TEXT AS A COMPOSITIONAL INTERFACE: TOWARD DECOMPOSABLE LEARNING WITH RECONSTRUCTED POLICIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Complex systems should not require every new combination of learned skills to be trained end to end. We study whether a language model can instead externalize a learned component as a local textual policy and use that text as a runtime interface for composition. Models are fine-tuned on short chains of opaque triggers, each denoting a two-branch conditional policy; the definitions are absent from the test prompt, and the model reconstructs the complete policy immediately before each use. We then ask which parts of the reconstructed text actually control the next computation. Exchanging only the two operations inside a written policy, with the upstream trace and state fixed, redirects the next operation on the majority branches of both LLaMA-3.2-3B-Instruct and Qwen2.5-3B-Instruct (82–85% pooled), but far less on rare branches. On a separate customer-service workflow, editing a written threshold on the *same record* produces a **+64.2 percentage-point** decision-directed effect beyond an equally sized decision-preserving edit (95% CI 59.4–68.9), and the redirected decision propagates to the final verdict. The interface is not uniformly semantic, however: writing an unfamiliar modular-arithmetic residue into a policy yields no value-specific discrimination (-0.35 **points**), which bounds what operation following can certify and leaves open whether the bottleneck is reading the predicate or computing it. Finally, a token- and position-matched control shows that informative policy content remains useful at a higher numeric range (**+23.3 points, 6/6 seeds**), with a six-point attenuation at the held-out depth. Together, the results distinguish policy reconstruction, interpretation, action selection, and continued invocation as requirements on a textual interface, and motivate reconstructed policies as a route *toward* decomposable learning rather than a demonstration of it.

1 INTRODUCTION

A useful modular learner should not have to relearn every workflow as a monolithic input–output mapping. If it has already learned components A, B, and C, a new workflow $A \to C \to B$ should ideally be assembled from those components rather than trained from scratch. Program synthesis makes this modularity explicit by inducing a symbolic rule system and executing it (Nye et al., 2020). Language models offer a different possibility: the model itself can *externalize* the component it needs into its own context, then condition later generation on that text. A scratchpad could therefore serve not merely as an explanation, but as an **interface between learned components**.

This possibility matters beyond synthetic reasoning. Tool-using and policy-constrained agents must combine multiple learned capabilities while respecting local policies and workflow state (Yao et al., 2024). Human operators may also want to inspect or correct the policy immediately before an action. If locally written rules are reliable interfaces, modular systems could learn skills independently, compose them at inference time, and expose the composition boundary for debugging or intervention. If they are not, a generated rule may be decorative, only partially read, or bound to the positions and shortcuts present during training.

We study this question in a controlled setting where each opaque trigger denotes a learned two-branch conditional policy (Figure 1). The model learns these policies during fine-tuning on chains of depth 1–2, but the standard test prompt provides only trigger identifiers and a record—not the policy

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.

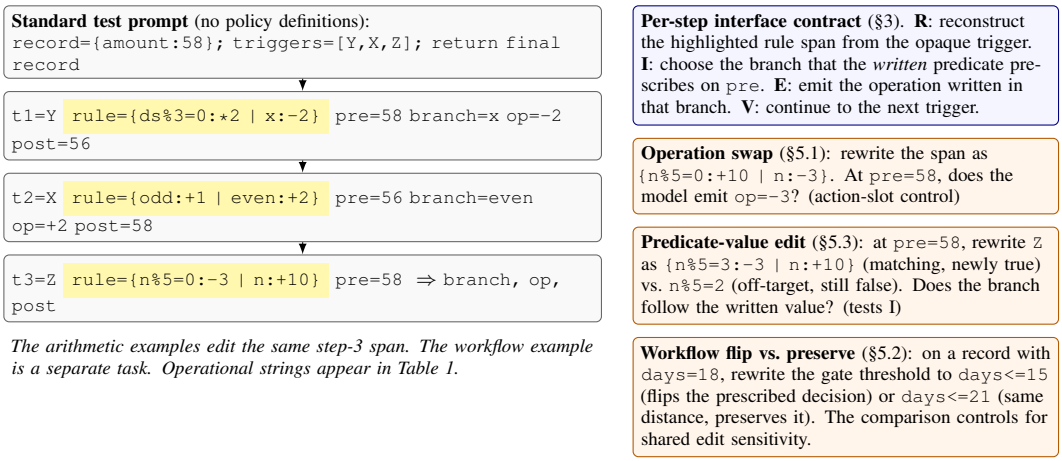

Figure 1: **Text as a compositional interface, and the interventions that test it.** Left: the model reconstructs each opaque trigger's complete policy (highlighted) immediately before using it; policy definitions never appear in the prompt. Right: the four-part contract and the three edit families. Each family probes a different aspect; successful operation selection alone certifies neither reconstruction nor numerical execution. Strings are schematic; Table 1 and Appendix C give the operational definitions.

definitions. At each application, a complete local policy is reconstructed into the assistant trace immediately before the decision, and the new workflows we evaluate are longer chains (depth 3) than any seen in training. We take this interface as the experimental starting point and ask a narrower reliability question:

When does locally reconstructed text actually function as a compositional interface between reusable learned components?

The naive answer is tempting: if the model writes the correct rule, it should follow it. Our experiments show why that is too coarse. A model can follow the *operation* written in a selected branch without demonstrably interpreting a newly written *predicate*. It can react both to an edit that changes the prescribed decision on a record and to one that preserves that decision. And training-time benefit can disagree with inference-time controllability. These are different properties, all hidden by a single end-to-end accuracy number. We therefore decompose the desired interface into four requirements—(1) reconstruct the relevant component, (2) interpret its condition on the current state, (3) execute the selected action, and (4) invoke the next component instead of terminating or reverting to a training shortcut—and test them with interventions that edit one part of the written policy while holding everything else fixed (Figure 1). This turns vague "reasoning failures" into measurable failures at a modular boundary.

Contributions.

1. **Controlled evidence for semantically directed local control.** Same-token operation swaps redirect the next operation on majority branches in two model families, and a stronger same-record workflow test isolates a **64.2-point** flip-versus-preserve effect that controls for shared edit sensitivity and propagates to the final verdict (§5.1–5.2).
2. **Boundaries that change the interpretation.** Rare branches resist operation swaps (2–5% following on LLaMA's two rare branches); unfamiliar modular-arithmetic predicates show no value-specific discrimination; decision-preserving workflow edits still cause 16.1% churn; and a token/position-matched informative-policy advantage remains positive at a higher numeric range but attenuates at the held-out depth (§5.1–5.4).
3. **A component-resolved measurement view.** We separate reconstruction, interpretation, execution, and invocation (§3) and show empirically that operation following, task-level training benefit, and predicate semantics are different estimands (§5.5). A textual interface should be tested both for decision-changing edits it follows and for edits that preserve the decision on the same record.

