COMPOSABLE CAUSAL CONTROL OF STRUCTURED CODE EDITING

Anonymous authors
Paper under double-blind review

ABSTRACT

When a language model edits code, it must apply an editing rule to the source program. We test whether internal-state interventions can control the rule and source features separately in Qwen3.5-122B-A10B. We keep the visible prompt and supplied Edit/old prefix fixed. We replace internal states with states recorded from other runs. These other runs are the donors. Replacing states at instruction tokens changes the editing rule but preserves source features. Replacing states at old-code tokens changes source features but preserves the rule. Applying both interventions produces the combined target. The generated replacement differs from either donor's complete output. Donor-background controls test whether the interventions also transfer unwanted donor information. We change the rule donor's source program and the source donor's editing rule. The outputs still retain the intended combination. The same procedure works across four tested code structures: straight-line code, multiple-return branches, nested branches and dataflow chains. We also reuse the same recorded states across two recipient programs. These interventions transfer a different data-dependency graph within the same statement layout. Restoring downstream residual states from a reference run recovers recipient source features while keeping the changed rule. In a separate study, we change states associated with externally supplied plans. These changes affect the model's choice between inspecting and editing code. We also test how the supplied action prefix affects sensitivity to the plan-state intervention. Supplying a policy branch reduces the sensitivity of the next-token distribution in 5 of 6 task families.

1 INTRODUCTION

Editing a program to square its return value requires more than selecting a squaring operator. The model must apply the transformation to the expressions, variables and data dependencies of the source code. We ask whether internal-state interventions can separately control the editing rule and source features, and whether these effects compose across donor and recipient contexts.

We test this question in Qwen3.5-122B-A10B. The model receives an Edit envelope and the complete `old_string`, then generates the replacement. We keep the visible prompt and supplied prefix fixed across conditions. We replace states at instruction positions, old-code positions, or both with states recorded from donor runs. The resulting programs reveal which rule and source features each intervention transfers.

Figure 1 illustrates the effect. The recipient task asks the model to multiply the return value of source A by 3. Instruction-position states from A/square change the operation to squaring while preserving A's variables and structure. Old-code-position states from B/multiply transfer B's source features while retaining multiplication. Applying both produces B/square. This joint output combines the selected factors and differs from either donor's complete replacement, despite the unchanged visible A/multiply task.

We test transfer beyond this pairing on new functions and non-additive rules. The tests cover 4 code structures: straight-line code, multiple-return branches, nested branches and dataflow chains. We keep the intervention layers and span-selection procedure fixed. We also reuse the same captured donor states across 2 recipients, transferring a different data-dependency graph within the same

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

(a) Baseline: A, multiply

```
def compute(amount):
    total = amount * 4
    return (total + 7) * 3
```

(b) Rule only: A, square

```
def compute(amount):
    total = amount * 4
    return (total + 7) ** 2
```

(c) Source only: B, multiply

```
def compute(index):
    result = index * 5
    return (result - 6) * 3
```

(d) Joint: B, square

```
def compute(index):
    result = index * 5
    return (result - 6) ** 2
```

Figure 1: **Separate rule and source interventions compose.** Actual new strings for source A with a multiply-by-3 instruction. Rule-only uses states from A/square; source-only uses states from B/multiply; their joint application produces B/square. All 4 runs receive the same visible recipient prompt and supplied Edit/old prefix.

statement layout. Controls for unwanted donor information change the rule donor's source program and the source donor's rule. The selected effects persist in these controls (Section 3).

Restoring downstream residual states recovers the recipient source features while retaining the changed rule. A narrower intervention at a single attention read produces only a small shift in source-token preference (Section 4). The restoration experiment evaluates complete programs. The attention-read experiment measures next-token distributions.

These content experiments supply the decision to edit and the old argument. To examine native action choice, we perturb states associated with externally supplied Plans in 2 repository cohorts. The perturbation changes inspection/Edit choices. In controlled serialization tests, next-token distributions become less sensitive to the perturbation after a policy branch is supplied in 5 of 6 task families (Section 5).

Our contributions are:

- Separate and joint control of editing-rule and source effects in complete generated programs, with tests across 4 code structures, reused donor states and changed donor backgrounds.
- Downstream restoration that reverses the induced source switch while retaining the changed rule, contrasted with the small preference shift from a specified attention read.
- Evidence that a Plan-state perturbation changes native action choices, while its influence on continuation depends on the supplied action context.

2 Studying Decisions and Content in Code Editing

Tasks and outcomes. All studies use Qwen3.5-122B-A10B (Qwen Team, 2026). A structured `Edit` specifies a file and two strings. The `old_string` locates the code to change, and the `new_string` replaces it. Our main experiments supply the Edit envelope and complete old argument, fixing action choice and old localization. The model generates the replacement, which we score for the requested rule, source features and applicability. Sections 3 and 4 use this setting to study composition and downstream restoration.

The Plan experiments in Section 5 address action choice and continuation under different visible contexts. Externally supplied plans select between 2 valid policies. In repository tasks, the model chooses whether to inspect or edit. In serialization experiments, we supply matched prefixes before or after the first token that distinguishes the policies. We report native action choice, next-token distributions and task verification as separate outcomes.

Internal-state interventions. Across these settings, the *recipient* is the run being modified and a *donor* supplies states from another task. For an aligned token span A at layer input ℓ, we replace each full residual vector:

$$\widetilde{H}_\ell^R[A_R[j]] \leftarrow H_\ell^D[A_D[j]], \qquad j = 0, \ldots, |A_R| - 1.$$

The replacement leaves visible recipient tokens unchanged. Donor and recipient spans have equal lengths, without padding or interpolation. Layers are zero-indexed; B17 and B18 denote inputs to decoder layers 17 and 18. Donor vectors are collected in their original task contexts.

Controls and experimental units. The factorial content experiments cross 2 sources with 2 rules and retain all 4 worlds in every arm. The reuse test first validates 3 sources crossed with 2 rules. It then applies fixed donor states to 2 recipients. The donor-background test validates 6 selected worlds and varies the donors for one recipient. Each content experiment uses a fresh model instance and its own donors. Its execution path combines a prefill with token-by-token processing of the supplied prefix. Plan experiments use full prefill instead.

Discovery and held-out confirmation use disjoint populations; the serialization follow-up reuses its original test population. Family means summarize the 6-family serialization study, and repository-task units define the 2 cohort denominators. The appendices give the intervention spans, numerical tolerances, and observer, repeat, self-replacement and cache controls for each experiment.

3 Composing Rule and Source Effects

Our factorial design tests separate control of the editing rule and source features, and their joint composition. We pair a donor that changes the rule with another that changes the source. Joint intervention tests whether the generated program applies the first donor's rule to the second donor's variables and computations.

3.1 A Factorial Intervention Design

Let $s, r \in \{0, 1\}$ index the source programs and editing rules, respectively. We denote by X_{sr} the task of applying rule R_r to source program S_s, and by T_{sr} its target program. The factorial experiments cross 2 sources with 2 rules. Within each recipient task, the visible prompt and supplied Edit/old prefix remain fixed across intervention conditions. The prompt displays the source program, and old contains the same complete function. All 4 unmodified baselines produce their corresponding targets.

The two interventions act at different token positions and layers. The *rule intervention*, G17, replaces the 8 instruction-token vectors at layer 17 input with states from $X_{s,1-r}$. This donor has the same source as the recipient but a different rule. The *source intervention*, O18, replaces old-string-token vectors at layer 18 input with states from $X_{1-s,r}$. This donor has the same rule as the recipient but a different source. We test each intervention separately and then combine them, with the following target programs:

Rule only:	$G17 \leftarrow X_{s,1-r}$,	target $T_{s,1-r}$,
Source only:	$O18 \leftarrow X_{1-s,r}$,	target $T_{1-s,r}$,
Joint:	$G17 \leftarrow X_{s,1-r}$, $O18 \leftarrow X_{1-s,r}$,	target $T_{1-s,1-r}$.

Both sets of donor states come from unmodified baselines in the same model instance. Joint intervention copies full vectors from these 2 donors at aligned positions. The O18 donor states are collected independently of the recipient's G17 intervention.

We use sites fixed in preceding source-entry and restoration experiments (Appendix E). Each run begins with a fresh prefill that stops before old, followed by forced single-token steps through old and the new-string opener. G17 acts during prefill; O18 acts later, as the model processes old. The model then generates greedily without further intervention. The initial and new-factor studies replace 35 old-token vectors; structural extensions select the full old function using the same span-selection procedure. Each factorial block has 24 validation calls followed by 12 intervention calls.

Scoring an edited program. We evaluate both patch applicability and the resulting program. After applying new at the supplied old anchor, we compare the program's return values with the target rule. Source checks compare variables, assignments, branches and statement order with the source expected for that intervention. The return expression may change as required by the rule. We also score Edit format, exact old, unique applicability, distractor preservation and literal equality to each donor's new string. The old checks validate the supplied anchor; they do not test whether the model can locate it. Each generated program is one outcome, evaluated on 65 integer inputs in the initial studies or 69 integer/rational inputs in the structural extension.

Source/rule setting	Rule only	Source only	Joint	Joint equals donor
Initial functions, add/subtract 7	4/4	4/4	4/4	0/4 G; 0/4 O
New functions, multiply/square	4/4	4/4	4/4	0/4 G; 0/4 O
Straight-line, multiply/square	4/4	4/4	4/4	0/4 G; 0/4 O
Multiple returns, multiply/square	4/4	4/4	4/4	0/4 G; 0/4 O
Nested branches, multiply/square	4/4	4/4	4/4	0/4 G; 0/4 O
Dataflow chain, multiply/square	4/4	4/4	4/4	0/4 G; 0/4 O

Table 1: **Rule and source effects compose across the tested settings.** Each fraction counts 4 source/rule worlds in one arm of a separately validated experiment. Success requires target semantics and expected source features; all scientific edits also have exact applicable old and preserve the distractor. The last column compares joint new strings with the selected rule donor (G) and source donor (O). All target worlds have baselines.

3.2 Separate Effects and Their Joint Composition

With add-7 and subtract-7 rules, G17 alone changes the rule and retains recipient source features in 4/4 worlds. O18 alone changes source features and retains the recipient rule in 4/4. Each single-arm output equals its selected donor's entire new string in 3/4 cases; the remaining case is an equivalent arithmetic rewrite.

