 1. Differences are in percentage points; p-values are unadjusted (multiple-testing corrections in Appendix I).

Task	Model	Arm 4: Closed-Book (1T)	Arm 4b: Reflect (2T)	Arm 2c: Echo	Arm 2a: Self-Sim	Arm 1: Real Tool	SSE (2c − 4)	Hallu. Effect	NPIG$_{\text{cond}}$ (1 − 2a)	Total Gain
calc-reach	Qwen2.5-3B	42.0% [33,52] (21%)	25.0% (23%)	12.0% (10%)	12.0% (9%)	**60.0%** [50,69] (46%)	**−30.0**% ($p < .0001$)	+0.0% ($p = 1.00$)	**+48.0**% ($p < .0001$)	+18.0% ($p = .0001$)
calc-reach	Qwen2.5-7B	47.0% [38,57] (46%)	**47.0%** (47%)	24.0% (10%)	11.0% (11%)	**92.0%** [85,96] (66%)	**−23.0**% ($p < .0001$)	−13.0% ($p = .0002$)	**+81.0**% ($p < .0001$)	+45.0% ($p < .0001$)
calc-reach	Gemma-3-27B	67.0% [57,75] (60%)	**71.0%** (65%)	44.0% (43%)	42.0% (42%)	**79.0%** [70,86] (78%)	**−23.0**% ($p < .0001$)	−2.0% ($p = .8388$)	**+37.0**% ($p < .0001$)	+12.0% ($p = .0018$)
calc-reach	Qwen2.5-72B	51.0% [41,61] (50%)	**52.0%** (51%)	38.0% (38%)	18.0% (18%)	**67.0%** [57,75] (67%)	**−13.0**% ($p = .0002$)	−20.0% ($p < .0001$)	**+49.0**% ($p < .0001$)	+16.0% ($p = .0015$)
gsm8k-easy	Qwen2.5-72B	**82.0%** [73,88] (71%)	**85.0%** (85%)	75.0% (72%)	72.0% (68%)	75.0% [66,83] (71%)	−7.0% ($p = .0923$)	−3.0% ($p = .5488$)	+3.0% ($p = .4531$)	−7.0% ($p = .0923$)
gsm8k-easy	R1-Qwen-7B	**72.0%** [62,80] (18%)	**68.0%** (5%)	49.0% (4%)	53.0% (4%)	53.0% [43,62] (4%)	**−23.0**% ($p = .0003$)	+4.0% ($p = .424$)	+0.0% ($p = 1.00$)	**−19.0**% ($p = .0013$)
toolbelt-multi	Coder-32B	54.0% [44,63] (53%)	**48.0%** (48%)	39.0% (5%)	36.0% (4%)	**78.0%** [69,85] (7%)	**−15.0**% ($p = .0015$)	−3.0% ($p = .5811$)	**+42.0**% ($p < .0001$)	+24.0% ($p = .0004$)
flight-easy	Coder-32B	8.0% [4,15] (8%)	7.0% (7%)	10.0% (0%)	0.0% (0%)	**94.0%** [88,97] (0%)	+2.0% ($p = .6875$)	−10.0% ($p = .0020$)	**+94.0**% ($p < .0001$)	**+86.0**% ($p < .0001$)
table-agg	Qwen2.5-72B	0.0% [0,4] (0%)	0.0% (0%)	0.0% (0%)	0.0% (0%)	**99.0%** [95,100] (99%)	+0.0% ($p = 1.00$)	+0.0% ($p = 1.00$)	**+99.0**% ($p < .0001$)	**+99.0**% ($p < .0001$)
wiki-mixed	Qwen2.5-7B	47.0% [38,57] (22%)	48.0% (48%)	47.0% (21%)	46.0% (41%)	**100.0%** [96,100] (52%)	+0.0% ($p = 1.00$)	−1.0% ($p = 1.00$)	**+54.0**% ($p < .0001$)	**+53.0**% ($p < .0001$)

5 Open-Ended Benchmarks & Limits of Tool Gating

Why did our prefix uncertainty gate (H_{corr}) succeed on controlled lookups (Section 4), yet fail to yield a generalizable gate when deployed on open-ended SOTA benchmarks (**Gating Attempt 2**)? Because task accuracy is binary, a tool call turns a wrong answer into a right one if and only if the model fails closed-book ($\text{Acc}_{\text{Closed}} = 0$) and succeeds after tool execution ($\text{Acc}_{\text{Real}} = 1$). This yields the exact probability factorization:

$$\Pr\left(\text{Acc}_{\text{Real}} > \text{Acc}_{\text{Closed}}\right) = \underbrace{\Pr\left(\text{Acc}_{\text{Closed}} = 0\right)}_{\text{Closed-Book Failure (Internal Uncertainty)}} \times \underbrace{\Pr\left(\text{Acc}_{\text{Real}} = 1 \,\middle|\, \text{Acc}_{\text{Closed}} = 0\right)}_{\text{Conditional Tool Recovery (External + Turn-1 Extraction)}} . \quad (4)$$

On the controlled lookups (`wiki-mixed`, `flight-easy`), the tool fixes most closed-book errors (85% on `wiki-mixed` and 71% on `flight-easy`, pooled over 10 models), so tool benefit largely reduces to closed-book failure, which H_{corr} tracks. In contrast, our attached tool fixes only 18.5% of closed-book errors (8.7%–38.7% per model) across the 12 Suite II text and vision benchmarks (GSM8K, MATH-500, PopQA, MMLU-Pro, HotpotQA, MBPP; MathVista, ChartQA, DocVQA, RealWorldQA, ScienceQA-IMG, MMStar; $N = 1{,}920$ trajectories across eight models; Table 11) and two routing suites (MetaTool, SMART-ER; $N = 6{,}480$ items across nine models). This contrast exposes three evaluation pitfalls (Tables 3–5):

Pitfall 1: Call-Only Grading and Oracle Returns (Table 3). Multi-turn tracks of BFCL and τ-bench execute tools and score final states or answers, but widely used single-turn tracks (BFCL AST;

Table 3: **Left (A):** Source-code audit of six open-source tool evaluation and routing harnesses classifying whether they execute and grade Turn-2 answers (Pattern A), check the Turn-1 call against a reference (Pattern B), or omit a Turn-1-matched schema-echo control (Pattern C); MetaTool scores tool selection against annotated labels. **Right (B):** Overstatement from assuming calls succeed ($\mathrm{Acc}_{\mathrm{Oracle}} - \mathrm{Acc}_{\mathrm{Real}}$, where the oracle returns the gold answer at Turn 2) on five representative text benchmarks ($n = 240$ per benchmark across six text models) and pooled means across all 12 benchmarks ($N = 1{,}920$; live: valid-call rate $\geq 10\%$; per-model results, which include MBPP and the six VLM benchmarks, are in Appendix G, Table 11; deltas computed from unrounded counts).

Public Repository / Track	What Is Scored in Source Code	Turn-2 Graded?	Counterfactual Gap
Gorilla / BFCL (Single-Turn AST)	AST match of Turn-1 call vs. reference	No (Turn-1 AST)	**Pattern B + C (No Echo)**
Gorilla / BFCL (Multi-Turn Exec)	State + output match after execution	Yes (Multi-turn)	**Pattern A + C (No Echo)**
DAMO API-Bank (L1–L3)	Teacher-forced call check + ROUGE-L reply	No (isolated steps)	**Pattern B + C (No Echo)**
HowieHwong / MetaTool (Tool selection)	Need/selection vs. annotated label	N/A (no task answer)	**External label only**
Open-SMARTAgent	Answer match + static routing F1	Yes (No Echo)	**Pattern A + C (Static F1)**
ToolBench & **sierra / τ-bench**	Judge win rate / final DB state + answer	Yes (Multi-turn)	**Pattern A + C (No Echo)**

SOTA Benchmark (n)	Closed-Book	Real Turn-2	Oracle (o_{ora})	Overstatement
PopQA (240)	8.8%	47.5%	85.8%	+**38.3** pp
MMLU-Pro (240)	27.5%	22.9%	60.4%	+**37.5** pp
GSM8K (240)	14.6%	27.1%	61.2%	+**34.2** pp
MATH-500 (240)	12.1%	12.5%	45.8%	+**33.3** pp
HotpotQA (240)	20.8%	36.2%	69.2%	+**32.9** pp
7-Run Live Mean (1,680)	**33.2%**	**38.1%**	**69.6%**	+**31.5** pp
All 8 Models Mean (1,920)	**29.6%**	**34.7%**	**64.6%**	+**29.9** pp

Table 4: **Text vs. VLM Tool Use When Non-Visual Symbolic Tools Are Attached:** Item-level decomposition of apparent tool gain ($\Delta_{\mathrm{total}} = \mathrm{SSE} + \mathrm{NPIG}$, where SSE reflects two-turn call scaffolding) across all eight SOTA models ($N = 1{,}920$: all six text LLMs [$N = 1{,}440$; five live text LLMs with valid-call rate $\geq 10\%$, $N = 1{,}200$] vs. both VLMs [Qwen2.5-VL-3B/7B, $N = 480$]; subset rows condition on Turn-1 tool outcomes, so NPIG is an exact comparison given c_1; deltas computed from unrounded counts).

Modality Cohort	Item Subset	Items (k)	Closed-Book	Null (o_{null})	Echo (o_{echo})	Real (o_{real})	Total Gain (Real − Closed)	SSE (Echo − Closed)	NPIG (Real − Echo)
Text LLMs (All 6 Models, $N = 1{,}440$; 5 Live, $N = 1{,}200$)	All benchmark items (6 Models / 5 Live)	1,440 (1,200)	19.5% (22.5%)	15.5% (16.5%)	17.1% (18.7%)	27.0% (30.2%)	+7.5 (+7.7)	−2.4 (−3.8)	+**9.9** (+**11.5**)
	Syntactically valid Turn-1 call	1,018 (1,016)	22.8% (22.8%)	17.7% (17.7%)	19.9% (20.0%)	33.7% (33.7%)	+10.9 (+10.9)	−2.8 (−2.9)	+**13.8** (+**13.7**)
	Clean tool execution (no error)	702 (701)	25.5% (25.5%)	21.5% (21.5%)	22.9% (23.0%)	45.6% (45.5%)	+20.1 (+20.0)	−2.6 (−2.6)	+**22.6** (+**22.5**)
	Tool return contains gold answer	**273 (272)**	**20.9%** (21.0%)	**17.6%** (17.6%)	**22.7%** (22.8%)	**78.4%** (78.3%)	+**57.5** pp	+**1.8** pp (**3.2**%)	+**55.7** pp (**96.8**%)
Multimodal VLMs (2 Models, $N = 480$)	All benchmark items	480	60.0%	57.7%	58.3%	57.9%	−2.1 pp	−1.7 pp	−0.4 pp
	Syntactically valid Turn-1 call	94	53.2%	51.1%	56.4%	53.2%	+0.0 pp	+3.2 pp	−3.2 pp
	Clean tool execution (no error)	**50**	**74.0%**	**66.0%**	**72.0%**	**70.0%**	−**4.0** pp	−**2.0** pp	−**2.0** pp
	Tool return contains gold answer	**24**	**91.7%**	**83.3%**	**87.5%**	**91.7%**	+**0.0** pp	−**4.2** pp	+**4.2** pp

API-Bank's teacher-forced steps) check the call against a reference instead of grading a final answer after real execution. Only Open-SMARTAgent runs a closed-book arm on the same items, and none of the six audited repositories includes a Turn-1-matched echo control. In our 40-item-per-benchmark Suite II test with `solve_or_lookup`, an oracle that returns the gold answer whenever the tool is called yields 64.6% Turn-2 accuracy across all eight models, whereas real execution yields 34.7%: in this setup, assuming calls succeed overstates accuracy by 29.9 pp (31.5 pp for the seven models with $\geq 10\%$ valid calls: 69.6% vs. 38.1%), and closed-book failure mispredicts actual tool benefit on 83–209 of 240 items per model (35%–87%).

Pitfall 2: Omitting Echo Controls Masks How Differently Text LLMs and VLMs Use Tool Outputs (Table 4). Across all eight SOTA models ($N = 1{,}920$), action-first JSON calling answered by a content-free echo slightly lowers accuracy below closed-book (SSE $= -2.2$ pp: -2.4 pp on text LLMs, -1.7 pp on VLMs), whereas real execution adds $+7.3$ pp (NPIG; under ReAct in the four live text runs, SSE $= +8.6$ pp and NPIG $= +9.7$ pp). With our non-visual symbolic tool (`solve_or_lookup`), however, text LLMs and Qwen2.5-VL respond very differently to the tool return. On gold-return calls, text LLMs ($k = 273$) jump from 22.7% under the echo to 78.4% with the real output (NPIG $= +55.7$ pp, McNemar $p = 2.7 \times 10^{-43}$; SSE $= +1.8$ pp), whereas Qwen2.5-VL-3B/7B ($k = 24$) already solves 91.7% closed-book from the image and gets similar accuracy under real (91.7%), echo (87.5%, 95% CI [69.0%, 95.7%]), and empty returns (83.3%; NPIG $= +4.2$ pp, SSE $= -4.2$ pp). Across all 50 clean VLM calls (SSE $=$ NPIG $= -2.0$ pp) and all 480 VLM items (SSE $= -1.7$ pp, NPIG $= -0.4$ pp), text LLMs fail under the echo while Qwen2.5-VL still answers from the image (87.5% vs. 22.8% on live gold-return calls, Fisher $p = 3.21 \times 10^{-10}$; 72.0% vs. 23.0% on clean calls, $p = 2.65 \times 10^{-12}$), matching Yang et al. (2026). Because `solve_or_lookup` is a text tool, getting a gold return on VLM items means the model already read the numbers from the image in Turn 1; testing visual-crop tools and other VLM families (Llama-3.2-Vision, Pixtral) is future work.

Pitfall 3: Shortcut Overfitting in Passive Logit-Lens Routers (Gating Attempt 3, Table 5). When output uncertainty failed on Suite II (Gating Attempt 2), our third attempt trained passive classifiers on 90 scalar Logit-Lens entropy and JSD features across 10 layers and nine models (projected to

Table 5: **Why Our Passive 90-Feature Logit-Lens Routers Fail Under Counterfactual and OOD Controls (Gating Attempt 3):** Evaluating our 10-layer scalar Logit-Lens probe bank across nine models shows that high within-dataset AUROC scores reflect static topic and dataset shortcuts and fall to chance (0.501) under zero-shot transfer to unseen models and benchmarks.