2 RELATED WORK

Composable modules, programs, and skills. A long line of work seeks reusable computation that can be recombined rather than relearned globally. Neural Module Networks assemble reusable neural modules according to compositional structure (Andreas et al., 2016); Neural Programmer-Interpreters learn callable subprograms and recursively compose them while using an external scratchpad (Reed & de Freitas, 2016); Neural Task Programming recursively decomposes hierarchical tasks into callable subprograms (Xu et al., 2018); and later modular networks learn both modules and routing (Kirsch et al., 2018). Meta sequence-to-sequence learning targets systematic reuse of learned primitives on new compositions (Lake, 2019), while CTL++ explicitly evaluates never-seen compositions of known functions (Csordás et al., 2022). Neural program synthesis induces symbolic rules that can be recombined (Nye et al., 2020), and least-to-most prompting decomposes harder problems into sequentially solved subproblems (Zhou et al., 2023). Recent language-centric work similarly studies composition of learned reasoning skills (Yao et al., 2026; Yin et al., 2026). Our focus is a different interface boundary: the reusable component is stored in model parameters, named by an opaque identifier, and must be reconstructed into text before the same decoder consumes it.

Language as an abstraction and a composition bus. Language has been used as the abstraction between levels of a hierarchy: a high-level policy can issue compositional language instructions to reusable low-level skills (Jiang et al., 2019), and latent-language planners can induce a combinatorial library of named skills and plan by generating instruction sequences (Sharma et al., 2022). Socratic Models compose distinct pretrained models by exchanging language without joint finetuning (Zeng et al., 2023), while SELF-DISCOVER selects atomic reasoning modules and composes them into an explicit reasoning structure for the same LLM to follow (Zhou et al., 2024). These works motivate the possibility that text can be an inter-module protocol. We ask a complementary reliability question: when the interface is generated from an internally learned policy, which parts of the written object actually control the next computation?

Scratchpads, programs, and external execution. Scratchpads and chain-of-thought make intermediate tokens available as serial computational state (Nye et al., 2021; Wei et al., 2022). Their content and representation strongly influence extrapolation: inductive scratchpads can improve out-of-distribution length generalization (Abbe et al., 2024), local statistical structure can make step-by-step reasoning more data-efficient (Prystawski et al., 2023), and Turing Programs obtain broad length generalization by organizing computation into repeated local copy-and-edit operations (Hou et al., 2025). Compositionality gaps can remain even when component questions are individually solvable (Press et al., 2023; Dziri et al., 2023). Program-aided approaches instead make the textual intermediate state executable by an external interpreter: PAL and Program of Thoughts generate programs and delegate computation to code execution (Gao et al., 2023; Chen et al., 2023). Our setting deliberately removes that external correctness mechanism: the same autoregressive model writes the local policy and must then consume it correctly.

Rule following and reasoning-text interventions. Rule-Following Fine-Tuning supplies explicit rules in the input and trains models to recite and follow them, substantially improving arithmetic extrapolation (Hu et al., 2024). Our ordinary test prompt does not supply the rule definition; the model must recover it from an opaque learned identifier. Generated reasoning is also not automatically a faithful description of the computation producing an answer (Turpin et al., 2023), and intervention-based work tests whether changing chain-of-thought changes the answer (Lanham et al., 2023); faithful-CoT methods instead couple reasoning to an executable symbolic representation (Lyu et al., 2023). Rule-following benchmarks further show that even simple natural-language constraints are not followed uniformly (Mu et al., 2023). We treat text as an externally editable interface rather than an explanation of an inaccessible internal mechanism. The central distinction is therefore between sensitivity to *some* edit and correct response to the *meaning* of that edit.

Tools, executable policies, and agents. Toolformer trains a model to decide when and how to call external APIs (Schick et al., 2023); Code as Policies and ProgPrompt expose program-like action interfaces for embodied control and task planning (Liang et al., 2023; Singh et al., 2023). Voyager stores acquired behaviors in an executable skill library and retrieves them for later reuse (Wang et al., 2023). Agent benchmarks such as τ-bench emphasize that useful agents must combine tools, user

interaction, and domain policies reliably (Yao et al., 2024). Our synthetic tasks do not demonstrate deployment reliability, but isolate whether locally reconstructed policy text can play that interface role without an external symbolic executor.

Relation to concurrent work. A concurrent manuscript (Anonymous, 2026) studies what scratch-pad supervision teaches, including complete conditional rules at the point of use, execution-versus-coordination failures, and initial behavioral rule perturbations. We treat that supervision scheme as context rather than as a contribution of this paper. What is new here is the component-resolved intervention suite: cross-model operation swaps, the same-record flip-versus-preserve control, the unfamiliar-predicate boundary, and the depth-by-range matched-content study.

3 A TEXTUAL INTERFACE FOR DECOMPOSABLE LEARNING

Let z_t denote an opaque component identifier at step t, s_t the current state, and r_t the reconstructed textual policy. Four clauses describe the desired *interface contract*:

$$R_t : \text{reconstruct the intended local policy,}$$
$$I_t : \text{select the branch prescribed by the written predicate,}$$
$$E_t : \text{select and execute the operation in that branch,}$$
$$V_t : \text{invoke the next requested step, or terminate at the end.}$$

These are requirements, not identified latent modules: satisfying them along a chain is sufficient for correct execution, but a correct final answer need not certify each clause, since errors can cancel. Operation swaps measure action-slot selection, not full numerical execution; matched predicate edits probe the value-specific response that agreement on an unedited policy cannot identify; and local rule-span completion after a supplied prefix, together with the rate of reaching a canonical editable workflow prefix, are *operational* proxies for reconstruction and continuation rather than semantic estimates of $R_t \cap V_t$.

3.1 WHY AN OPERATION SWAP IS INSUFFICIENT

A successful operation-swap intervention may look like proof that the model reads the written policy. It proves local control by the written *operation*, but not use of the written predicate. Let $r = (p, u_1, u_0)$. Compare

$$A_{\text{interp}}(s; r) = (p(s), u_{p(s)}), \tag{1}$$
$$A_{\text{fam}}(s, t; r) = (p_t(s), u_{p_t(s)}), \tag{2}$$

where p_t is the familiar predicate associated with trigger t. If an intervention changes only (u_1, u_0) while keeping $p = p_t$, the two response functions are observationally identical for any operation pair; an edit is diagnostically visible only where the two branch operations (or, when only the next state is scored, the two updates $u_1(s) \neq u_0(s)$) differ. Thus operation swaps are a positive control for textual action-slot use, not a predicate test. A matched predicate intervention on states where the edited predicate and familiar predicate disagree is required to distinguish the two (Appendix B).

This motivates the sequence below: operation swaps test action-slot control; predicate-value edits test a stronger semantic property; same-record preserving edits estimate generic edit sensitivity; and range controls ask whether informative policy content remains useful as states move.