Together, G17 and O18 produce $T_{1-s,1-r}$ in 4/4 worlds, combining the selected rule and source features. Every joint new string differs from both selected donors and matches the expected semantics, variables and structure. All 4 targets have unmodified baselines for comparison; 3 initial joint strings equal that unselected target-world baseline (Appendix D.2).

All 12 edits have exact, uniquely applicable old strings and preserve the unrelated function. In the O18 and joint arms, old still matches the recipient even though new uses the other source's variables and structure. The source switch therefore appears in the generated replacement within a valid edit envelope.

3.3 Transfer to New Rules and Code Structures

The initial results use additive rules. To test transfer beyond them, we replace both functions and use 2 non-additive rules: multiply every return by 3, or square it. We keep the intervention sites and donor design fixed. These factors were not used to select the sites. All 3 arms again produce the expected target and source features in 4/4 cases each. Joint strings differ from both selected donors in all 4 cases and equal their corresponding target-world baselines (Appendix D.3).

The structural extension applies the same procedure to 4 structures: straight-line arithmetic with one return, sequential conditions with multiple returns, nested branches, and dataflow chains with dependent intermediate assignments. Each crosses 2 sources with multiplication and squaring. Intervention layers, rule span and donor contrasts are fixed; the old span follows a prescribed full-function selection rule. Each structure receives a fresh model instance, its own donors and all validation controls.

Each arm succeeds in all 4 worlds per structure (Table 1). Single-arm strings equal their selected donors; all joint strings differ from both. Every edit is complete and applicable and passes the 69-input semantic checks. In the dataflow chain, the selected source determines the intermediate variables and their assignment order. The selected rule determines how the final expression changes. Appendix D.4 provides all source functions, per-structure counts and independent rescoring records.

3.4 Reuse Across Recipients and Donor Backgrounds

The preceding experiments use donors matched to the recipient on one factor, with new states captured for each source/rule setting. Reusing a fixed pair of donor states tests transfer across recipients. Changing the donors' other factors tests whether the interventions also transfer unwanted information.

Fixed states transfer a different dependency graph. We collect G17 once from A/square and O18 once from D/multiply, then apply the same captured states separately and jointly to A/multiply and B/multiply. A and B have assignment chains; D computes 2 intermediate values directly from the input before combining them. Figure 2 shows both recipients and the generated joint program.

Recipient source A

```
def compute(x):
    a = x * 2
    b = a + 3
    c = b * 4
    return c - a
```

$x \to a \to b \to c$

Recipient source B

```
def compute(y):
    a = y - 40
    b = a * a
    c = b + 7
    return c + a
```

$y \to a \to b \to c$

Joint output for A and B

```
def compute(z):
    a = z * 30
    b = z + 2
    c = a * b
    return (c - b) ** 2
```

$z \to \{a, b\} \to c$

Figure 2: **Identical donor states transfer a different dependency graph to 2 recipients.** Left and middle: the source programs supplied as old. Right: the actual joint new string generated for both recipients using G17 from A/square and O18 from D/multiply. The diagrams show assignment dependencies; the code also shows the return dependencies. D's fork-and-join dependencies replace the recipients' chains within the same statement layout, while the return applies squaring.

Rule-only produces A/square and B/square; source-only produces D/multiply in both recipients. Both joint outputs are D/square, preserving D's dependencies and differing from either selected donor's complete replacement. On recipient B/multiply, a same-rule control using G17 from A/multiply and a same-source control using O18 from B/square both retain B/multiply. All 8 conditions pass semantic and source checks (Appendix D.5).

The selected effects survive changes in donor background. A separate straight-line experiment tests whether donor backgrounds affect the selected effects. We vary the donors' unselected factors while keeping recipient A/multiply fixed. G17 from either A/square or C/square produces A/square (2/2), while O18 from either B/multiply or B/add-7 produces B/multiply (2/2). All 4 joint pairings produce B/square (Table 2). When G17 comes from C/square and O18 from B/add-7, the generated program uses B's source features and squares its return, importing neither C's source features nor the add-7 rule. Thus composition persists when both donors differ from the recipient in both factors. Appendix D.6 gives the source functions and all 8 outcomes. Selectivity is not uniform across settings: an earlier additive-rule test loses the rule in 1 of 4 conditions when both factors of the O18 donor change (Appendix D.1).

Recipient	G17 donor	O18 donor	Generated target
A/multiply	A/square	B/multiply	B/square
A/multiply	A/square	B/add-7	B/square
A/multiply	C/square	B/multiply	B/square
A/multiply	**C/square**	**B/add-7**	**B/square**

Table 2: **Rule and source effects compose across donor backgrounds.** All 4 joint outputs pass the target semantics and source checks, and differ from both selected donors' complete new strings. The final row changes both donors' non-target factors relative to the first row. These straight-line A/B/C sources belong to a separate experiment from Figure 2.

4 Interrupting Induced Source Shifts via Downstream Restoration

Joint intervention changes both the editing rule and source features. Can a downstream intervention recover the recipient's source features while retaining the changed rule? We test this in the multiply/square setting by restoring residual states downstream of O18 and evaluating complete generated programs. A separate, more localized experiment substitutes old-code keys and values at one attention read during generation and measures the resulting next-token preferences.

4.1 Residual Restoration Recovers Recipient Source Features

Our restoration reference is a matched G17-only trajectory, which produces the changed rule with the recipient's source features. At each of the 35 old-token positions, restoration F replaces the complete output residual of decoder layer 18 with the reference vector. Layer 19 then receives that vector. This places F downstream of the O18 input intervention. The experiment uses one fresh model instance for 28 validation calls followed by 12 intervention calls (Appendix E.3).

G17-only and G17+O18 reproduce their expected targets in 4/4 conditions each. Adding F to the joint intervention returns all 4 outputs to the G17-only target: the recipient's variables and structure, with the changed rule. Complete generated token sequences and new strings match the reference continuations exactly. All $4 \times 35 = 140$ replacements change the residual. The records confirm that each resulting vector equals the G17-only reference and reaches layer 19 unchanged.

Restoring the output residual is therefore sufficient to interrupt the induced source shift while retaining the changed rule. The restoration leaves layer 18's convolutional and recurrent cache unchanged. It replaces the full residual vector at every old position, allowing subsequent layers to respond to the restored state.

4.2 One Attention Read Produces a Small Preference Shift

Full residual restoration replaces complete vectors. A separate experiment tests a narrower intervention at one layer-19 attention read. For a joint trajectory, we substitute keys and values at the 35 old-code positions with those from its matched G17-only reference. The query, other attention operands and persistent cache stay unchanged. At the source-discriminating token position, the intervention raises the recipient-versus-donor log-probability margin by approximately 0.125 nats in all 4 conditions. Every margin remains negative, so the donor-source token is still preferred. Repeat, clone-only and self-read distributions exactly match their recorded references.

Appendix E.4, Table 6 reports the complete margins and the rule and scope landmarks. This substitution shifts the measured preference toward the recipient source but does not reverse it. Unlike the residual-restoration experiment, it ends at next-token measurement; no continuation is generated. The interventions also differ in width and timing, so their outcomes do not rank the importance of residual and attention pathways.

5 Plan Influence Depends on Action Context

The composition and restoration experiments supply the Edit action and its old argument. We now turn to earlier decisions: how does an externally supplied Plan influence action choice, and how does that influence change as an action is serialized? We first establish a fixed Plan-span perturbation. We then test native inspection/Edit choices in repositories without supplying the next action. Finally, we compare continuation sensitivity before and after a policy branch is supplied in controlled prefixes.

5.1 A Frozen Perturbation Changes Editing Behavior

We first check behavioral steering on 32 Python and Go templates from 8 semantic families. Plan A and Plan B each elicit their corresponding oracle-valid policy in 32/32 cases. Using a separate discovery set of 2 families and 8 aligned templates, we compare 49 residual boundaries and select B17 for whole-plan replacement. This boundary remains fixed in the subsequent experiments. On 2 disjoint held-out families, all 8 template comparisons per direction favor the donor policy. All 8/8 generated edits per direction are applicable and satisfy donor-side semantics (Appendix B.2).

Equal-span shuffled and neutral controls on 6 separate families also retain substantial effects. The registered specificity criterion is not met (Appendix B.2.3). We use whole-plan replacement as a fixed upstream perturbation. These controls do not identify a Plan-specific representation.

5.2 Native Inspection/Edit Choices in Repositories

We apply the same perturbation to 2 repository-disjoint cohorts containing 24 and 64 tasks from an internal SWE pool, selected under recorded structural rules. Here the model chooses its next action.

The perturbation reverses the registered inspection/Edit contrast in both directions in 10/11 and 17/22 informative tasks, respectively. Every switched edit is applicable (Figure 3).

Only 5/10 and 5/17 switched edits pass the canonical verifier. Cohort 2 therefore falls below its recorded criterion of 8 verifier-valid outcomes. The intervention changes native action choice more consistently than it achieves verified task success. Appendix C gives the full acquisition and evaluation funnels; Appendix F describes the historical protocol records.

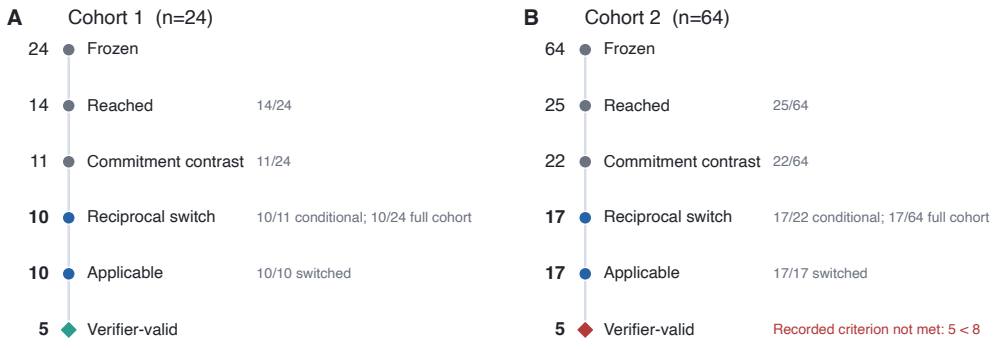

Figure 3: **Native action switching and task success in real repositories.** Both cohorts retain the full selection funnel, from all tasks through Edit reachability, informative contrast, reciprocal switching, applicability and verification. Cohort 2 has 5 verifier-valid switched tasks, below its recorded criterion of 8.