Diagnostic Experiment	What Is Tested	Reported / IID Metric	Counterfactual / OOD Control	Null / Baseline	Finding
1. Static vs. Counterfactual Labels	MetaTool label vs. closed-book failure ($N = 2{,}160$)	0.8354 (Within-MetaTool AUROC)	$\phi = \mathbf{-0.0414}$ (vs. CB failure)	0.4698 (Int.→Ext. AUROC)	**No correlation w/ model's failures**
2. Cross-Domain Topic Shortcut	SMART-ER pooled vs. within-domain ($N = 4{,}320$)	0.8848 (Pooled Math/Time/Intent.)	**0.5063** (Within SMART-ER math)	+0.3785 Shortcut Gap	**Classifies topic, not item need**
3. Lexical TF-IDF Baseline	Prompt words alone vs. 10-layer probe bank	0.8403 (White-Box on MetaTool)	**0.8163** (**Prompt TF-IDF alone**)	−0.0016 Δ on SMART-ER	**Probe features redundant w/ text**
4. Dataset-Identity Shortcut	One-hot dataset name vs. 10-layer probe bank ($N = 1{,}680$)	0.6646 (White-Box IID split)	**0.7317** (**1-Hot Dataset Name alone**)	0.5848 (LOBO per-fold)	**Dataset name beats probe bank**
5. Planted Signal Power Check	Transferable vs. bench-local planted signal ($\alpha = 2..4$)	0.7101 (Trans.) / 0.7030 (Local) IID	0.6781 (Trans.) vs. **0.5136** (Local)	Real (0.5848) ∈ Local regime	**Low power unlikely**
6. Joint OOD Transfer	Hold out model + benchmark	0.6713 (IID) / 0.5256 (new model)	**0.5006** (95% CI [0.442, 0.558])	0.4885 (50-Perm Floor)	**Chance on new model + task**
7. Zero-Loss Deployment Frontier	Call savings at zero accuracy loss across 7 live runs	1/7 models save any calls (5%)	**0.7% mean call saving**	+0.81% vs. random routing	**Statistical noise in deployment**

scalar summaries so a single probe can be tested across models with different hidden dimensions; Appendix G). These passive probes do not generalize to actual tool benefit when tools are fallible (Eq. 4). MetaTool's annotated need labels are uncorrelated with the model's own closed-book failure ($\phi = -0.0414$ on 159 graded items per model); prompt TF-IDF alone nearly matches the probe bank (0.8163 vs. 0.8403 AUROC); the dataset name alone beats it in-distribution (0.7317 vs. 0.6646); and SMART-ER routing drops from 0.8848 across domains to 0.5063 within math. A planted positive control makes low power an unlikely explanation: a transferable signal survives leave-one-benchmark-out (LOBO) transfer (0.6781 AUROC), whereas the real label behaves like a benchmark-local signal (0.5848 vs. 0.5136). On unseen models and benchmarks together, probe AUROC falls to 0.5006 (95% CI [0.442, 0.558]; permutation floor 0.4885), saving just 0.7% of calls at zero accuracy loss (this does not rule out learned routers trained on full residual vectors or task-specific data).

End-to-End Latency and Repair. We also timed complete agent episodes, including serving and tool overheads, for eight models (216 items, 18–36 per model; Appendix J). An ungated tool agent takes 7.2 s per item, and the $K = 3$ probe adds 0.17 s (2.4%). The default gate sends 0%–93% of items to the tool depending on the model (46% on average) and cuts latency by 44%, but with these fallible tools it keeps only 59% of the accuracy gain (28.2% vs. 31.3% ungated and 23.8% closed-book), barely above random routing at the same rates (26.9%). One retry after a malformed call or tool error raises valid calls from 68% to 88% and accuracy by 7.3 pp, for 2.6 s per item.

6 Conclusion

Evaluating and routing tool-augmented language models requires three disciplines. First, benchmarks should grade Turn-2 answers after real execution; in our 12-benchmark Suite II setup, assuming that calls succeed overstates accuracy by 29.9–31.5 pp. Second, evaluations should include a Turn-1-matched echo control to separate structural scaffolding (SSE) from external information (NPIG). This control exposes the loss from action-first JSON on parametric text math—not explained by prompt schemas or echo wording and largely removed under the echo by post-call verification—and shows that while text LLMs rely on real tool outputs (NPIG $= +55.7$ pp on $k = 273$ gold-return calls), Qwen2.5-VL ($k = 24$) already reads the answer from the image across real, echo, and empty returns. Third, the benefit factorization (Eq. 4) marks the limit of pre-generation gates: H_{corr} predicts closed-book errors on lookups (AUROC 0.76–0.96) when calibrated per task, but does not beat simpler prefix scores on tool benefit, and when tools are fallible, prefix uncertainty is near chance and our passive 90-feature Logit-Lens probes fall to chance (0.501) on unseen models and benchmarks. Our evaluation is offline and single-query; extending Turn-1-matched decomposition to multi-turn, streaming, and interactive agents is future work.

Limitations. Suite I tasks are synthetic, and near-perfect routing on `wiki-mixed` partly reflects its synthetic tail facts. Suite II uses 40 fixed-seed items per benchmark and model with a local lookup/solver tool, and its closed-book accuracies on text benchmarks are below published results for these models, so its SSE and oracle-overstatement values are specific to this setup. The VLM result covers one model family (Qwen2.5-VL) and 24 gold-return items with a non-visual tool and without a two-turn natural-language control. Our white-box falsification covers passive 90-feature Logit-Lens probes rather than end-to-end learned routers. The prompt and format control ablations cover five 7B–8B models, the main matrix uses placeholder echo template B1 (the smallest loss of the four we tested), and all models are open-weight and at most 72B.

AI USE STATEMENT

In this work, we used generative AI tools to assist with implementing evaluation scripts, formatting figures and tables, and editing prose for readability. We have reviewed all AI-assisted work: all mathematical claims, proofs, benchmark items, code, and empirical results were verified and tested for correctness through deterministic execution and paired statistical testing by the authors. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work evaluates open-weights models on public and synthetic benchmarks, involving no human subjects, personal data, or proprietary assets. A critical practical deployment consideration raised by our findings is the **two-sided risk of tool routing**: while an overly aggressive epistemic gate can suppress necessary external lookups on items where a model is confidently wrong ($\mathrm{NPIG} > 0$), requiring a tool call under rigid action-first schemas can lower accuracy on solvable reasoning. In practice, we suggest grading final answers after real execution, adding one retry that returns parse or tool errors to the model, and gating only reliable lookup tools, with thresholds calibrated per model and task on held-out data. Post-call verification prompts recovered losses under the content-free echo, but with real tools they lowered accuracy by 2.4 pp and doubled Turn-2 time, so they should be tested before deployment.

REPRODUCIBILITY STATEMENT

Prompt templates, threshold selection, and decoding hyperparameters are detailed in Appendix E, and theoretical proofs appear in Appendix C. Appendix A provides the complete 70-row decomposition table, Appendix B reports the full $K \times k$ grid, and Appendices H–J report the prompt and format control ablations, held-out thresholds, confidence intervals, and end-to-end timing. We release all prompts (including every echo wording and both self-simulation conditionings), datasets, strict and lenient parsers, seeds, trajectories, and result tables in the supplementary material for all prompt and echo configurations.

REFERENCES

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *International Conference on Learning Representations (ICLR)*, 2024.

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

Nora Belrose, Igor Ostrovsky, Lev McKinney, Zach Furman, Logan Smith, Danny Halawi, Stella Biderman, and Jacob Steinhardt. Eliciting latent predictions from transformers with the tuned lens. *arXiv preprint arXiv:2303.08112*, 2023.

Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. FireAct: Toward language agent fine-tuning. *arXiv preprint arXiv:2310.05915*, 2023a.

Lin Chen, Jinsong Li, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Zehui Chen, Haodong Duan, Jiaqi Wang, Yu Qiao, Dahua Lin, and Feng Zhao. Are we on the right way for evaluating large vision-language models? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research (TMLR)*, 2023b.

Jiali Cheng, Rui Pan, and Hadi Amiri. Investigating tool-memory conflicts in tool-augmented LLMs. *arXiv preprint arXiv:2601.09760*, 2026.

Yung-Sung Chuang, Yujia Xie, Hongyin Luo, Yoon Kim, James R. Glass, and Pengcheng He. DoLa: Decoding by contrasting layers improves factuality in large language models. In *International Conference on Learning Representations (ICLR)*, 2024.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Ekaterina Fadeeva, Roman Vashurin, Akim Tsvigun, Artem Vazhentsev, Sergey Petrakov, Kirill Fedyanin, Daniil Vasilev, Elizaveta Goncharova, Alexander Panchenko, Maxim Panov, Timothy Baldwin, and Artem Shelmanov. LM-Polygraph: Uncertainty estimation for language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP): System Demonstrations*, pp. 446–461, 2023.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. PAL: Program-aided language models. In *International Conference on Machine Learning (ICML)*, pp. 10764–10799, 2023.

Zhicheng Guo, Sijie Cheng, Hao Wang, Shihao Liang, Yujia Qin, Peng Li, Zhiyuan Liu, Maosong Sun, and Yang Liu. StableToolBench: Towards stable large-scale benchmarking on tool learning of large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 11143–11156, 2024.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021.

Yushi Hu, Weijia Shi, Xingyu Fu, Dan Roth, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Ranjay Krishna. Visual sketchpad: Sketching as a visual chain of thought for multimodal language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Yue Huang, Jiawen Shi, Yuan Li, Chenrui Fan, Siyuan Wu, Qihui Zhang, Yixin Liu, Pan Zhou, Yao Wan, Neil Zhenqiang Gong, and Lichao Sun. MetaTool benchmark for large language models: Deciding whether to use tools and which to use. In *International Conference on Learning Representations (ICLR)*, 2024.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C. Park. Adaptive-RAG: Learning to adapt retrieval-augmented large language models through question complexity. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pp. 7036–7050, 2024.

Zhengbao Jiang, Frank F. Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active Retrieval Augmented Generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 7969–7992, 2023.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

Jannik Kossen, Jiatong Han, Muhammed Razzak, Lisa Schut, Shreshth Malik, and Yarin Gal. Semantic entropy probes: Robust and cheap hallucination detection in LLMs. *arXiv preprint arXiv:2406.15927*, 2024.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *International Conference on Learning Representations (ICLR)*, 2023.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. API-Bank: A comprehensive benchmark for tool-augmented LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 3102–3116, 2023.

Wenjun Li, Dexun Li, Kuicai Dong, Cong Zhang, Hao Zhang, Weiwen Liu, Yasheng Wang, Ruiming Tang, and Yong Liu. Adaptive tool use in large language models with meta-cognition trigger. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 13346–13370, 2025.

Zuxin Liu, Thai Hoang, Jianguo Zhang, Ming Zhu, Tian Lan, Shirley Kokane, Juntao Tan, Weiran Yao, Zhiwei Liu, Yihao Feng, Rithesh Murthy, Liangwei Yang, Silvio Savarese, Juan Carlos Niebles, Huan Wang, Shelby Heinecke, and Caiming Xiong. APIGen: Automated pipeline for generating verifiable and diverse function-calling datasets. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, 2024.

Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. MathVista: Evaluating mathematical reasoning of foundation models in visual contexts. In *International Conference on Learning Representations (ICLR)*, 2024.

Yaxing Lyu, Shengjie Zhou, Binbin Toh, Pengyu Zhu, and Lijun Li. KC-Bench: A dynamic interactive benchmark for evaluating knowledge conflicts in LLM agents. *arXiv preprint arXiv:2609.03588*, 2026.

Qinghe Ma, Zhen Zhao, Yiming Wu, Jian Zhang, Lei Bai, and Yinghuan Shi. Are tools always beneficial? learning to invoke tools adaptively for dual-mode multimodal LLM reasoning. In *International Conference on Machine Learning (ICML)*, 2026.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 9802–9822, 2023.

Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. ChartQA: A benchmark for question answering about charts with visual and logical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2022*, 2022.

Minesh Mathew, Dimosthenis Karatzas, and C. V. Jawahar. DocVQA: A dataset for VQA on document images. In *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2021.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. WebGPT: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

nostalgebraist. Interpreting GPT: The logit lens. LessWrong blog post, https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens, 2020.

Aaron Parisi, Yao Zhao, and Noah Fiedel. TALM: Tool augmented language models. *arXiv preprint arXiv:2205.12255*, 2022.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive APIs. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, 2024.

Shishir G. Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The Berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *International Conference on Machine Learning (ICML)*, volume 267, pp. 48371–48392, 2025.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *International Conference on Learning Representations (ICLR)*, 2024.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 68539–68551, 2023.

Claude Elwood Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

Yifan Song, Weimin Xiong, Dawei Zhu, Wenhao Wu, Han Qian, Mingbo Song, Hailiang Huang, Cheng Li, Ke Wang, Rong Yao, et al. RestGPT: Connecting large language models with real-world RESTful APIs. *arXiv preprint arXiv:2306.06624*, 2023.

Chung-En Sun, Linbo Liu, Ge Yan, Zimo Wang, and Tsui-Wei Weng. LLM agents already know when to call tools — even without reasoning. *arXiv preprint arXiv:2605.09252*, 2026.

Dídac Surís, Sachit Menon, and Carl Vondrick. ViperGPT: Visual inference via Python execution for reasoning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

Zhi Rui Tam, Cheng-Kuang Wu, Yi-Lin Tsai, Chieh-Yen Lin, Hung-yi Lee, and Yun-Nung Chen. Let me speak freely? A study on the impact of format restrictions on performance of large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP): Industry Track*, pp. 1218–1236, 2024.

Qiaoyu Tang, Ziliang Deng, Hongyu Lin, Xianpei Han, Qiao Liang, Boxi Cao, and Le Sun. ToolAlpaca: Generalized tool learning for language models with 3000 simulated cases. *arXiv preprint arXiv:2306.05301*, 2023.

Qixun Wang, Yang Shi, Letian Cheng, Zhuoran Zhang, Yan He, Yuqi Tang, Qi Zhang, Xinlei Yu, Ruizhe Chen, Tianrun Xu, Yuanxing Zhang, Pengfei Wan, Haotian Wang, and Xianghua Ying. Beacon: Knowing when and how to perform agentic visual reasoning. *arXiv preprint arXiv:2607.28595*, 2026.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. Executable code actions elicit better LLM agents. In *International Conference on Machine Learning (ICML)*, 2024a.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2024b.

Brandon T. Willard and Remi Louf. Efficient guided generation for large language models. *arXiv preprint arXiv:2307.09702*, 2023.

Konstantin Yakovlev, Sergey Nikolenko, and Andrey Bout. Toolken+: Improving LLM tool usage with reranking and a reject option. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 5967–5974, 2024.

Wenjie Yang, Siqi Zhu, and Zengfeng Huang. Position: Your VLM may not be thinking with interleaved images. In *International Conference on Machine Learning (ICML) Position Paper Track*, 2026.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In *International Conference on Learning Representations (ICLR)*, 2025.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, et al. SGLang: Efficient execution of structured language model programs. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, 2024.

Yuchen Zhuang, Yue Yu, Kuan Wang, Haotian Sun, and Chao Zhang. ToolQA: A dataset for LLM question answering with external tools. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, volume 36, 2023.

A Complete 70-Row Task-by-Model Decomposition Table

Table 6 presents the complete, machine-verified 3-way decomposition across all 7 tasks and all 10 full-suite open-weights models ($N = 100$ items per task per model; $N = 700$ items per model; 70 task-model combinations), including 2-Turn Natural Language Reflection (`Arm 4b`), 95% Wilson confidence intervals, and exact paired binomial McNemar p-values.

Table 6: Complete Task-by-Model 3-Way Decomposition Matrix ($N = 100$ items per row across 10 full-suite models $\times$ 7 tasks = 70 rows). Values show `Acc@lenient` [95% Wilson CI] (`Acc@strict`) and exact paired McNemar p-values for SSE = Echo − wo-tool, Hallucination Effect = SelfSim − Echo, $\text{NPIG}_{\text{cond}}$ = Real − SelfSim (where direct NPIG = Real − Echo = Halluc. Effect + $\text{NPIG}_{\text{cond}}$), and Total Gain = Real − wo-tool.