4 CONTROLLED SETTINGS AND EVALUATION

Table 1 distinguishes the prompt and the training interfaces; Figure 1 shows the interventions. The strings are schematic for readability; the actual experiments use frozen canonical renderers and, where claimed, verify exact token counts and downstream positions.

4.1 ARITHMETIC POLICIES

Each trigger X, Y, Z denotes a deterministic two-branch arithmetic policy over an integer state n: X adds 1 to an odd state and 2 to an even state; Y doubles the state when its digit sum is divisible

Condition	Illustrative prompt / local trace	What the comparison asks
Arithmetic test prompt	`record={amount:56}; triggers=[X,Z,Y]; return final amount`	Policy definitions are *not* supplied. Can opaque learned components be recovered and composed?
Realized trace (C)	`t1=X pre=56 branch=even op=+2 post=58`	A worked transition states only what happened; it does not externalize the counterfactual branch.
Local policy interface (D)	`t1=X rule={odd:+1 | even:+2} pre=56 branch=even op=+2 post=58`	The model writes the complete component immediately before using it. Does informative local policy content help composition?
Matched empty interface (N)	`t1=X rule={q:v | w:zzz} pre=56 branch=even op=+2 post=58`	Same rule-shaped span length/position, no policy semantics: separates useful content from merely having extra local tokens.

Table 1: **What "text as an interface" means operationally.** C illustrates a realized-only reference format; the new content comparisons use D versus N. Figure 1 shows the distinct edit operations, and Appendix C specifies their logic. Strings and punctuation here are schematic, not token-matching evidence.

by 3 and otherwise subtracts 2; Z subtracts 3 if $n \bmod 5 = 0$ and otherwise adds 10. The training sequences are the three single triggers and six ordered pairs of distinct triggers; testing adds the six distinct-trigger triples. Models are trained on depths 1–2 over starting states 1–50; the standard test prompt contains the trigger sequence and record but not the rule definitions, and depth 3 is held out. The main D/N training sets contain 900 rows per seed (450 direct-answer and 450 trace rows). We use 4-bit LLaMA-3.2-3B-Instruct and Qwen2.5-3B-Instruct with LoRA $r = 16$, $\alpha = 16$ on the seven attention/MLP projection matrices (Hu et al., 2021): LLaMA uses learning rate 5×10^{-5} and Qwen 10^{-4}, batch 4 with gradient accumulation 4, 20 warmup steps, at most 40 epochs, 10% validation, AdamW-8bit, and early stopping with patience 3. Evaluation here is greedy. Exact adapter reuse and seed mapping for each intervention are listed in Appendix A.

The operation-edit tests hold the prompt, preceding two steps, third-step incoming state, predicate, token count, and downstream positions fixed, while exchanging the two operations. Six independently trained adapters are tested per family, 288 items per seed. A response follows the edit only if it emits the gold branch and the operation written in that branch after the swap.

The predicate test keeps the operations fixed and changes numeric predicate values. A main edit changes the truth value on every selected item. On originally false items, a second off-target value remains false. Value-specific interpretation predicts activation of the positive branch for the matching value but not the off-target value. This is a stronger test than merely moving away from the original action.

4.2 Same-record workflow edits

A separate synthetic returns workflow applies four known gates, G1–G4: return window, final sale, opened/defective status, and price/tier review. A gate can terminate the workflow early. Its training set contains all four single gates and twelve ordered pairs, but no triples; thus *position three*, not a new gate identity, is held out. We edit G1 or G4 when it occurs in that third position (Appendix A). We reuse three complete-policy LLaMA adapters. The successor experiment constructs 288 *common-support* records per seed, i.e., records on which both of the following edits can be made at the *same token indices* and with the same absolute margin from the record's own decision boundary:

- a **flip edit** moves the written threshold across the boundary and changes the policy-prescribed decision;
- a **preserve edit** moves it equally far while remaining on the same side, so the prescribed decision is unchanged.

The primary estimand is the probability of the flip-prescribed decision under flip minus preserve. This is a same-record comparison: generic sensitivity to editing appears in both arms. We report a gold-prefix regime and an unconditional regime starting from the model's own saved free run. The latter assigns zero paired edit effect to runs that do not meet the predeclared canonical-prefix criterion, rather than conditioning those runs away.

Model	Canonical policy	Swapped policy	Branch-balanced swap
LLaMA-3.2-3B-Instruct	87.2%	**82.0%**	66.6%*
Qwen2.5-3B-Instruct	92.2%	**85.2%**	75.8%

Table 2: Operation interventions, six seeds and 288 items per seed per family. Canonical and swapped columns score the *emitted* correct branch together with the operation prescribed by the respective written span. Seed-level 95% intervals for pooled swap following are [80.8,83.2]% (LLaMA) and [82.1,88.4]% (Qwen). The balanced average weights six trigger–branch groups equally; *LLaMA's swap-specific macro-average is a *post-hoc descriptive recount*, not its prespecified correct-minus-neutral macro statistic. See Appendix H.

4.3 Matched policy-content and numeric-range control

To ask whether the usefulness of informative policy text survives a state-range shift, we reuse six pairs of LLaMA adapters: D reconstructs an informative policy span, while N emits a same-length, same-position, rule-shaped but semantically empty span. We evaluate a fully crossed low/high starting-value design at familiar depths 1–2 and held-out depth 3. Low starts are 51–70 and high starts 351–370. The primary effect is the D–N difference in final accuracy at depth 3 in each range, and the difference of those differences.

Conventions. We report percentage-point differences as "pp". "Native" final-record accuracy compares the format-specific parsed output record with the oracle record after canonicalization; it does not require every intermediate step to be correct. "Mixed-format" averages the equally sized direct-answer and trace subsets. The prefix regime—unassisted prompt-start generation, supplied gold history, or saved model-generated history—is stated separately. All new evaluation pools, scorers, token-position checks, adapters, and analysis procedures were frozen before the corresponding model outputs. Intervals are across training seeds plus record-cluster bootstraps where specified. We preserve unsuccessful protocol gates rather than redefining them after outcomes.

5 Results

5.1 Written operations act as a local control interface

Table 2 gives the cleanest positive control. With upstream computation and incoming state fixed, the operation tokens written inside the local policy strongly redirect the next operation in both model families.

Following is heterogeneous in both families: the pooled rates are carried by the majority branches. On the two rare positive branches (Y:s and Z:d), LLaMA follows the swapped operation in only 4.5% and 2.0% of cases and Qwen in 48.4% and 24.6%, so the equal-branch averages fall to 66.6% (LLaMA, descriptive recount) and 75.8% (Qwen, prespecified). A high pooled rate is therefore not branch-wise controllability: where a branch is rare in training, the written operation is rarely followed. Reconstruction itself is also imperfect. Releasing the step-3 rule opening for free generation, with gold earlier steps, reproduces the exact canonical span in 82.6% (LLaMA) and 61.3% (Qwen) of cases (Appendix H).