5.3 Supplying a Policy Token Reduces Plan Sensitivity

The repository results concern action choice. The serialization study asks how supplying a policy branch changes sensitivity to the Plan-span perturbation. The serialization study uses 6 fixed families, 24 templates and 96 factorial cells. We compare next-token distributions before the first policy-specific token and after supplying either policy's branch token. At each landmark, the compared runs receive identical visible prefixes. A 4-condition design crosses Plan-state replacement and action-prefix state replacement: Original, Plan-only, Prefix-only and Double. Plan-only versus Original measures Plan sensitivity without replacing action-prefix states; Double versus Prefix-only measures it with that replacement (Appendix B.3.3).

Without action-prefix state replacement, mean full-vocabulary Jensen–Shannon divergence (JSD) falls from 0.04447 nats before the branch to 0.00709 after the correct token and 0.00636 after the wrong token. With action-prefix state replacement, mean JSD falls from 0.10807 to 0.00476 and 0.00409 (Figure 4A). Average sensitivity decreases after either supplied branch identity, including the one that conflicts with the intended policy. The family-level results differ: 5 of 6 show attenuation, while recursive-list merge reverses. Resource charge has small contrasts that change sign under phase-matched donor recapture (Appendix Figure 7).

Because supplying a branch token also changes the visible context, we test whether matched surface forms produce the same attenuation. We change the cue's meaning but keep its token count, digit forms and syntax fixed. The cue specifies either the policy branch or a policy-irrelevant nuisance bit. The nuisance cue retains 3.06 times as much divergence as the policy cue, with the same direction in 5/6 families (Figure 4B). Policy cues therefore produce greater attenuation under matched surface conditions. Policy information and local predictability still co-vary in this comparison (Section 7).

From token preferences to complete edits. At the unforced baseline, the registered target has the highest candidate score in 67/96 cells, but generation is semantically valid in only 4/96. Longer supplied prefixes improve valid continuation. A single shared-token residual transplant, however, yields 0/96 valid restorations at each of 4 tested boundaries (Appendix B.3).

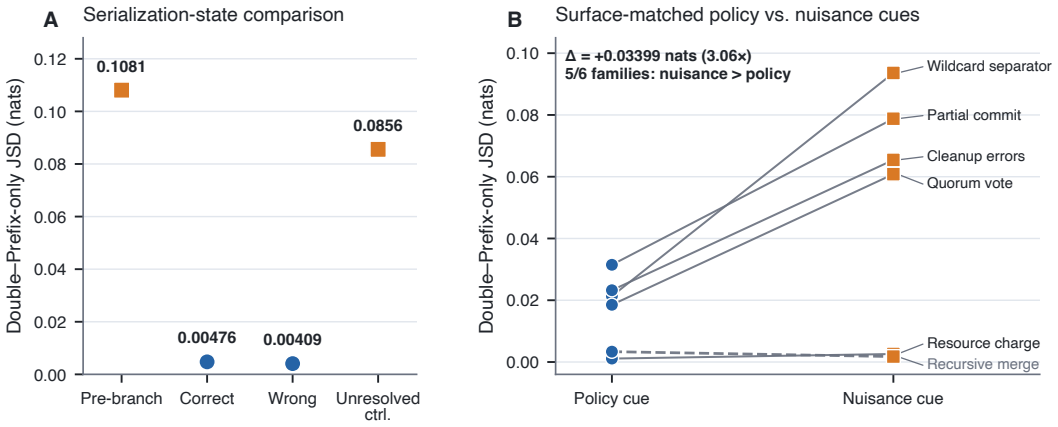

Figure 4: **Plan sensitivity falls after branch identity is supplied.** (A) Full-vocabulary Double- versus-Prefix-only JSD before the branch and after correct, wrong or unresolved-control tokens, with action-prefix state replacement. (B) A surface-matched policy-versus-nuisance comparison. Mean attenuation also occurs without action-prefix state replacement; Appendix Figure 7 reports all 6 families, including the reversal.

6 Related Work

Task and context composition. The closest precedent is request/context composition in retrieval. Variengien & Winsor (2025) show across models and domains that residual patching can combine a request from one input with the context of another. Function Vectors induce task behavior across prompts, including natural text completions (Todd et al., 2024). We study composition in complete structured code edits. The replacement must follow the selected rule, preserve the selected source features and apply at the supplied old anchor. Using 2 donors, we test separate and joint effects, reuse across recipients, and selectivity across donor backgrounds.

Interchange interventions and causal abstraction. Interchange interventions replace a neural state with one from another input and test the resulting behavior. Causal abstraction formalizes how such interventions can relate neural states to high-level computational variables (Geiger et al., 2021). Activation-patching results depend on corruption methods, evaluation metrics and whether layers are patched individually or jointly (Zhang & Nanda, 2024). In code generation, residual patching has also been used to study competing semantic cues (Abdelsalam et al., 2026). Our factorial comparisons test whether rule and source effects compose in generated programs.

Agent decisions and tool use. Chen et al. (2025) distinguish the choice between code execution and textual reasoning from task-solving ability. In single-turn, fixed-menu settings, tool identities can be steered while arguments adapt to the schema (Wu et al., 2026). TACT and AgentLens intervene in coding-agent activations (Sui et al., 2026; Luo et al., 2026). Sun & Kazakov (2026) decode tool-call dependencies; their patches change downstream probe readouts without changing the actual next tool call. Silva et al. (2026) probe properties of current and future programs. Our Plan experiments distinguish native inspection/Edit switching from verifier success. Our content experiments hold the action fixed and test rule/source control over its replacement argument.

Visible prefixes and contextualized states. Tao et al. (2024) distinguish task inference from mapping an inferred answer to output labels during in-context learning. Our Plan study asks how supplying policy information in an action prefix changes sensitivity to an earlier state perturbation. Memento compares retained summary KV states with states recomputed from the same summary text after removing the original reasoning blocks. Retaining the states yields higher accuracy in its Qwen3-8B ablation on AIME24 (Kontonis et al., 2026). Our content experiments test state transfer across donor contexts while holding the recipient's visible prompt and Edit/old prefix fixed.

7 DISCUSSION

Our experiments show that rule and source effects on replacement code can be separately controlled and composed in Qwen3.5-122B-A10B. With recipient A/multiply, instruction states from C/square and old-code states from B/add-7 produce B/square. The output combines the selected factors without adopting C's source features or the add-7 rule, and differs from both selected donors' complete replacements. Results across 4 code structures and reuse of the same captured states across 2 recipients extend this evidence beyond matched donor pairs. Downstream residual restoration then recovers recipient source features while retaining the changed rule.

The scope of selective control. Selectivity depends on the tested context. In the earlier additive-rule experiment, changing both the source and rule of the O18 donor yields an untransformed donor-source program in 1 of 4 conditions (Appendix D.1). Thus the successful donor-background controls coexist with a setting in which the source intervention drops the rule. G17 and O18 replace complete contextualized vectors and differ in layer, timing and span width. These experiments establish effective control at the selected sites. They leave open how the model binds a rule to source variables. The donor and recipient spans have equal token lengths and aligned positions. Transfer across different span lengths remains untested. The content experiments also supply Edit and old, so they do not explain natural anchor selection or its link to action choice.

Policy information and local predictability. The Plan experiments raise a different question: why does supplying a policy branch reduce sensitivity to an earlier perturbation? The token specifies the branch and also makes the next token more predictable. Without intervention, maximum next-token probability rises from 0.737 before the branch to 0.946 or 0.939 after the correct or wrong branch token, respectively. Policy cues produce greater attenuation than policy-irrelevant nuisance cues under matched surface forms. This comparison does not separate policy information from local entropy, salience and prior expectations. Varying policy information while matching continuation predictability would help distinguish their contributions.

REPRODUCIBILITY STATEMENT

The supplementary archive `supplementary_code.zip` provides offline rescoring of saved synthetic Edit responses, recomputation of attention-read distributions and aggregation of Plan measurements. Its 6 entry points pass in a clean Python environment; `README.md` and `docs/REPRODUCTION.md` explain how to run them. Internal repository tasks, their verifier and GPU generation code are excluded. Appendices A, D.4–D.6 specify the execution paths, programs, donor maps and scoring rules, while Appendix F.3 distinguishes the checks completed for each study.

AI-ASSISTED LANGUAGE EDITING

We used Codex (OpenAI) to improve the clarity, grammar, and flow of the manuscript. The authors reviewed and revised the AI-assisted edits and take responsibility for the final text.

REFERENCES

Youssef Abdelsalam, Norman Peitek, Anna-Maria Maurer, Marvin Wyrich, and Sven Apel. A mechanistic lens on semantic conflicts: Using activation patching to understand LLM behavior, 2026. URL https://arxiv.org/abs/2607.05587.

Yongchao Chen, Harsh Jhamtani, Srinagesh Sharma, Chuchu Fan, and Chi Wang. Steering large language models between code execution and textual reasoning. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 99676–99707, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/f763f7c9a6599e14b07add5937d8189c-Paper-Conference.pdf.

Atticus Geiger, Hanson Lu, Thomas Icard, and Christopher Potts. Causal abstractions of neural networks. In *Advances in Neural Information Processing Systems 34*, pp. 9574–9586, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/4f5c422f4d49a5a807eda27434231040-Abstract.html.

Vasilis Kontonis, Yuchen Zeng, Shivam Garg, Lingjiao Chen, Hao Tang, Ziyan Wang, Ahmed Awadallah, Eric Horvitz, John Langford, and Dimitris Papailiopoulos. MEMENTO: Teaching LLMs to manage their own context, 2026. URL https://arxiv.org/abs/2604.09852.

Weidi Luo, Qiming Zhang, Yihao Quan, Mingyu Jin, Jie Cai, Chaowei Xiao, Jingcheng Niu, and Zhen Xiang. AgentLens: Interpretable safety steering via mechanistic subspaces for multi-turn coding agent, 2026. URL https://arxiv.org/abs/2606.22673.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

André Silva, Han Tu, and Martin Monperrus. Latent programming horizons in coding agents, 2026. URL https://arxiv.org/abs/2607.05188.

Yuan Sui, Yulin Chen, Yibo Li, Xue Jiang, Yufei He, Yihong Dong, Xiaoxin He, Tianyu Gao, and Bryan Hooi. TACT: Mitigating overthinking and overacting in coding agents via activation steering, 2026. URL https://arxiv.org/abs/2605.05980.

Tianda Sun and Dimitar Kazakov. Tool-call dependency structure is linearly decodable in LLM agent residual streams, 2026. URL https://arxiv.org/abs/2605.25310.