Task	Model	Arm 4: wo-tool (1T)	Arm 4b: Reflect (2T)	Arm 2c: Echo	Arm 2a: Self-Sim	Arm 1: Real Tool	SSE (2c − 4)	Halluc. Effect	$\text{NPIG}_{\text{cond}}$ (1 − 2a)	Total Gain
flight-easy	Qwen2.5-3B-Instruct	0.0% [0,4] (0.0%)	0.0% (0.0%)	3.0% (2.0%)	1.0% (0.0%)	**49.0%** [39,59] (8.0%)	+3.0% ($p = .2500$)	−2.0% ($p = .5000$)	+**48.0**% ($p < .0001$)	+**49.0**% ($p < .0001$)
flight-easy	Qwen2.5-7B-Instruct	1.0% [0,5] (0.0%)	2.0% (2.0%)	0.0% (0.0%)	6.0% (0.0%)	**90.0%** [83,94] (4.0%)	−1.0% ($p = 1.000$)	+**6.0**% ($p = .0312$)	+**84.0**% ($p < .0001$)	+**89.0**% ($p < .0001$)
flight-easy	Qwen2.5-Coder-7B-Inst	11.0% [6,19] (11.0%)	8.0% (8.0%)	2.0% (2.0%)	0.0% (0.0%)	**82.0%** [73,88] (49.0%)	−**9.0**% ($p = .0117$)	−2.0% ($p = .5000$)	+**82.0**% ($p < .0001$)	+**71.0**% ($p < .0001$)
flight-easy	Gemma-3-27B-IT	1.0% [0,5] (1.0%)	2.0% (2.0%)	0.0% (0.0%)	0.0% (0.0%)	**91.0%** [84,95] (90.0%)	−1.0% ($p = 1.000$)	+0.0% ($p = 1.000$)	+**91.0**% ($p < .0001$)	+**90.0**% ($p < .0001$)
flight-easy	Qwen2.5-Coder-32B-Inst	8.0% [4,15] (8.0%)	7.0% (7.0%)	10.0% (0.0%)	0.0% (0.0%)	**94.0%** [88,97] (0.0%)	+2.0% ($p = .6875$)	−**10.0**% ($p = .0020$)	+**94.0**% ($p < .0001$)	+**86.0**% ($p < .0001$)
flight-easy	Qwen2.5-72B-Inst-AWQ	0.0% [0,4] (0.0%)	0.0% (0.0%)	0.0% (0.0%)	0.0% (0.0%)	**88.0%** [80,93] (74.0%)	+0.0% ($p = 1.000$)	+0.0% ($p = 1.000$)	+**88.0**% ($p < .0001$)	+**88.0**% ($p < .0001$)
flight-easy	DeepSeek-R1-Qwen-7B	0.0% [0,4] (0.0%)	4.0% (4.0%)	1.0% (1.0%)	1.0% (1.0%)	**40.0%** [31,50] (1.0%)	+1.0% ($p = 1.000$)	+0.0% ($p = 1.000$)	+**39.0**% ($p < .0001$)	+**40.0**% ($p < .0001$)
flight-easy	DeepSeek-R1-Llama-8B	0.0% [0,4] (0.0%)	2.0% (2.0%)	3.0% (3.0%)	8.0% (7.0%)	**70.0%** [60,78] (68.0%)	+3.0% ($p = .2500$)	+5.0% ($p = .1797$)	+**62.0**% ($p < .0001$)	+**70.0**% ($p < .0001$)
flight-easy	Llama-3.1-Tulu-3-8B	0.0% [0,4] (0.0%)	2.0% (2.0%)	0.0% (0.0%)	1.0% (1.0%)	**71.0%** [62,79] (55.0%)	+0.0% ($p = 1.000$)	+1.0% ($p = 1.000$)	+**70.0**% ($p < .0001$)	+**71.0**% ($p < .0001$)
flight-easy	Mistral-7B-Inst-v0.3	0.0% [0,4] (0.0%)	1.0% (1.0%)	4.0% (4.0%)	3.0% (3.0%)	**40.0%** [31,50] (31.0%)	+4.0% ($p = .1250$)	−1.0% ($p = 1.000$)	+**37.0**% ($p < .0001$)	+**40.0**% ($p < .0001$)
table-agg	Qwen2.5-3B-Instruct	0.0% [0,4] (0.0%)	0.0% (0.0%)	0.0% (0.0%)	0.0% (0.0%)	**20.0%** [13,29] (20.0%)	+0.0% ($p = 1.000$)	+0.0% ($p = 1.000$)	+**20.0**% ($p < .0001$)	+**20.0**% ($p < .0001$)
table-agg	Qwen2.5-7B-Instruct	0.0% [0,4] (0.0%)	0.0% (0.0%)	0.0% (0.0%)	0.0% (0.0%)	**79.0%** [70,86] (79.0%)	+0.0% ($p = 1.000$)	+0.0% ($p = 1.000$)	+**79.0**% ($p < .0001$)	+**79.0**% ($p < .0001$)
table-agg	Qwen2.5-Coder-7B-Inst	0.0% [0,4] (0.0%)	0.0% (0.0%)	0.0% (0.0%)	1.0% (0.0%)	**16.0%** [10,24] (0.0%)	+0.0% ($p = 1.000$)	+1.0% ($p = 1.000$)	+**15.0**% ($p = .0003$)	+**16.0**% ($p < .0001$)
table-agg	Gemma-3-27B-IT	0.0% [0,4] (0.0%)	0.0% (0.0%)	0.0% (0.0%)	0.0% (0.0%)	**99.0%** [95,100] (99.0%)	+0.0% ($p = 1.000$)	+0.0% ($p = 1.000$)	+**99.0**% ($p < .0001$)	+**99.0**% ($p < .0001$)
table-agg	Qwen2.5-Coder-32B-Inst	0.0% [0,4] (0.0%)	0.0% (0.0%)	0.0% (0.0%)	0.0% (0.0%)	**98.0%** [93,99] (0.0%)	+0.0% ($p = 1.000$)	+0.0% ($p = 1.000$)	+**98.0**% ($p < .0001$)	+**98.0**% ($p < .0001$)
table-agg	Qwen2.5-72B-Inst-AWQ	0.0% [0,4] (0.0%)	0.0% (0.0%)	0.0% (0.0%)	0.0% (0.0%)	**99.0%** [95,100] (99.0%)	+0.0% ($p = 1.000$)	+0.0% ($p = 1.000$)	+**99.0**% ($p < .0001$)	+**99.0**% ($p < .0001$)
table-agg	DeepSeek-R1-Qwen-7B	0.0% [0,4] (0.0%)	0.0% (0.0%)	0.0% (0.0%)	0.0% (0.0%)	0.0% [0,4] (0.0%)	+0.0% ($p = 1.000$)	+0.0% ($p = 1.000$)	+0.0% ($p = 1.000$)	+0.0% ($p = 1.000$)
table-agg	DeepSeek-R1-Llama-8B	0.0% [0,4] (0.0%)	0.0% (0.0%)	0.0% (0.0%)	1.0% (1.0%)	1.0% [0,5] (1.0%)	+0.0% ($p = 1.000$)	+1.0% ($p = 1.000$)	+0.0% ($p = 1.000$)	+1.0% ($p = 1.000$)
table-agg	Llama-3.1-Tulu-3-8B	0.0% [0,4] (0.0%)	0.0% (0.0%)	0.0% (0.0%)	0.0% (0.0%)	**36.0%** [27,46] (35.0%)	+0.0% ($p = 1.000$)	+0.0% ($p = 1.000$)	+**36.0**% ($p < .0001$)	+**36.0**% ($p < .0001$)
table-agg	Mistral-7B-Inst-v0.3	0.0% [0,4] (0.0%)	0.0% (0.0%)	0.0% (0.0%)	0.0% (0.0%)	**36.0%** [27,46] (36.0%)	+0.0% ($p = 1.000$)	+0.0% ($p = 1.000$)	+**36.0**% ($p < .0001$)	+**36.0**% ($p < .0001$)
wiki-mixed	Qwen2.5-3B-Instruct	46.0% [37,56] (24.0%)	48.0% (48.0%)	48.0% (5.0%)	48.0% (15.0%)	**97.0%** [92,99] (9.0%)	+2.0% ($p = .6250$)	+0.0% ($p = 1.000$)	+**49.0**% ($p < .0001$)	+**51.0**% ($p < .0001$)
wiki-mixed	Qwen2.5-7B-Instruct	47.0% [38,57] (22.0%)	48.0% (48.0%)	47.0% (21.0%)	46.0% (41.0%)	**100.0%** [96,100] (52.0%)	+0.0% ($p = 1.000$)	−1.0% ($p = 1.000$)	+**54.0**% ($p < .0001$)	+**53.0**% ($p < .0001$)
wiki-mixed	Qwen2.5-Coder-7B-Inst	47.0% [38,57] (46.0%)	46.0% (46.0%)	44.0% (43.0%)	44.0% (43.0%)	**93.0%** [86,97] (88.0%)	−3.0% ($p = .3750$)	+0.0% ($p = 1.000$)	+**49.0**% ($p < .0001$)	+**46.0**% ($p < .0001$)
wiki-mixed	Gemma-3-27B-IT	48.0% [39,58] (48.0%)	49.0% (49.0%)	46.0% (45.0%)	44.0% (44.0%)	**99.0%** [95,100] (99.0%)	−2.0% ($p = .5000$)	−2.0% ($p = .6250$)	+**55.0**% ($p < .0001$)	+**51.0**% ($p < .0001$)
wiki-mixed	Qwen2.5-Coder-32B-Inst	48.0% [38,58] (48.0%)	46.0% (46.0%)	46.0% (0.0%)	48.0% (0.0%)	**98.0%** [93,99] (0.0%)	−2.0% ($p = .5000$)	+2.0% ($p = .5000$)	+**50.0**% ($p < .0001$)	+**50.0**% ($p < .0001$)
wiki-mixed	Qwen2.5-72B-Inst-AWQ	50.0% [40,60] (50.0%)	49.0% (48.0%)	48.0% (13.0%)	47.0% (46.0%)	**98.0%** [93,99] (97.0%)	−2.0% ($p = .5000$)	−1.0% ($p = 1.000$)	+**51.0**% ($p < .0001$)	+**48.0**% ($p < .0001$)
wiki-mixed	DeepSeek-R1-Qwen-7B	45.0% [36,55] (1.0%)	48.0% (33.0%)	45.0% (2.0%)	45.0% (2.0%)	**55.0%** [45,64] (2.0%)	+0.0% ($p = 1.000$)	+0.0% ($p = 1.000$)	+**10.0**% ($p = .0020$)	+**10.0**% ($p = .0129$)
wiki-mixed	DeepSeek-R1-Llama-8B	43.0% [34,53] (0.0%)	42.0% (35.0%)	46.0% (5.0%)	46.0% (5.0%)	**91.0%** [84,95] (46.0%)	+3.0% ($p = .2500$)	+0.0% ($p = 1.000$)	+**45.0**% ($p < .0001$)	+**48.0**% ($p < .0001$)
wiki-mixed	Llama-3.1-Tulu-3-8B	46.0% [37,56] (46.0%)	48.0% (48.0%)	46.0% (45.0%)	45.0% (45.0%)	**87.0%** [79,92] (85.0%)	−1.0% ($p = 1.000$)	+1.0% ($p = 1.000$)	+**41.0**% ($p < .0001$)	+**41.0**% ($p < .0001$)
wiki-mixed	Mistral-7B-Inst-v0.3	47.0% [38,57] (47.0%)	45.0% (45.0%)	41.0% (39.0%)	43.0% (35.0%)	**85.0%** [77,91] (78.0%)	−6.0% ($p = .0703$)	+2.0% ($p = .6250$)	+**42.0**% ($p < .0001$)	+**38.0**% ($p < .0001$)
gsm8k-easy	Qwen2.5-3B-Instruct	58.0% [48,67] (54.0%)	56.0% (56.0%)	50.0% (35.0%)	40.0% (30.0%)	**58.0%** [48,67] (35.0%)	−8.0% ($p = .1338$)	−**10.0**% ($p = .0414$)	+**18.0**% ($p = .0001$)	+0.0% ($p = 1.000$)
gsm8k-easy	Qwen2.5-7B-Instruct	**73.0%** [64,81] (70.0%)	66.0% (65.0%)	66.0% (39.0%)	56.0% (37.0%)	69.0% [59,77] (50.0%)	−7.0% ($p = .2100$)	−**10.0**% ($p = .0213$)	+**13.0**% ($p = .0023$)	−4.0% ($p = .5034$)
gsm8k-easy	Qwen2.5-Coder-7B-Inst	**74.0%** [65,82] (66.0%)	67.0% (66.0%)	69.0% (66.0%)	69.0% (66.0%)	70.0% [60,78] (66.0%)	−5.0% ($p = .3323$)	+0.0% ($p = 1.000$)	+1.0% ($p = 1.000$)	−4.0% ($p = .4807$)
gsm8k-easy	Gemma-3-27B-IT	82.0% [73,88] (62.0%)	82.0% (78.0%)	**85.0%** (78.0%)	84.0% (77.0%)	**85.0%** [77,91] (78.0%)	+3.0% ($p = .2500$)	−1.0% ($p = 1.000$)	+1.0% ($p = 1.000$)	+3.0% ($p = .2500$)
gsm8k-easy	Qwen2.5-Coder-32B-Inst	**83.0%** [74,89] (81.0%)	80.0% (79.0%)	80.0% (13.0%)	73.0% (13.0%)	76.0% [67,83] (13.0%)	−3.0% ($p = .5078$)	−**7.0**% ($p = .0391$)	+3.0% ($p = .4531$)	−7.0% ($p = .0654$)
gsm8k-easy	Qwen2.5-72B-Inst-AWQ	**82.0%** [73,88] (71.0%)	**85.0%** (85.0%)	75.0% (72.0%)	72.0% (68.0%)	75.0% [66,83] (71.0%)	−7.0% ($p = .0923$)	−3.0% ($p = .5488$)	+3.0% ($p = .4531$)	−7.0% ($p = .0923$)
gsm8k-easy	DeepSeek-R1-Qwen-7B	**72.0%** [62,80] (18.0%)	68.0% (5.0%)	49.0% (4.0%)	53.0% (4.0%)	53.0% [43,62] (4.0%)	−**23.0**% ($p = .0003$)	+4.0% ($p = .4240$)	+0.0% ($p = 1.000$)	−**19.0**% ($p = .0013$)
gsm8k-easy	DeepSeek-R1-Llama-8B	51.0% [41,61] (3.0%)	43.0% (33.0%)	52.0% (8.0%)	52.0% (8.0%)	**52.0%** [42,62] (8.0%)	+1.0% ($p = 1.000$)	+0.0% ($p = 1.000$)	+0.0% ($p = 1.000$)	+1.0% ($p = 1.000$)
gsm8k-easy	Llama-3.1-Tulu-3-8B	64.0% [54,73] (64.0%)	60.0% (59.0%)	**67.0%** (63.0%)	65.0% (60.0%)	66.0% [56,74] (64.0%)	+3.0% ($p = .6636$)	−2.0% ($p = .5000$)	+1.0% ($p = 1.000$)	+2.0% ($p = .8318$)
gsm8k-easy	Mistral-7B-Inst-v0.3	**29.0%** [21,38] (29.0%)	27.0% (26.0%)	9.0% (9.0%)	10.0% (10.0%)	11.0% [6,19] (10.0%)	−**20.0**% ($p = .0005$)	+1.0% ($p = 1.000$)	+1.0% ($p = 1.000$)	−**18.0**% ($p = .0005$)
calc-reach	Qwen2.5-3B-Instruct	42.0% [33,52] (21.0%)	25.0% (23.0%)	12.0% (10.0%)	12.0% (9.0%)	**60.0%** [50,69] (46.0%)	−**30.0**% ($p < .0001$)	+0.0% ($p = 1.000$)	+**48.0**% ($p < .0001$)	+**18.0**% ($p = .0001$)
calc-reach	Qwen2.5-7B-Instruct	47.0% [38,57] (46.0%)	47.0% (47.0%)	24.0% (10.0%)	11.0% (11.0%)	**92.0%** [85,96] (66.0%)	−**23.0**% ($p < .0001$)	−**13.0**% ($p = .0002$)	+**81.0**% ($p < .0001$)	+**45.0**% ($p < .0001$)
calc-reach	Qwen2.5-Coder-7B-Inst	30.0% [22,40] (28.0%)	38.0% (38.0%)	28.0% (28.0%)	20.0% (20.0%)	**43.0%** [34,53] (41.0%)	−2.0% ($p = .7744$)	−8.0% ($p = .0574$)	+**23.0**% ($p < .0001$)	+**13.0**% ($p = .0241$)
calc-reach	Gemma-3-27B-IT	67.0% [57,75] (60.0%)	71.0% (65.0%)	44.0% (43.0%)	42.0% (42.0%)	**79.0%** [70,86] (78.0%)	−**23.0**% ($p < .0001$)	−2.0% ($p = .8388$)	+**37.0**% ($p < .0001$)	+**12.0**% ($p = .0018$)
calc-reach	Qwen2.5-Coder-32B-Inst	50.0% [40,60] (46.0%)	49.0% (49.0%)	45.0% (0.0%)	27.0% (0.0%)	**80.0%** [71,87] (0.0%)	−5.0% ($p = .0625$)	−**18.0**% ($p < .0001$)	+**53.0**% ($p < .0001$)	+**30.0**% ($p < .0001$)
calc-reach	Qwen2.5-72B-Inst-AWQ	51.0% [41,61] (50.0%)	52.0% (51.0%)	38.0% (38.0%)	18.0% (18.0%)	**67.0%** [57,75] (67.0%)	−**13.0**% ($p = .0002$)	−20.0% ($p < .0001$)	+**49.0**% ($p < .0001$)	+16.0% ($p = .0015$)
calc-reach	DeepSeek-R1-Qwen-7B	**60.0%** [50,69] (7.0%)	53.0% (6.0%)	58.0% (0.0%)	49.0% (0.0%)	58.0% [48,67] (0.0%)	−2.0% ($p = .7539$)	−**9.0**% ($p = .0117$)	+**9.0**% ($p = .0117$)	−2.0% ($p = .7539$)
calc-reach	DeepSeek-R1-Llama-8B	**55.0%** [45,64] (13.0%)	52.0% (29.0%)	30.0% (11.0%)	30.0% (11.0%)	30.0% [22,40] (11.0%)	−**25.0**% ($p < .0001$)	+0.0% ($p = 1.000$)	+0.0% ($p = 1.000$)	−**25.0**% ($p < .0001$)
calc-reach	Llama-3.1-Tulu-3-8B	25.0% [18,34] (25.0%)	40.0% (40.0%)	35.0% (35.0%)	31.0% (31.0%)	**68.0%** [58,76] (68.0%)	+**10.0**% ($p = .0063$)	−4.0% ($p = .2188$)	+**37.0**% ($p < .0001$)	+**43.0**% ($p < .0001$)
calc-reach	Mistral-7B-Inst-v0.3	8.0% [4,15] (8.0%)	23.0% (23.0%)	1.0% (0.0%)	1.0% (1.0%)	**57.0%** [47,66] (25.0%)	−**7.0**% ($p = .0156$)	+0.0% ($p = 1.000$)	+**56.0**% ($p < .0001$)	+**49.0**% ($p < .0001$)
hop-necessity	Qwen2.5-3B-Instruct	24.0% [17,33] (24.0%)	24.0% (24.0%)	3.0% (3.0%)	3.0% (3.0%)	**46.0%** [37,56] (6.0%)	−**21.0**% ($p < .0001$)	+0.0% ($p = 1.000$)	+**43.0**% ($p < .0001$)	+**22.0**% ($p = .0081$)
hop-necessity	Qwen2.5-7B-Instruct	26.0% [18,35] (25.0%)	25.0% (25.0%)	10.0% (1.0%)	8.0% (6.0%)	**79.0%** [70,86] (5.0%)	−**16.0**% ($p < .0001$)	−2.0% ($p = .7266$)	+**71.0**% ($p < .0001$)	+**53.0**% ($p < .0001$)
hop-necessity	Qwen2.5-Coder-7B-Inst	25.0% [18,34] (25.0%)	25.0% (25.0%)	25.0% (25.0%)	24.0% (24.0%)	**58.0%** [48,67] (33.0%)	+0.0% ($p = 1.000$)	−1.0% ($p = 1.000$)	+**34.0**% ($p < .0001$)	+**33.0**% ($p < .0001$)
hop-necessity	Gemma-3-27B-IT	25.0% [18,34] (25.0%)	25.0% (25.0%)	25.0% (25.0%)	25.0% (25.0%)	**100.0%** [96,100] (100.0%)	+0.0% ($p = 1.000$)	+0.0% ($p = 1.000$)	+**75.0**% ($p < .0001$)	+**75.0**% ($p < .0001$)
hop-necessity	Qwen2.5-Coder-32B-Inst	25.0% [18,34] (25.0%)	26.0% (26.0%)	26.0% (0.0%)	16.0% (0.0%)	**83.0%** [74,89] (0.0%)	+1.0% ($p = 1.000$)	−**10.0**% ($p = .0020$)	+**67.0**% ($p < .0001$)	+**58.0**% ($p < .0001$)
hop-necessity	Qwen2.5-72B-Inst-AWQ	27.0% [19,36] (27.0%)	27.0% (27.0%)	23.0% (23.0%)	15.0% (15.0%)	**97.0%** [92,99] (97.0%)	−4.0% ($p = .1250$)	−8.0% ($p = .0215$)	+**82.0**% ($p < .0001$)	+**70.0**% ($p < .0001$)
hop-necessity	DeepSeek-R1-Qwen-7B	**25.0%** [18,34] (1.0%)	**25.0%** (0.0%)	**25.0%** (0.0%)	21.0% (0.0%)	**25.0%** [18,34] (0.0%)	+0.0% ($p = 1.000$)	−4.0% ($p = .1250$)	+4.0% ($p = .1250$)	+0.0% ($p = 1.000$)
hop-necessity	DeepSeek-R1-Llama-8B	25.0% [18,34] (1.0%)	25.0% (25.0%)	25.0% (15.0%)	25.0% (15.0%)	**44.0%** [35,54] (34.0%)	+0.0% ($p = 1.000$)	+0.0% ($p = 1.000$)	+**19.0**% ($p < .0001$)	+**19.0**% ($p < .0001$)
hop-necessity	Llama-3.1-Tulu-3-8B	26.0% [18,35] (26.0%)	26.0% (26.0%)	25.0% (25.0%)	2.0% (2.0%)	**72.0%** [62,80] (72.0%)	−1.0% ($p = 1.000$)	−**23.0**% ($p < .0001$)	+**70.0**% ($p < .0001$)	+**46.0**% ($p < .0001$)
hop-necessity	Mistral-7B-Inst-v0.3	4.0% [2,10] (4.0%)	**20.0%** (20.0%)	1.0% (1.0%)	0.0% (0.0%)	4.0% [2,10] (4.0%)	−3.0% ($p = .3750$)	−1.0% ($p = 1.000$)	+4.0% ($p = .1250$)	+0.0% ($p = 1.000$)
toolbelt-multi	Qwen2.5-3B-Instruct	30.0% [22,40] (22.0%)	31.0% (31.0%)	28.0% (11.0%)	27.0% (11.0%)	**61.0%** [51,70] (14.0%)	−2.0% ($p = .7905$)	−1.0% ($p = 1.000$)	+**34.0**% ($p < .0001$)	+**31.0**% ($p < .0001$)
toolbelt-multi	Qwen2.5-7B-Instruct	38.0% [29,48] (29.0%)	39.0% (39.0%)	31.0% (12.0%)	30.0% (10.0%)	**74.0%** [65,82] (16.0%)	−7.0% ($p = .1892$)	−1.0% ($p = 1.000$)	+**44.0**% ($p < .0001$)	+**36.0**% ($p < .0001$)
toolbelt-multi	Qwen2.5-Coder-7B-Inst	51.0% [41,61] (46.0%)	45.0% (45.0%)	37.0% (36.0%)	37.0% (36.0%)	**61.0%** [51,70] (59.0%)	−**14.0**% ($p = .0005$)	+0.0% ($p = 1.000$)	+**24.0**% ($p < .0001$)	+10.0% ($p = .1433$)
toolbelt-multi	Gemma-3-27B-IT	44.0% [35,54] (39.0%)	45.0% (44.0%)	42.0% (41.0%)	42.0% (41.0%)	**93.0%** [86,97] (92.0%)	−2.0% ($p = .6875$)	+0.0% ($p = 1.000$)	+**51.0**% ($p < .0001$)	+**49.0**% ($p < .0001$)
toolbelt-multi	Qwen2.5-Coder-32B-Inst	54.0% [44,63] (53.0%)	48.0% (48.0%)	39.0% (5.0%)	36.0% (4.0%)	**78.0%** [69,85] (7.0%)	−**15.0**% ($p = .0015$)	−3.0% ($p = .5811$)	+**42.0**% ($p < .0001$)	+**24.0**% ($p = .0004$)
toolbelt-multi	Qwen2.5-72B-Inst-AWQ	42.0% [33,52] (37.0%)	43.0% (42.0%)	37.0% (31.0%)	34.0% (34.0%)	**84.0%** [76,90] (78.0%)	−5.0% ($p = .1797$)	−3.0% ($p = .4531$)	+**50.0**% ($p < .0001$)	+**42.0**% ($p < .0001$)
toolbelt-multi	DeepSeek-R1-Qwen-7B	39.0% [30,49] (8.0%)	40.0% (11.0%)	39.0% (2.0%)	30.0% (2.0%)	**47.0%** [38,57] (2.0%)	+0.0% ($p = 1.000$)	−**9.0**% ($p = .0117$)	+**17.0**% ($p = .0015$)	+8.0% ($p = .2153$)
toolbelt-multi	DeepSeek-R1-Llama-8B	31.0% [23,41] (0.0%)	30.0% (25.0%)	37.0% (24.0%)	36.0% (23.0%)	**63.0%** [53,72] (50.0%)	+6.0% ($p = .2379$)	−1.0% ($p = 1.000$)	+**27.0**% ($p < .0001$)	+**32.0**% ($p < .0001$)
toolbelt-multi	Llama-3.1-Tulu-3-8B	34.0% [26,44] (34.0%)	39.0% (39.0%)	34.0% (33.0%)	34.0% (31.0%)	**69.0%** [59,77] (63.0%)	+0.0% ($p = 1.000$)	+0.0% ($p = 1.000$)	+**35.0**% ($p < .0001$)	+**35.0**% ($p < .0001$)
toolbelt-multi	Mistral-7B-Inst-v0.3	23.0% [16,32] (23.0%)	24.0% (23.0%)	12.0% (10.0%)	19.0% (11.0%)	**30.0%** [22,40] (21.0%)	−**11.0**% ($p = .0034$)	+**7.0**% ($p = .0391$)	+**11.0**% ($p = .0433$)	+7.0% ($p = .2810$)