5.2 Same-record workflow edits isolate a large decision-directed effect

The workflow experiment strengthens the intervention in two ways: it edits predicate *values*, not operation labels, and compares decision-flipping and decision-preserving edits on the exact same record and token positions. The result is large (Table 3).

Decision-preserving edits are not inert. They change 16.1% of decisions and reduce gold accuracy by 12.2 points on common support. This is why flip–preserve is more informative than the raw flip effect (76.4 points). Subtracting the preserving-edit response removes about 12 points of shared edit sensitivity without identifying a unique mechanism. Even after this control, the directed effect remains 64.2 points, and when a flip is followed, the final workflow verdict follows it in 99.9% of cases, so local interface control propagates downstream in this substrate.

Regime	Flip–preserve effect	95% bootstrap CI	Seeds
Controlled gold prefix	$+64.2\,\mathrm{pp}$	[59.4, 68.9]	3/3 positive
Model's saved free run, unconditional	$+46.5\,\mathrm{pp}$	[41.9, 51.2]	3/3 positive
Saved run, valid editable prefix only	$+65.7\,\mathrm{pp}$	—	71.5% valid

Table 3: Same-record customer-service policy-value edits. The unconditional free-run estimand does not discard failures to produce an editable canonical third-gate policy. On the valid subset, the saved prefixes happen to be token-for-token identical to the gold prefix; the result therefore does not establish robustness to erroneous natural prefixes.

Predicate endpoint	Mean	Seed-level 95% CI	Interpretation
Edited-decision compliance shift	$+11.23\,\mathrm{pp}$	[8.46, 13.99]	positive but below target
Matching–off-target discrimination	$-0.35\,\mathrm{pp}$	$[-1.24, 0.55]$	no value-specific support

Table 4: LLaMA predicate-value intervention, six seeds. Among 864 originally false cases, the positive branch appears once with the matching novel residue and four times with an off-target residue.

The saved-run result measures *coverage of an editable format*, not robustness to erroneous histories. Let c_s be seed s's canonical-prefix coverage and $\Delta_s^{\mathrm{valid}}$ the paired effect among those prefixes. The reported unconditional effect is exactly $\frac{1}{3}\sum_s c_s \Delta_s^{\mathrm{valid}} = 46.5\,\mathrm{pp}$, not the product of the two separately averaged quantities. Coverage averages 71.5%, and every qualifying saved prefix is token-identical to its gold counterpart. The remaining 28.5% fail the canonical-prefix criterion: they include noncanonical or wrong spans, unparsed lines, wrong gate identifiers, and termination. They are not all certified semantic reconstruction or invocation failures; their edit response was not tested.

5.3 Operation following does not certify novel predicate interpretation

The arithmetic predicate intervention provides the critical negative result. The main edit produces a positive average compliance shift of $+11.23\,\mathrm{pp}$ (6/6 seeds), but it fails the frozen $20\,\mathrm{pp}$ practical target. More importantly, the matching-versus-off-target control shows why that aggregate shift cannot be called predicate interpretation.

The model mainly changes behavior when an originally true familiar predicate is edited; it almost never activates a newly true branch under an unfamiliar residue. A system that falls back to the otherwise branch for any unfamiliar residue can therefore obtain a positive aggregate compliance shift without evaluating the written value. The experiment therefore establishes a *boundary*: operation following does not demonstrate novel-predicate execution. Two features of the contrast with §5.2 deserve emphasis. First, the threshold edits that the workflow model follows and the residue edits to which the arithmetic model lacks value-specific response differ in predicate *type*, not only in substrate: written threshold values direct decisions in the workflow model, whereas the arithmetic models show no newly-true residue activation on the tested pool. Predicate family, training, state distribution, and model instance differ, so this contrast does not isolate the cause. Second, the arithmetic result therefore does *not* distinguish failure to consume the written predicate from failure to compute an unfamiliar modular-arithmetic predicate; a familiar-predicate transplant (writing one trigger's known predicate under another trigger) or a direct-rule capability probe (supplying the novel predicate in a simpler prompt) is needed for that separation, and both are inference-only follow-ups on the existing adapters.

5.4 Policy content remains useful under a numeric-range shift, but less so

The range experiment gives a complementary boundary. D and N differ only in whether the matched local span contains the true policy content. At held-out depth 3, D–N remains positive in every seed in both ranges (Table 5).

Three features of Table 5 matter. First, the depth-3 advantage is not an artifact of one start-value range: it is positive in every seed in both ranges. Second, the attenuation is depth-specific. At familiar depths the informative span helps *more* at the high range (+11.7/+12.8 vs. +3.9/+8.9 pp), where the

Mixed-format native accuracy	d1	d2	d3
D, low (51–70)	84.4	72.5	38.8
N, low	80.6	63.6	9.5
D–N, low	+3.9	+8.9	**+29.2**
D, high (351–370)	76.1	59.4	31.0
N, high	64.4	46.5	7.7
D–N, high	+11.7	+12.8	**+23.3**

Table 5: Absolute accuracy (%) and D–N gaps (pp), six matched adapter pairs. At d3, seed-level 95% CIs are [22.19,36.28] pp (low) and [17.88,28.64] pp (high), both positive in 6/6 seeds. High minus low at d3 is $-5.97\,\mathrm{pp}$ $[-9.25, -2.69]$, negative in 6/6 seeds.

underlying arithmetic is harder; only at the held-out depth does the gap shrink, and it shrinks because D falls (38.8 to 31.0) while N is already near floor (9.5 to 7.7). The depth-by-range interaction on the D–N gap is accordingly $-11.20\,\mathrm{pp}$ (seed-level 95% CI $[-17.80, -4.61]$). Third, absolute depth-3 accuracy is low even with the informative span (39% and 31%); the claim is about the contribution of policy content to held-out composition, not that composition is solved. Because every range contrast is a paired, within-adapter comparison, differences in level between adapter populations (§5.5) do not confound it. Restricting the low-range side to locally novel state–trigger pairs preserves the qualitative result.

5.5 Training utility and edit controllability are distinct

A final comparison prevents another conflation. In a prospective six-pair LLaMA confirmation, informative policy spans beat equal-length neutral spans by $+20.76\,\mathrm{pp}$ at depth 3, 6/6 positive. This is not the same estimate as the $+29.24\,\mathrm{pp}$ low-range result above: the confirmation used freshly trained seed pairs 101–606 on a separate band (278–312), whereas the range study reuses an earlier six-pair adapter set (seeds 0/1/7/13/21/42) on twin starts 51–70 and 351–370. On Qwen the same prospective confirmation yields $+11.25\,\mathrm{pp}$ on average but only four positive seeds, one negative seed, and one tie; its seed-level interval includes zero, so the frozen 6/6 consistency criterion fails. Yet the Qwen D models follow operation swaps at 85.2%.