Junyi Tao, Xiaoyin Chen, and Nelson F. Liu. Inference and verbalization functions during in-context learning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 16394–16421, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.957. URL https://aclanthology.org/2024.findings-emnlp.957/.

Eric Todd, Millicent L. Li, Arnab Sen Sharma, Aaron Mueller, Byron C. Wallace, and David Bau. Function vectors in large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=AwyxtyMwaG.

Alexandre Variengien and Eric Winsor. Look before you leap: Universal emergent mechanism for retrieval in language models. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 69352–69394, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/ad36c2cfc423e75c6d68d751a955b22e-Paper-Conference.pdf.

Zekun Wu, Ze Wang, Seonglae Cho, Yufei Yang, Adriano Koshiyama, Sahan Bulathwela, and Maria Perez-Ortiz. Tool calling is linearly readable and steerable in language models, 2026. URL https://arxiv.org/abs/2605.07990.

Fred Zhang and Neel Nanda. Towards best practices of activation patching in language models: Metrics and methods. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=9eJv5PS27Q.

A SHARED METHODS AND EVALUATION PROTOCOLS

This appendix specifies the execution paths, state replacements, controls and evaluation measures shared across experiments. The map below locates the task-specific procedures and results.

Appendix	Contents
A	Shared execution settings, state replacement and evaluation measures
B	Plan populations, intervention selection, specificity and serialization controls
C	Repository acquisition, native action choice and verifier outcomes
D	Separate and joint rule/source effects; transfer across programs, recipients and donors
E	Source-entry comparisons, residual restoration and a generation-time attention read
F	Study records, compute, verification coverage and backend sensitivity

A.1 MODEL, INPUTS AND EXECUTION PATHS

All experiments use Qwen3.5-122B-A10B. Layer indices are zero-based: B17 and B18 denote residual-stream inputs to decoder layers 17 and 18. State replacement copies complete residual vectors between equal-length aligned token spans while leaving visible recipient tokens unchanged. Donor states come from unmodified trajectories, except where a restoration experiment specifies an intervened reference trajectory.

The Plan studies intervene during full prefill. Content studies use an initial prefill ending before old, followed by forced single-token steps through the complete old argument and new-string opener. Donors, recipients and controls share this segmented schedule within each experiment. Full-generation content experiments then decode greedily, capped at 384 generated tokens including EOS. The first EOS is retained and not fed back. The final forced forward supplies the first generated token. A call with f forced single-token steps and n generated tokens therefore uses $1+f+(n-1)=f+n$ forwards: one prefill, f forced steps and $n-1$ decoding steps. The attention-read experiment instead measures next-token distributions at specified landmarks without generating a continuation.

Content experiments use bfloat16 computation on 8 NVIDIA H20 GPUs, the resolved `grouped_mm` expert dispatcher and Torch chunk/recurrent DeltaNet routines. The model has 12 full-attention and 36 linear-attention layers. Linear-attention state consists of convolutional and recurrent containers; it is not a position-indexed key/value cache. Each content experiment uses a fresh model instance for its donors, recipients and controls. Source functions, token spans and condition counts are specified with the corresponding results. A preliminary full-prefill/segmented comparison in the source-entry setting yielded TV 0.00253 and 0.000554, exceeding the 10^{-5} transport tolerance despite matching greedy suffixes. The content results therefore describe the validated segmented execution path.

A.2 PLAN-STATE REPLACEMENT DURING FULL PREFILL

Plan-state interventions in serialization and repository experiments replace tensors during full `generate(use_cache=True)` prefill at B17. Let

$$H_{17}^{R} \in \mathbb{R}^{1\times L_R\times d}$$

denote the recipient residual tensor. P_R covers the complete external Plan. S_R covers the action prefix from `<tool_call>` through the new-string opener, including the path and old argument. The spans are disjoint: $P_R \cap S_R = \varnothing$.

For a donor span A_D and matched recipient span A_R, the implementation clones the recipient hidden tensor and replaces the registered slice position-wise:

```
replacement = hidden.clone()
replacement[:, start:end] = donor_state
```

The donor tensor has shape `[span_tokens, hidden_dim]` and is converted to the recipient device and dtype before replacement.

Let D_S denote the action-prefix donor and D_P the Plan donor. Unmodified donor prefills provide

$$Z^{D_S}_{17,S} = H^{D_S}_{17}[S_{D_S}], \qquad Z^{D_P}_{17,P} = H^{D_P}_{17}[P_{D_P}].$$

The 4-condition extension crosses Plan-state and action-prefix state replacement:

$$\begin{aligned}
\textsc{Original}: &\quad \widetilde{H}_{17} = H^R_{17}, \\
\textsc{Plan-only}: &\quad \widetilde{H}_{17}[P_R] \leftarrow Z^{D_P}_{17,P}, \\
\textsc{Prefix-only}: &\quad \widetilde{H}_{17}[S_R] \leftarrow Z^{D_S}_{17,S}, \\
\textsc{Double}: &\quad \widetilde{H}_{17}[S_R] \leftarrow Z^{D_S}_{17,S}, \qquad \widetilde{H}_{17}[P_R] \leftarrow Z^{D_P}_{17,P}.
\end{aligned}$$

Other recipient positions remain unchanged. Plan-only versus Original measures Plan sensitivity without prefix-state replacement; Double versus Prefix-only measures it with the same prefix replacement. The disjoint writes commute and give bitwise-identical tensors in either order.

For the primary 4-condition analysis, scientific donor tensors are captured once at the unextended generation prefix and reused across all supplied phases. A separate sensitivity analysis recaptures donors after appending each phase prefix. Exact-input self replacements use phase-matched captures from the same recipient input and are kept separate from the transported scientific donors.

Donor state j replaces recipient span position j, without padding or interpolation. Synthetic alignment is specified in Appendix B.1; repository alignment is in Appendix C.

Replacement occurs only in the B17 prefill input hook. Subsequent decoding is unpatched; cache changes follow from the intervened computation.

A.3 Content Controls and Scoring

Content experiments use four types of control. Baselines omit state replacement. Hook-off references check observation effects, and repeated runs check stability. Self replacements exercise the intervention code with a world's own states. Hook-off references omit the internal observation hooks; instrumented baselines retain them. Comparisons cover generated tokens, distributions and cache values at registered boundaries. The attention-read study also includes clone-only and self-read controls. Each experiment specifies its comparison tolerances and control counts.

During the forced prefix, state checks verify replacement at the registered positions, preservation of non-target residual rows and agreement of unaffected lower-layer states with the recipient baseline. Cache checks cover 48 layers and 4 container types, including empty slots. Donors are copied by value. Joint intervention applies G17 during prefill and O18 during subsequent old-token processing, using independently captured payloads. The O18 donor states are not recomputed after the recipient receives G17. Appendix F.3 summarizes verification coverage and the records retained for review.

Replacement-code outcomes. We evaluate each generated program against the rule oracle and the source expected for its intervention condition. Source checks cover variable names, assignments, branches and statement order, allowing the registered return-value transformation. We separately record Edit format, exact parsed old, unique applicability, distractor preservation and literal equality to each selected donor's complete new string. Every target has an unmodified baseline. In joint conditions, correctness against the combined target and inequality to both selected donors distinguish the composite output from either donor's full answer.

Initial content studies evaluate 65 integers in $[-32, 32]$. Structural and donor follow-ups add 4 rational inputs specified with each experiment. These inputs evaluate one generated program and do not add model trials. Success counts describe the constructed source/rule conditions and interventions, without population-level confidence intervals. Supplying Edit and old fixes action selection and old localization.

A.4 Plan and Repository Endpoints and Statistical Units

The table below lists the Plan and repository endpoints, their observables and units of analysis. Content-generation outcomes are defined in Appendix A.3.

Endpoint	Observable	Measurement unit	Independent scientific unit
Complete-candidate preference	Relative log-score of registered complete-sequence alternatives	Registered template / factorial comparison	Semantic family
Branch choice	Registered policy-token decision at L_1	Factorial cell	Semantic family
Distributional sensitivity	Full-vocabulary Plan-only-vs-Original and Double-vs-Prefix-only JSD / TV under matched prefixes	Matched cell pair	Semantic family
Semantic realization	Satisfaction of the frozen semantic oracle by a generated `Edit`	Generated cell	Semantic family
Native action commitment	Registered reciprocal inspection/`Edit` switch	Repository-task	Repository-task
Patch applicability	Existing target path, uniquely applicable `old_string` anchor, and nonempty target-only patch	Switched `Edit` within repository-task	Repository-task, conditional on switch
Official outcome	Canonical issue-specific verifier	Switched `Edit` within repository-task	Repository-task, conditional on switch

For a matched serialization condition c, let

$$P_c^{\mathrm{orig}} = p(\text{next token} \mid c, \text{Original}), \qquad P_c^{\mathrm{plan}} = p(\text{next token} \mid c, \text{Plan-only}),$$
$$P_c^{\mathrm{dbl}} = p(\text{next token} \mid c, \text{Double}), \qquad P_c^{\mathrm{pfx}} = p(\text{next token} \mid c, \text{Prefix-only}).$$

For vocabulary distributions P and Q, the primary measure is Jensen–Shannon divergence (JSD):

$$\mathrm{JSD}(P, Q) = \frac{1}{2} D_{\mathrm{KL}}(P \| M) + \frac{1}{2} D_{\mathrm{KL}}(Q \| M), \qquad M = \frac{1}{2}(P + Q).$$

Natural logarithms are used, so JSD is reported in nats and lies in $[0, \ln 2]$. The secondary distributional metric is total variation,

$$\mathrm{TV}(P, Q) = \frac{1}{2} \sum_v |P(v) - Q(v)|.$$

The two Plan-intervention sensitivities are

$$D_0(c) = \mathrm{JSD}\left(P_c^{\mathrm{plan}}, P_c^{\mathrm{orig}}\right), \qquad D_1(c) = \mathrm{JSD}\left(P_c^{\mathrm{dbl}}, P_c^{\mathrm{pfx}}\right).$$

JSD and TV are computed for each matched cell pair. We average cells within templates, templates within families, and then weight the 6 family means equally. Because the design is balanced, this equals the arithmetic mean across the 96 cells. Lower divergence measures weaker sensitivity to the Plan-state replacement at that supplied prefix; it is not a test of statistical independence, information loss, or causal mediation.

Inferential summaries use the declared units of analysis. Where family-level bootstrap resampling is used, template results are first aggregated within their semantic family. Candidate preference, generated semantics, patch applicability and verifier success remain separate endpoints.

