CALLING IS NOT USING: DO POST-TRAINING GAINS INCREASE DEPENDENCE ON TOOL RETURNS?

Anonymous authors
Paper under double-blind review

ABSTRACT

Tool-using language models are judged by task success, which does not separate using a tool from using what the tool returns: a model can score higher by calling more often, executing more reliably, and stopping in a scorable format, none of which requires reading a return. We trace interactions through five stages: routing, execution, acquisition, integration, and termination. Only integration leaves no trace, since a response can agree with a return without being caused by it; we measure it by intervention instead. Replaying each episode twice from the same history and first action, Natural replay preserves tool returns while matched-null replay replaces every executed return with a length-matched uninformative observation. Their paired accuracy difference measures the return's contribution to task success, and its change relative to the frozen base model is what post-training did to that dependence. Across our training grid, returns are worth most on the interface a model trained against: elsewhere accuracy rises, but most of the gain survives deleting them. Transfer does occur: among the single-source settings, supervised fine-tuning on mathematics yields the only confirmed increase on a debugging interface where every policy starts from the same failing program. But acquiring a return is not using it: on a diagnostic answerable only from the retrieved passage, all three Search-SFT runs retrieve the answer in every example yet answer only 7.3% of items correctly; given it in context, they are almost always right. Scoring well, acquiring information, and using it are three different things; tool-use transfer should be tested by the contribution of returns to success, not inferred from task performance.

1 INTRODUCTION

Language models increasingly solve tasks by interacting with external tools. They search the web, execute code, query databases, and operate software (Yao et al., 2023; Schick et al., 2023; Qin et al., 2024; Nakano et al., 2021; Jimenez et al., 2024; Mialon et al., 2024; Zhou et al., 2024; Yao et al., 2024), and benchmarks have followed them there, scoring models over multi-turn interactions with tools and environments (Wang et al., 2024; Chen et al., 2025). What nearly all of these evaluations report is a single number, whether the final answer was right, sometimes accompanied by how often a tool was called. That summary treats the interaction as a means to an endpoint. It is silent about the one step that makes a tool worth having: whether the model used what came back.

This omission changes how post-training gains are interpreted. We build a diagnostic in which the answer exists nowhere but in the retrieved passage: each item asks for a nonce string that the model cannot know and cannot guess. All three of our Search-SFT runs retrieve the passage containing that string on every single item, and then answer correctly on only 7.3% of items on average. Placed directly in context, the same string is reproduced almost every time (Figure 1(b)). Thus the failure occurs after answer-bearing content has been retrieved, rather than in the ability to report the nonce.

What this diagnostic shows in the extreme, an endpoint score hides in general. Accuracy can rise without greater reliance on return content: a policy may call a tool on more examples, dispatch calls that execute without error, or stop in a format the grader accepts. The ambiguity is sharpest immediately after post-training, when accuracy moves and we would like to say what changed, and sharper still in claims of transfer, where a gain on an unfamiliar interface is read as evidence that the model has learned to work with a new kind of return.

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

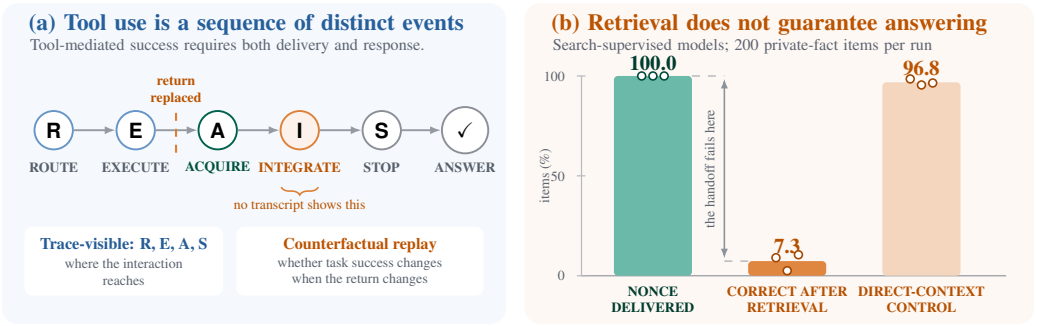

Figure 1: **Separating information delivery from downstream response.** **(a)** The stage analysis records where a tool interaction reaches; counterfactual replay measures whether changing the delivered return changes task success. **(b)** In the private-fact diagnostic, all three Search-SFT runs retrieve the target nonce on every item but answer only 7.3% correctly on average (2.5%–10.5% across runs). Direct placement yields 96.8% accuracy. Bars show means and dots show training runs.

Interactive evaluations come close to this question without reaching it. MINT scores behavior across multiple turns, and BrowseComp-Plus goes further by fixing the retrieval corpus so that search quality can be separated from answer quality (Wang et al., 2024; Chen et al., 2025). What such designs control is the information a model has available. Whether the answer it gave was caused by that information is a different question, and arranging the environment does not settle it; the return itself has to be intervened on.

We therefore study the interaction rather than its endpoint, and ask where two policies first behave differently. Does the policy route to a tool, execute the action, acquire answer-bearing content, let that content shape what follows, and stop with a scorable answer? Figure 1(a) labels these stages Route, Execute, Acquire, Integrate, and Stop. Transcripts settle four of them but cannot settle integration, because a response may agree with a return it never consulted; transcript agreement alone does not identify causal dependence on the return.

Integration is instead tested by intervention. We replay each episode twice from the same recorded history and the same first action. Natural replay preserves tool returns, while matched-null replay replaces every executed return with a length-matched uninformative observation. Their paired accuracy difference measures how much the return contributes to success, and comparing that contribution against the frozen base model measures what post-training did to it. When the training source and the evaluation interface differ, this Base-relative change is our estimate of transfer. One complication is that policies which differ in when they call a tool also differ in how often they receive a return to depend on, so on the naturalistic interfaces we report effects separately for examples on which the frozen base model does and does not call. We also add CodeDebug, a held-out debugging interface on which every policy starts each item from the same failing program, first action, and initial diagnostic, fixing opportunity by construction rather than by partition.

Our main experiment crosses training on Code, Math, and Search with supervised fine-tuning, group relative policy optimization, and on-policy distillation, and evaluates every trained policy on both source-matched and cross-domain interfaces; further experiments vary the model scale and family, the training-data mixture, and the trajectory teacher. Across this grid the two quantities come apart. Source-matched accuracy improves broadly, yet most of the gain survives deleting the returns, and it is not accompanied by a matching rise in acquisition. Cross-domain effects even point both ways, with search supervision raising dependence on Math and code supervision pushing it below the base model, while fixing the initial feedback opportunity on CodeDebug produces a further increase after mathematics supervision. The confirmed transfer effects are therefore conditional: they occur under supervised fine-tuning for particular source–regimen–target combinations and vary with the mixture, scale, model, and teacher. We consequently treat endpoint accuracy and return dependence as separate outcomes when evaluating whether post-training carries reliance to a new interface.

In summary, we contribute (1) matched-null replay, which turns the one stage a transcript cannot see into a measurable paired effect, embedded in a five-stage audit of the interaction; (2) CodeDebug,

a held-out interface that equalizes the opportunity to receive a return, so that dependence stays comparable across policies that call at different rates; and (3) evidence, across the grid above and its scale, family, mixture, and teacher variations, that endpoint gains and return dependence move independently.

2 Related Work

Tool-using agents and what their evaluations report. ReAct interleaves reasoning with environment actions, Toolformer learns when to call an API from self-supervised annotations, and ToolLLM instruction-tunes over thousands of REST APIs (Yao et al., 2023; Schick et al., 2023; Qin et al., 2024); others put the return to work directly, rewarding a browsing policy for answers humans prefer, feeding tool output back as a critique, or prepending retrieved passages (Nakano et al., 2021; Gou et al., 2024a; Lewis et al., 2020). More recent systems train the interaction itself from outcome reward or execution feedback (Jin et al., 2025; Wang et al., 2025; Gou et al., 2024b). Benchmarks score these policies the same way: MINT reports task success across interaction budgets, τ-bench and WebArena report completion on realistic workflows, and BrowseComp-Plus, closest to us in motivation, fixes the retrieval corpus so that search quality can be credited separately from answer quality (Wang et al., 2024; Yao et al., 2024; Zhou et al., 2024; Chen et al., 2025). Fixing the corpus separates the recall of a retriever from the accuracy of an answer, but the answer is still scored on its own, and a policy that retrieves the right passage and then ignores it scores like one that never had it. We take the same episodes these suites would score and re-run them without the returns.

Counterfactual tests of whether information is used. Closest to our method is a line that intervenes on an intermediate variable and asks whether the output follows. In knowledge-conflict question answering, Longpre et al. (2021) and Neeman et al. (2023) substitute the entity in a supplied passage to see whether a model answers from context or from parameters, while Liu et al. (2024) and Shi et al. (2023) show that supplied context is used unevenly or not at all; the same question has been asked of a model's own stated reasoning, which Turpin et al. (2023) and Lanham et al. (2023) perturb while finding the answer often unmoved, and interpretability work makes the intervention internal (Vig et al., 2020; Meng et al., 2022). These studies supply the intermediate variable directly. For a tool-using policy, the policy's own action decides whether an observation exists and which one. Yang et al. (2026) intervene upstream by swapping the cue that motivates a call and show that final accuracy can hide calls made for the wrong reason. We intervene downstream, replacing the observation returned by a realized action; the private-fact diagnostic isolates cases in which answer-bearing content is retrieved but does not reach the answer. Within each replay pair, fixing the realized history and first action ensures that the contrast is not confounded by a policy that simply calls more often. Because policies still differ in how often such pairs occur, we partition by Base routing when comparing them.