Thus a model can be highly controllable by an edited interface even when the training intervention that produced that interface has not shown a consistent task-level advantage. *Training utility, textual controllability, and semantic interpretation are three different estimands.*

6 What these results mean for decomposable learning

The experiments support a constructive but bounded view. Text can function as a local interface between learned components. It is not merely decorative: controlled policy edits redirect subsequent computation, the effect replicates across two model families for operation assignments, and same-record predicate-value edits control a separate workflow relative to decision-preserving edits on those records. A matched semantic-content advantage also persists under a sizeable state-range shift.

But "write the rule and compose" is not yet a complete recipe. The arithmetic predicate failure shows that the model can expose a rule-shaped object while consuming only part of its semantics—or while lacking the capability to evaluate the part it exposes. The workflow preserving-edit churn shows that irrelevant changes can perturb behavior. Range attenuation shows that interface reliability itself can be distribution-sensitive. And 28.5% of free workflow runs fail the operational canonical-prefix criterion, exposing a coverage limitation distinct from the response to an edit once applied. These are separable failure modes, not one monolithic reasoning score.

This leads to a practical design target for modular training. Rather than train every global composition, one can aim to train components independently and enforce a local contract: *reconstruct the right component, interpret the entire written policy on the current state, execute it, and continue*. Such an interface is attractive because it is visible and editable. A human or another model could inspect the reconstructed component before an irreversible action, and failures can be localized to retrieval, interpretation, execution, or invocation.

Why this is not simply rule following. Classic rule-following tests often supply the rule at inference and ask whether the model applies it (Hu et al., 2024). Here the rule is a learned component that must be externalized on demand from an opaque identifier. The compositional question is therefore two-level: can the model recover the component, and can later generation reliably use the recovered object? This second level is what makes a textual policy an interface rather than merely a prompt.

Why the negative results matter. If operation swaps alone had been taken as proof of general rule use, the 82–85% results would suggest a nearly solved interface. The predicate-specificity test overturns that interpretation. Likewise, a 76-point raw workflow flip effect looks cleaner than it is until a same-record preserving edit reveals a 12-point loss in canonical decision accuracy. The contribution is therefore a decomposition of which edits produce meaning-specific control and which reveal shortcutting, not a demonstration that editing text can force arbitrary behavior.

7 Limitations and boundaries

The evidence is deliberately controlled and therefore narrow. The arithmetic setting uses two 3B model families, three triggers with simple deterministic policies, and held-out chains of depth 3, so the space of novel compositions is small and absolute depth-3 accuracy remains low; the workflow is synthetic and tested on one family with three seeds. No model is trained exclusively on isolated components: training already includes depth-2 chains, so the new workflows are longer chains rather than compositions of separately trained modules. The numeric-range study changes state values but is not a pure length control because composition depth and trace length co-vary. The valid free-run workflow prefixes are identical to their gold prefixes, so the unconditional result tests whether the model naturally reaches an editable interface, not robustness to erroneous upstream reasoning. The arithmetic predicate result bounds what operation following certifies but does not separate interface failure from arithmetic failure (§5.3). We make behavioral claims about externally edited text, not internal circuits or natural mediation.

A preregistered familiarity×position×range factorial was stopped before any decode because its frozen matched-item quotas were unfillable; no model output was produced (Appendix F). The point-of-use complete-policy supervision underlying our interface is analyzed in concurrent work (Anonymous, 2026); the contribution here is the component-resolved controls and boundaries, not that supervision recipe.

8 Conclusion

Locally reconstructed policy text can redirect a model's later computation: exchanging written operations redirects majority-branch decisions in two model families, and a same-record threshold edit produces a 64-point decision-directed effect that survives a non-inert preserving-edit control and propagates to the final verdict. The same tests bound the claim. Rare branches resist operation swaps, unfamiliar residues receive no value-specific response, only about 72% of free runs reach an editable policy at the held-out position, and the matched-content advantage attenuates at the held-out depth. Operation following, value-specific predicate response, editable-prefix coverage, and training benefit are therefore distinct measurements on which a textual interface should be certified; that decomposition, not a claim that independently composable learning has been achieved, is our route toward it.

Reproducibility statement

All evaluation pools, scorers, token-position checks, and analysis procedures used for the new intervention studies were frozen before decoding. Appendix A lists training recipes and adapter provenance; the remaining appendices give intervention logic, seed-level results, and stopped-protocol details. The supplementary archive includes the aggregated per-cell counts (`data/`) and a script (`scripts/verify_statistics.py`) that recomputes the reported rates and seed-level intervals from them; it is an aggregate audit rather than a full reproduction package, since the training pipeline, adapters, and per-record outputs needed to regenerate the bootstrap intervals are not included.

THE USE OF LARGE LANGUAGE MODELS (LLMS)

Generative AI tools were used to propose and refine hypotheses; provide feedback on experimental methodology; assist with implementation, analysis, and experiment-orchestration code; and interpret quantitative and qualitative model outputs. They were also used in the construction of synthetic research data: AI tools helped design, implement, and audit deterministic synthetic-data generators and to clean, match, filter, and reformat training and evaluation pools. The reported synthetic examples themselves were produced by the resulting deterministic programs rather than sampled as free-form LLM generations. Generative AI additionally helped develop the conceptual framework, formulate mathematical claims, and draft and revise the elementary arguments and proofs used in the manuscript; translate research ideas and draft text between the authors' first language and English; search, summarize, and verify related literature and references; and draft and edit manuscript text, tables, figures, structure, titles, and keywords. AI-assisted numerical summaries were recomputed against saved experimental outputs, and primary or official sources were consulted for citation checks. Protocol changes were frozen before outcome inspection where stated; later descriptive recounts are labelled as such. The authors take responsibility for the final text, claims, code, data-processing choices, and artifacts included in the submission.

REFERENCES

Emmanuel Abbe, Samy Bengio, Aryo Lotfi, Colin Sandon, and Omid Saremi. How far can transformers reason? the globality barrier and inductive scratchpad. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 39–48, 2016. URL https://openaccess.thecvf.com/content_cvpr_2016/html/Andreas_Neural_Module_Networks_CVPR_2016_paper.html.

Anonymous. Teaching a step is not teaching a procedure: Separating execution from coordination in scratchpad supervision, 2026. Concurrent manuscript under review; not an accepted publication.

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*, 2023. URL https://openreview.net/forum?id=YfZ4ZPt8zd.