B UPSTREAM PLAN STEERING AND SERIALIZATION DYNAMICS

This appendix details the Plan experiments in Section 5. We describe the task populations and token alignment, report boundary selection and specificity controls, then examine how supplied prefixes affect distributions and generated edits. Appendix A defines the shared intervention and evaluation methods.

B.1 TASK POPULATIONS AND TOKEN ALIGNMENT

Figure 5 shows the Plan intervention and the matched-prefix comparisons used in the serialization experiments.

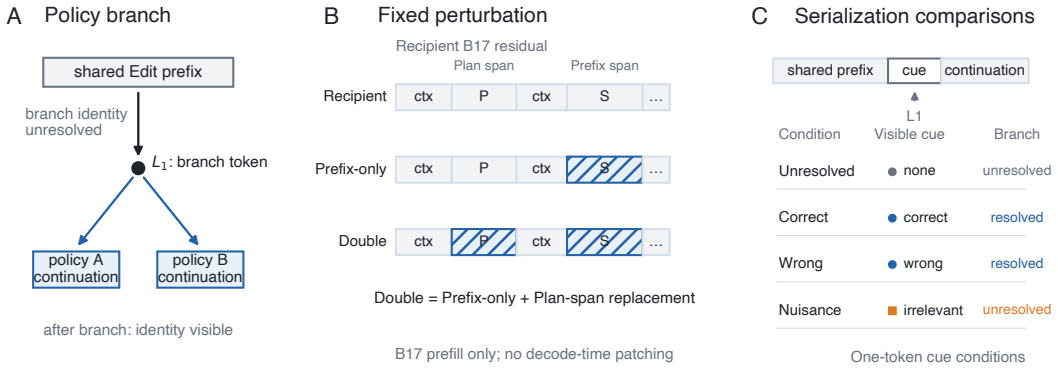

Figure 5: **Experiment overview.** Controlled `Edit` trajectories share a prefix before diverging at a known policy branch. We compare the same Plan-span perturbation before the branch is visible and after correct, wrong, or surface-matched nuisance cues. Prefix-only and Double share the action-prefix replacement; Double additionally replaces the Plan span.

B.1.1 BEHAVIORAL CONTRAST POPULATION

The behavioral study contains 32 templates from 8 semantic families, split evenly between Python and Go. Semantic policy identity is counterbalanced across Plan A/B labels (16/16). We validate both registered target implementations against separate frozen semantic oracles before manipulating Plan states. The Plan is supplied as an experimental input; it is not private reasoning or hidden chain-of-thought.

B.1.2 BOUNDARY DISCOVERY AND HELD-OUT CONFIRMATION

The boundary study uses the 4 behavioral families whose complete Plan A and Plan B spans satisfy the exact token-alignment requirement.

We divide this population into 2 discovery families with 8 templates and 2 disjoint held-out families with 8 templates. The discovery scan applies the registered selection criterion to 49 residual boundaries. We then fix the selected boundary and evaluate both transfer directions on the held-out families. Boundary selection uses neither the held-out outcomes nor the 6-family serialization population or repository cohorts.

Frozen family assignments are:

- Discovery: `duplicate_resolution`, `case_normalization`;
- Held-out confirmation: `integer_rounding`, `ordering_direction`.

We order the 4 family names by the SHA-256 hash of a frozen, study-specific preimage recorded in the study records. The first 2 families form the discovery population and the final 2 form the held-out population. This ordering is computed without using boundary effects.

B.1.3 SERIALIZATION POPULATION

The serialization population contains 6 semantic families, 4 templates per family, and 4 registered factorial cells per template. Each family contributes 16 repeated factorial cells. The 4-condition extension reuses this fixed population; it is a same-population follow-up rather than a held-out replication.

Semantic family	Templates	Language balance	Factorial cells
cleanup error reporting	4	2 Python / 2 Go	16
partial batch commit	4	2 Python / 2 Go	16
quorum vote basis	4	2 Python / 2 Go	16
recursive list merge	4	2 Python / 2 Go	16
resource charge basis	4	2 Python / 2 Go	16
wildcard separator	4	2 Python / 2 Go	16
Total	**24**	**12 Python / 12 Go**	**96**

The semantic family is the primary unit of aggregation and inference. Templates, factorial cells, phases and intervention conditions provide repeated measurements within this design. The 6 constructed families are not assumed to be independent random samples from a broader task population.

B.1.4 SYNTHETIC SPAN ALIGNMENT

For each serialization template, we construct the 4 registered policy/donor factorial worlds $J_{AP}, J_{AQ}, J_{BP}, J_{BQ}$ and freeze them before model evaluation. The first subscript records the grounding factor and the second records the policy obligation. For each recipient, the Plan donor changes only the obligation factor, while the action-prefix donor changes only the grounding factor:

Recipient	Plan donor	Action-prefix donor	Missing corner
J_{AP}	J_{AQ}	J_{BP}	J_{BQ}
J_{AQ}	J_{AP}	J_{BQ}	J_{BP}
J_{BP}	J_{BQ}	J_{AP}	J_{AQ}
J_{BQ}	J_{BP}	J_{AQ}	J_{AP}

Their tokenization must satisfy the following alignment constraints:

- the complete rendered prefix token counts are equal;
- the Plan spans have the same absolute `[start,end)` offsets;
- the complete action-prefix spans have the same absolute `[start,end)` offsets;
- corresponding donor/recipient spans contain the same number of tokens;
- Plan and action-prefix spans do not overlap.

Interchange therefore uses the local-ordinal mapping

$$Z_D[j] \to H_R[\text{start}_R + j], \qquad j = 0, \ldots, m-1.$$

Here m is the number of tokens in the replaced span. The synthetic conditions also share absolute span offsets, so donor and recipient vectors match in both ordinal and absolute token position.

B.2 SELECTING AND TESTING THE PLAN PERTURBATION

B.2.1 BOUNDARY SELECTION

The discovery scan evaluates all 49 residual boundaries on 2 discovery semantic families (8 templates) in both reciprocal transfer directions. The protocol-specified selection rule selects B17, with a mean bidirectional gain of $+2.533686$. Effects are positive in all 16/16 repeated template-direction comparisons.

The scan shows a broad high-gain region. B28 is the runner-up under the same statistic, with mean bidirectional gain $+2.532900$ and positive effects in all 16/16 template-direction comparisons. Its mean trails B17 by 0.000786. We retain B17 as the maximum under the selection rule, while recognizing that the near tie provides little evidence for uniquely localizing the effect there.

Figure 6 shows the complete 49-boundary discovery curve. All registered self/no-op identity controls are exact.

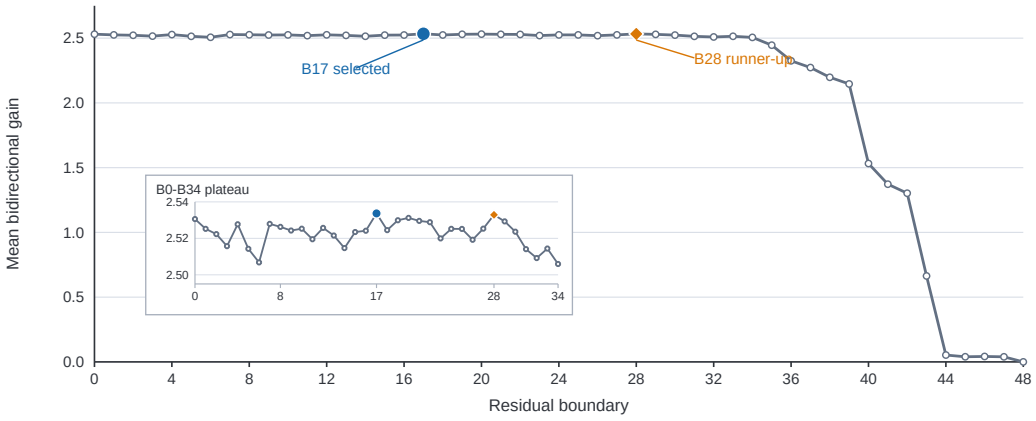

Figure 6: **Complete 49-boundary discovery scan.** The curve has a broad high-gain plateau; the inset expands B0–B34. B17 is the literal argmax ($+2.533686$) and B28 is the runner-up ($+2.532900$), a difference of 0.000786.

B.2.2 Held-Out Confirmation

At the fixed B17 boundary, the intervention remains positive in all 8/8 templates from the 2 held-out families in each direction. Mean gains are $+1.759126$ for A$\rightarrow$B and $+1.750524$ for B$\rightarrow$A. These counts describe consistency across templates within the held-out families.

Generated edits on the held-out families. In each reciprocal direction, all 8/8 generated actions are closed Edits with the registered path, a uniquely applicable old anchor and donor-side semantics; 5/8 exactly equal the registered donor target.

B.2.3 Equal-Span Specificity Controls

The specificity study uses a separate 6-family population to test whether whole-plan interchange depends on preserving the ordered semantic structure of the donor Plan.

Aligned, shuffled, and neutral conditions replace the same number of B17 residual states at the same recipient Plan-span positions. Within a template, all A, B, and neutral Plan spans have the same rendered start, end, and token length, and tokens outside the Plan span are identical. Aligned interchange copies donor state i to recipient position i. Shuffled interchange applies a frozen deterministic derangement to the same donor-state multiset. Donor-state bytes, count, dtype, source-position set, and recipient coordinates remain fixed, while order and source-position correspondence change. Neutral interchange copies the ordered residual states of an equal-length, semantically neutral Plan to the same recipient positions. None of the conditions shuffle or rerender visible text.

Let Y denote the registered binary exclusive donor-policy success indicator. It equals one only when the Edit is present, closed and schema-valid, with the registered path. The old string must apply uniquely, and the single-file patch must be nonempty. The output must also pass the donor-policy oracle exclusively and pass the shared regression checks. All other outcomes score zero. For each template, the intervention effect relative to the recipient-identity baseline is

$$\Delta = Y_{\text{condition}} - Y_{\text{recipient identity}}.$$

The 2 templates within each semantic family are averaged first; the 6 resulting family-level effects are then equally weighted to obtain the aggregate Δ.

The frozen specificity rule uses

$$R_{\text{signed}} = \frac{\Delta_{\text{control}}}{\Delta_{\text{aligned}}}, \qquad R_{\text{positive}} = \frac{\max(0, \Delta_{\text{control}})}{\Delta_{\text{aligned}}},$$

when $\Delta_{\text{aligned}} > 0$. Registered specificity requires retention strictly below 0.25. Because the aggregate control effects are nonnegative here, the 2 ratios coincide.