Post-training objectives and their transfer. The objectives we compare supervise trajectories at different resolutions: SFT imitates a teacher's tokens, GRPO and its variants reweight whole trajectories by outcome reward, and on-policy distillation scores the student's own trajectory under the teacher (Chu et al., 2025; Shao et al., 2024; Liu et al., 2025; Zheng et al., 2025; Agarwal et al., 2024). Evidence on what they carry across domains is mixed and increasingly negative. Yue et al. (2025) argue that verifiable-reward RL sharpens sampling within the base model's reach rather than extending it; Hao et al. (2026) find outcome RL destabilizes multi-step invocation without supervisory signal; Xi et al. (2026) report that RL gains survive changes of task far better than changes of interface; and Li et al. (2026) measure interference between reasoning and tool-use updates that share one optimizer. Against these, endpoint suites still report asymmetric wins, such as code accuracy improving after training on mathematics (Huan et al., 2025; Hu et al., 2025). These studies measure whether post-training expands the set of problems a model can solve, commonly through pass@k on static problems. Our replay contrast instead measures whether external returns contribute to correctness. The two quantities can vary independently: on source-matched interfaces, accuracy rises while most of the gain remains under return replacement. We capture this distinction with a Base-relative measure of the return's contribution after post-training.

3 METHOD

A tool-using episode can go wrong in several places, and an endpoint score reports only their conjunction. We therefore measure each policy twice. A stage trace follows an episode from its first action to its submitted answer and locates where policies behave differently. A paired replay then fixes that first action and changes only what comes back, testing whether such a difference reaches the answer.

Three interfaces share one action grammar and differ in the task, the returned observation, and the scorer (Table 7). Math and Search are the naturalistic targets, each a fixed panel of 200 ordered items: a Math answer is checked by a frozen symbolic verifier with normalized exact matching as a fallback, and a Search answer by alias-normalized exact equality. CodeDebug asks for a repair and scores a fresh-process execution of the submitted program. Accuracy on such a panel is the quantity every contrast below is built from. Decoding is deterministic, so any divergence between two replays of an episode originates in the observations they received.

3.1 STAGES OF A TOOL-USING EPISODE

We describe each trajectory with the five stages of Figure 1(a). ROUTE asks whether the first action is a tool call. EXECUTE asks whether that call returns without an outer protocol error. ACQUIRE asks whether the return carries answer-bearing content. INTEGRATE asks whether correctness depends on it. STOP asks whether the policy submits a scorable answer.

Route, Execute, and Stop are read off the transcript with the same criteria on every interface. Acquire needs a target-specific notion of answer-bearing content and is reported only where one exists: an answer-bearing passage on Search, a payload containing the gold value on Math. CodeDebug diagnostics never reveal hidden-test outputs, so Acquire is unavailable there.

Integrate is the stage a transcript cannot settle. A policy may repeat a return it did not rely on, or change its response without becoming correct, so agreement between the two is not evidence of dependence. We therefore use the final-outcome contrast after taking the return away as an operational measure of downstream dependence, not as an isolated mediator effect.

3.2 MATCHED-NULL REPLAY

Whenever an episode calls a tool, we record its history and its first tool action and replay the continuation twice. The *Natural* run proceeds with ordinary returns. The *matched-null* run replaces every executed return—the replayed first return and any return from a later, freely generated call—with an uninformative repeated-token sequence of the same token length. Later returns are replaced too, since a policy denied one return could otherwise recover the information by calling again. Matching length holds the observation's position and size fixed, so the runs differ in what a return says and not in whether one arrived. They are identical until the first return and may diverge afterward. An episode that never calls contributes nothing, since the two runs coincide. We denote accuracy in this all-return matched-null arm by N_0.

Let Y_i^N and Y_i^0 be binary correctness in the two arms. For model m on target t, matched-null dependence is

$$U(m,t) = \mathbb{E}_i[Y_i^N - Y_i^0] = N - N_0, \tag{1}$$

positive when the ordinary return buys accuracy over its matched null, negative when it costs accuracy. Post-training changes this quantity, so we subtract the contrast for the frozen Base:

$$T(m, s{\to}t) = U(m,t) - U(\mathrm{Base}, t), \tag{2}$$

where s and t denote the training source and evaluation target. We call T a transfer effect when $s \neq t$. The endpoint gain then decomposes as

$$N_{\mathrm{trained}} - N_{\mathrm{Base}} = \big(N_{0,\mathrm{trained}} - N_{0,\mathrm{Base}}\big) + T. \tag{3}$$

The first term is the gain that remains under matched null, and the second is the change in return dependence. Because replay continuations may diverge, U is an intention-to-treat, intervention-specific outcome effect: it includes any return-induced path through later calls, response, and Stop, rather than isolating semantic integration as a latent mechanism.

3.3 SEPARATING ACCESS, RETURN CONTENT, AND RESPONSE

A policy that calls more often also receives more returns, so an aggregate U mixes dependence with the opportunity to obtain a return. We separate these factors using route stratification, fixed-opportunity replay, alternative returns, and the private-fact diagnostic. Figure 2 summarizes the first two.

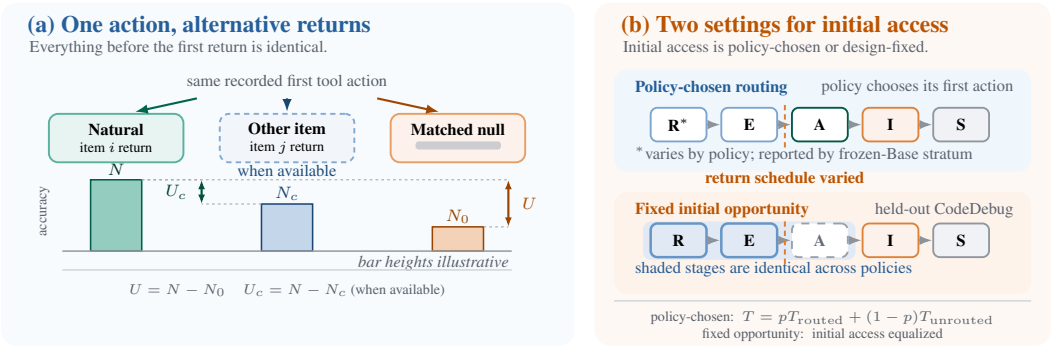

Figure 2: **Matched-null replay varies the return separately from initial access.** **(a)** We replay the same recorded first action, then provide Natural returns or replace every executed return with matched null. Their endpoint gap is U; for an alternative return c, its gap from Natural is U_c. Subtracting the corresponding Base contrast gives T or T_c. **(b)** In one setting, each policy chooses its first action and effects are stratified by whether the frozen Base routes. In CodeDebug, the initial action and feedback opportunity are fixed; later behavior may still diverge. The dashed Acquire gate marks the answer-bearing stage as unavailable on this hidden-output interface.

Policy-chosen routing. Math and Search leave the first action to the policy. Rather than override it, we partition items by whether the frozen Base calls a tool. With p the Base call rate,

$$T = p\,T_{\text{routed}} + (1-p)\,T_{\text{unrouted}}, \tag{4}$$

where T_{routed} and T_{unrouted} are the effects on items where the Base does and does not call. On the first stratum both policies had a chance to read a return; on the second only the trained policy did. The partition shows which stratum an aggregate effect comes from, while leaving each trained policy's action unchanged.

Fixed-opportunity replay. CodeDebug removes the access difference instead of accounting for it. Its 199 held-out items (106 from Agentica's DeepCoder-Preview-Dataset and 93 from TACO) are disjoint from Code training (Agentica Project, 2025; Li et al., 2023). A deterministic, outcome-blind AST mutation makes each program fail a public test; every policy then executes the same supplied failing program and first action, and so reads the same diagnostic. Natural preserves that diagnostic, whereas matched-null replaces every executed return as above. Appendix E gives the full contract.

Alternative returns. A matched null removes an observation's form along with its content. To keep the form and remove only item-specific information, we substitute a well-formed return from another item wherever the interface admits one. For each comparator c we replace N_0 by N_c and recompute $U_c = N - N_c$ and $T_c = U_{c,\text{trained}} - U_{c,\text{Base}}$. The Search-to-Math audit uses these comparator-indexed quantities.

Private-fact diagnostic. The last control holds the return fixed and isolates a different handoff, from acquisition to answer. Each of 200 generated two-hop items asks for a nonce absent from the question, so the retrieved passage is the only place the answer exists. We record whether retrieval delivers it and score the final answer; an oracle condition instead places the same nonce directly in context. Table 11 and Appendix B give the construction and both conditions.

3.4 MODELS, TRAINING, AND INFERENCE

The primary panel crosses three training sources with three methods, then evaluates every trained policy on two naturalistic targets, Math and Search. Math- and Search-source rows therefore include

a source-matched target, while Code-source rows are cross-domain by construction. CodeDebug is held out from training and supplies the fixed-opportunity comparison. Its separate family contains 12 contrasts: four sources (Code, Math, Search, and the mixed-source extension) crossed with three methods. Mix is not part of the three-source primary panel.

The frozen Base and all students are Qwen2.5-3B-Instruct, and Qwen2.5-72B supplies the teacher signal used by SFT and OPD (Qwen Team, 2024). The training sources differ in the return they expose: execution diagnostics on TACO and Agentica's DeepCoder-Preview-Dataset problems for Code, symbolic results from DeepMath for Math, and HotpotQA passages retrieved from a frozen Wikipedia-2018 E5 index for the primary Search source. Natural Questions appears in the held-out Search evaluation panel, not in the primary atomic Search training pool (Li et al., 2023; Agentica Project, 2025; He et al., 2025; Kwiatkowski et al., 2019; Yang et al., 2018; Wang et al., 2022).

The methods differ in the signal they learn from. SFT imitates teacher trajectories retained only when the task-specific scorer marks the terminal answer correct ("verified trajectories"); GRPO learns from terminal reward; and OPD uses token-level teacher feedback on student trajectories. We match the prompt count, exposure budget, update count, and training seeds, and evaluate the terminal checkpoint. Exact schedules appear in Appendix B.