B 2D Hyperparameter Sensitivity Grid ($K \times k$) & Dynamic Gate Verification

Table 7 reports the complete 2D hyperparameter sensitivity grid sweeping prefix token length $K \in \{1, 2, 3, 5, 10\}$ and top-k softmax tail cutoff $k \in \{5, 10, 20, 50\}$ across all four model-benchmark panels (`Llama-3.1-Tulu-3-8B` and `Qwen2.5-7B-Instruct` on `wiki-mixed` and `calc-reach`), empirically validating Proposition 1. Whereas H_{exact} already achieves 0.991–0.994 AUROC on `wiki-mixed` at $K = 3, k = 20$ (where P_{tail} remains below the non-monotonic turning point), applying the Grouping-Corrected bound H_{corr} ($V = 128{,}256$) boosts `calc-reach` AUROC at $K = 3, k = 20$ from 0.667 to **0.824** on `Llama-3.1-Tulu-3-8B` and from 0.819 to **0.852** on `Qwen2.5-7B-Instruct`. Over all seven tasks, H_{corr} reaches 0.862 on `DeepSeek-R1-Llama-8B` and 0.870 on `Gemma-3-27B-IT` (Appendix I). Across independent runs at $K = 3, k = 20$, the grouping correction consistently improves diffuse-tail Tulu-3-8B

(0.622→0.777) while changing Qwen2.5-7B less (0.690 vs. 0.689). Parenthetical latencies reflect the single-pass $K_{\max} = 10$ prefill+decode linear attribution model ($0.4\,T_{10} + 0.6\,(K/10)\,T_{10}$); direct isolated wall-clock GPU timing per K on A100-80GB measures $\mathbf{1.02}$ ms ($K = 1$), $\mathbf{1.18}$ ms ($K = 2$), $\mathbf{1.33}$ ms ($K = 3$), $\mathbf{1.66}$ ms ($K = 5$), and $\mathbf{2.45}$ ms ($K = 10$). These kernel timings are not per-request latency; Appendix J reports end-to-end time (0.17 s per probe).

Table 7: Complete 2D Hyperparameter Sensitivity Grid ($K \times k$) for Exact Tail-Bucketed Entropy H_{exact} (with H_{corr} at $K = 3, k = 20$ achieving $\mathbf{0.824}$ on `Tulu-3-8B calc-reach` and $\mathbf{0.852}$ on `Qwen2.5-7B calc-reach`) and modeled probe latency (ms; $0.4\,T_{10} + 0.6\,(K/10)\,T_{10}$ from measured A100 T_{10}).

Model & Benchmark	Softmax Cutoff k	$K = 1$ Token	$K = 2$ Tokens	$K = 3$ Tokens	$K = 5$ Tokens	$K = 10$ Tokens
Llama-3.1-Tulu-3-8B `wiki-mixed` ($N = 100$)	$k = 5$ (Narrow Top-k)	0.977 (10.49 ms)	0.962 (11.86 ms)	0.966 (13.22 ms)	0.951 (15.96 ms)	0.944 (22.80 ms)
	$k = 10$ (Medium Top-k)	0.980 (10.49 ms)	0.972 (11.86 ms)	0.981 (13.22 ms)	0.966 (15.96 ms)	0.964 (22.80 ms)
	$k = 20$ **(Default Tail Cutoff)**	**0.980** (10.49 ms)	**0.986** (11.86 ms)	**0.991** (13.22 ms)	**0.982** (15.96 ms)	**0.975** (22.80 ms)
	$k = 50$ (Wide Top-k)	0.980 (10.49 ms)	0.986 (11.86 ms)	**0.991** (13.22 ms)	**0.991** (15.96 ms)	0.986 (22.80 ms)
Llama-3.1-Tulu-3-8B `calc-reach` ($N = 100$)	$k = 5$ (Narrow Top-k)	0.560 (2.65 ms)	0.603 (2.99 ms)	0.617 (3.34 ms)	0.638 (4.02 ms)	0.619 (5.75 ms)
	$k = 10$ (Medium Top-k)	0.611 (2.65 ms)	0.653 (2.99 ms)	0.651 (3.34 ms)	0.661 (4.02 ms)	0.655 (5.75 ms)
	$k = 20$ **(Default Tail Cutoff)**	**0.628** (2.65 ms)	**0.674** (2.99 ms)	**0.667** [**0.824** H_{corr}] (3.34 ms)	**0.660** (4.02 ms)	**0.656** (5.75 ms)
	$k = 50$ (Wide Top-k)	0.628 (2.65 ms)	**0.682** (2.99 ms)	**0.678** (3.34 ms)	**0.682** (4.02 ms)	**0.673** (5.75 ms)
Qwen2.5-7B-Instruct `wiki-mixed` ($N = 100$)	$k = 5$ (Narrow Top-k)	0.994 (10.92 ms)	0.996 (12.34 ms)	0.994 (13.77 ms)	0.993 (16.62 ms)	0.993 (23.74 ms)
	$k = 10$ (Medium Top-k)	0.994 (10.92 ms)	0.996 (12.34 ms)	0.994 (13.77 ms)	0.993 (16.62 ms)	0.993 (23.74 ms)
	$k = 20$ **(Default Tail Cutoff)**	**0.994** (10.92 ms)	**0.996** (12.34 ms)	**0.994** (13.77 ms)	**0.993** (16.62 ms)	**0.993** (23.74 ms)
	$k = 50$ (Wide Top-k)	0.994 (10.92 ms)	0.996 (12.34 ms)	0.994 (13.77 ms)	0.993 (16.62 ms)	0.993 (23.74 ms)
Qwen2.5-7B-Instruct `calc-reach` ($N = 100$)	$k = 5$ (Narrow Top-k)	0.895 (3.03 ms)	0.907 (3.43 ms)	0.818 (3.82 ms)	0.780 (4.61 ms)	0.799 (6.59 ms)
	$k = 10$ (Medium Top-k)	0.894 (3.03 ms)	0.909 (3.43 ms)	0.819 (3.82 ms)	0.784 (4.61 ms)	0.800 (6.59 ms)
	$k = 20$ **(Default Tail Cutoff)**	**0.894** (3.03 ms)	**0.909** (3.43 ms)	**0.819** [**0.852** H_{corr}] (3.82 ms)	**0.784** (4.61 ms)	**0.800** (6.59 ms)
	$k = 50$ (Wide Top-k)	0.894 (3.03 ms)	0.909 (3.43 ms)	0.819 (3.82 ms)	0.784 (4.61 ms)	0.800 (6.59 ms)