Róbert Csordás, Kazuki Irie, and Juergen Schmidhuber. CTL++: Evaluating generalization on never-seen compositional patterns of known functions, and compatibility of neural representations. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 9758–9767, 2022. doi: 10.18653/v1/2022.emnlp-main.662. URL https://aclanthology.org/2022.emnlp-main.662/.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D. Hwang, Soumya Sanyal, Sean Welleck, Xiang Ren, Allyson Ettinger, Zaid Harchaoui, and Yejin Choi. Faith and fate: Limits of transformers on compositionality, 2023. URL https://arxiv.org/abs/2305.18654. arXiv:2305.18654.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. PAL: Program-aided language models. In *Proceedings of the 40th International Conference on Machine Learning*, 2023. URL https://proceedings.mlr.press/v202/gao23f.html.

Kaiying Hou, David Brandfonbrener, Sham M. Kakade, Samy Jelassi, and Eran Malach. Universal length generalization with turing programs. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 23873–23893. PMLR, 2025.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685. arXiv:2106.09685.

Yi Hu, Xiaojuan Tang, Haotong Yang, and Muhan Zhang. Case-based or rule-based: How do transformers do the math? In *Proceedings of the 41st International Conference on Machine Learning*, pp. 19438–19474, 2024. URL https://proceedings.mlr.press/v235/hu24n.html.

Yiding Jiang, Shixiang Gu, Kevin Murphy, and Chelsea Finn. Language as an abstraction for hierarchical deep reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://papers.nips.cc/paper_files/paper/2019/hash/0af787945872196b42c9f73ead2565c8-Abstract.html.

Louis Kirsch, Julius Kunze, and David Barber. Modular networks: Learning to decompose neural computation. In *Advances in Neural Information Processing Systems*, volume 31, pp. 2414–2423, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/310ce61c90f3a46e340ee8257bc70e93-Abstract.html.

Brenden M. Lake. Compositional generalization through meta sequence-to-sequence learning. In *Advances in Neural Information Processing Systems*, volume 32, pp. 9791–9801, 2019. URL https://papers.nips.cc/paper_files/paper/2019/hash/f4d0e2e7fc057a58f7ca4a391f01940a-Abstract.html.

Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, Kamilė Lukošiūtė, Karina Nguyen, Newton Cheng, Nicholas Joseph, Nicholas Schiefer, Oliver Rausch, Robin Larson, Sam McCandlish, Sandipan Kundu, Saurav Kadavath, Shannon Yang, Thomas Henighan, Timothy Maxwell, Timothy Telleen-Lawton, Tristan Hume, Zac Hatfield-Dodds, Jared Kaplan, Jan Brauner, Samuel R. Bowman, and Ethan Perez. Measuring faithfulness in chain-of-thought reasoning, 2023. URL https://arxiv.org/abs/2307.13702. arXiv:2307.13702.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *2023 IEEE International Conference on Robotics and Automation*, pp. 9493–9500, 2023. doi: 10.1109/ICRA48891.2023.10160591. URL https://dblp.org/rec/conf/icra/LiangHXXHIFZ23.

Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. Faithful chain-of-thought reasoning. In *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 305–329, 2023. doi: 10.18653/v1/2023.ijcnlp-main.20. URL https://aclanthology.org/2023.ijcnlp-main.20/.

Norman Mu, Sarah Chen, Zifan Wang, Sizhe Chen, David Karamardian, Lulwa Aljeraisy, Basel Alomair, Dan Hendrycks, and David Wagner. Can llms follow simple rules?, 2023. URL https://arxiv.org/abs/2311.04235. arXiv preprint; RuLES benchmark.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. Show your work: Scratchpads for intermediate computation with language models, 2021. URL https://arxiv.org/abs/2112.00114. arXiv:2112.00114.

Maxwell I. Nye, Armando Solar-Lezama, Joshua B. Tenenbaum, and Brenden M. Lake. Learning compositional rules via neural program synthesis. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 5687–5711, 2023. doi: 10.18653/v1/2023.findings-emnlp.378. URL https://aclanthology.org/2023.findings-emnlp.378/.

Ben Prystawski, Michael Y. Li, and Noah D. Goodman. Why think step by step? reasoning emerges from the locality of experience. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/e0af79ad53a336b4c4b4f7e2a68eb609-Abstract-Conference.html.

Scott E. Reed and Nando de Freitas. Neural programmer-interpreters. In *International Conference on Learning Representations*, 2016. URL https://arxiv.org/abs/1511.06279.

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*, volume 36, 2023. doi: 10.52202/075280-2997. URL https://proceedings.neurips.cc/paper/2023/hash/d842425e4bf79ba039352da0f658a906-Abstract-Conference.html.

Pratyusha Sharma, Antonio Torralba, and Jacob Andreas. Skill induction and planning with latent language. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1713–1726, 2022. doi: 10.18653/v1/2022.acl-long.120. URL https://aclanthology.org/2022.acl-long.120/.

Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. Progprompt: Generating situated robot task plans using large language models. In *2023 IEEE International Conference on Robotics and Automation*, pp. 11523–11530, 2023. doi: 10.1109/ICRA48891.2023.10161317. URL https://ieeexplore.ieee.org/document/10161317/.

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*, 2023. URL https://arxiv.org/abs/2305.04388.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models, 2023. URL https://arxiv.org/abs/2305.16291. arXiv:2305.16291.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2022. URL https://arxiv.org/abs/2201.11903. arXiv:2201.11903.

Danfei Xu, Suraj Nair, Yuke Zhu, Julian Gao, Animesh Garg, Li Fei-Fei, and Silvio Savarese. Neural task programming: Learning to generalize across hierarchical tasks. In *2018 IEEE International Conference on Robotics and Automation*, 2018. doi: 10.1109/ICRA.2018.8460689. URL https://arxiv.org/abs/1710.01813.

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

Xinhao Yao, Ruifeng Ren, Yun Liao, Lizhong Ding, and Yong Liu. Compositional generalization from learned skills via CoT training: A theoretical and structural analysis for reasoning. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/9990a18179e862042c18dfc856b0e53f-Abstract-Conference.html.

Fangcong Yin, Zeyu Leo Liu, Liu Leqi, Xi Ye, and Greg Durrett. Learning composable chains-of-thought. In *Findings of the Association for Computational Linguistics: EMNLP 2026*, 2026. URL https://arxiv.org/abs/2505.22635v2. Author-posted version 2; arXiv:2505.22635.

Andy Zeng, Maria Attarian, Brian Ichter, Krzysztof Marcin Choromanski, Adrian Wong, Stefan Welker, Federico Tombari, Aveek Purohit, Michael S. Ryoo, Vikas Sindhwani, Johnny Lee, Vincent Vanhoucke, and Pete Florence. Socratic models: Composing zero-shot multimodal reasoning with language. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2204.00598.

Denny Zhou, Nathanael Sch
"arli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V. Le, and Ed H. Chi. Least-to-most prompting enables complex reasoning in large language models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WZH7099tgfM.