Inference across items and training runs. The confirmatory unit is a full training run; each cell has three, with seed-specific data order and training stochasticity under a fixed contract. Crossed-bootstrap intervals are descriptive, not seed-level confidence intervals (Efron & Tibshirani, 1993). Confirmation uses the three seed effects, a two-sided t-test, and Holm correction (Holm, 1979), plus same sign and a same-sided descriptive interval. With $df = 2$, the test is sensitive to small between-run variance, so p_H is a low-resolution gate. The primary family has 12 estimands but 11 finite p-values because one nonzero constant row is untestable; CodeDebug has 12 defined tests.

4 Results

We organize the results around the distinction introduced above: obtaining a return, benefiting from it, and transferring that benefit are separate questions. We begin with a direct acquisition diagnostic, then turn to the training grid and the controls that separate routing from downstream use.

4.1 Successful retrieval does not guarantee a correct answer

Every Search-SFT run retrieves the private nonce on every item, yet average accuracy is only 7.3%. The same models are almost perfectly accurate when the nonce is placed directly in context and fail under matched null. The gap in Figure 1 and Table 11 therefore arises after acquisition: the answer reaches the trajectory but not the submitted response.

4.2 Endpoint gains do not imply stronger return dependence

Table 1: **Source matching consistently improves endpoint accuracy; cross-domain changes are mixed.** Entries count positive and negative changes in Natural accuracy relative to the frozen Base. Each cell averages three runs; both rows cover Math and Search.

Training–target relation	Positive ΔN	Negative ΔN	Range of ΔN
Source-matched	6/6	0/6	$[+.010, +.393]$
Cross-domain	6/12	6/12	$[-.087, +.255]$

All six source-matched cells improve, while the cross-domain cells split evenly, as summarized in Table 1. Within each method–target group, source matching also gives the largest Natural–null gap, yet several accuracy gains survive replacement. The complete grid appears in Table 12.

Three SFT rows are confirmed in Table 2: Search-SFT raises dependence on Math, Code-SFT lowers it, and Math-SFT raises dependence on CodeDebug under fixed access. Constant across seeds, Code-SFT→Search is not seed-testable, as shown in Table 8.

All three policies still outperform the Base after replacement, although their T values have opposite signs. The table makes the central dissociation concrete: endpoint improvement does not determine whether the ordinary return contributes more or less to success.

Table 2: **Confirmed changes in return dependence have both signs.** Retained is the share of Natural accuracy preserved under matched null. Intervals are descriptive, and p_H is corrected within each opportunity regime. The full ledger includes the seed-untestable Code-SFT→Search effect. Blue and orange shading mark increases and decreases.

Opportunity	Training → target	Method	U_{Base}	U_{trained}	Retained	T [descriptive 95% CI]	p_H
Policy-chosen	Search → Math	SFT	.125	.212	.47	$+.087\,[+.020, +.153]$	.026
Policy-chosen	Code → Math	SFT	.125	.018	.69	$-.107\,[-.160, -.055]$	.011
Fixed initial	Math → CodeDebug	SFT	.040	.199	.36	$+.159\,[+.094, +.226]$	.0489

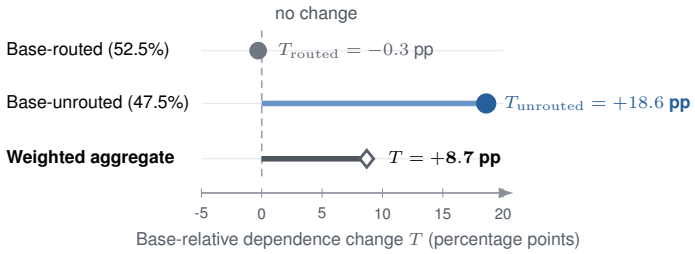

Figure 3: **The Search-SFT→Math increase comes almost entirely from Base-unrouted items.** Points show the two Base-defined strata; the diamond is their weighted aggregate.

4.3 Stage profiles reveal where endpoint gains arise

On Search, the stage profile records transcript events; replay tests whether they changed the outcome.

Search-SFT raises acquisition from .320 to .617 and improves return dependence with it. Math-SFT improves endpoint accuracy while remaining near the Base on both quantities. Code-SFT executes nearly every dispatched call but obtains no answer-bearing passage and shows no Natural–null gap. Successful execution is therefore a prerequisite for use, not evidence that the return helped.

Paired cases show the same distinction at the trajectory level. In one Search case, Code-SFT prints the question instead of retrieving, so both arms fail before acquisition. In one Math case, the two replays begin with the same calculation, but only the Natural return leads Math-SFT to the correct answer. The full episodes are in Appendix D. They illustrate failure paths rather than their prevalence, and naturalistic evaluation still lets each policy choose which items receive a return.

4.4 Controlling access separates routing from use

Policy-chosen routing. Figure 3 decomposes the Search-SFT-to-Math effect. Return dependence changes little on items that the Base already routes, and the aggregate increase comes almost entirely from items that the Base leaves unrouted. The same asymmetry recurs across the cross-domain Math cells: changes tend to be negative in the routed stratum and positive where the Base does not call. Table 10 reports the per-cell counts, and Table 9 gives the corresponding stage rates.

A 3,000-item known-mechanism calibration checks the stratification in Table 3.

Table 3: **Known-mechanism calibration separates routing from downstream use.** Each 3,000-item control changes one component from the same simulated Base; evidence-free accuracy is fixed at $q = .05$.

Control	Route r	Use u	Raw T	T_{routed}
Route only	$.50 \to 1.00$	$.60 \to .60$	$+.285$	$+.016$
Use only	$.50 \to .50$	$.60 \to .90$	$+.137$	$+.273$

The routed contrast suppresses a planted change in calling behavior while retaining a planted change in downstream use. Appendix G gives the full simulation.

Fixed initial opportunity. Table 4 asks the same question after access is fixed. Every policy starts from the same failing program, action, and diagnostic before repair.

Only Math-SFT has a confirmed increase. Its two additional runs preserve the positive direction, although this stability check does not replace the original family decision. Mix-OPD has a larger point

Table 4: **On CodeDebug, only SFT on mathematics has a confirmed increase in matched-null dependence.** Trained rows average three seeds and share one deterministic frozen Base. N_0 is accuracy after replacing every executed return; $U = N - N_0$ and $T = U - U_{\text{Base}}$, while N_0/N is the share of Natural accuracy that survives replacement, and Δ_{resp} is the fraction of paired episodes whose submitted answer changes. *ref.* denotes a reference condition. Crossed intervals are descriptive summaries of effect magnitude, while p_H is Holm-adjusted within this family. Green shading and bold type mark the confirmed row.

Source	Method	Endpoint arms		Dependence	Under replacement		Change from Base	Adjusted
		N	N_0	U	N_0/N	Δ_{resp}	T [95% CI]	Holm p_H
Base	Reference	.045	.005	.040	.11	*ref.*	*ref.*	*ref.*
Code	SFT	.228	.211	.017	.93	.08	$-.023\,[-.067, +.018]$	.198
Code	GRPO	.258	.214	.044	.83	.63	$+.003\,[-.094, +.090]$	1.000
Code	OPD	.018	.000	.018	.00	.02	$-.022\,[-.057, +.012]$	1.000
Math	SFT	.310	.111	**.199**	**.36**	.58	$\mathbf{+.159\,[+.094, +.226]}$	**.0489**
Math	GRPO	.039	.000	.039	.00	.24	$-.002\,[-.055, +.062]$	1.000
Math	OPD	.186	.034	.152	.18	.68	$+.112\,[+.049, +.178]$	.364
Search	SFT	.119	.025	.094	.21	.56	$+.054\,[+.002, +.106]$	.137
Search	GRPO	.059	.055	.003	.93	.69	$-.037\,[-.090, +.015]$	1.000
Search	OPD	.027	.007	.020	.26	.27	$-.020\,[-.060, +.030]$	1.000
Mix	SFT	.260	.214	.045	.82	.09	$+.005\,[-.044, +.052]$	1.000
Mix	GRPO	.285	.246	.039	.86	.66	$-.002\,[-.052, +.049]$	1.000
Mix	OPD	.308	.075	.233	.24	.47	$+.193\,[+.055, +.313]$	.995

estimate but is not confirmed because its run-level effects vary substantially. Code-SFT and Math-SFT reach similar Natural accuracy, yet Code-SFT retains most of that accuracy after replacement while Math-SFT does not. Appendix E.1 reports the extension.

The panel is difficult, and several trained cells remain near the floor. Their effect sizes can therefore turn on only a few repaired programs. Moreover, non-answer endings score zero, so U includes any return-induced change in whether the policy stops with a scorable submission. Response-change and nonce controls do not remove this path; U is not a Stop-free measure of semantic use.

Choice of replacement. In the Search-to-Math audit, T remains positive against hard-negative and corrupted returns, is near zero without injection, and is uncertain against an oracle. Appendix F.1 gives the comparison. The detectable matched null also elicits 2.06 queries per item, compared with 1.46 under Natural. Thus U is not a pure content effect under this comparator; Appendix J discusses the boundary.

4.5 Dependence varies across models and training choices

Table 5 tests the pattern across data mixtures, Qwen2.5 scales, model setups spanning Qwen2.5, Gemma 3, and Llama 3.1 (Qwen Team, 2024; Gemma Team et al., 2025; Grattafiori et al., 2024), and trajectory teachers. Each block centers one factor. Each block is a focused comparison whose exact contract and jointly varying factors are stated in the appendix.

These panels delimit generality. On Math, Appendix H shows that Mix-GRPO leads N, but Mix-SFT leads T. Across Qwen2.5 sizes, trained Search-SFT dependence stays near its 3B level while T shrinks as the Base becomes stronger; Table 13 gives the full scale results. Across model families, the Math effect moves from positive for Qwen2.5 to near zero for Gemma 3 and negative for Llama 3.1. The teacher comparison behaves differently: Math remains positive with either teacher, whereas Search reverses sign. Appendix I reports that comparison. The cross-setup divergence in Table 6 arises at different stages. For Llama, routing does not change, but execution, acquisition, and answer submission all decline. Gemma instead routes more often, while a reduction in invalid or otherwise unscored endings accounts for the residual change. These focused comparisons mark the boundaries of the tested grid rather than a universal ordering of models or teachers.