C Mathematical Proofs & Derivations

C.1 Proof of Proposition 1 (Top-k Softmax Re-Normalization Bias & Grouping-Corrected Entropy)

Proof. Let $\mathbf{p}_t = (p_{t,1}, \ldots, p_{t,V})$ be the true softmax distribution over vocabulary $\mathcal{V}$ at step t, ordered such that $p_{t,1} \geq p_{t,2} \geq \cdots \geq p_{t,V}$. Let $S_k(t) = \sum_{i=1}^{k} p_{t,i}$ be the total probability mass captured by the top-k tokens, and let $P_{\text{tail}}(t) = 1 - S_k(t) = \sum_{j=k+1}^{V} p_{t,j}$ be the residual tail mass. By definition, the top-k re-normalized probabilities are $\tilde{p}_{t,i} = p_{t,i}/S_k(t)$ for $i \in \{1, \ldots, k\}$. Substituting $p_{t,i} = S_k(t)\tilde{p}_{t,i}$ into the single-step Exact Tail-Bucketed Entropy formula ($H_{\text{exact}}(t) = -\sum_{i=1}^{k} p_{t,i} \ln p_{t,i} - P_{\text{tail}}(t) \ln P_{\text{tail}}(t)$, with natural logarithms ln):

$$
\begin{aligned}
H_{\text{exact}}(t) &= -\sum_{i=1}^{k} \big(S_k(t)\tilde{p}_{t,i}\big) \ln \big(S_k(t)\tilde{p}_{t,i}\big) - P_{\text{tail}}(t) \ln P_{\text{tail}}(t) \\
&= -S_k(t) \sum_{i=1}^{k} \tilde{p}_{t,i} \big(\ln S_k(t) + \ln \tilde{p}_{t,i} \big) - P_{\text{tail}}(t) \ln P_{\text{tail}}(t) \\
&= -S_k(t) \ln S_k(t) \underbrace{\sum_{i=1}^{k} \tilde{p}_{t,i}}_{=1} + S_k(t) \underbrace{\left(-\sum_{i=1}^{k} \tilde{p}_{t,i} \ln \tilde{p}_{t,i} \right)}_{=H_{\text{top-norm}}(t)} - P_{\text{tail}}(t) \ln P_{\text{tail}}(t).
\end{aligned}
$$

Since $S_k(t) = 1 - P_{\text{tail}}(t)$, grouping the first and third terms yields the single-bucket decomposition:

$$
H_{\text{exact}}(t) = (1 - P_{\text{tail}}(t))\, H_{\text{top-norm}}(t) + H_{\text{binary}}(P_{\text{tail}}(t)).
$$

Differentiating this expression with respect to $P_{\text{tail}}(t) \in (0, 1)$ while holding $H_{\text{top-norm}}(t)$ fixed gives:

$$
\frac{\partial H_{\text{exact}}(t)}{\partial P_{\text{tail}}(t)} = -H_{\text{top-norm}}(t) + \ln\left(\frac{1 - P_{\text{tail}}(t)}{P_{\text{tail}}(t)} \right),
$$

which is strictly positive on the low-tail regime $P_{\text{tail}}(t) < (1 + e^{H_{\text{top-norm}}(t)})^{-1}$ and becomes negative when $P_{\text{tail}}(t)$ approaches 1 because $H_{\text{exact}}(t)$ collapses the $V - k$ tail tokens into a single bucket.

By the Shannon grouping axiom, the exact full-vocabulary entropy satisfies $H(p_t) = H_{\text{exact}}(t) + P_{\text{tail}}(t)\, H(p_{\text{tail}})$. Under the maximum-entropy uniform tail distribution ($H(p_{\text{tail}}) = \ln(V-k)$), the Grouping-Corrected Entropy $H_{\text{corr}}(t) = H_{\text{exact}}(t) + P_{\text{tail}}(t)\ln(V-k)$ has partial derivative $\frac{\partial H_{\text{corr}}(t)}{\partial P_{\text{tail}}(t)} = \ln(V-k) - H_{\text{top-norm}}(t) + \ln\left(\frac{1-P_{\text{tail}}(t)}{P_{\text{tail}}(t)}\right) > 0$ for all non-degenerate tail regimes ($P_{\text{tail}}(t) < 1 - \frac{k}{V}$ since $H_{\text{top-norm}}(t) \le \ln k$), guaranteeing strict monotonicity in $P_{\text{tail}}(t)$. □

C.2 Potential-Outcomes View of the 8-Arm Design

Let $x \in \mathcal{X}$ denote the query, $S \in \{0,1\}$ denote the Turn-1 schema regime ($S=0$ for free-form natural language, $S=1$ for action-first tool schema $\mathcal{P}_{\text{tool}}$), $H \in \{1,2\}$ denote the conversational turn horizon, $C_1(x,s,h)$ denote the potential Turn-1 string generated under (s,h), $O(x,c_1,e)$ denote the Turn-2 injected observation under environment regime $e \in \{\emptyset, \text{reflect}, \text{echo}, \text{sim}, \text{null}, \text{real}, \text{adv}\}$, $\hat{y}(x,s,h,c_1,o)$ denote the generated final answer string, and $A(x,s,h,c_1,o) \equiv \mathbb{I}[\hat{y}(x,s,h,c_1,o) = y^*(x)] \in \{0,1\}$ denote binary task accuracy. Under greedy decoding (temperature $= 0.0$), $C_1(x,1,2) = c_1$ is deterministically identical across all observation-intervention arms (`Arms 1, 2a, 2b, 2c, 3`). The potential accuracy outcomes $A^{(m)}(x) \equiv \mathbb{I}[\hat{y}^{(m)}(x) = y^*(x)]$ evaluated by the primary arms are:

$$A^{(\text{wo-tool})}(x) = A(x,0,1,C_1(x,0,1),\emptyset), \quad A^{(\text{reflect})}(x) = A(x,0,2,C_1(x,0,2),o_{\text{reflect}}), \quad A^{(\text{echo})}(x) = A(x,1,2,c_1,o_{\text{echo}}(c_1)),$$

$$A^{(\text{direct-cot})}(x) = A(x,1,2,c_1,o_{\text{echo}}(c_1),\mathcal{P}_{\text{verify}}), \quad A^{(\text{self-sim})}(x) = \mathbb{E}_{o_{\text{sim}}\sim\mathcal{M}_\theta(\cdot|x,c_1,\mathcal{P}_{\text{sim}})}\big[A(x,1,2,c_1,o_{\text{sim}})\big],$$

$$A^{(\text{real})}(x) = A(x,1,2,c_1,o_{\text{real}}(c_1)),$$

with strict zero-context simulation $o_{\text{sim}} \sim \mathcal{M}_\theta(\cdot \mid c_1, \mathcal{P}_{\text{sim}})$ evaluated in our c_1-only ablation (Section 3). When $C_1(x,1,2) = \emptyset$ (Turn-1 Non-Invocation), $O(x,\emptyset,e) = \emptyset$ across all tool arms (`Arms 1, 2a, 2b, 2c, 3`), yielding NPIG = Halluc. Effect = 0. Taking expectations over $x \sim \mathcal{D}$ yields the exact telescoping identity (Definition 1):

$$\mathbb{E}_{x\sim\mathcal{D}}[A^{(\text{real})}(x) - A^{(\text{wo-tool})}(x)] = \underbrace{\mathbb{E}[A^{(\text{echo})} - A^{(\text{wo-tool})}]}_{\text{SSE (Composite Structural Scaffolding)}} + \underbrace{\mathbb{E}[A^{(\text{self-sim})} - A^{(\text{echo})}]}_{\text{Halluc. Effect}} + \underbrace{\mathbb{E}[A^{(\text{real})} - A^{(\text{self-sim})}]}_{\text{NPIG}_{\text{cond}}},$$

where composite SSE partitions against Two-Turn Reflection ($A^{(\text{reflect})}$) into horizon extension and the schema-first contrast:

$$\text{SSE} = \underbrace{\mathbb{E}[A^{(\text{reflect})} - A^{(\text{wo-tool})}]}_{\Delta_{\text{horizon}}} + \underbrace{\mathbb{E}[A^{(\text{echo})} - A^{(\text{reflect})}]}_{\text{SSE}_{\text{pure}}}.$$

Intervening with explicit Turn-2 chain-of-thought verification ($A^{(\text{direct-cot})}$) further splits the schema-first contrast SSE_{pure} into a residual protocol contrast (labeled Stage 1) and the effect of the verification instruction (labeled Stage 2):

$$\text{SSE}_{\text{pure}} = \underbrace{\mathbb{E}[A^{(\text{direct-cot})} - A^{(\text{reflect})}]}_{\Delta_{\text{schema,Stage1}}} + \underbrace{\mathbb{E}[A^{(\text{echo})} - A^{(\text{direct-cot})}]}_{\Delta_{\text{delib,Stage2}}}.$$

These identities are algebraic and hold for any set of arms; what they identify depends on which inputs differ between arms. $\Delta_{\text{schema,Stage1}}$ compares the tool protocol (with verification) against free-form reflection, so it bundles JSON serialization with the other differences between the two protocols (instructions, tool definitions, output format); Appendix H tests these differences separately. $\Delta_{\text{delib,Stage2}}$ measures the effect of the verification instruction given the same call and echo. Only contrasts among `Arms 2c, 2a, 1`, which hold ($S=1, H=2, C_1=c_1$) fixed, are controlled interventions on the observation: they measure the contribution of the external observation $o_{\text{real}}(c_1)$ (NPIG = Halluc. Effect + $\text{NPIG}_{\text{cond}}$) given the call the model actually made.

D Qualitative Transcript Forensics of Premature Schema Compression

To illustrate Premature Schema Compression, Figure 2 compares actual Turn-1 generations from `Qwen2.5-7B-Instruct` on a multi-step word problem from `gsm8k-easy` across `Arm 4` (`wo-tool`) and `Arm 2c`/`Arm 1` (`Schema Echo` / `Real Tool`).

Query (x): *"A merchant buys 15 crates of apples at $24 per crate. During transport, 3 crates are damaged and discarded. He sells the remaining crates at a markup of 35% above his total initial expenditure. What is the selling price per remaining crate?"*

Arm 4 (`wo-tool` — Free-Form Closed-Book CoT) → CORRECT ($40.50):

```
Let's break down the steps clearly:
1.  Total initial expenditure = 15 crates * $24/crate = $360.
2.  Remaining crates after 3 damaged = 15 - 3 = 12 crates.
3.  Target total revenue with 35% markup = $360 * 1.35 = $486.
4.  Selling price per remaining crate = $486 / 12 = $40.50.
<answer>40.5</answer>
```

Arm 1 / Arm 2c (Turn-1 Tool Schema Prompting) → PREMATURE SCHEMA COMPRESSION ($32.40 — WRONG):

```
<tool_call>
{"name": "calculator", "arguments": {"expression": "(24 * 1.35)"}}
</tool_call>
```

Mechanism: Because the system prompt requires the model to emit a structured `<tool_call>` JSON object on Turn 1 before generating prose equations, the model prematurely compresses the multi-step word problem into a single arithmetic string `"(24 * 1.35)"`, forgetting that the 35% markup applies to the *total expenditure divided by the 12 surviving crates* ($15 \times 24 \times 1.35/12 = 40.50$). When Turn 2 receives the calculator output (`32.4`), the model blindly copies `<answer>32.4</answer>`.

Figure 2: Representative transcript demonstrating how Turn-1 JSON schema enforcement induces Premature Schema Compression by suppressing intermediate algebraic variable binding.

E System Prompts, Tool Schemas & Reproducibility Specifications

Inference Infrastructure & Hyperparameters. All 10 open-weights models were served using `vLLM v0.7.2` on NVIDIA A100-SXM4-80GB GPUs with tensor parallelism (`TP=1` for 3B–8B models, `TP=2` for `Gemma-3-27B-IT`, `Qwen2.5-Coder-32B-Instruct`, and AWQ-4bit `Qwen2.5-72B-Instruct-AWQ`), `bfloat16` precision, greedy decoding (`temperature=0.0`) for deterministic counterfactual trajectory comparison, and `max_tokens=512` per turn (the `DeepSeek-R1-Distill` chat templates always open a think block, so our released runner suppresses thinking by pre-filling an empty one; the reasoning-budget ablation lets these models think, with `max_tokens=2048`). Prefix entropy probing (H_{exact} and H_{corr}) extracted `logprobs=20` over $K = 3$ greedy tokens following the `<answer>` prefix trigger on $\mathcal{P}_{\text{closed}}$.

Counterfactual Prompt Templates. Across all tool-enabled arms (`Arms 1, 2a, 2b, 2c, 3`), Turn 1 uses identical system instructions specifying the JSON tool schema inside `<tool_call>...</tool_call>` tags. In `Arm 2a` (`Self-Sim`), an isolated model instance receives `SELF_SIM_SYSTEM_PROMPT`: *"Predict the exact JSON response that the tool environment would return for the following function call based on your internal knowledge"* (conditioned on (x, c_1) in the main 10-model matrix and on c_1 alone when `include_question=False`). In `Arm 2c` (`Echo`), our runner supports both placeholder template B1 (`REDACTED_BY_MOCK_ECHO_PROBE`, `legacy_redacted=True`) and neutral templates B2–B3 (`legacy_redacted=False`; Appendix H). The `Arm 2c-DirectCoT` control (Table 8) instead keeps the closed-book system prompt, shows the model its Turn-1 call, states that tool execution is disabled, and asks it to evaluate the exact expression or query in that call step by step in plain text before answering inside `<answer>...</answer>` tags (`max_tokens=768`); it is thus a verification instruction, like wording B4, not the neutral default. Note that the action-space ablation in Table 9A executes all three protocols (`Action-First JSON`, `ReAct`, and `CodeAct`) under the strict unified multi-statement AST sandbox (`safe_exec_python`), which rejects prose/comments inside calculator arguments and therefore yields lower baseline `Action-First JSON` accuracies (17.0%–37.0% on `gsm8k-easy`) than the permissive single-expression parser in Table 6 (66.0%–69.0%). In `Arm 4` (`wo-tool`), tool definitions are omitted and the model is instructed to reason step-by-step and enclose its final answer in `<answer>...</answer>` (adding an inactive JSON tool schema to the `wo-tool` prompt without requiring a call does not lower closed-book accuracy, +2.6 pp; Appendix H). In `Arm 4b` (`Reflection`), Turn 1 prompts *"Step 1 (Decomposition): Write down the*

exact algebraic equations or logical sub-goals needed to solve this problem in natural language. Do not output the final answer tag yet," and Turn 2 prompts *"Step 2 (Resolution): Now evaluate your step-by-step equations carefully and state your final answer clearly inside* `<answer>...</answer>` *tags."* All code, benchmark datasets, and trajectory logs are released under an open-source license.