Direction	Aligned Δ	Shuffled Δ	Neutral Δ	Shuffled / Aligned	Neutral / Aligned
A→B	0.166667	0.250000	0.166667	1.50	1.00
B→A	0.166667	0.166667	0.083333	1.00	0.50

Neither direction meets the registered specificity criterion.

A→B.

Semantic family	Aligned	Shuffled	Neutral
cache_admission_threshold	0.0	0.0	0.0
circuit_breaker_half_open_probe	0.5	0.5	0.5
environment_expansion_depth	0.0	0.0	0.0
sequence_gap_handling	0.5	0.5	0.0
symlink_resolution_basis	0.0	0.0	0.0
transactional_batch_failure	0.0	0.5	0.5

B→A.

Semantic family	Aligned	Shuffled	Neutral
cache_admission_threshold	0.0	0.0	0.0
circuit_breaker_half_open_probe	0.5	0.0	0.0
environment_expansion_depth	0.0	0.0	0.0
sequence_gap_handling	0.5	0.5	0.5
symlink_resolution_basis	0.0	0.5	0.0
transactional_batch_failure	0.0	0.0	0.0

The aligned effect is positive in 2/6 semantic families in each direction. Shuffled and neutral controls are positive in some of the same families and in additional families where aligned interchange has no effect. All 36/36 registered identity interventions reproduce their corresponding baselines with byte-identical response text, identical generated token IDs, and a maximum chosen-token log-probability difference of 0.

Steering effects persist when donor states are reordered or replaced by neutral-Plan states. The measured effect is therefore not specific to an ordered donor Plan under these controls.

B.3 Sensitivity During Action Serialization

We first compare family-level branch choices and continuation outcomes, then test one-token prefixes, action-prefix state replacement and surface-matched cues. The final analyses examine longer supplied prefixes and single-token residual transplants.

B.3.1 FAMILY-LEVEL EFFECTS

Semantic family	Pre-L_1 target-choice effect	Pre-L_1 mean margin effect	Post-L_1 oracle effect
cleanup error reporting	+9/16	+1.008	+3/16
partial batch commit	+11/16	+3.047	-8/16
quorum vote basis	+11/16	+3.555	0/16
recursive list merge	-3/16	+0.039	0/16
resource charge basis	+1/16	+0.586	0/16
wildcard separator	+8/16	+3.258	+1/16

Pre-L_1 target-choice effects range from $-3/16$ to $+11/16$. The pre-L_1 policy-margin effect is positive in all 6 families, whereas post-L_1 semantic-continuation effects range from $-8/16$ to $+3/16$.

Across the 96 repeated factorial cells, the mean pre-branch policy-margin effect is $+1.915$ nats, with positive effects in 85/96 cells. The corresponding branch-choice counts are:

Pre-branch target choice	Count
Double only	40
Prefix-only only	3
Both	26
Neither	27

Double therefore selects the registered target in 37 more cells than Prefix-only, a paired descriptive difference of $(40 - 3)/96 = 37/96 = 0.385$.

B.3.2 REGISTERED ONE-TOKEN SERIALIZATION CONDITIONS

The registered one-token conditions are:

- **Correct-resolved:** the first registered branch-resolving token of the target policy;
- **Wrong-resolved:** the corresponding branch-resolving token of the reciprocal policy;
- **Neutral-unresolved:** a protocol-specified single token that does not explicitly instantiate either registered A/B branch identity.

The Neutral token round-trips exactly through the tokenizer as _. It matches the resolved conditions in token count and registered position, but not in byte length, syntactic category, lexical content, frequency, or entropy reduction. In Python instances, for example, correct and wrong branch tokens decode as `_a` and `_b`, whereas the Neutral token decodes as _.

The pooled one-token results are:

Condition	JSD (nats)	Total variation	Same greedy token	JSD retention
Branch unresolved	0.10807	0.32748	53/96	1.000
Correct-resolved	0.00476	0.02730	96/96	0.044
Wrong-resolved	0.00409	0.02515	96/96	0.038
Neutral-unresolved	0.08556	0.28458	44/96	0.792

At the independent-unit level, Neutral-unresolved JSD exceeds Correct-resolved JSD in all 6/6 semantic families. Correct- and Wrong-resolved divergences are small across all 6 families.

Metric definitions are given in Appendix A.4.

Neutral continuation outputs. We inspect the 192 stored Double and Prefix-only Neutral continuations without issuing additional model requests. No greedy identifier exactly matches either registered A/B branch identity. Across 96 matched pairs, the first emitted token is identical in 44 pairs and differs in 52. In the Neutral condition, the registered A-vs-B log-probability margin is a fixed probe rather than a greedy branch classifier.

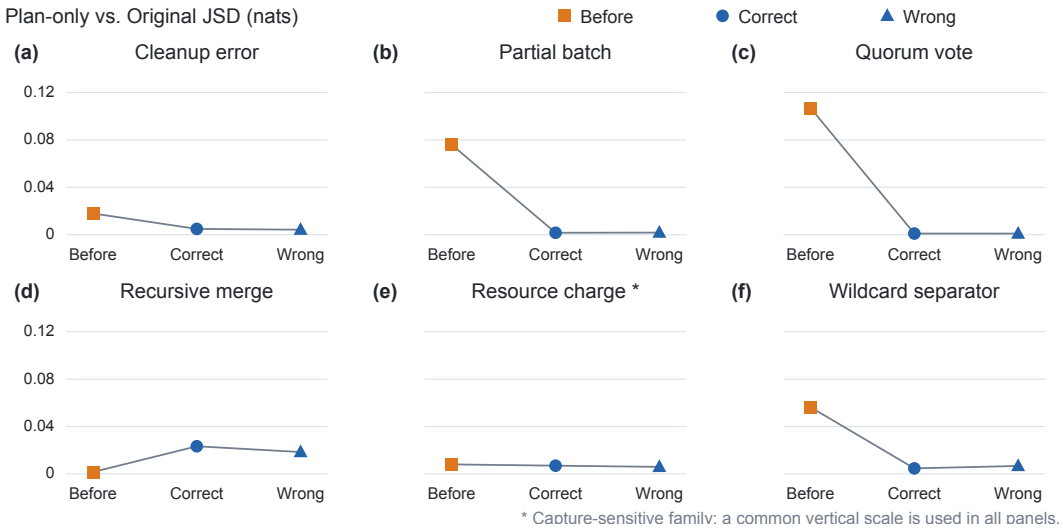

Figure 7: **Family-level Plan sensitivity without action-prefix state replacement.** JSD between Plan-only and Original, averaged within each family. Orange squares mark the unresolved prefix; blue circles and triangles mark the correct and wrong supplied branches. All panels share a vertical scale. Recursive-list merge reverses; resource charge is capture-sensitive. Each family contains 4 templates and 16 factorial cells.

B.3.3 4-Condition Plan and Prefix-State Comparison

The follow-up crosses Plan-state and action-prefix state replacement. Original and Plan-only retain the native action-prefix states; Prefix-only and Double use the same donor span. Plan-only versus Original measures Plan sensitivity without prefix-state replacement, and Double versus Prefix-only measures it with that replacement. All 4 conditions receive the same supplied visible prefix, including those without prefix-state replacement.

Scientific donor tensors are captured once at the unextended generation prefix and reused across phases. A separate sensitivity analysis recaptures donors after each phase prefix. The fixed-donor results are:

	No prefix-state replacement		With prefix-state replacement	
Supplied phase	JSD	TV	JSD	TV
Before policy token	0.04446544	0.20972930	0.10807453	0.32747593
Correct policy token	0.00709390	0.03531105	0.00475657	0.02729822
Wrong policy token	0.00635618	0.03366174	0.00409036	0.02515014
Neutral underscore	0.05167528	0.21935383	0.08556376	0.28457970

The mean attenuation after either policy token appears in both backgrounds. In 5 of 6 families, before-minus-correct and before-minus-wrong JSD contrasts are positive in both backgrounds:

	No prefix-state replacement			With prefix-state replacement		
Semantic family	Before	Correct	Wrong	Before	Correct	Wrong
cleanup error reporting	.01796	.00492	.00426	.04584	.00427	.00400
partial batch commit	.07593	.00167	.00180	.18422	.00116	.00069
quorum vote basis	.10697	.00100	.00097	.22973	.00092	.00084
recursive list merge	.00159	.02334	.01845	.00241	.01578	.01195
resource charge basis	.00803	.00696	.00594	.00755	.00203	.00255
wildcard separator	.05631	.00468	.00672	.17869	.00439	.00452

Recursive-list merge reverses the aggregate pattern. The no-prefix-state resource-charge contrasts are small and change sign when donors are recaptured at each phase. Across the 72 specified family-

by-background-by-post-phase-by-metric contrasts, 4 change sign between fixed and phase-matched capture. The 2-capture range is a sensitivity diagnostic, not a confidence interval or error bound; no cells are removed.

The formal run contains 384/384 cell-phase records and 768 primary matched comparisons. All 384/384 exact-input identity checks and 3,072 fixed-donor tensor-hash checks pass, with no missing or duplicate cell-phase rows. The 5,088 one-token generation calls use Qwen3.5-122B-A10B in bfloat16 across 8 H20 GPUs, SDPA attention, and the Transformers Torch DeltaNet fallback.

The unintervened Original condition also becomes more locally predictable after a policy token is supplied:

Supplied phase	Entropy (nats)	Maximum probability
Before policy token	0.54954731	0.73748070
Correct policy token	0.30827461	0.94571547
Wrong policy token	0.31626078	0.93912890
Neutral underscore	2.99016948	0.33680786

Supplying a branch token thus changes both policy information and local continuation predictability. These measurements establish their co-occurrence under the supplied-prefix conditions, rather than identifying which accounts for the attenuation.

B.3.4 Surface-Matched Policy-versus-Nuisance Cue Control

A separate experiment compares policy-relevant and nuisance cues under a matched one-token surface construction. The policy-cue and nuisance-bit conditions use the same one-token bandwidth, numerical surface family, surface-rotation scheme, and syntax. The policy cue explicitly resolves the registered policy branch, whereas the nuisance cue serializes a policy-irrelevant binary variable.