Table 5: **Endpoint accuracy and dependence rank methods differently across settings.** Each block changes one focal setting and reports three-seed means. E is the protocol-return rate among dispatched calls; bold marks the leader in the Ranking block.

Setting	N	U	T	E
Ranking: mixed-source training, evaluated on Math				
Mix-GRPO	**.515**	.263	+.138	.889
Mix-SFT	.475	.310	**+.185**	.693
Scale: Search-SFT across Qwen2.5 sizes, evaluated on Search				
Qwen2.5-1.5B	.372	.285	+.285	.989
Qwen2.5-3B	.400	.260	+.150	.993
Qwen2.5-7B	.433	.257	+.062	.991
Setup: Search-SFT across model families, evaluated on Math				
Qwen2.5-3B	.400	.212	+.087	.657
Gemma-3-4B	.142	.082	+.027	.495
Llama-3.1-8B	.167	.070	−.160	.702
Teacher: matched 86-prompt training set; evaluated on the 200-item Search panel				
Llama-3.1 teacher	.065	.030	−.080	.300
Qwen2.5-72B teacher	.330	.192	+.082	.966

Table 6: **Cross-model Math effects diverge at different stages.** Percentage-point changes from each Base to Search-SFT; C_A is acquisition contribution and Stop is Answer/Budget.

Setup	ΔR	ΔE	ΔA	ΔC_A	ΔS ans./bud.	T
Qwen2.5-3B	+47.5	+1.7	+13.2	+1.2	+43.3/ + 13.2	+8.7
Gemma-3-4B	+52.5	+19.1	+7.8	−1.2	+20.0/ + 31.8	+2.7
Llama-3.1-8B	+0.0	−7.9	−1.5	−10.2	−25.3/ + 25.3	−16.0

5 Discussion

Post-training can alter tool-use stages separately. Source matching improves accuracy, but much of the gain survives return replacement. Search-SFT retrieves every private nonce yet answers only 7.3% correctly on average; individual runs range from 2.5% to 10.5%. Several GRPO policies change CodeDebug submissions but retain almost all accuracy after replacement. Acquisition, response change, and outcome dependence are distinct. Traces locate behavioral changes; paired replay tests whether the return affects success.

Opportunity also matters. Search-SFT's Math effect lies mostly on Base-unrouted items; CodeDebug instead fixes the initial diagnostic, where only Math-SFT is confirmed. Thus free routing includes when to call, whereas fixed opportunity asks whether an available return matters more.

The results do not establish an objective ranking. All four confirmed dependence effects come from SFT: three in the primary/CodeDebug families and Mix-SFT→Math in the mixed-source family. Their signs differ, and the regimens differ in supervision and compute. Dependence also varies with data mixture, model setup, trajectory teacher, and return comparator. It belongs to a policy under an interface, opportunity regime, and intervention, not to a checkpoint alone. Appendix J details these boundaries.

6 Conclusion

Tool-use evaluations usually leave one claim implicit: higher post-training accuracy indicates that a model has learned to use its tools' returns. Pairing a five-stage audit with matched-null replay tests this inference. Endpoint performance and return dependence separate: a policy can retrieve an answer without reporting it, and accuracy gains can survive return replacement. The dependence change varies with mixture, scale, model, and teacher in both sign and magnitude. We therefore contribute an evaluation protocol, not a training prescription: report accuracy with the interaction path, return opportunity, and paired replacement effect.

Reproducibility Statement

The appendix specifies the replay procedure, evaluation contracts, training settings, and inference used for the primary 3-source $\times$ 2-target panel, CodeDebug, and the stage summaries. It records the construction of token-matched null returns, the shared interaction budget, the frozen-Base route split, the dependence contrasts, the response rate, and the crossed bootstrap with its draw count and fixed seed schedule. Complete evaluation and inference settings are given in Appendix B.

AI Use Statement

We used a general-purpose language model as a writing aid for copy-editing and for tightening prose in the manuscript. It was not used to generate research ideas, design experiments, produce or analyse results, or write the analysis code, and every claim and number in the paper was produced by the authors' own pipeline and checked by the authors.

Ethics Statement

This work post-trains and evaluates publicly released language models (Qwen2.5, Gemma-3, Llama-3.1) on publicly released datasets (TACO, DeepCoder-Preview-Dataset, DeepMath, Natural Questions, HotpotQA) together with a frozen Wikipedia retrieval index. All code execution occurs inside a network-disabled sandbox created fresh for each call. The study involves no human subjects, no personal data, and no deployment of the trained checkpoints.

References

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=3zKtaqxLhW.

Agentica Project. DeepCoder-Preview-Dataset, 2025. URL https://huggingface.co/datasets/agentica-org/DeepCoder-Preview-Dataset.

Zijian Chen, Xueguang Ma, Shengyao Zhuang, Ping Nie, Kai Zou, Andrew Liu, Joshua Green, Kshama Patel, Ruoxi Meng, Mingyi Su, Sahel Sharifymoghaddam, Yanxi Li, Haoran Hong, Xinyu Shi, Xuye Liu, Nandan Thakur, Crystina Zhang, Luyu Gao, Wenhu Chen, and Jimmy Lin. BrowseComp-Plus: A more fair and transparent evaluation benchmark of deep-research agent. *arXiv preprint arXiv:2508.06600*, 2025.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V. Le, Sergey Levine, and Yi Ma. SFT memorizes, RL generalizes: A comparative study of foundation model post-training. *arXiv preprint arXiv:2501.17161*, 2025.

Bradley Efron and Robert J. Tibshirani. *An Introduction to the Bootstrap*. Chapman and Hall/CRC, 1993.

Gemma Team et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *International Conference on Learning Representations*, 2024a.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Minlie Huang, Nan Duan, and Weizhu Chen. ToRA: A tool-integrated reasoning agent for mathematical problem solving. In *International Conference on Learning Representations*, 2024b. arXiv:2309.17452.

Aaron Grattafiori et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Yupu Hao, Zhuoran Jin, Huanxuan Liao, Kang Liu, and Jun Zhao. Why multi-step tool-use reinforcement learning collapses and how supervisory signals fix it. *CoRR*, abs/2606.26027, 2026. doi: 10.48550/ARXIV.2606.26027. URL https://doi.org/10.48550/arXiv.2606.26027.

Zhiwei He, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xingyu Chen, Yue Wang, Linfeng Song, Dian Yu, Zhenwen Liang, Wenxuan Wang, et al. DeepMath-103K: A large-scale, challenging, decontaminated, and verifiable mathematical dataset for advancing reasoning. *arXiv preprint arXiv:2504.11456*, 2025.

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

Chuxuan Hu, Yuxuan Zhu, Antony Kellermann, Caleb Biddulph, Suppakit Waiwitlikhit, Jason Benn, and Daniel Kang. Breaking barriers: Do reinforcement post training gains transfer to unseen domains? *CoRR*, abs/2506.19733, 2025. doi: 10.48550/ARXIV.2506.19733. URL https://doi.org/10.48550/arXiv.2506.19733.

Maggie Huan, Yuetai Li, Tuney Zheng, Xiaoyu Xu, Seungone Kim, Minxin Du, Radha Poovendran, Graham Neubig, and Xiang Yue. Does math reasoning improve general LLM capabilities? understanding transferability of LLM reasoning. *arXiv preprint arXiv:2507.00432*, 2025.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations*, 2024.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural Questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

Tamera Lanham, Anna Chen, Ansh Radhakrishnan, et al. Measuring faithfulness in chain-of-thought reasoning. *arXiv preprint arXiv:2307.13702*, 2023.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, 2020.

Rongao Li, Jie Fu, Bo-Wen Zhang, Tao Huang, Zhihong Sun, Chen Lyu, Guang Liu, Zhi Jin, and Ge Li. TACO: Topics in algorithmic COde generation dataset. *arXiv preprint arXiv:2312.14852*, 2023.

Yu Li, Mingyang Yi, Xiuyu Li, Ju Fan, Fuxin Jiang, Binbin Chen, Peng Li, Jie Song, and Tieying Zhang. Reasoning and tool-use compete in agentic rl:from quantifying interference to disentangled tuning. *CoRR*, abs/2602.00994, 2026. doi: 10.48550/ARXIV.2602.00994. URL https://doi.org/10.48550/arXiv.2602.00994.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics (TACL)*, 2024. arXiv:2307.03172.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding R1-Zero-Like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025.

Shayne Longpre, Kartik Perisetla, Anthony Chen, Nikhil Ramesh, Chris DuBois, and Sameer Singh. Entity-based knowledge conflicts in question answering. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2021. arXiv:2109.05052.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. arXiv:2202.05262.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: A benchmark for general AI assistants. In *International Conference on Learning Representations*, 2024.

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.

Ella Neeman, Roee Aharoni, Or Honovich, Leshem Choshen, Idan Szpektor, and Omri Abend. DisentQA: Disentangling parametric and contextual knowledge with counterfactual question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*, 2023. arXiv:2211.05655.

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*, 2024.

Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/d842425e4bf79ba039352da0f658a906-Abstract-Conference.html.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed Chi, Nathanael Schärli, and Denny Zhou. Large language models can be easily distracted by irrelevant context. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023. arXiv:2302.00093.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. arXiv:2305.04388.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Simas Sakenis, Jason Huang, Yaron Singer, and Stuart Shieber. Causal mediation analysis for interpreting neural NLP: The case of gender bias. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. arXiv:2004.12265.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*, 2022.

Xingyao Wang, Zihan Wang, Jiateng Liu, Yangyi Chen, Lifan Yuan, Hao Peng, and Heng Ji. MINT: Evaluating LLMs in multi-turn interaction with tools and language feedback. In *International Conference on Learning Representations*, 2024. arXiv:2309.10691.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, Yiping Lu, Kyunghyun Cho, Jiajun Wu, Li Fei-Fei, Lijuan Wang, Yejin Choi, and Manling Li. RAGEN: Understanding self-evolution in LLM agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*, 2025.