Sequential Value-of-Information (VoI) Calibration Objective. On a calibration split $\mathcal{D}_{\text{train}}$, the optimal gating threshold τ^* is selected to maximize tool call pruning while guaranteeing non-inferiority to ungated real execution (Arm 1):

$$\tau^* = \max\left\{\tau \in \mathcal{T} \mid \mathbb{E}_{x\sim\mathcal{D}_{\text{train}}}[\text{Acc}(x;\tau)] \geq \mathbb{E}_{x\sim\mathcal{D}_{\text{train}}}[\text{Acc}(\text{Arm 1})]\right\}, \tag{5}$$

where $\text{Acc}(x;\tau) \equiv \mathbb{I}[H_{\text{corr}}(x) < \tau]\,\text{Acc}^{(\text{wo-tool})}(x) + \mathbb{I}[H_{\text{corr}}(x) \geq \tau]\,\text{Acc}^{(\text{real})}(x)$. On the calibration split, this keeps the gate at least as accurate as mandatory tool calling. Fit on all items, it prunes about a quarter of calls (60.6% vs. 81.8% call rate; Table 9B); held-out results are in Appendix I.

Table 8: **Turn-1 c_1 Validity Taxonomy and `Arm 2c-DirectCoT` Recovery Across All 10 Models ($N = 7{,}000$ items).** We partition all 462 echo-loss items ($\hat{y}_i^{(\text{wo-tool})} = y_i^* \wedge \hat{y}_i^{(\text{2c})} \neq y_i^*$) into Stage-2 deliberation suppression, Stage-1 argument corruption, and Turn-1 non-invocation. The rightmost column reports echo-arm accuracy on `calc-reach` before and after adding the Turn-2 verification instruction (`Arm 2c-DirectCoT`; prompt in Appendix E), from separate A100 reruns; where two starting values are given, the first is from Table 6 and the second from the rerun.

Model (Scale)	Total Echo Loss Items	Stage-2 Deliberation Suppression (c_1 Valid in `Arm 1`)	Stage-1 Schema Compression (c_1 Flawed in `Arm 1`)	Turn-1 Non-Invocation (No `<tool_call>` Emitted)	Echo → Echo + Verification (`calc-reach`)
Qwen2.5-3B-Instruct (3B)	75	**36** (48.0%)	12 (16.0%)	27 (36.0%)	12.0% → **39.0%** (+27.0 pp)
Qwen2.5-7B-Instruct (7B)	71	**49** (69.0%)	20 (28.2%)	2 (2.8%)	24.0%/28.0% → **45.0%** (+17.0–21.0 pp)
Qwen2.5-Coder-7B-Inst (7B)	48	15 (31.2%)	4 (8.3%)	29 (60.4%)	28.0% → **41.0%** (+13.0 pp)
DeepSeek-R1-Qwen-7B (7B)	47	12 (25.5%)	**27** (57.4%)	8 (17.0%)	58.0% → **68.0%** (+10.0 pp)
DeepSeek-R1-Llama-8B (8B)	45	0 (0.0%)	0 (0.0%)	**45** (100.0%)	30.0% → **52.0%** (+22.0 pp)
Llama-3.1-Tulu-3-8B (8B)	19	1 (5.3%)	**16** (84.2%)	2 (10.5%)	35.0%/34.0% → **41.0%** (+6.0–7.0 pp; no echo drop)
Mistral-7B-Inst-v0.3 (7B)	56	18 (32.1%)	**38** (67.9%)	0 (0.0%)	1.0% → **17.0%** (+16.0 pp)
Gemma-3-27B-IT (27B)	32	**28** (87.5%)	3 (9.4%)	1 (3.1%)	44.0% → **64.0%** (+20.0 pp)
Qwen2.5-Coder-32B-Inst (32B)	33	**21** (63.6%)	5 (15.2%)	7 (21.2%)	45.0% → **63.0%** (+18.0 pp)
Qwen2.5-72B-Inst-AWQ (72B)	36	**23** (63.9%)	13 (36.1%)	0 (0.0%)	38.0% → **55.0%** (+17.0 pp)
Pooled 10-Model Total ($N = 7{,}000$)	**462**	**203 (43.9%)**	**138 (29.9%)**	**121 (26.2%)**	+6 **to** +27 **pp**

Table 9: **Top (A):** Action-Space Expressivity Ablation ($N = 100$ items per task under the strict `safe_exec_python` AST sandbox) comparing **Action-First Declarative JSON** ($B = 0$) against **`ReAct` (Thought-First JSON)** and **`CodeAct` (Imperative Python Execution)** under Real Execution, Passive Echo (`RedEcho`), and Post-Observation Verification (`DirectCoT`). [§]**Cond. Stage-1 Loss Share** reports $\Pr(\text{flawed } c_1 \mid \text{echo loss item})$: under placeholder echo B1, `ReAct` reduces Stage-2 losses and shrinks total loss items by $> 65\%$, so residual errors concentrate in Stage 1 while absolute flawed-c_1 items drop (with neutral echo B2, pre-call reasoning narrows the loss less; Appendix H). Across models and tasks, `CodeAct` and `ReAct` recover $+5.0$ to $+46.0$ pp and $+4.0$ to $+34.0$ pp on `Real Exec` (`Qwen2.5-7B CodeAct` $37.0\% \rightarrow 64.0\%$, `Tulu-3-8B ReAct` $17.0\% \rightarrow 51.0\%$, `Qwen2.5-3B CodeAct` $53.0\% \rightarrow 99.0\%$). **Bottom (B):** 10-Model Sequential Value-of-Information (VoI) Routing comparing Static Uncalibrated $H_{\text{exact}}(\tau_0)$ against VoI-Calibrated $H_{\text{corr}}(\tau^*)$ and Dynamic Per-Hop Routing (`Arm 5-Dynamic`). Here τ^* is fit on all items, so these rows are optimistic upper bounds; held-out (5-fold) results are in Appendix I. In Panel B, Arm 5-Static reports the initial live gate ($\tau = 0.14$ on Qwen2.5-Coder-32B; $\tau_0 = 0.35$ on the other nine models), whereas Table 1 applies a uniform $\tau_0 = 0.35$ across all 10 models (64.3% accuracy at 68.2% CR vs. 66.0% at 82.6% CR for Arm 1).

A. Action-Space Expressivity & Variable-Binding Ablation: Action-First JSON vs. `ReAct` vs. `CodeAct` Python ($N = 100$/cell)

Model	Task (1T `wo-tool`)	Action Protocol	Real Exec Acc	Passive `RedEcho` Acc	`DirectCoT` Echo Acc	Cond. Stage-1 Loss Share[§]	Turn-1 Gen Latency	Pre-Turn-1 H_{corr}, Gate Acc @ CR (Probe)
Qwen2.5-7B-Inst	`gsm8k-easy` (70.0%)	Action-First JSON ($B = 0$)	37.0%	21.0%	57.0%	68.5%	18.1 ms	65.0% @ 11.0% CR (16.9 ms)
		`ReAct` (Thought → JSON)	41.0%	49.0%	**58.0%**	82.8%	72.8 ms	66.0% @ 11.0% CR (16.9 ms)
		`CodeAct` (Python Let-Bindings)	**64.0% (+27.0 pp)**	26.0%	55.0% (**+29.0 pp**)	**35.3%** (**-33.2 pp**)	73.2 ms	**67.0% @ 11.0% CR** (16.9 ms)
	`toolbelt-multi` (38.0%)	Action-First JSON ($B = 0$)	17.0%	22.0%	36.0%	77.8%	8.4 ms	36.0% @ 13.0% CR (10.1 ms)
		`ReAct` (Thought → JSON)	23.0%	**29.0%**	**37.0%**	81.8%	25.8 ms	35.0% @ 13.0% CR (10.1 ms)
		`CodeAct` (Python Let-Bindings)	**39.0% (+22.0 pp)**	25.0%	32.0%	**38.9%** (**-38.9 pp**)	68.4 ms	**37.0% @ 13.0% CR** (10.1 ms)
Tulu-3-8B	`gsm8k-easy` (62.0%)	Action-First JSON ($B = 0$)	17.0%	22.0%	37.0%	86.7%	42.3 ms	49.0% @ 43.0% CR (16.3 ms)
		`ReAct` (Thought → JSON)	**51.0% (+34.0 pp)**	**52.0% (+30.0 pp)**	**57.0%**	100.0%	88.1 ms	**56.0% @ 43.0% CR** (16.3 ms)
		`CodeAct` (Python Let-Bindings)	29.0% (+12.0 pp)	15.0%	39.0% (**+24.0 pp**)	**76.0%** (−10.7 pp)	82.4 ms	49.0% @ 43.0% CR (16.3 ms)
Qwen2.5-3B-Inst	`calc-reach` (42.0%)	Action-First JSON ($B = 0$)	53.0%	10.0%	20.0%	75.0%	13.3 ms	49.0% @ 57.0% CR (2.5 ms)
		`ReAct` (Thought → JSON)	57.0%	**25.0%**	**29.0%**	41.2%	32.2 ms	42.0% @ 57.0% CR (2.5 ms)
		`CodeAct` (Python Let-Bindings)	**99.0% (+46.0 pp)**	8.0%	0.0%	**0.0%** (−75.0 **pp**)	12.0 ms	**82.0% @ 57.0% CR** (2.5 ms)
R1-Llama-8B	`calc-reach` (54.0%)	Action-First JSON ($B = 0$)	29.0%	29.0%	26.0%	100.0%	87.9 ms	43.0% @ 59.0% CR (3.4 ms)
		`ReAct` (Thought → JSON)	41.0% (+12.0 pp)	**39.0%**	**40.0%**	87.5%	173.9 ms	46.0% @ 59.0% CR (3.4 ms)
		`CodeAct` (Python Let-Bindings)	**55.0% (+26.0 pp)**	30.0%	29.0%	**29.2%** (−70.8 **pp**)	176.5 ms	**57.0% @ 59.0% CR** (3.4 ms)
	`gsm8k-easy` (49.0%)	Action-First JSON ($B = 0$)	43.0%	43.0%	43.0%	100.0%	93.2 ms	48.0% @ 47.0% CR (22.2 ms)
		`ReAct` (Thought → JSON)	**51.0% (+8.0 pp)**	**51.0%**	**51.0%**	100.0%	179.0 ms	**50.0% @ 47.0% CR** (22.2 ms)
		`CodeAct` (Python Let-Bindings)	48.0% (+5.0 pp)	45.0%	47.0%	**80.0%** (−20.0 pp)	176.6 ms	**50.0% @ 47.0% CR** (22.2 ms)

B. 10-Model Sequential Value-of-Information (VoI) Routing Summary ($N = 7{,}000$ Total Items Across 10 Models × 7 Tasks)

Routing Policy	`hop-necessity` Reason-to-Lookup (Cell B) Mean Acc	`hop-necessity` Reason-to-Reason (Cell D) Mean Acc	Full `hop-necessity` ($N = 100$) Mean Acc	10-Model 7-Task Macro-Average Acc @ CR
Arm 4: 1-Turn Closed-Book (`wo-tool`)	0.0%	**91.2%**	23.5%	31.6% @ 0.0% CR
Arm 1: Ungated Action-First Tool	43.2% (Cross-Hop Compression)	76.4% (−14.8 pp Compression)	60.8%	66.0% @ 81.8% CR
Arm 5-Static (H_{corr}, $\tau_0 = 0.35$)	35.2% (Horizon-1 Masking)	77.6%	59.1%	64.5% @ 68.8% CR
Arm 5-Static VoI (H_{corr}, τ^* **Calibrated**)	43.2%	90.8% (+13.2 pp)	64.7% (+5.6 pp)	**67.4% @ 60.6% CR (+1.4 pp** vs. `Arm 1`)
Arm 5-Dynamic Per-Hop VoI (H_{corr}, τ^*)	**78.8% (+43.6 pp)**	**90.8% (+13.2 pp)**	**72.5% (+13.4 pp)**	**68.7% @ 58.9% CR (+2.7 pp** vs. `Arm 1`)

F Multi-Model Generalization Across Action Spaces, Catalogs, and Prompts

To distinguish findings that hold across models from design-dependent moderators, we evaluate seven diverse 7B–8B models across 13,480 additional rollouts. This cohort spans generalists (`Qwen2.5-7B-Instruct`, `Mistral-7B-Instruct-v0.3`), code and tool specialists (`Qwen2.5-Coder-7B-Instruct`, `Hermes-3-Llama-3.1-8B`, `Salesforce/xLAM-7b-r`), and RLVR/reasoning post-trained models (`Llama-3.1-Tulu-3-8B`, `DeepSeek-R1-Distill-Llama-8B`). The primary generalization suite (7,680 rollouts) tests four orthogonal axes: call-graph depth **E1** ($d \in \{2, 3, 4\}$, $n = 40$/cell $\times 5$ seeds), catalog scale **E2** and statefulness **E3** ($n = 40$/cell greedy), and prompt surface variation **E5** (9 surfaces, $n = 120$/condition split into 60 parallel and 60 chained items), while Action-Space Expressivity (**E4**) is reported in Table 9A. We complement these with 5,800 robustness rollouts covering fixed-sandbox action spaces (Table 9A; $n = 100$, 1,800 rollouts), greedy cache-cleared repeatability (**E6**; 3 repeats at $T = 0.0$, 2,400 rollouts), and pre-call deliberation checks (**E7**; 1,600 action-first vs. 1,600 `ReAct` rollouts). All `CodeAct` snippets execute in a deterministic hardened sandbox (`exec_python_with_tools` / `safe_exec_python`) using a unified `globals` dictionary, an AST import guard (`math`, `fractions`, `decimal`), a 5,000-step `sys.settrace` limit (`SandboxCallLimit(BaseException)`), a 2.0 s `signal.ITIMER_REAL` wall-clock timeout, and `finally:` trace restoration (`sys.settrace(prev_trace)`). Each cell is filtered by tool-invocation compliance ($\tau_{\text{comp}} \in \{25\%, 50\%, 75\%\}$, where all four attributes remain CONDITIONAL across all three gates).

G Per-Model Suite II Results and White-Box Probe Diagnostics

While our controlled suite (§4, Appendix A) tests deterministic lookup engines that fix most closed-book errors (85% on `wiki-mixed` and 71% on `flight-easy`), open-ended tool execution is often noisy or incomplete. We therefore conducted a full two-turn counterfactual replication across six multimodal VLM benchmarks (`MathVista`, Lu et al., 2024; `ChartQA`, Masry et al., 2022; `DocVQA`, Mathew et al., 2021; `RealWorldQA`; `ScienceQA-IMG`, Lu et al., 2022; and `MMStar`, Chen et al., 2024), six text LLM benchmarks (`GSM8K`, Cobbe et al., 2021; `MATH-500`, Hendrycks et al., 2021; `PopQA`, Mallen et al., 2023; `MMLU-Pro`, Wang et al., 2024b; `HotpotQA`, Yang et al., 2018; and `MBPP`, Austin et al., 2021), and two external routing suites (`MetaTool`, Huang et al., 2024; and `SMART-ER`; $N = 6{,}480$ items across 9 models). For each item and model ($N = 1{,}920$ two-turn SOTA trajectories across 8 models; $N = 1{,}680$ across the 7 live tool-calling models with valid-call rate $\geq 10\%$), we hold Turn-1 c_1 byte-identical across Real, Echo, Null, Conflict, and Oracle Turn-2 arms and grade the model's actual Turn-2 response against gold ($y = \mathbb{I}[\text{Acc}_{\text{Real}} > \text{Acc}_{\text{Closed}}]$). To test whether internal hidden activations can detect counterfactual tool need prior to generation, we extracted a bank of 90 scalar probe features across 10 normalized layer deciles $L \in \{0.25, 0.40, 0.50, 0.60, 0.67, 0.75, 0.80, 0.85, 0.90, 1.00\}$. At each layer decile, the 9 extracted scalar signals are: (1) `k0_last_h_corr`: Logit Lens H_{corr} on prompt-final token ($K = 0$); (2) `k0_last_h_exact`: Logit Lens H_{exact} ($K = 0$); (3) `k0_jsd_to_final`: Jensen-Shannon divergence between intermediate layer logits and the final layer output distribution; (4) `k0_composite`: linear combination of prompt-final entropy and JSD divergence; (5) `k3_mean_h_corr`: mean H_{corr} over $K = 3$ prefix tokens; (6) `k3_noncopy_h_corr`: mean H_{corr} over non-prompt tokens at $K = 3$; (7) `k10_mean_h_corr`: mean H_{corr} over $K = 10$ prefix tokens; (8) `k10_noncopy_h_corr`: mean H_{corr} over non-prompt tokens at $K = 10$; and (9) `hybrid_k0_plus_k10`: joint combination of prompt-final hidden state and $K = 10$ prefix entropy (10 deciles $\times$ 9 signals $=$ 90 features). Table 11 presents the complete model-by-model two-turn decomposition and white-box gating falsification matrix.