The cue is placed in a language-specific comment slot, `# cue=[d]` for Python and `// cue=[d]` for Go. The single-token digit surfaces `2`, `3`, `5`, and `7` rotate across the 4 codebook roles by template ordinal:

Template	`2`	`3`	`5`	`7`
1	policy P	policy Q	nuisance X	nuisance Y
2	policy Q	nuisance X	nuisance Y	policy P
3	nuisance X	nuisance Y	policy P	policy Q
4	nuisance Y	policy P	policy Q	nuisance X

The pooled Double-versus-Prefix-only JSD values are:

Cue type	JSD (nats)	Relative to policy cue
Policy-branch cue	0.01651	1.00
Nuisance-bit cue	0.05050	3.06

The matched contrast is

$$\Delta_{\text{nuisance}-\text{policy}} = +0.03399 \text{ nats.}$$

At the semantic-family level, the nuisance-minus-policy contrast is positive in 5/6 independent families. The remaining family is near zero with a small reversal.

Semantic family	Policy JSD	Nuisance JSD	Difference
cleanup error reporting	0.02324	0.06544	+0.04220
partial batch commit	0.03148	0.07878	+0.04730
quorum vote basis	0.01853	0.06083	+0.04230
recursive list merge	0.00335	0.00176	-0.00159
resource charge basis	0.00111	0.00260	+0.00148
wildcard separator	0.02134	0.09360	+0.07226

The correct and wrong policy-cue values are close, as are the X and Y nuisance-bit values. Double and Prefix-only choose the same first continuation token in 94/96 cells for the correct policy cue and 95/96 for the wrong cue. The corresponding counts for nuisance cues X and Y are 64/96 and 67/96.

The stored entropy diagnostic shows that the cue types are not matched in local predictability:

Cue role	Prefix-only entropy	Double entropy
Nuisance X	0.616738	0.612769
Nuisance Y	0.572045	0.564153
Correct policy	0.392412	0.249586
Wrong policy	0.249976	0.391949

Policy-branch cues produce lower Double-versus-Prefix-only sensitivity than the tested nuisance cues under matched token count, numerical surfaces, rotation and syntax. This direction holds in 5 of 6 families. The entropy measurements show that surface matching leaves local predictability unequal; semantic salience and model prior are also unmatched. The contrast therefore measures the difference between these cue types without isolating branch relevance from predictability.

B.3.5 Secondary Continuation Endpoints

Landmark	Double oracle-valid	Prefix-only oracle-valid	Prefix-only exact registered target
L_1: first correct policy token visible	88/96	92/96	84/96
L_2: complete correct policy call visible	88/96	94/96	86/96

After the correct L_1 token is visible, the aggregate Double-minus-Prefix-only oracle effect is $-4/96$, while family-level effects range from $-8/16$ to $+3/16$. We report the full-vocabulary next-token distribution as the phase-matched readout defined for every cell; semantic continuation is a separate, more heterogeneous endpoint.

The registered secondary L_1/L_2 suffix sequence-score readout was not collected, so no result is reported for that endpoint.

B.3.6 First Divergence and Prefix Forcing

To measure how supplied target text affects completion, we provide 0, 1, 2, 4, 8, 16, or all tokens of the registered target `new_string`, then continue greedily. Every level retains all 96 cells:

Supplied target tokens	Semantic target, all cells	Exact target, all cells	Semantic target, candidate-top1 stratum
0	4/96	0/96	4/67
1	5/96	0/96	5/67
2	3/96	0/96	3/67
4	5/96	0/96	5/67
8	5/96	0/96	5/67
16	87/96	87/96	61/67
Complete target	96/96	96/96	67/67

The complete-target condition supplies the answer and serves as a transport and closure ceiling. At 16 supplied tokens, 87/96 trajectories complete the registered target byte-exactly, comprising 44/48 Python and 43/48 Go instances and spanning all 4 transfer conditions. The candidate-top1 stratum is descriptive and does not replace the 96-cell denominator.

At zero forcing, 94/96 outputs first diverge from the registered target around suffix-token indices 9–10 and the transition into the second target line. This count includes all 5 semantic successes from the earlier historical run. Among the 91 semantic failures in that run, the corresponding count is 89/91. These divergence counts localize a mismatch in generated strings; they are not natural arrival rates for the supplied prefixes.

B.3.7 Single-Token Pre-Branch Transplants

The single-token sufficiency test uses 4 residual boundaries listed in the transplant protocol: B17, B32, B43, and B44. B17 is the frozen whole-plan intervention boundary. An earlier diagnostic showed a sharp change between adjacent boundaries B43 and B44, so we included both. B43 was positive in 8/8 repeated cases in both directions. B44 retained only 1.74% of the absolute mean B43 effect. These observations were not used to re-select the whole-plan intervention boundary.

At each boundary, only the residual state of the final shared L_0 action token is transplanted between matched Double and Prefix-only conditions.

At B17, B32, B43, and B44, the transplant produces 0/96 oracle-valid restorations at every boundary. Mean policy-margin changes over Prefix-only are $+0.0117$, $+0.0000$, $+0.0039$, and $+0.0104$, respectively.

Reciprocal Prefix-only-to-Double transplants leave the Double execution baseline unchanged. All 768/768 registered self-identity controls are exact.

Replacing the final shared action-token residual is insufficient to recover the broader pre-branch effect at these 4 boundaries. The result does not distinguish a representation at another site from one distributed across multiple positions.

C NATIVE ACTION SELECTION IN SOFTWARE REPOSITORIES

The repository experiments in Section 5.2 test whether Plan-state perturbations change native inspection/Edit choices. Interpreting these results requires tracking both task selection and the tasks that reach an informative action contrast. We describe cohort acquisition and Plan-span alignment, then report action switching, patch applicability and verifier outcomes with their respective denominators.

C.1 TASK COHORT ACQUISITION

Cohort 1. Cohort 1 starts from a frozen census of 992 candidate repository-task rows in an internal SWE pool. Each task has a frozen repository snapshot, execution image and evaluation script. This is an internal cohort, not SWE-bench Verified. Repository-level deduplication, prior-use exclusions and structural eligibility filters retain 133 rows from 124 repositories. The acquisition record specifies 24 evaluation repositories and 4 disjoint smoke repositories; Appendix F describes the available chronology.

Selection uses structural and prior-use criteria available before model or intervention outcomes. Every frozen task remains in the full-cohort denominator. This includes tasks that never reach the registered Edit point or lack an informative action contrast. It also includes inapplicable patches and verifier failures.

Cohort 2. Cohort 2 reuses the frozen 992-row structural census. After excluding the 28 repositories frozen for the preceding smoke and confirmatory cohorts, the source pool contains 105 eligible rows from 96 previously unused repositories.

At the repository level, this remaining pool contains 20 Python, 15 PHP, 15 TypeScript, 12 Ruby, 9 C, 8 Go, 6 Java, 5 JavaScript, 3 C++, and 3 Rust repositories.

A salted deterministic ranking selects 1 issue per selected repository and allocates 4 repositories to smoke testing and 64 to evaluation. The groups are disjoint. Smoke tasks validate the protocol and do not enter the evaluation denominators. These allocations follow the recorded selection protocol, with its chronology described in Appendix F.

The 64-unit confirmatory cohort contains 16 Python, 9 Ruby, 8 PHP, 8 TypeScript, 6 C, 6 Go, 5 Java, 4 JavaScript, and 2 Rust repositories. All 64 repository identities are frozen before confirmatory outcomes.

All frozen units remain in the full-cohort denominator regardless of later `Edit` reachability, switching, applicability, or verifier outcome.

C.2 PLAN-SPAN ALIGNMENT WITHIN REPOSITORY TASKS

Within each repository task, Plan A, Plan B and the registered neutral Plan N have equal token lengths and identical absolute start and end offsets. Prefix token IDs outside the Plan span are unchanged. Donor vector j replaces recipient Plan vector j at the same coordinate, without padding, rerendering or interpolation. This alignment applies to the within-task action-switching experiments in Section 5.2.

C.3 REPOSITORY-LEVEL EVALUATION FUNNELS

The table tracks both cohorts from their frozen task sets through reachability, reciprocal action switching, patch applicability and verification. Conditional and full-cohort counts are reported separately.

Endpoint	Cohort 1	Cohort 2
Frozen repository-task units	24	64
Reached registered `Edit` point	14/24	25/64
Informative native commitment contrast	11/14	22/25
Strict reciprocal commitment switch	10/11 (10/24 full)	17/22 (17/64 full)
Applicable among switched Edits	10/10	17/17
Verifier-valid among switched Edits	5/10	5/17
Verifier-valid switched `Edit`, full cohort	5/24	5/64

Cell-level execution accounting. The same repository task contributes multiple evaluation cells. These repeated measurements do not add independent repository-task outcomes.

Cohort 1 has 168 registered cells, including 62 informative `Edit` cells. All 62/62 produce applicable target-only patches, and 33/62 pass the canonical verifier.

Cohort 2 has 448 registered cells: 342 retain the recorded host-zero status, and 106 contain informative model-produced `Edit` calls. All 106/106 responses are present, closed and schema-valid. Of these, 104/106 target an existing path, contain a uniquely applicable `old_string`, and yield a clean, nonempty target-only patch. The remaining 2/106 fail the unique-applicability check and remain in the denominator; neither was rerun. The canonical verifier jointly checks issue obligations, fail-to-pass (F2P) and pass-to-pass (P2P) tests, with 30/106 cells satisfying this combined endpoint.

Cohort 2 records only the joint verifier reward, so the 76 reward-zero cells cannot be decomposed post hoc into separate issue-obligation, F2P, or P2P failures.

The 33/62 and 30/106 verifier-passing cells are nested within the 24 and 64 frozen repository-task units. They report cell-level outcomes; the full-cohort counts of verifier-valid switched edits remain 5/24 and 5/64.

Figure 3 visualizes these cohort funnels in the main text.

D RULE–SOURCE COMPOSITION AND STRUCTURAL GENERALIZATION

This appendix gives the source programs, donor pairings and complete outcomes for the content experiments in Section 3. We first test rule and source effects separately, then combine them and transfer the intervention procedure to new programs, rules and code structures. The final studies test reuse of fixed donor states across recipients and selectivity when donor backgrounds change. Appendix A defines the shared controls and evaluation units.

G17 denotes full-vector replacement at instruction positions entering layer 17; O18 denotes replacement at old-code positions entering layer 18. Their selection is motivated by the source-entry and restoration comparisons in Appendix E. All experiments below keep the visible recipient prompt and supplied Edit/old prefix fixed within an intervention comparison.