Zhiheng Xi, Xin Guo, Jiaqi Liu, Jiazheng Zhang, Yutao Fan, Zhihao Zhang, Shichun Liu, Mingxu Chai, Xiaowei Shi, Yitao Zhai, Xunliang Cai, Tao Gui, Qi Zhang, and Xuanjing Huang. Can RL improve generalization of LLM agents? an empirical study. *CoRR*, abs/2603.12011, 2026. doi: 10.48550/ARXIV.2603.12011. URL https://doi.org/10.48550/arXiv.2603.12011.

Yiwei Yang, Haoxiang Zhang, Bingbing Wen, Yao Lu, Yuchen Wu, Lei Zhang, Julian McAuley, Pan Lu, and Bill Howe. Spurious tool use: When rl agents learn the wrong reason to act, 2026. URL https://arxiv.org/abs/2609.16268.

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*, 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*, 2023.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/537d5aa768c2d534016a4d06f87bc8fb-Abstract-Conference.html.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, 2024.

A NOTATION

Symbol	Definition
Outcome conditions	
N	Natural accuracy: success when the tool's real return is delivered
N_0	matched-null accuracy when every executed return is replaced
Return and Base-relative contrasts	
$U = N - N_0$	Natural–null dependence under the specified replay schedule
$T = U_{\text{trained}} - U_{\text{Base}}$	Base-relative change in matched-null dependence after training
$U_c,\ T_c$	comparator-indexed dependence $U_c = N - N_c$ and Base-relative change $T_c = U_{c,\text{trained}} - U_{c,\text{Base}}$ for alternative return c
Route, response, and inference diagnostics	
$R_i,\ p = \mathbb{E}_i[R_i]$	R_i indicates whether the *frozen Base*'s first action is a tool call on item i; p is its routing rate
$T_{\text{routed}},\ T_{\text{unrouted}}$	T restricted to $R_i = 1$ and $R_i = 0$, respectively; $1 - p$ is the Base's *route headroom*
Δ_{resp}	label-free fraction of paired episodes whose normalized submitted answer changes after return replacement
N_0/N	fraction of Natural accuracy retained under matched-null replacement
p_H	Holm-adjusted p within a frozen comparison family
n/a	not defined for a reference condition
ref.	reference condition
Item-level counts	pooled counts are descriptive; all inference is at the training-run level

B SHARED INTERACTION PROTOCOL AND DOMAIN-SPECIFIC EVALUATION

The Math and Search evaluations hold the system prompt, action grammar, interaction budget, replay implementation, and failure handling fixed while changing the task, returned observation, and scorer. Code is a training source, not a primary naturalistic evaluation target. CodeDebug retains the same action grammar but fixes the initial program and first action and uses the shorter continuation budget specified in Appendix E.

Literal shared action contract. The policy-chosen Math and Search evaluations use the following prompt.

```
You are solving one task with optional tools.

Every assistant turn must contain exactly one complete action and no
    other text. The first non-whitespace characters must be either <
    python> or <answer>, and the last characters must be the matching
    closing tag.

Use this action to run Python or search, then stop and wait for the real
     tool observation:

<python>
# Python code. search(query, top_k=5) is available.
</python>

Use this action to give the final answer and terminate:

<answer>
your actual answer
</answer>

This rule still applies after a tool observation: a final answer such as
     Paris must be emitted as <answer>Paris</answer>, never as bare text
    .

Syntax-only interaction example:

Assistant: <python>print(1)</python>
```

```
Tool: STDOUT:
1
Assistant: <answer>1</answer>

Never copy the example's content as the task answer. Never invent a tool
     observation. Use at most 5 Python actions and at most 6 assistant
    turns. Do not emit <think>, <final>, prose outside tags, bare search
    (...), or multiple actions in one turn. Before sending each turn,
    verify that it is exactly one complete tagged action.
```

The common user wrapper is exactly

```
Task:
{question}
```

where {question} is the benchmark task body. Tags are case-sensitive; an assistant turn must be a full match to exactly one <python>...</python> or <answer>...</answer> action, with no prose outside the tags. The closing stop string is restored only when the generation otherwise contains one unclosed canonical action.

Evaluation panel and replay. The 112 retained primary Natural/null files comprise four Base files and 108 trained files (18 cells $\times$ three seeds $\times$ two arms). They use identical ordered 200-item panels, first-action replay, observation-token parity, and the same scorer. In the null arm, the recorded first action is replayed and every executed return, including returns from freely generated later calls, is replaced independently by repetitions of the tokenizer token `null`, including its leading space, with equal token length. Observations are deterministically truncated to 1,024 tokens before matching, and an assertion requires zero post-construction token-count difference. Natural later calls receive their ordinary returns. Every run uses its terminal checkpoint. Evaluation uses temperature zero, top-$p = 1$, and decoding seed zero, with 2,048 prompt, 4,096 response, and 8,192 context token limits. Each tool call runs in a fresh process with a 30-second timeout; invalid actions, context overflow, and tool failures score zero and remain in the denominator. The temperature .7 below applies only to online-training rollouts.

Table 7: **Domain-specific evaluation contracts.** All interfaces use the same action grammar; the task, returned observation, and scorer differ.

Domain	User task	Natural observation	Terminal scorer
Math	DeepMath problem	Sandboxed Python/SymPy output	Frozen symbolic verifier, with normalized exact matching as fallback
Search	NQ or HotpotQA question	Top-5 passages from the frozen Wikipedia-2018 E5 retriever	Alias-normalized exact equality for a short answer
CodeDebug	Repair a deterministic single-mutation failing draft; 106 Agentica DeepCoder-Preview and 93 TACO held-out items	Frozen diagnostic from the forced public-test execution	Stateless fresh-process execution of the final program on the first 15 frozen tests

Data construction. Code begins with 19,550 normalized-unique reference candidates: 9,704 from TACO and 9,846 from Agentica's DeepCoder-Preview-Dataset (Li et al., 2023; Agentica Project, 2025). Each reference solution must pass the frozen executable verifier. After 69 overlength exclusions, 19,481 items remain eligible; the deterministic split selects 10,000 training and 200 disjoint validation IDs. Math retains 30,000 deterministic candidates from 103,022 DeepMath rows; after 175 near-duplicate exclusions, 14,290 IDs have a qualifying teacher trajectory, of which 10,000 training and 200 disjoint validation IDs are selected. The primary Search source retains 30,000 candidates from 90,447 HotpotQA rows; after 62 exclusions, 12,467 IDs qualify, with 10,000 training and 200 disjoint validation IDs. Natural Questions is used in the held-out Search evaluation panel, not in this atomic Search training pool. The fixed 200-item Search evaluation panel takes the first 200 entries of the frozen validation order and comprises 97 Natural Questions and 103 HotpotQA questions. Both components come from their respective upstream training splits but are held out by the outcome-blind SHA-256 item split. No source pads its training set, and the final manifests report zero train–validation overlap.

The private-fact diagnostic uses the first 200 records from a deterministic 500-item validation split (generator seed 20260730). Each two-hop item has an isolated synthetic index containing an entity–alias document and an alias–nonce document; no nonce is inserted into Wikipedia. The oracle places the same nonce directly in context, and both conditions use alias-normalized exact match.

Training procedures. SFT uses global batch 64, microbatch 2 per GPU, an 8,192-token full-trace limit, cosine decay, and 3% warmup. GRPO and OPD use 40 prompts per rollout round, eight rollouts per prompt, 250 rounds, actor minibatches of eight prompt groups, temperature .7, a 2,048-token prompt cap, a 4,096-token response cap, zero warmup, and a constant schedule. All methods use optimizer learning rate 10^{-6}. GRPO uses binary task reward and reference KL coefficient 10^{-3}; OPD disables task reward and policy KL and uses coefficient-one, top-64, k=3 teacher-token distillation. These schedules match prompts, trajectory counts, and updates, but not tokens, FLOPs, teacher compute, or visited-state diversity; method contrasts therefore compare complete training procedures, not objectives in isolation.

Each training cell has 10,000 unique prompts, 80,000 sequences or trajectories, and 1,250 updates.

Inference across items and training runs. Multiplicity is fixed separately for each reported experiment: 12 cross-domain effects in the primary family, 18 pairwise method contrasts in the primary within-setting family, 12 CodeDebug source–method effects, and six mixed-source target–method effects. Within each family, we apply Holm correction to the defined two-sided one-sample t-tests over the three training-seed estimates. We retain an effect when the adjusted test rejects, all three estimates have the same strict sign, and a crossed seed–item interval lies strictly on that side of zero.

Each crossed interval uses 10,000 deterministic draws. Every draw independently resamples the three trained-seed rows and one common item-index vector, then applies that vector jointly to the paired Natural/null outcomes and the shared, fixed Base checkpoint. The training run is the unit of inference, so the seed-level tests have $df = 2$. When all three effects are exactly zero we set $p = 1$ by convention. A nonzero constant has zero estimated standard error, making the one-sample t statistic undefined; we therefore report its p-value as undefined and omit it from Holm ordering. The cross-domain inference ledger follows.

B.1 Complete Cross-Domain Inference Results

The table reports all 12 cross-domain Base-relative effects in the reported Holm family. Seed columns give the three training-run estimates, crossed-item intervals describe magnitude, and p_{H} gives the Holm-adjusted seed-level test. Exact-zero rows use the stated $p = 1$ convention. Nonzero constant rows have zero estimated standard error, so their one-sample t statistic and p-value are undefined.

Table 8: **All 12 cross-domain Base-relative effects; one constant row is not testable by the seed-level t-test.** The family contains 12 estimands; one nonzero-constant row has no defined seed-level t statistic or p-value, so Holm correction orders the 11 defined tests and marks that row *undef.* Blue and orange rows survive correction. The final column records direction, localization, or the limiting condition.