H Controls for Prompt and Format Confounds

The Arm 2c vs. Arm 4 contrast changes the whole protocol, not only the output format. To test which part of the protocol causes the echo loss, we ran a control ablation study on five models (Qwen2.5-7B-Instruct, Qwen2.5-Coder-7B-Instruct, Llama-3.1-Tulu-3-8B, DeepSeek-R1-Distill-Llama-8B, and Mistral-7B-Instruct-v0.3) and four tasks (`calc-reach`, `gsm8k-easy`, `wiki-mixed`, and

Table 10: **7-Model Generalization Summary** ($N = 13{,}480$ **Generalization & Robustness Rollouts; Action-Space axis E4 is detailed separately in Table 9A).** Classifications are evaluated under a $\geq 50\%$ tool-call compliance gate (with identical CONDITIONAL classifications at 25%, 50%, and 75% gates, and identical E1 verdicts across the $d \in \{2, 3, 4\}$ macro-average and $d = 4$ max-depth slice).

Experiment & Design Axis	**Ledger Verdict** ($25/50/75\%$)	**Universal Invariant Claim (Holds Across Compliant Models)**	**Design-Dependent Moderator (Where Sign or Magnitude Shifts)**	**Per-Model Empirical Summary (E1: $n = 40 \times 5$; E2/E3: $n = 40$; E5: $n = 120$; E6/E7: $n = 100$)**
E1. Call-Graph Dependency ($d \in \{2, 3, 4\}$ Parallel vs. Chained; 3 Action Spaces) (7/7 compliant; COND at $25/50/75\%$)	**CONDITIONAL** `CodeAct` Python variable bindings eliminate or halve the magnitude of the parallel-vs.-chained gap across all 7 models ($\lvert\Delta_{\text{CodeAct}}\rvert \leq 7.5$ pp on 5/7 models).	**Action Space × Post-Training:** Generalist/RLVR models suffer Chained-Binding Collapse in Action-First JSON (par $\gg$ chn), whereas code/tool specialists exhibit Multi-Block JSON Batching Overhead (par $<$ chn).	**Generalist/RLVR** (par − chn **in JSON → CodeAct,** $d \in \{2, 3, 4\}$ **mean):** `Tulu-3-8B`: $+78.0 \rightarrow +\mathbf{4.5}$ pp; `R1-Llama-8B`: $+33.0 \rightarrow -\mathbf{6.8}$ pp; `Mistral-7B`: $+19.2 \rightarrow +\mathbf{3.2}$ pp; `Hermes-3-8B`: $+17.0 \rightarrow +\mathbf{7.5}$ pp; `Qwen2.5-7B`: $+9.0 \rightarrow +\mathbf{2.0}$ pp. **Code/Tool Spec:** `Coder-7B`: $-35.8 \rightarrow -\mathbf{15.7}$ pp (93.2% chn); `xLAM-7b-r`: -8.7 pp (`ReAct`).	
E2. Tool Inventory Scale ($\lvert\mathcal{T}\rvert \in \{5, 25, 100, 300\}$ Distractor Tools) (6/7 compliant [6/6/5 at $25/50/75\%$]; COND at all gates)	**CONDITIONAL** **Argument Binding Given Selection is INVARIANT** (0.0%)**:** Under `action_first_json`, `binding_error_pct` $= \mathbf{0.0}\%$ across all 7 models at all catalog sizes $\lvert\mathcal{T}\rvert \in [5, 300]$.	**Action Space × Catalog Size** ($\lvert\mathcal{T}\rvert = 300$)**:** Declarative JSON preserves high tool selection on non-coders at 300 tools (77.5%–100%), whereas `CodeAct` suffers namespace crowding on generalists while excelling on `Coder-7B`.	**JSON** ($\lvert\mathcal{T}\rvert = 5 \rightarrow 300$)**:** `Qwen2.5-7B`: $97.5\% \rightarrow \mathbf{100}\%$; `R1-Llama-8B`: $97.5\% \rightarrow \mathbf{97.5}\%$; `Tulu-3-8B`: $85.0\% \rightarrow \mathbf{77.5}\%$; `Hermes-3-8B`: $15.0\% \rightarrow \mathbf{85.0}\%$ (25% non-call at $\lvert\mathcal{T}\rvert = 5$ eliminated). **CodeAct** ($\lvert\mathcal{T}\rvert = 5/25 \rightarrow 300$)**:** `Coder-7B`: $100\% \rightarrow \mathbf{95.0}\%$; `Qwen2.5-7B`: $57.5\%/77.5\% \rightarrow 42.5\%$ (30% sel err); `Hermes-3-8B`: $72.5\%/97.5\% \rightarrow 40.0\%$; `Tulu-3-8B`: $37.5\%/67.5\% \rightarrow 5.0\%$.	
E3. Tool Semantics & Statefulness (`closed_book`, `read`, `write`, `must_not_write`) across tool types (6/7 compliant); **INVARIANT** within `closed_book`	**CONDITIONAL** **Closed-Book Non-Positivity** (NPIG ≤ 0)**:** Real execution is **never positive** ($\Delta_{\text{real-echo}} \leq 0$) on `closed_book` items across all 7 models (0.0% `unsafe_write` across all 7 models).	**Epistemic Necessity** (NPIG > 0)**:** On external `read`, stateful `write`, and safety-gated `must_not_write`, real execution reverses sign from negative/null to $+\mathbf{82.5}$ to $+\mathbf{100.0}$ pp.	**`closed_book`** (Real − Echo)**:** `Mistral-7B`: $-\mathbf{32.5}$ pp (5% vs. 37.5%); `R1-Llama-8B`: $-\mathbf{17.5}$ pp (30% vs. 47.5%); `Hermes-3-8B`: $-\mathbf{12.5}$ pp; `Tulu-3-8B`: -2.5 pp. **Stateful** (NPIG > 0)**:** `R1-Llama-8B`: $\mathbf{100}\%$ (`must_not_write`), $\mathbf{87.5}\%$ (`read`); `Mistral-7B`: $\mathbf{100}\%$ (`read`), $\mathbf{90.0}\%$ (`write`); `Coder-7B`: $\mathbf{100}\%$ (`read`).	
E5. Prompt Presentation (9 Surface Forms: Zero-Shot, Native Tool, CoT-Hint, 1-Shot Demo) (5/7 compliant; COND at $25/50/75\%$)	**CONDITIONAL** On RLVR-aligned `Tulu-3-8B`, the parallel-over-chained bottleneck is **invariant across all 8 active prompt surfaces** ($+70.0$ to $+100.0$ pp).	**In-Context Demonstration Moderation:** On `Qwen2.5-7B` and `Hermes-3-8B`, providing a 1-shot variable-binding demonstration (`fewshot_binding_demo`) collapses the chained penalty.	**`Tulu-3-8B`:** par $= 71.7\%$–100% vs. chn $= 0.0\%$–13.3% across all 8 surfaces. **`Qwen2.5-7B`:** gap drops from $+26.7$ pp (`native_tool`) and $+15.0$ pp (`zeroshot`) to $+\mathbf{1.7}$ pp (83.3% vs. 81.7%) under `fewshot_binding_demo`.	
E6 & E7. Greedy Nondeterminism & Manipulation Check ($n = 100$, 4 Models × 2 Tasks) (1,600 E7 rollouts; 2,400 E6 rollouts)	**VERIFIED** **100% Action-First Isolation:** Across all 1,600 E7 rollouts, `action_first_thinking_rate` $= \mathbf{0.0}\%$ (0.0 pre-call tokens), confirming a clean manipulation of $B(t_{\text{commit}}) = 0$.	**Metric Variance:** Raw accuracy greedy drift is small (0.0–4.0 pp on 3/4 models), whereas small-denominator conditional loss-share ratios drift by 5.0–23.6 pp.	**E7 Pre-Call Deliberation:** `Action-First` $= \mathbf{0.0}\%$ vs. `ReAct` $= \mathbf{97.0}\%$–$\mathbf{100.0}\%$ on `Qwen2.5-3B/7B` & `Tulu-3-8B`. **E6 Greedy Drift (Acc):** 0.0–4.0 pp (`R1-Llama-8B` 0.0–2.0, `Qwen2.5-7B` 0.0–4.0, `Qwen2.5-3B` 2.0–5.0), 0.0–7.0 pp (`Tulu-3-8B`).	

`hop-necessity`), with 100 items per cell (20 cells) and greedy decoding. Within each cell,

Table 11: **Complete Two-Turn Counterfactual SOTA Replication ($N = 1{,}920$ Trajectories Across 8 Models / 12 SOTA Benchmarks) & Passive White-Box Gating Falsification Ledger ($N = 6{,}480$ External Items Across 9 Models). Top (A):** Per-model two-turn accuracy across Closed-Book, Null, Echo, Real, and Oracle arms (`AF-JSON` primary), decomposing Total Gain into $\text{SSE} = \text{Acc}_{\text{Echo}} - \text{Acc}_{\text{Closed}}$ and $\text{NPIG} = \text{Acc}_{\text{Real}} - \text{Acc}_{\text{Echo}}$, plus the item-level Text vs. VLM tool-use subsets (percentage-point deltas computed from unrounded raw counts). **Bottom (B):** Diagnostic falsification of 10-layer ($K = 0$) and prefix ($K = 10$) white-box gates on counterfactual benefit labels $y = \mathbb{I}[\text{Acc}_{\text{Real}} > \text{Acc}_{\text{Closed}}]$ (LOMO_z denotes within-model z-scored leave-one-model-out transfer).

A. Two-Turn Counterfactual Decomposition & Text vs. VLM Tool Use on Gold-Return Items

Model / Subset	Suite (N)	Closed-Book	Null (o_{null})	Echo (o_{echo})	Real (o_{real})	Oracle (o_{ora})	Total Gain	SSE (Echo−Closed)	NPIG (Real−Echo)	Overstatement (Ora − Real)
Qwen2.5-3B-Instruct	6 Text SOTA (240)	17.9%	19.2%	19.6%	24.2%	81.2%	+6.2 pp	+1.7 pp	+4.6 pp	+**57.1** pp
Qwen2.5-7B-Instruct	6 Text SOTA (240)	30.0%	22.9%	23.8%	48.8%	82.5%	+18.8 pp	−6.2 pp	+**25.0** pp	+**33.8** pp
Qwen2.5-Coder-7B-Inst	6 Text SOTA (240)	27.9%	14.6%	15.8%	27.5%	68.8%	−0.4 pp	−12.1 pp	+11.7 pp	+**41.2** pp
Llama-3.1-Tulu-3-8B	6 Text SOTA (240)	18.8%	15.8%	19.2%	17.9%	27.1%	−0.8 pp	+0.4 pp	−1.2 pp	+9.2 pp
Mistral-7B-Inst-v0.3	6 Text SOTA (240)	17.9%	10.0%	15.0%	32.5%	72.1%	+14.6 pp	−2.9 pp	+**17.5** pp	+**39.6** pp
DeepSeek-R1-Llama-8B[‡]	6 Text SOTA (240)	4.6%	9.6%	9.2%	11.2%	30.0%	+6.7 pp	+4.6 pp	+2.1 pp	+**18.8** pp
Qwen2.5-VL-3B-Instruct	6 VLM SOTA (240)	60.0%	55.4%	56.7%	55.8%	78.3%	−4.2 pp	−3.3 pp	−0.8 pp	+**22.5** pp
Qwen2.5-VL-7B-Instruct	6 VLM SOTA (240)	60.0%	60.0%	60.0%	60.0%	77.1%	+0.0 pp	+0.0 pp	+0.0 pp	+**17.1** pp
All 8 Models `AF-JSON`	8 Runs (1,920)	29.6%	26.0%	27.4%	34.7%	64.6%	+5.1 pp	−2.2 pp	+**7.3** pp	+**29.9** pp
Pooled Live `AF-JSON`	7 Runs (1,680)	33.2%	28.3%	30.0%	38.1%	69.6%	+4.9 pp	−3.2 pp	+**8.1** pp	+**31.5** pp
Pooled Live `ReAct`	4 Runs (960)	23.6%	32.3%	32.3%	42.0%	64.5%	+18.3 pp	+**8.6** pp (**47.1**%)	+9.7 pp (52.9%)	+**22.5** pp
Text (Gold Return Subset)	6 Text LLMs ($k = 273$; 5 Live $k = 272$)	20.9% (21.0%)	17.6% (17.6%)	22.7% (22.8%)	**78.4%** (78.3%)	91.6% (91.5%)	+57.5 pp	+1.8 pp (3.2%)	+**55.7** pp (**96.8**%)	+13.2 pp
VLM (Gold Return Subset)	2 VLMs ($k = 24$)	**91.7%**	83.3%	**87.5%**	**91.7%**	95.8%	+0.0 pp	−4.2 pp	+4.2 pp	+4.2 pp

[‡] *`DeepSeek-R1-Llama-8B` is included in `All 8 Models AF-JSON` ($N = 1{,}920$) and `6 Text LLMs` ($k = 273$), and omitted only from the valid-call $\geq 10\%$ `Pooled Live` subset (valid-call rate $= 0.8\%$).*

B. Diagnostic Evaluation of Intermediate-Layer ($K = 0$) and Prefix ($K = 10$) White-Box Tool Gates

Diagnostic Suite	Diagnostic Control / Transfer Regime	Evaluated Feature Set	Within-Dist. AUROC	Controlled / OOD	Permuted Floor	Empirical Verdict
Diag. 1 (ID Shortcut)	Benchmark-Identity Shortcut vs. White-Box	90 White-Box vs. 1-Hot Benchmark ID	0.6646 vs. **0.7317**	0.5848 (`LOBO`)	0.5010 (+0.0837)	**1-Hot ID Beats Hidden States**
Diag. 2 (Planted Power)	Planted Positive Control at Matched IID	Transferable vs. Bench-Local Planted	0.7101 (Trans.) / 0.7030 (Loc.)	**0.6781** vs. 0.5136	Real (0.5848) ∈ Local	**Signal Is Benchmark-Local**
Diag. 3 (Domain Shortcut)	External `SMART-ER` Domain Shortcut	10-Layer × $K = 0..10$ White-Box	**0.8848** (Pooled)	**0.5063** (Math)	+0.3785 Gap	**Collapses Within Domain**
Diag. 4 (Lexical TF-IDF)	External `MetaTool` & Lexical TF-IDF Control	White-Box (0.8403) vs. Prompt TF-IDF	0.8354 ($\phi = -0.0414$)	**0.8163** (TF-IDF)	Int.→Ext. = 0.4698	**Lexical Prior; Orthogonal Label**
Diag. 5 (Joint OOD)	Full 90-Feature Cross-Axis Transfer Matrix	10 Layers × 9 Shared Signals ($d = 90$)	0.6713 (IID) / 0.5973	0.5256 / **0.5006**	0.4885 ($[0.442, 0.558]$)	**Zero Joint Transfer** (cos = +0.049)
Diag. 6 (Zero-Loss Frontier)	Deployment Accuracy–Call Frontier (`LOBO`)	Out-of-Fold `LOBO` Gate Routing	1/7 Runs Save Calls	**0.7% Mean Saving**	+0.0081 vs. Random	**Negligible Call Saving** ($< 1\%$)

every echo wording and both simulator conditionings reuse the same Turn-1 call c_1. The echo wordings are: B1, the placeholder string used in Table 1, which returns a placeholder result (`REDACTED_BY_MOCK_ECHO_PROBE`); B2, neutral JSON that only repeats the call arguments; B3, a status field with the note "Compute answer directly" (the default in our released runner); and B4, the B2 echo followed by an explicit instruction to verify the answer step by step. Table 12 reports mean accuracy over cells, the mean paired difference from a stated reference, and the number of cells in which that difference is negative.