Pei Zhou, Jay Pujara, Xiang Ren, Xinyun Chen, Heng-Tze Cheng, Quoc V. Le, Ed H. Chi, Denny Zhou, Swaroop Mishra, and Huaixiu Steven Zheng. SELF-DISCOVER: Large language models self-compose reasoning structures. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-4004. URL https://papers.nips.cc/paper_files/paper/2024/hash/e41efb03e20ca3c231940a3c6917ef6f-Abstract-Conference.html.

A Training recipes and adapter provenance

The main arithmetic D/N adapters use 900 training rows per seed from starts 1–50 and depths 1–2: 450 direct-answer and 450 trace rows. Both model families use 4-bit base weights and LoRA $r = 16$, $\alpha = 16$ on $q, k, v, o, \text{gate}, \text{up}, \text{down}$ projections. LLaMA uses learning rate 5×10^{-5}; Qwen uses 10^{-4}. Both use batch 4 with gradient accumulation 4, 20 warmup steps, 10% validation, AdamW-8bit, a 40-epoch cap, and early stopping with patience 3. The intervention studies are inference-only and assert zero trainable parameters. Training includes all three single-trigger sequences and all six ordered distinct-trigger pairs; the held-out distinct triples are the six permutations of X/Y/Z, not 27 unrestricted sequences. The workflow adapters instead use 960 rows (30 records $\times$ 16 workflows $\times$ two formats), all four singleton gates and twelve distinct ordered pairs, the same LLaMA learning rate and LoRA rank, and no depth-three workflows. Thus the third *position* is untrained even though every gate identity and adjacent ordered pair is familiar.

Study	Adapter provenance	Evaluation use
LLaMA matched-content / predicate / op-swap	The same six NCONF D adapters (101,202,303,404,505,606) reused for operation and predicate interventions	Greedy, frozen pools; operation and predicate edits at step 3
Qwen matched-content / op-swap	Six prospective D/N pairs (111,222,333,444,555,666), with recipe chosen using depths 1–2 only	Greedy; Qwen D/N confirmation plus op-swap
Workflow same-record control	Three persistent complete-policy workflow adapters (0,1,42)	Same-record threshold flip/preserve at held-out position 3 (known G1/G4)
Range control	Earlier six matched LLaMA D/N pairs (0,1,7,13,21,42)	Twin low/high starting-value evaluation across depths 1–3

Table 6: Adapter provenance for the inference-only interventions. Distinct studies that estimate the same conceptual D–N contrast can therefore use different seeds and evaluation populations.

B Formal note on operation-swap identification

For a fixed trigger t with familiar predicate p_t, an interpreter and a familiar-predicate selector are observationally identical under operation-only interventions:

$$A_{\text{interp}}(s; (p_t, u_1, u_0)) = (p_t(s), u_{p_t(s)}) = A_{\text{fam}}(s, t; (p_t, u_1, u_0)).$$

Exchanging (u_1, u_0) preserves the equality. A predicate edit p' can distinguish the two only on states where $p'(s) \neq p_t(s)$ and where the two branch actions differ. This is an identification observation, not a learning theorem.

C Exact intervention logic

C.1 Arithmetic operation and predicate edits

At the held-out third transition, upstream steps and incoming state are fixed. The operation-swap family exchanges the two operation tokens but leaves the predicate unchanged. The predicate family

holds both operations fixed while modifying numeric predicate content. For a Z-style rule,

$$\{n \bmod 5 = 0 : -3 \mid \text{else} : +10\},$$

a state with residue 1 can be paired with a matching edit $= 1$ and an off-target edit such as $= 3$. The two edits introduce similarly unfamiliar values but prescribe different branches on the same state.

The main predicate compliance shift is positive ($+11.23$ pp) because corrupting an originally true familiar predicate often suppresses the positive branch. The stronger newly-true test almost never activates the positive branch: 1 of 864 matching cases versus 4 of 864 off-target cases. We therefore interpret the result as a failure to demonstrate value-specific interpretation, not as zero sensitivity to the text.

C.2 Same-record workflow control

For each eligible workflow record, choose two threshold values at the same distance from that record's decision boundary: one changes the prescribed decision, and one preserves it. This match is relative to the record's boundary, not necessarily an equal displacement from the original threshold. Both change the same token positions and preserve token count. The primary estimand

$$\Delta_{EP} = \mathbb{P}(\text{flip decision} \mid \text{flip edit}) - \mathbb{P}(\text{flip decision} \mid \text{preserve edit})$$

compares decision-directed response against a same-record preserving-edit control; it does not establish a unique mechanism for either response.

On the controlled prefix, $\Delta_{EP} = 64.2$ pp with a record-cluster bootstrap 95% interval [59.4,68.9]. The raw flip-versus-canonical effect is 76.4 points, while the preserving edit itself moves gold accuracy by 12.2 points and produces 16.1% decision churn. This is why we headline Δ_{EP} rather than the raw flip effect.

The free-run regime starts from each model's saved generation. Only 71.5% of runs contain a canonical editable third-gate policy. Invalid runs count as zero in the unconditional primary, producing $+46.5$ pp [41.9,51.2]. On the valid subset the effect is $+65.7$ pp, but every valid prefix is token-for-token identical to the gold prefix. The result therefore measures natural arrival at the interface, not tolerance to upstream textual errors.

D Depth-by-range control

Six matched D/N adapter pairs are evaluated at low starts 51–70 and high starts 351–370. D writes the informative policy; N writes an equal-length, same-position, rule-shaped but semantically empty span. The primary mixed-format depth-three gaps are:

	Seed 0	1	7	13	21	42	Mean
Low D–N (pp)	25.42	30.42	29.17	18.75	38.33	33.33	29.24
High D–N (pp)	17.08	22.50	25.83	17.50	29.17	27.50	23.26
High–low (pp)	-8.33	-7.92	-3.33	-1.25	-9.17	-5.83	-5.97

Table 7: Seed-level primary mixed-format range results. All six policy-content advantages remain positive in both ranges; all six attenuate at high range.

The seed-level $t(5)$ intervals are [22.19,36.28] pp (low), [17.88,28.64] pp (high), and $[-9.25, -2.69]$ pp for high-minus-low. Record-cluster bootstrap intervals are [26.46,31.67], [19.93,26.32], and $[-8.26, -3.68]$ pp. The depth-by-range interaction on the D–N gap is -11.20 pp with $t(5)$ interval $[-17.80, -4.61]$.

Trace-channel effects are larger: $+54.17$ pp at low range and $+49.31$ pp at high range. This does not replace the mixed-format primary. In the direct channel the D–N gaps are +4.31 pp (low) and -2.78 pp (high; seed interval includes zero). This format dependence is evidence about access through the trained response channel, not proof that an internal algorithm is absent.

The design is not a pure numerical extrapolation theorem. High values are larger than the matched low starts, while composition depth still changes trace length. The correct conclusion is that informative policy content remains useful under this controlled range shift, with measurable attenuation.