		Seed estimates (three training runs)			Descriptive crossed items	Adjusted test	Direction/localization
Method	Source→target	T_{11}	T_{22}	T_{33}	Mean [95% CI]	Holm p_{H}	Outcome
SFT	Code→Math	−.110	−.110	−.100	−.107 [−.160, −.055]	.011	decrease
SFT	Code→Search	−.110	−.110	−.110	−.110 [−.160, −.065]	undef.	acquisition failure; trained $U = 0$
SFT	Math→Search	−.015	−.025	+.015	−.008 [−.073, +.058]	1.000	not retained
SFT	Search→Math	+.085	+.080	+.095	+.087 [+.020, +.153]	.026	increase; Base-unrouted
GRPO	Code→Math	+.160	−.095	+.095	+.053 [−.085, +.180]	1.000	not retained
GRPO	Code→Search	−.115	−.090	+.045	−.053 [−.137, +.045]	1.000	not retained
GRPO	Math→Search	−.110	−.090	−.120	−.107 [−.160, −.057]	.061	not retained
GRPO	Search→Math	+.045	+.020	−.135	−.023 [−.135, +.080]	1.000	not retained
OPD	Code→Math	+.000	−.040	−.035	−.025 [−.088, +.037]	1.000	not retained
OPD	Code→Search	−.005	−.020	−.035	−.020 [−.073, +.033]	1.000	not retained
OPD	Math→Search	+.010	−.055	−.030	−.025 [−.087, +.037]	1.000	not retained
OPD	Search→Math	−.125	−.005	+.045	−.028 [−.128, +.068]	1.000	not retained

B.2 A Zero Trained-Side Contrast Can Hide an Acquisition Failure

Natural and matched null yield the same outcome for all 600 Code-SFT-on-Search checkpoint–item pairs, so $U_{\text{trained}} = 0$. Its $T = -.110$ therefore comes entirely from the Base term and coincides with an acquisition failure. Table 9 places this case alongside the remaining route-specific effects.

Table 9: **From paired outcome changes to route-specific effects.** The columns follow the analysis order: Help/Harm count the paired outcomes that change; U/T measure matched-null dependence and its Base-relative change; and the final columns split T by the Base's route. Trained-policy counts pool 600 checkpoint–item pairs; Base counts cover the single 200-item Base evaluation. Blue and orange mark the positive effect in the Base-unrouted stratum and acquisition failure discussed in the main text; in-domain rows are bold. *n/a* denotes the empty Base-unrouted Search stratum; every Base item routed on that target, so no conditional effect exists there.

	Outcome changes	Matched-null contrasts		Route-specific effects	
Source–method	Help/Harm	U	T	T_{routed}	T_{unrouted}
Target: Math	Base 27 Help / 2 Harm				
Code-GRPO	114/7	+.178	+.053	−.035	+.151
Math-GRPO	189/9	+.300	+.175	+.067	+.295
Search-GRPO	77/16	+.102	−.023	−.143	+.109
Code-OPD	60/0	+.100	−.025	−.133	+.095
Math-OPD	204/6	+.330	+.205	+.124	+.295
Search-OPD	69/11	+.097	−.028	−.124	+.077
Code-SFT	11/0	+.018	−.107	−.232	+.032
Math-SFT	213/8	+.342	+.217	+.121	+.323
Search-SFT	139/12	+.212	+.087	−.003	+.186
Target: Search	Base 23 Help / 1 Harm				
Code-GRPO	43/9	+.057	−.053	−.053	*n/a*
Math-GRPO	10/8	+.003	−.107	−.107	*n/a*
Search-GRPO	168/22	+.243	+.133	+.133	*n/a*
Code-OPD	54/0	+.090	−.020	−.020	*n/a*
Math-OPD	58/7	+.085	−.025	−.025	*n/a*
Search-OPD	60/4	+.093	−.017	−.017	*n/a*
Code-SFT	0/0	+.000	−.110	−.110	*n/a*
Math-SFT	75/14	+.102	−.008	−.008	*n/a*
Search-SFT	175/19	+.260	+.150	+.150	*n/a*

Table 10: **Cross-domain outcome changes align with Base-route strata.** Entries pool the six cross-domain source–method cells and three checkpoints per target, counting positive/negative item-level differences; ties are omitted. The final column reports directional agreement across cells. Search has no Base-unrouted stratum because its Base routes on every item.

Target	Base routed positive/negative	Base unrouted positive/negative	Agreement across cells
Math	161/376	205/20	6/6 routed negative; 6/6 unrouted positive
Search	203/391	*n/a*	6/6 routed negative

Table 11 isolates the same acquisition prerequisite: a response to a nonce can be measured only when retrieval delivers it.

Table 11: **Private-fact response requires nonce delivery.** The columns separate the retrieval precondition, the observed Natural–null response, and a direct-context positive control. Search-SFT delivers the nonce on every item and has positive U in all three seeds (green). The Base and Code groups almost never deliver it. Their oracle scores measure response to a nonce supplied directly in context, outside the retrieval path. The wide Code-OPD oracle range (.110–.655) is checkpoint heterogeneity; the oracle is diagnostic and is not used for a transfer claim.

	Retrieval precondition	Observed response		Positive control
Group	nonce delivered	N natural	U vs. null	N oracle
Base	.040	.000	+.000	.040
Code-OPD s11	.000	.000	+.000	.655
Code-OPD s22	.000	.000	+.000	.110
Code-OPD s33	.000	.000	+.000	.600
Code-SFT s11	.000	.000	+.000	.965
Code-SFT s22	.000	.000	+.000	.915
Code-SFT s33	.000	.000	+.000	.830
Search-SFT s11	1.000	.090	+.090	.985
Search-SFT s22	1.000	.025	+.025	.955
Search-SFT s33	1.000	.105	+.105	.965

C Complete Stage Diagnostics

Across the full 3-source × 2-target panel, interpreting dependence requires a stage profile, not one upstream rate. Figure 3 gives the main route decomposition; Table 12 reports every primary cell under one shared column contract, followed by the scale comparison. The contract for the main-text cross-model mechanism comparison follows the atlas.

C.1 Primary and Scale Stage Atlas

Route, Execute, Acquire, and Stop describe transcript events, whereas Integrate is measured by the paired outcome contrast $U = N - N_0$ and its training-induced change $T = U - U_{\text{Base}}$. The primary panel is reported once in Table 12; the scale table uses the same definitions.

Table 12: **Qwen2.5-3B primary 10,000-prompt stage atlas.** Rows cover the Base and every trained cell. Columns $R/E/A/S$ denote Route, Execute, Acquire, and Stop; Acquire is the answer-bearing item rate on Search and Math. $U = N - N_0$ represents Integrate, and $T = U - U_{\text{Base}}$ uses the target block's Base. Values are three-seed means over 200 items; *n/a* denotes a field that the interface does not define, and *ref.* marks the Base-relative contrast. gray marks Base and green source-matched training.

Source	Method	R	E	A	S	N	N_0	U	T
Target: Math									
n/a	Base	.525	.640	.115	.435	.145	.020	+.125	*ref.*
Code	SFT	.998	.882	.058	.998	.058	.040	+.018	−.107
Math	SFT	1.000	.668	.313	.937	.473	.132	+.342	+.217
Search	SFT	1.000	.657	.247	.868	.400	.188	+.212	+.087
Code	GRPO	.630	.737	.153	.620	.242	.063	+.178	+.053
Math	GRPO	1.000	.898	.317	.978	.538	.238	+.300	+.175
Search	GRPO	1.000	.883	.202	.980	.200	.098	+.102	−.023
Code	OPD	.993	.808	.263	.155	.100	.000	+.100	−.025
Math	OPD	1.000	.655	.257	.753	.402	.072	+.330	+.205
Search	OPD	.667	.677	.135	.418	.133	.037	+.097	−.028
Target: Search									
n/a	Base	1.000	.726	.320	.510	.115	.005	+.110	*ref.*
Code	SFT	.940	.975	.000	1.000	.083	.083	+.000	−.110
Math	SFT	1.000	.915	.340	.972	.282	.180	+.102	−.008
Search	SFT	.988	.993	.617	.982	.400	.140	+.260	+.150
Code	GRPO	.535	.779	.183	.962	.148	.092	+.057	−.053
Math	GRPO	1.000	.439	.002	.990	.108	.105	+.003	−.107
Search	GRPO	.998	.999	.677	.907	.392	.148	+.243	+.133
Code	OPD	.993	.525	.142	.390	.090	.000	+.090	−.020
Math	OPD	1.000	.598	.423	.715	.173	.088	+.085	−.025
Search	OPD	.667	.238	.200	.328	.125	.032	+.093	−.017

Table 13: **Qwen2.5 stage atlas across model size: naturalistic targets.** Rows cover 1.5B, 3B, and 7B on Search and Math. Route is $1 - P(\text{zero tool calls})$, Execute is protocol return among dispatched calls, and Stop is format adherence. Acquire is comparable across model sizes only on Search and is *n/a* otherwise; unlike the primary 3B atlas, the scale release has no comparable Math Acquire field. U_{B} is the same-size, target-matched Base contrast; T is the three-seed mean with its descriptive crossed seed–item 95% interval. *n/a* marks an undefined conditional rate or a field not stored comparably across model sizes. Green marks source-matched training; OPD was not run.