Findings. (1) Tool definitions in the prompt do not hurt closed-book accuracy (+2.6 pp), so the echo loss is not a prompt-distraction effect; requiring the Turn-1 call is what costs accuracy (−11.3 pp against the same prompt with tool definitions). (2) Placeholder string B1 does not create the loss: neutral wordings, including the released default B3, make it larger (−8.7 and −8.8 pp vs. −4.7 pp), while an explicit verification instruction removes most of it (−2.1 pp; Qwen2.5-7B on `calc-reach` rises from 13% to 47%, its closed-book level). This matches Stage-2 deliberation suppression: after any tool return, models tend to stop reasoning unless asked to check. These reruns match Table 6 on average (B1 −4.7 vs. −5.1 pp on the same 20 cells) but differ in some cells (Qwen2.5-7B `calc-reach` echo: 13% vs. 24%). (3) Well-formed calls are not the fix: grammar-constrained decoding keeps the echo loss and lowers `gsm8k-easy` accuracy below closed-book for all five models (Qwen2.5-7B 71%→43%, Coder-7B 75%→26%, Tulu-3-8B 64%→44%, R1-Llama-8B 52%→17%, Mistral-7B 27%→10%). With the real tool, grammar-constrained calls, which are valid JSON by construction, reach 64.1%, close to the 65.6% of our default tag format, so the lenient parser does not inflate real-tool accuracy. (4) Reasoning before the call (ReAct) narrows the echo loss only slightly (−6.9 pp). Only R1-Llama-8B has a thinking mode, so for the other four models the D3 row is a 2,048-token rerun of B2 (−6.5 pp overall). With native thinking enabled, R1-Llama-8B's real-tool

Table 12: **Controls for Prompt and Format Confounds** (5 models $\times$ 4 tasks, 100 items per cell). Accuracy is the mean lenient accuracy over cells; Δ is the mean paired difference from the reference, in percentage points; the last column counts cells with $\Delta < 0$. Echo rows use the default custom-tag call format with a 512-token budget. Native chat-template calls (A2) are available for three models (12 cells).

Factor	Condition	Acc. (%)	Δ (pp)	Reference	Cells with $\Delta < 0$
Closed-book	Arm 4, 512 tokens (reference)	39.5	–	–	–
	Arm 4, 1,024 tokens	39.5	+0.0	Arm 4 (512)	5/20
	Arm 4 + tool definitions, no call required (1,024)	42.0	+2.6	Arm 4 (1,024)	6/20
	Arm 4b two-turn reflection (1,024)	40.3	+0.8	Arm 4 (1,024)	10/20
	Required Turn-1 call, neutral echo (B2)	30.8	−11.3	Arm 4 + tool definitions	15/20
Echo wording (Arm 2c)	B1: placeholder string (Table 1)	34.8	−4.7	Arm 4 (512)	13/20
	B2: neutral JSON repeating the arguments	30.8	−8.7	Arm 4 (512)	15/20
	B3: status field + "compute directly" note	30.6	−8.8	Arm 4 (512)	15/20
	B4: B2 + explicit verification instruction	37.4	−2.1	Arm 4 (512)	12/20
Call format (echo B2)	A1: custom tag (default)	30.8	−8.7	Arm 4 (512)	15/20
	A2: native chat-template tool call	22.1	−15.6	Arm 4 (512)	10/12
	A3: grammar-constrained JSON	20.1	−19.4	Arm 4 (512)	17/20
Call format (real tool)	A1: custom tag (default)	65.6	+26.2	Arm 4 (512)	5/20
	A2: native chat-template tool call	53.8	+16.2	Arm 4 (512)	4/12
	A3: grammar-constrained JSON	64.1	+24.6	Arm 4 (512)	6/20
Pre-call reasoning (echo B2)	D2: thought before the call (ReAct, 1,024)	32.6	−6.9	Arm 4 (1,024)	12/20
	D3: native thinking for R1-Llama-8B (2,048)	33.6	−6.5	Arm 4 (2,048)	13/20
Self-simulation (Arm 2a)	Simulator sees question and call	31.3	+0.15	Call only	5/20

accuracy on `calc-reach` rises from 31% to 65% (and from 50% to 64% on `gsm8k-easy`), but it calls the tool on only 11% and 0% of these items, so its gains come from internal reasoning rather than tool use; its non-invocation rows in Table 1 should be read as specific to our no-thinking setting. (5) The simulator's conditioning set changes accuracy by only 0.15 pp, so the difference between conditioning on question plus call (C1) and call only (C2; Appendix E) does not affect our conclusions. We release both configurations.

I Uncertainty Baselines, Held-Out Thresholds, and Multiple Testing

Matched Baselines. Table 13 compares prefix scores computed from the same $K = 3$ prefix and the same top-$k = 20$ log-probabilities, with 95% paired bootstrap confidence intervals (2,000 resamples) over Suite I items. AUROCs are computed within each model and averaged over the 10 models; p-values are unadjusted, and $p < 0.001$ means that no resample reversed the sign. We report two targets: closed-book error ($\mathrm{Acc}_4 = 0$) and actual tool benefit ($\mathrm{Acc}_1 > \mathrm{Acc}_4$). On closed-book error, H_{corr} is best overall (0.794 vs. 0.764 for top-1 confidence, paired difference $+0.030$, $p < 0.001$; vs. 0.781 for raw tail mass, $p = 0.046$). On actual tool benefit, all scores are similar (0.64–0.66), and H_{corr} is slightly below top-1 confidence (-0.007, $p = 0.014$), as Eq. 4 predicts: many items fail both with and without the tool, and no closed-book signal can see that. On `wiki-mixed`, where the tool usually works, H_{corr} predicts tool benefit best (0.922). Raw tail mass is weak on the `wiki-mixed`, `flight-easy`, and `hop-necessity` lookups (0.856 vs. 0.919 on closed-book error), and top-1 confidence is slightly better than H_{corr} for tool benefit there (0.751 vs. 0.726). On the four panels of Table 7, ten-sample semantic entropy is 10–42$\times$ slower than the modeled probe time (about 140 ms vs. 3–14 ms) and is more accurate only on Tulu-3-8B `calc-reach` (0.840 vs. 0.824 for H_{corr}).

Held-Out Thresholds. The uncalibrated gate (Arm 5, Table 1) applies one H_{exact} threshold, $\tau_0 = 0.35$, to every model, including Qwen2.5-Coder-32B (85.3% [82.7, 87.9] at a 77.7% call rate; the recorded live run used $\tau = 0.14$ for this model, and the change moves the 10-model macro-average by only -0.2 pp to 64.3% [63.2, 65.3] at 68.2%); H_{corr} at the same τ_0 gives 64.6% at 68.7%. For calibrated thresholds, on four folds we choose the τ that maximizes mean accuracy minus 0.05 times the routing rate, and apply it to the fifth, stratified by task and sub-regime, so no item is scored with a threshold fit on itself. With one threshold per model and task, H_{corr} reaches 66.7% [65.7, 67.7] at a 58.2% [57.1, 59.3] call rate, versus 66.0% at 82.6% for ungated Arm 1 ($+0.7$ pp [$+0.2$, $+1.3$]; 29.6% fewer calls). With one threshold per model across tasks, gating matches ungated accuracy (65.8% vs. 66.0%) with 15% fewer calls, so the accuracy gain requires task-level calibration;

Table 13: **Matched Prefix-Uncertainty Baselines** ($K = 3$, $k = 20$; AUROC with 95% paired bootstrap CIs for top-1 confidence $1 - p_1$ and H_{corr}). "CB error" is closed-book error; "Benefit" is actual tool benefit ($\text{Acc}_1 > \text{Acc}_4$). AUROCs are computed within each model and averaged over the 10 models; the lookup row pools `wiki-mixed`, `flight-easy`, and `hop-necessity`.

Items (N)	**Target**	$1 - p_1$	**Tail mass** P_{tail}	$H_{\text{top-norm}}$	H_{exact}	H_{corr} **(ours)**
All 7 tasks (7,000)	CB error	0.764 [0.750, 0.778]	0.781	0.765	0.779	**0.794** [0.781, 0.806]
	Benefit	0.652 [0.638, 0.665]	0.644	0.658	0.657	0.644 [0.631, 0.657]
Lookup: wiki, flight, hop (3,000)	CB error	0.911 [0.896, 0.923]	0.856	0.918	0.927	0.919 [0.906, 0.932]
	Benefit	0.751 [0.732, 0.770]	0.657	0.758	0.755	0.726 [0.707, 0.745]
`wiki-mixed` (1,000)	CB error	0.976 [0.967, 0.985]	0.864	0.978	0.980	0.980 [0.972, 0.988]
	Benefit	0.916 [0.900, 0.931]	0.833	0.911	0.915	**0.922** [0.906, 0.937]
`calc-reach` (1,000)	CB error	0.689 [0.651, 0.727]	0.538	0.689	0.689	**0.709** [0.669, 0.750]
	Benefit	0.519 [0.474, 0.555]	0.454	0.504	0.494	0.515 [0.474, 0.550]

thresholds transferred from the other nine models do worse than ungated Arm 1 (-0.8 pp [-1.0, -0.6] at a 76.0% call rate). Under the same held-out protocol, the other prefix scores reach similar operating points (top-1 confidence $+0.8$ pp at 58.3%; H_{exact} $+0.5$ pp at 56.7%), and raw tail mass reaches $+1.6$ pp [$+1.1$, $+2.2$] at a higher 62.7% call rate. Thus the practical benefit of gating comes from prefix uncertainty in general; H_{corr}'s advantage is better ranking of closed-book errors and a monotone response to tail mass, not a larger held-out accuracy gain. The VoI rows in Appendix Table 9B fit τ^* on all items and should be read as optimistic upper bounds.

Multiple Testing. Across the 70 task–model cells of Table 6, we test the echo contrast $\text{Acc}(\text{Arm 2c}) - \text{Acc}(\text{Arm 4})$ with exact McNemar tests and correct for multiplicity. Under Holm–Bonferroni ($M = 70$), 10 cells remain significantly negative and none is significantly positive (the largest positive cell, Tulu-3-8B on `calc-reach` at $+10.0$ pp, has $p_{\text{Holm}} = 0.37$); under Benjamini–Hochberg, 12 cells are significantly negative and this one cell is positive. $\text{NPIG}_{\text{cond}}$ is significantly positive in 55 of 70 cells after Holm correction. Pooled by task (1,000 items each, Holm over 7 tasks), the echo contrast is significantly negative on `calc-reach` (-12.0 pp, $p_{\text{Holm}} = 1.6 \times 10^{-21}$), `gsm8k-easy` ($-6.6$ pp, 2.5×10^{-5}), `toolbelt-multi` (-5.0 pp, 1.5×10^{-4}), and `hop-necessity` (-4.4 pp, 2.2×10^{-10}), and not significant on `flight-easy`, `table-agg`, and `wiki-mixed`. For SSE_{pure} itself (Arm 2c minus Arm 4b), the macro difference is -4.2 pp [-4.9, -3.5] (exact McNemar test on pooled items, $p = 8.7 \times 10^{-30}$); 9 of 70 cells remain significantly negative after Holm correction and none is positive, and pooled by task it is significant on `calc-reach` (-13.5 pp), `hop-necessity` (-6.0 pp), and `toolbelt-multi` (-4.8 pp). In the prompt and format control study, the neutral-echo version of this contrast (B2 minus Arm 4b) is -9.6 pp, negative in 14 of 20 cells.

J End-to-End Latency and Tool-Error Repair

To measure deployment cost rather than probe kernel time, we timed complete agent episodes for eight models, the Suite II cohort with DeepSeek-R1-Distill-Qwen-7B in place of R1-Distill-Llama-8B (six text LLMs: Qwen2.5-3B/7B, Qwen2.5-Coder-7B, Mistral-7B-v0.3, Tulu-3-8B, and DeepSeek-R1-Distill-Qwen-7B; two VLMs: Qwen2.5-VL-3B/7B), on 216 items stratified across the Suite II benchmarks (18–36 per model). Each episode includes model serving, Turn-1 generation, parsing, tool execution, and Turn-2 generation, timed as synchronized wall-clock time without prefix caching. We compare action-first JSON calls with thought-first (ReAct) calls, and add one repair turn that returns the parse or tool error to the model and lets it retry once. The gate uses the $K = 3$ closed-book prefix and routes to the tool only when $H_{\text{corr}} \geq \tau$.

Findings. Generation dominates episode time: for action-first JSON, Turn-1 and Turn-2 generation take 4.23 s and 2.99 s on average, while parsing (0.1 ms) and tool execution (2.2 ms) are negligible. The probe costs 0.17 s end to end (2.4% of an ungated episode). Gating cuts mean latency by 44% (action-first) and 49% (ReAct), but on these fallible tools it keeps only part of the tool's accuracy gain (28.2% vs. 31.3% ungated for action-first), so it is a latency–accuracy trade-off here rather than a free improvement; random routing at the same per-model rates reaches 26.9% (action-first) and 28.6% (ReAct), close to the gate's 28.2% and 31.1%. One repair retry is triggered on 52% of action-first

Table 14: **End-to-End Latency and Multi-Turn Repair** (8 models, 216 items; macro-averaged across models). Here the call rate is the gate's routing rate, which is the same for both call formats because it depends only on the closed-book prefix.

	Action-first JSON		Thought-first (ReAct)	
Configuration	**Latency (s)**	**Acc. (%)**	**Latency (s)**	**Acc. (%)**
Closed-book (Arm 4)	0.84	23.8	0.84	23.8
$K = 3$ probe only	0.17	–	0.17	–
Ungated tool agent (Arm 1)	7.22	31.3	11.75	36.4
Arm 1 + one repair retry	9.78	38.5	16.89	39.9
Gated, $\tau_0 = 0.35$ (46% of items call the tool)	4.04	28.2	5.95	31.1
Gated, $\tau_0 = 0.35$, + one repair retry	5.46	28.9	8.86	30.9
Random routing at the gate's per-model rates	–	26.9	–	28.6
Gated, median threshold (50% call)	4.24	29.8	6.90	33.5

episodes; it raises valid calls from 68.2% to 88.3% and successful executions from 48.4% to 63.9%, adding 7.3 pp accuracy for 2.6 s per item. For ReAct, where short Turn-1 budgets often truncate the call, repair raises valid calls from 28.4% to 67.3% and adds 3.5 pp for 5.1 s. Even with repair, accuracy stays far below the oracle-return ceiling of Section 5, so most of that gap reflects tool and extraction quality rather than malformed syntax. Adding a post-call verification instruction with real tools changes accuracy by -2.4 pp (action-first) and -2.6 pp (ReAct) and nearly doubles Turn-2 time (action-first: 2.99 to 5.80 s). These measurements are offline; we do not evaluate streaming, interruptions, or user-facing outcomes.