E TRAINING UTILITY VERSUS EDIT CONTROLLABILITY

Six-seed matched-span training comparisons provide a useful non-equivalence result. LLaMA D–N at held-out depth is $+20.76\,\mathrm{pp}$ with all six differences positive. The analogous Qwen confirmation is $+11.25\,\mathrm{pp}$ with four positive differences, one negative, and one tie; its seed-level 95% interval is [−1.81,24.31] pp and the frozen all-positive criterion fails. Nevertheless, those Qwen D models follow operation swaps at 85.2%. A model can therefore expose a highly editable local operation interface without a confirmed task-level benefit from the training contrast that created it.

F AN ATTEMPTED FACTORIAL TEST THAT COULD NOT BE RUN

A preregistered follow-up attempted to cross trained-versus-held-out position, familiar-versus-novel predicate form, and numeric range. The protocol required 288 matched items per cell, a fresh starting-state pool, token-count matching, and the inherited predicate quotas. An input-only fill audit showed that none of the eight cells could be filled under these frozen constraints: the in-range level conflicted with the fresh-pool exclusion, and the above-range one-token level provided only 38 of the required 72 Z-positive examples at positions 2 and 3. The experiment stopped before decoding and produced zero model outputs. We do not relax the pool or quota after seeing this failure. A future factorial study requires a prospectively redesigned, feasible support distribution. This is a design infeasibility, not a negative model-behavior result.

G PROTOCOL AND STATISTICAL NOTES

All main edit experiments use greedy decoding and paired conditions within a single loaded adapter. Adapter hashes are checked before and after inference, trainable parameter counts are zero, and within-load determinism checks pass for the reported cells. Reloaded adapters need not reproduce old greedy strings byte-for-byte; cross-load identity is not used as a validity gate.

For six-seed contrasts, the paper reports means and $t(5)$ intervals over seed-level effects. The workflow same-record study has three independently trained seeds; its primary interval is a record-cluster bootstrap over the common-support records and is accompanied by per-seed effects. Small numbers of training seeds limit population-level claims; item-level counts are not treated as independent training replications.

H BRANCH RESPONSE AND LOCAL RECONSTRUCTION PROFILES

The two families have different evaluation pools and branch prevalences; their rows are not a matched between-family effect. Each within-family rate is paired under the same loaded adapter. Table 8 reports all six groups rather than treating a high micro-average as uniform following. The LLaMA macro swap rate below is a post-hoc descriptive recount of saved generations. Its preregistered balanced statistic concerned *correct versus neutral* branch/action accuracy, not swapped-policy following; the endpoints must not be interchanged.

Reconstructing a local span is not completing a workflow. A separate `free`-span condition in the operation studies releases generation after the third trigger's `rule` opening, with earlier steps still supplied correctly. LLaMA reconstructs the exact canonical span in 1428/1728 cases (82.6%); Qwen in 1060/1728 (61.3%). Their branch/action accuracies under this release are 73.2% and 68.6%. Exact-span failure need not imply semantic inequivalence. These predeclared endpoints provide a local reconstruction diagnostic; they do not measure unassisted traversal from the initial prompt or independently identify invocation competence. The compact data archive retains every seed's counts.

Canonical workflow coverage. The saved workflow prefixes are valid in 237/288, 195/288, and 186/288 runs. The corresponding paired effect numerators are 134, 137, and 131, so

$$\frac{1}{3}\left(\frac{237}{288}\frac{134}{237}+\frac{195}{288}\frac{137}{195}+\frac{186}{288}\frac{131}{186}\right)=46.53\%.$$

	LLaMA		Qwen	
Trigger:branch	Items/seed	Swap following (%)	Items/seed	Swap following (%)
X:even	61	100.0	63	97.1
X:odd	35	97.1	33	100.0
Y:x	63	97.4	65	88.7
Y:s	33	4.5	31	48.4
Z:n	79	98.7	77	96.1
Z:d	17	2.0	19	24.6
Equal-branch average		66.6		75.8
Item-weighted average		82.0		85.2

Table 8: Saved operation-swap responses, six seeds per family. Following requires both the original predicate's correct branch and that branch's *edited* operation. The macro swap intervals across seeds are [65.42,67.82]% (LLaMA, descriptive) and [70.68,80.94]% (Qwen, prespecified).

This is the mean of within-seed products, not the product of the mean coverage and mean conditional effect. Of the 246 runs outside the canonical-prefix criterion, 139 have a noncanonical third span, 49 an unparsed third line, 46 a wrong earlier gate identifier, and 12 another failure such as earlier denial or too few lines. These operational categories do not by themselves establish semantic R_t or V_t failures.

I CONCRETE SAVED-RESPONSE CASES

The examples below are descriptive cases selected after the results, not frequency estimates or new experiments. Earlier steps are gold in all three cases; the `free` cases additionally force the third trigger opening but let the model write its policy. Strings below reproduce the relevant saved span or decision, with the final record omitted.

Case	Expected local behavior	Saved response and implication
LLaMA seed 101, item 1; free Y span at $n = 487$	Digit sum 19 is not divisible by three: `branch=x op=-2 post=485`.	`{odd:+10|even:+5}` followed by `branch=odd op=+10 post=497`. The reconstructed policy is wrong; this is not a failure to compute the requested subtraction.
Qwen seed 111, item 1; free Y span at $n = 637$	Digit sum 16 is not divisible by three: `branch=x op=-2 post=635`.	`{ds%3=0:-6|n:+12}` followed by `branch=n op=+12 post=649`. The model writes a different operation pair and follows its emitted alternative.
Qwen seed 444, item 13; Y operation swap at $n = 654$	Digit sum 15 is divisible by three. Under `{ds%3=0:-2|x:*2}`, the requested output is `branch=s op=-2 post=652`.	`branch=s op=*2 post=1308`. The branch is correct but the operation agrees with the original trained policy rather than the edited one. This does not uniquely identify an internal mechanism.

Table 9: Three concrete failure cases from the archived local-span experiments. The first two isolate wrong reconstructed content; the third exhibits a wrong response to supplied edited content.

J CLAIM BOUNDARIES

Supported	Not established
Written operation assignments redirect many local decisions in two 3B families.	General semantic interpretation of every written rule.
Same-record workflow value edits have a large decision-directed effect beyond a preserving-edit control.	A safe or invariant policy editor for deployed agents.
Novel arithmetic residues lack value-specific activation on the tested pool.	A unique internal mechanism such as literal slot copying.
Matched informative policy content remains useful at higher numeric values, with attenuation.	A pure length-generalization or unrestricted numerical-extrapolation result.
Textual interfaces expose separable reconstruction, interpretation, execution, and invocation requirements.	Arbitrary-depth composability has been solved.

Table 10: The intended scope of the paper.