Size	Source	Method	R	E	A	S	N	U_{B}	U	T	desc. 95% CI
Target: Search											
1.5B	Search	GRPO	.000	*n/a*	.000	1.000	.145	+.000	+.000	+.000	[+.000, +.000]
3B	Search	GRPO	.998	.999	.677	.907	.392	+.110	+.243	+.133	[+.047, +.220]
7B	Search	GRPO	1.000	1.000	.752	.997	.472	+.195	+.235	+.040	[−.035, +.117]
1.5B	Search	SFT	1.000	.989	.578	.977	.372	+.000	+.285	+.285	[+.215, +.355]
3B	Search	SFT	.988	.993	.617	.982	.400	+.110	+.260	+.150	[+.077, +.225]
7B	Search	SFT	1.000	.991	.665	.983	.433	+.195	+.257	+.062	[−.023, +.145]
Target: Math											
1.5B	Search	GRPO	.000	*n/a*	*n/a*	1.000	.102	+.155	+.000	−.155	[−.205, −.105]
3B	Search	GRPO	1.000	.883	*n/a*	.980	.200	+.125	+.102	−.023	[−.135, +.078]
7B	Search	GRPO	1.000	.896	*n/a*	.963	.270	+.310	+.127	−.183	[−.348, +.005]
1.5B	Search	SFT	1.000	.608	*n/a*	.747	.305	+.155	+.155	+.000	[−.065, +.065]
3B	Search	SFT	1.000	.657	*n/a*	.868	.400	+.125	+.212	+.087	[+.020, +.153]
7B	Search	SFT	1.000	.722	*n/a*	.953	.562	+.310	+.298	−.012	[−.100, +.077]

Scale separates trained dependence from the Base-relative effect. On Search, Search-SFT's trained U remains near .26 across sizes, but T shrinks from $+.285$ to $+.062$ as same-size Base dependence rises. The scale slice therefore does not support a monotone size effect.

No stage alone identifies integration. In the 18-cell primary panel, 10 cells have Stop$\geq$ .90; among them, 2 have $|U| \leq .01$ and 4 have $U \geq .15$. None of the 18 has Route$<$ .20. Thus low stage completion can expose a missing opportunity, but high completion cannot establish dependence. The full profile is more informative than any single-stage proxy or global model ranking.

C.2 Cross-Model Stage Localization

Each setup compares its same-family Base with three frozen Search-SFT checkpoints (seeds 11/22/33) trained on the 10,000-prompt Search source, using the model's native tokenizer and chat template, and selected at the terminal checkpoint. We re-evaluate all rows on the same ordered 200-item Math panel, with temperature zero, decode seed zero, and the same Natural/matched-null replay. The original training realization and model-specific representation still vary across rows.

The main-text comparison in Table 6 uses the following shared definitions. R is first-tool-call rate, E is protocol return among dispatched calls, A is the answer-bearing item rate, and $C_A = P(A)\,\mathbb{E}[Y^N - Y^0 \mid A]$ is the full-item acquisition contribution. Stop reports Answer/Budget changes. These descriptive profiles compare complete model setups and localize the divergence without attributing it to model family alone.

D Two Paired Episode Summaries

On HotpotQA item 100014, Code-SFT executes `print(question)` rather than retrieving a passage, so both Natural and matched-null arms fail at Acquire and answer incorrectly. On DeepMath item 158, Math-SFT begins both arms with the same calculation, but the Natural return leads to the correct answer whereas matched null triggers another call and an incorrect answer.

E CODEDEBUG DEPENDENCE DOES NOT FOLLOW SOURCE PROXIMITY

CodeDebug tests whether source proximity predicts dependence when every model receives the same initial opportunity. We compare the existing Code-, Math-, Search-, and Mix-trained Qwen2.5-3B checkpoints on one common 199-item Natural/matched-null panel. The main panel (Table 4) applies one Holm correction to all 12 source–method contrasts; stage profiles then locate why endpoint and dependence rankings differ.

Panel, intervention, and scorer. The panel is constructed from an independent held-out code pool. A deterministic 200-item candidate split is formed before evaluation, with one identifier excluded by a predeclared data-build rule. The retained set has 106 items from Agentica's DeepCoder-Preview-Dataset and 93 TACO items, with no identifier overlap with the 10,000 Code training prompts. Starting from each reference solution, the builder searches a fixed order of single-point AST mutations and accepts the first candidate that fails a public test; it never reads hidden-test outcomes. The resulting panel contains 50 arithmetic-operator, 43 comparison, 52 integer off-by-one, and 54 variable-typo mutations, producing 81 runtime-error and 118 wrong-output diagnostics. Every policy executes the same supplied failing program and public input as its first action. Natural receives the frozen diagnostic; N_0 replaces every executed return, including those from later policy-chosen calls. The continuation allows at most three assistant turns and two Python calls. The stateless strict scorer runs the final submitted program in a fresh process against the first 15 frozen input–output tests. These choices define the held-out debugging interface used throughout the paper.

Endpoint accuracy and dependence produce different rankings. Code-SFT and Code-GRPO have high Natural accuracy, but retain most of it under matched null. Math-SFT, Math-OPD, Search-SFT, and Mix-OPD instead have positive T in all three seeds and descriptive intervals above zero. Only Math-SFT meets the 12-test seed-level Holm criterion; Mix-OPD's large mean varies substantially across seeds. Thus neither Natural accuracy nor source similarity predicts which effects survive correction.

Stage profiles locate two bottlenecks behind the mismatch. A payload-bearing call is a dispatched call with non-error, nonempty stdout or a last value; traceback diagnostics do not count. Code-SFT almost always answers, yet only .013 of its calls are payload-bearing and its aggregate T is negative: high Stop does not repair missing payload-bearing execution. Code-OPD reaches .496 payload-bearing calls but exhausts the trajectory budget on .958 of items and answers on only .022: payload-bearing execution does not ensure Stop. Math-SFT, Math-OPD, and Search-SFT pair payload-bearing rates of .428–.448 with positive U.

The mixed-source rows complete the comparison. Mix-OPD leads both Natural accuracy and T, whereas Mix-SFT and Mix-GRPO reverse order across those measures. Mix-OPD's larger contrast also coexists with frequent payload-bearing calls and substantial budget exhaustion. Across single- and mixed-source training, execution is therefore an intermediate condition; dependence further asks whether the return changes correctness and whether the policy stops with an answer.

E.1 Additional SFT Seeds Preserve the Source Ordering

Two independently trained seeds extend each SFT source from three to five runs. They use the same 199-item panel, evaluator, budgets, and intervention pair and leave the original twelve-comparison Holm family unchanged.

Table 14: **Five-seed sensitivity of CodeDebug SFT cells.** Per-seed T follows seed order 11/22/33/44/55. The mean uses all five seeds; the descriptive crossed seed–item interval uses only the two added seeds and 10,000 draws. All contrasts use the same Base ($U_{\mathrm{Base}} = .040$).

Source	per-seed T (11/22/33/44/55)	five-seed mean T	new-seed 95% CI
Code	$-.020, -.020, -.030, -.005, -.005$	$-.016$	$[-.035, +.025]$
Math	$+.141, +.176, +.161, +.141, +.141$	$+.152$	$[+.085, +.201]$
Search	$+.050, +.065, +.045, +.060, +.050$	$+.054$	$[+.010, +.106]$

Both added Math-SFT seeds are positive, while Code-SFT U never exceeds .035 and its mean remains below Math-SFT. Search-SFT stays positive and between them, preserving the source ordering. Because only SFT is extended, this is within-method stability and does not replace the primary Holm decisions.

F Matched-Null Contrasts Are Stable to Null Form but Sensitive to Information

Across the evaluated cells, matched-null contrasts preserve direction under changes to null identity and padding, whereas partial information and semantically different comparators can change the contrast. Matched-null replay holds the model's first action, return position, token count, and budget fixed while replacing the observation. Accordingly, we first verify these replay invariants and then vary the comparator's form and information content, including alternative returns on the positive Search-SFT→Math edge.

For binary scores, observation-invariant continuations contribute exactly zero, so $U = (N_{\mathrm{Help}} - N_{\mathrm{Harm}})/n_{\mathrm{items}}$. We test the remaining properties on Base for Math and Search and on three-seed Code-OPD→Math, Code-SFT→Search, and Search-SFT→Search cells. The following results summarize these checks. Across all 11 Natural/null pairs, first-action replay produces zero call-rate gap. All six direct-answer strata have exactly zero paired effect, as required when no return is observed. Three alternative null tokens preserve the sign and rank of all three evaluated cells. Finally, all nine padding-seed curves preserve direction, with a largest within-seed range of $.065$.

F.1 Comparator Content Separates Robustness from Invariance

For Search-SFT→Math, comparator content changes T magnitude and, under the oracle, its point-estimate direction. Hard-negative and corrupted intervals remain above zero and no auto-injection is near zero. The oracle return has a negative point estimate whose interval crosses zero.

All controls use the same three checkpoints and ordered 200-item Math prompt IDs, but were executed in a separate retrospective artifact. Its Base Natural file scores 28/200, whereas the primary release scores 29/200. The duplicate matched-null analysis gave $U_{\text{Base}} = .120$ and a mean contrast of $+.092\,[+.027, +.157]$. The retrospective artifact has no launch-time source receipt, so we treat these results as a descriptive sensitivity analysis rather than an independent replication. Table 15 reports the primary matched-null row ($U_{\text{Base}} = .125$, mean $T = +.087$) alongside alternative-comparator rows that remain internally paired with the retrospective Base.

Table 15: **Search-SFT→Math under alternative comparators.** Rows vary the return and report Base and mean trained U, seed-level T, and mean T with its crossed 95% interval. Blue marks positive alternatives, orange the negative oracle point estimate, and gray the null reference. The matched-null row is copied from the confirmatory primary release; the four alternative rows retain their retrospective artifact's internally paired Base.

Comparator	Dependence state		Seed-resolved change			Summary
Control return	Base U	mean trained U	T_{11}	T_{22}	T_{33}	mean T [crossed 95% CI]
Matched null (primary)	$+.125$	$+.212$	$+.085$	$+.080$	$+.095$	$+.087\,[+.020, +.153]$
Hard negative	$+.140$	$+.262$	$+.125$	$+.105$	$+.135$	$+.122\,[+.055, +.188]$
Corrupted	$+.140$	$+.293$	$+.175$	$+.135$	$+.150$	$+.153\,[+.087, +.220]$
Oracle	$-.175$	$-.215$	$-.020$	$-.060$	$-.040$	$-.040\,[-.118, +.042]$
No auto-injection	.000	$+.003$	$-.005$	$+.010$	$+.005$	$+.003\,[-.012, +.020]$

Hard-negative and corrupted results show that the positive contrast is not tied to one repeated-token null. No auto-injection anchors T near zero without an observation; the oracle reverses its point estimate but its interval crosses zero. Thus the evidence supports robustness to the tested imperfect returns, not invariance to information content.

Comparator effects across cells. Across cells, the controls separate changes in null form from changes in information content. Replacing the repeated-token null with three alternative nonempty nulls preserves all three cell signs and their rank order, while every padding seed curve preserves direction. The separate Search-SFT→Math comparator audit above reports the effect of changing information content rather than null form.

These checks begin only after a return is observed, so they do not isolate the opportunity to receive useful evidence. The Base-route decomposition in Section 4.4 addresses that separate question, and the calibration below shows what the decomposition recovers when the two mechanisms are varied one at a time under a known data-generating process.

G Synthetic Controls Separate Routing from Downstream Use

In simulation, known mechanisms allow a direct test of whether the route-stratified contrast separates routing from downstream use. We first change those mechanisms independently within one simulated cell, then test a planted domain-status difference across Base–target clusters. For the cell-level calibration, r is the probability of routing, u the probability of using returned evidence, and q the probability of answering correctly without evidence. Both controls begin from the same simulated Base and change one parameter over 3,000 items (Table 3).

The cell-level contrasts respond differently to the two interventions. Route expansion produces a sizable raw T but leaves T_{routed} near zero; increasing downstream use with routing fixed instead produces a large positive T_{routed}. Thus the route-stratified estimator suppresses the route-only contrast while retaining the planted-use contrast in the simulator.

Aggregation across cells. The aggregate calibration asks whether the same estimator preserves a planted in-domain/cross-domain difference. A second simulation plants higher use in-domain and lower use cross-domain in four Base–target clusters, each with three cells and three seeds per domain status. The cluster bootstrap recovers a contrast of $+.337$ $[+.320, +.356]$; all four cross-domain cluster means are negative (one-sided exact sign $p = .0625$). The positive aggregate contrast and four negative cluster signs recover both components of that design.

Together, the two calibrations establish the estimator's behavior under known mechanisms. We next return to observed policies and ask how mixed-source training distributes dependence across familiar interfaces.

H MIXED-SOURCE EFFECTS CONCENTRATE ON MATH

Mixed-source training does not distribute return-dependence gains evenly. The only confirmed effect within this six-test family is Mix-SFT→Math; for GRPO, Math has a larger T than Search ($+.138$ versus $+.097$), whereas OPD ties the two targets at $+.100$. We therefore restrict the paper-level analysis to endpoint and dependence effects.

The experiment exposes each policy to all three familiar interfaces. It therefore tests target-wise dependence under mixed training, not transfer to an unseen domain. We hold the total source budget at 10K prompts, allocated as 3,334 Code, 3,333 Math, and 3,333 Search prompts, and otherwise fix the model, three seeds, terminal-checkpoint rule, paired Natural/matched-null evaluation, and 200-item Math and Search target panels. The six target–method effects form a separate Holm family.

The null control changes the Math ranking reported by endpoint accuracy. GRPO leads in Natural accuracy, but its stronger matched-null performance leaves SFT with the larger Base-relative contrast. The endpoint leader is therefore not the dependence leader. Because Mix includes Math data, this is a mixed-training effect rather than unseen-domain transfer.

Table 16: **Mixed training: from endpoint accuracy to confirmed dependence.** The columns report endpoint accuracy (N and N_0), dependence ($U = N - N_0$ and $T = U - U_{\text{Base}}$), crossed-item uncertainty, and the Holm-corrected seed test. Only Mix-SFT→Math satisfies the same-sign, crossed-interval, and six-test Holm criteria; blue marks this row.

Experimental cell		Endpoint arms		Dependence contrasts		Training runs	Descriptive crossed items	Confirmatory seeds
Target	Model	N	N_0	U	T	T (11/22/33)	95% CI	Holm p
Math	Base	.145	.020	.125	*ref.*	*n/a*	*n/a*	*n/a*
Math	Mix-SFT	.475	.165	.310	$+.\mathbf{185}$	(+.190/+.170/+.195)	$[+.\mathbf{117}, +.\mathbf{255}]$	**.0102**
Math	Mix-GRPO	.515	.252	.263	+.138	(+.120/+.170/+.125)	[+.067,+.212]	.0648
Math	Mix-OPD	.382	.157	.225	+.100	(+.095/+.060/+.145)	[+.022,+.177]	.136
Search	Base	.115	.005	.110	*ref.*	*n/a*	*n/a*	*n/a*
Search	Mix-SFT	.350	.165	.185	+.075	(+.080/+.100/+.045)	[+.002,+.148]	.136
Search	Mix-GRPO	.353	.147	.207	+.097	(+.035/+.130/+.125)	[+.010,+.185]	.136
Search	Mix-OPD	.308	.098	.210	+.100	(+.135/+.070/+.095)	[+.028,+.175]	.136

The ranking reversal is visible directly: Mix-GRPO leads in Natural accuracy, whereas Mix-SFT has the larger dependence change. The next analysis tests another source of target variation: which model supplies the verified training trajectories.

I TEACHER CHOICE SEPARATES SEARCH FROM MATH

Both teachers preserve positive Math T but reverse Search T. We fix the Qwen2.5-3B student and the same 86 Search prompt IDs, changing only the model that supplies scorer-verified SFT trajectories. The shared IDs are those for which both teachers produced a strictly correct trajectory; each set is then deterministically padded and repeated to 7,616 training rows for one epoch. These rows estimate teacher sensitivity on the common subset. The primary Search-SFT result instead uses 10,000 qualified Qwen2.5-72B trajectories under a different training contract. The table pairs target contrasts with Search execution and stopping.

Table 17: **Teacher-conditioned trajectories change Search dependence and reported behavior.** The student and 86 selected Search prompts are fixed across the two teacher rows. Target cells report three-seed mean T with descriptive crossed seed–item 95% intervals. Natural-arm Search columns report protocol return among dispatched calls and Answer/Budget/Other terminal-state rates.

Training source	Cross-domain T		Search behavior	
Trajectory teacher	Math T	Search T	Execute	Stop: Answer/Budget/Other
Llama-3.1-8B	$+.130\,[+.058, +.200]$	$-.080\,[-.133, -.027]$	.300	.628/.353/.019
Qwen2.5-72B	$+.140\,[+.075, +.207]$	$+.082\,[+.012, +.152]$	.966	.990/.010/.000

Both Math intervals lie above zero, whereas Search moves from below zero with Llama to above zero with Qwen. This reversal coincides with low execution and frequent budget exhaustion under Llama versus reliable execution and almost universal answering under Qwen. With the student and prompts fixed, teacher choice changes both the downstream path and the sign of Search T, not the Math direction.

J Interpretation Boundaries

Our results identify policy-level dependence under the evaluated return interventions. Five design choices bound what that dependence can mean: return opportunity, comparator content, target-specific response and scoring, statistical resolution, and the coupled configurations used to test generalization.

Opportunity is policy-relative. Base and trained policies can choose different actions, so policy-level T may include learning where to call. Search-SFT→Math is the clearest case: training expands routing from roughly half the items to all of them. The Base-route split fixes the item partition rather than a common action and applies only when Base leaves route headroom. In the primary Qwen2.5-3B Search-SFT→Math cell, the estimate within the routed stratum is $-.003$, whereas the Base-unrouted estimate is $+.186$ (Table 9). On Base-unrouted Math items, T combines learning to call with using the return. The split therefore localizes opportunity without isolating action-fixed content use.

Comparator choice shapes measured dependence. An equal-length matched null controls length but can still be recognized as abnormal. In the retrospective Search-SFT→Math audit, pooled over three checkpoints and 600 episodes, it elicits 2.06 queries per item, compared with 1.46 under Natural. Thus U includes the policy's response to the intervention. The same audit makes this sensitivity visible: hard-negative and corrupted returns produce larger positive effects than the matched null, whereas withholding the observation brings the effect near zero. The private-fact control adds a delivery check: groups that do not retrieve the nonce have no opportunity to use the planted fact. Search-SFT retrieves the gold nonce on every item, but answers correctly on only $.025$–$.105$ of items; when the same nonce is placed directly in context, accuracy is $.955$–$.985$ (Table 11). The Code-OPD oracle range ($.110$–$.655$) further shows checkpoint heterogeneity rather than a stable retrieval effect. Thus U and T measure net task effects under a specified return intervention; Help and Harm can cancel, and later actions can diverge after the first return.

Response is distinct from task utility. On CodeDebug, changing a submitted program does not imply a change in task success. The response-change rates in Table 4 are scorer-independent, whereas U measures the signed change in correctness. Conversely, because an unscorable ending receives zero, a return can change U by changing whether the policy reaches Stop. The response-change and nonce controls do not decompose this termination path, so U should be read as the net downstream outcome effect of the specified return, not as Stop-free semantic integration.

Statistical resolution. The primary 3-source $\times$ 2-target panel evaluates one 3B student on 200-item panels with three training seeds. With only three seed effects, the t tests have $df = 2$ and are sensitive to small estimated between-run variance. Crossed seed–item intervals are descriptive and are not inferential intervals for those tests; effects that miss the confirmation criterion remain unresolved.

Generalization comparisons bundle factors. Training sources jointly change task population, tool affordance, feedback process, and data qualification. Cross-model setup panels also co-vary family, tokenizer, template, Base capability, and optimization. Qwen scale comparisons hold the family fixed but still change capacity and route headroom together. These panels show how results vary without isolating domain, architecture, scale, or headroom. The evaluated sources also omit return types such as the language feedback scored by MINT (Wang et al., 2024).

Together, these boundaries define the evidential ladder: read endpoint changes against the observed path, interpret T under its named opportunity and return comparator, and treat cross-setting differences as sensitivity rather than isolated causal factors.