D.1 SEPARATING RULE AND SOURCE EFFECTS

This study tests whether each site's effect depends on which factors differ between donor and recipient. Table 3 summarizes the 4 arms. The subsequent joint experiment uses donors matched to the recipient on one factor to test composition.

D.1.1 SOURCES, RULES AND DONORS

Source S_0

```
def adjust(amount):
    offset = amount + 2
    if offset < 0:
        return offset - 3
    return offset + 4
```

Source S_1

```
def adjust(count):
    shifted = count - 2
    if shifted < 0:
        return shifted + 3
    return shifted - 4
```

The initial source-by-rule experiment uses these 2 functions and the distractor `def untouched(z): return z * 11`. The 2 instruction strings are *Increase the returned value by exactly seven units* and *Reduce the returned value by exactly seven units*. Rule $r = 0$ replaces every return expression e by $e + 7$; rule $r = 1$ replaces it by $e - 7$. AST construction gives 4 targets T_{sr}. Independent numeric oracles evaluate the original source plus or minus 7 on every integer in $[-32, 32]$. The 4 complete target profiles are distinct. An output must also pass separate variable and normalized structure checks; semantic profile agreement alone does not imply source preservation or general program equivalence.

The 4 worlds X_{sr} cross source S_s with rule r, distinct from the earlier W_{so} source–old worlds. In each world, the visible source and supplied old both contain S_s. All 4 serve as recipients and donor baselines. The forced prefix specifies Edit, `calc.py`, the complete old argument and the new-string opener. The model generates the replacement content; action choice and old localization are fixed.

Entry	Layer input / half-open positions	Donor for X_{sr}	Changed factors
G	B17 / $[512, 520)$	$X_{1-s,1-r}$	source and rule
O (rule)	B18 / $[618, 653)$	$X_{s,1-r}$	rule only
O (source)	B18 / $[618, 653)$	$X_{1-s,r}$	source only
O (both)	B18 / $[618, 653)$	$X_{1-s,1-r}$	source and rule

B17 and B18 are inputs to zero-indexed decoder layers 17 and 18. Each replacement copies full hidden vectors at aligned token positions without interpolation. G replaces the 8-token instruction span during prefill; O replaces the old span during 35 forced single-token forwards. Donors come from unmodified baselines in the same model instance. Here the G donor changes both source and rule, so its result tests rule transfer under both donor changes. This study includes no joint G+O intervention, same-rule G control or further layer search.

D.1.2 EXECUTION AND CONTROLS

Each call begins with an empty cache and one 618-token prefill, followed by 45 one-token forwards at positions 618–662. The final forced forward supplies the first free token. Greedy argmax continues for at most 384 generated tokens; EOS IDs 248044/248046 are retained and not fed back. Thus each

Entry	Donor changes	Output source	Output rule	Donor-new equal
Rule / B17	source + rule	recipient 4/4	donor 4/4	0/4
Old / B18	rule	shared 4/4	recipient 4/4	0/4
Old / B18	source	donor 4/4	shared 4/4	3/4
Old / B18	source + rule	donor 4/4	recipient 3/4; none 1/4	0/4

Table 3: **Source-by-rule intervention outcomes.** Each row has 4 fixed recipient worlds, not 4 independent population replications. Source denotes variable and normalized structure identity; rule denotes the applied program's target semantics. Shared factors cannot identify transfer. "None" is an untransformed donor-source program. Donor-new equality is literal equality to the donor baseline output, not evidence of an internal copy operation.

call has at most $46 + 383 = 429$ forwards. The 20 validation and 16 science calls give a fixed cap of 15,444 forwards. Actual attempt/completion records report 1,895 validation plus 1,505 science forwards, totaling 3,400; all 36 calls completed without failure files.

The execution settings follow Appendix A. The run took approximately 6.27 hours, including evidence collection and persistence.

Validation contains 4 hook-off references, 4 instrumented baselines, 4 independent repeats, 4 G self replacements and 4 O self replacements. All 20 pass before science. Reference/baseline and repeat/self checks compare complete generated token sequences and complete float32 log-probability tensors at boundaries 653 and 663. They also compare all 192 boundary cache slots, including expected empty slots. Corresponding values are compared exactly. Each baseline must yield its own source/rule target, one complete Edit, an exact applicable old, recipient variables/structure and the unchanged distractor. Self checks exercise the actual replacement code path. Hook-off references have no internal event evidence; the shared outer ledger and boundary collection are outside the observer non-perturbation contrast.

State checks preserve non-patched residual rows and compare unaffected lower layers with the recipient baseline (Appendix A.3). G protects layers below 17; O protects layers below 18 and the complete pre-old cache. Snapshots are collected at cut 618 and boundaries 653/663.

D.1.3 Scoring and Results

We parse the full response and score Edit format, fields and file, exact old, unique applicability, distractor preservation, source features and applied semantics on the 65-value domain. The structure check normalizes variable names and masks return expressions before comparing assignments and branches; variable identity is scored separately. Donor-new equality is a literal comparison with the decoded donor baseline, not a test of the internal computation that produced the string.

The primary portable-rule endpoint for G, O (rule), and O (both) requires $T_{s,1-r}$ while preserving recipient source invariants. The same-rule O (source) condition has no rule-flip primary endpoint: it is reported as N/A, not a zero-success estimate. We retain all 4 worlds in each condition, including the output that drops the rule. The 16 conditions are repeated comparisons on 2 constructed sources and 2 rules, not independent samples from a population; no population-level confidence interval is inferred from these counts.

Entry / change	Recipient	Donor	Applied target	Recipient invariants	Donor-new equal
G / both	X_{00}	X_{11}	T_{01}	pass	no
G / both	X_{01}	X_{10}	T_{00}	pass	no
G / both	X_{10}	X_{01}	T_{11}	pass	no
G / both	X_{11}	X_{00}	T_{10}	pass	no
O / rule	X_{00}	X_{01}	T_{00}	pass	no
O / rule	X_{01}	X_{00}	T_{01}	pass	no
O / rule	X_{10}	X_{11}	T_{10}	pass	no
O / rule	X_{11}	X_{10}	T_{11}	pass	no
O / source	X_{00}	X_{10}	T_{10}	fail	yes
O / source	X_{01}	X_{11}	T_{11}	fail	no
O / source	X_{10}	X_{00}	T_{00}	fail	yes
O / source	X_{11}	X_{01}	T_{01}	fail	yes
O / both	X_{00}	X_{11}	base S_1	fail	no
O / both	X_{01}	X_{10}	T_{11}	fail	no
O / both	X_{10}	X_{01}	T_{00}	fail	no
O / both	X_{11}	X_{00}	T_{01}	fail	no

All 16 outputs contain one complete Edit with exact forced recipient old, a unique applicable anchor and a preserved distractor. These old-string results validate the supplied prefix; they do not measure natural copying or addressing. The sole "base S_1" output is an untransformed donor-source function, classified as other semantics. Algebraic inspection agrees with the bounded scorer on these affine branches.

D.1.4 Observed Code Example

For recipient X_{00}, the visible source is S_0 with `amount`/`offset`, and the instruction adds 7. G receives a donor from X_{11}, with source S_1 and the subtract-7 rule. The measured G output below preserves the recipient code organization and applies the donor rule. O (both) uses the same donor world at a different entry and instead produces the untransformed S_1. Both continue from the exact forced S_0 old argument. The listings are actual decoded `new_string` values, not synthetic target examples.

G output: recipient source, donor rule

```
def adjust(amount):
    offset = amount + 2
    if offset < 0:
        return offset - 3 - 7
    return offset + 4 - 7
```

O (both): donor source, no transformation

```
def adjust(count):
    shifted = count - 2
    if shifted < 0:
        return shifted + 3
    return shifted - 4
```

The G output retains recipient source features and applies the donor rule. The O output uses the donor source but applies neither editing rule. These different effects motivate testing G and O together with factor-matched donors (Appendix D.2).

D.2 Joint Factorial Composition

D.2.1 Design and Donors

The single-site experiments establish separate rule and source effects. Applying G17 and O18 in the same run tests whether these effects compose. The 4 X_{sr} worlds retain sources S_0, S_1, add-7 and subtract-7 rules, the distractor and the 65-input oracle. Each source is displayed in the prompt and supplied as old. All 4 targets T_{sr} have unmodified baseline runs collected before intervention.

For recipient X_{sr}, G17 receives states from $X_{s,1-r}$ and O18 from $X_{1-s,r}$. The respective targets are $T_{s,1-r}$, $T_{1-s,r}$ and, for joint replacement, $T_{1-s,1-r}$. The joint target differs from both selected donor targets. Unlike the preceding study, the G17 donor now differs only in rule; its single-arm condition checks whether this donor contrast produces the intended effect. Every arm retains all 4 worlds.

G17 is the full 8-token rule span at zero-based decoder layer 17 input, positions $[512, 520)$, replaced during the initial 618-token prefill. O18 is the full 35-token old span at layer 18 input, positions $[618, 653)$, replaced during successive forced one-token steps. B17/B18 refer to the 18th/19th decoder blocks. Every replacement copies complete hidden vectors at aligned positions; no low-rank operator,

interpolation, learned direction or additional layer search is used. Visible instruction and old text stay fixed.

Both donors are unmodified baselines from the same model instance. G17 first changes the recipient prefill. At each old token, the model computes the layer-17 output, which must equal the layer-18 input before replacement. We then substitute the aligned O18 donor vector, and layer 18 processes that vector. O18 donor states are collected independently of the recipient's G17 intervention. Records bind the native output, pre-replacement input, donor vector and post-replacement input to the same call, model load, cache owner and token position. Checks verify their order and actual downstream consumption.

D.2.2 Execution and Controls

Each call starts with an empty cache, one 618-token prefill, then 45 single-token forwards at positions 618–662. The position-662 forward supplies the first free-generation token. Greedy argmax continues for at most 384 generated tokens; the first EOS in the frozen set $\{248044, 248046\}$ is retained and never fed back. A call uses $46 + (n - 1)$ forwards for n generated tokens, at most 429. No free-generation hidden state is replaced.

The registry contains 4 hook-off references, 4 instrumented baselines, 4 fresh repeats, and 4 self replacements for each of G17, O18 and joint: 24 validation calls. All must pass before the 12 scientific conditions. The respective forward caps are 10,296 and 5,148, totaling 15,444. Actual ledger counts are 2,274 validation and 1,132 science forwards, totaling 3,406, with no incomplete or failed attempt. The single formal run took 24,261.10 seconds (approximately 6.74 hours), including evidence collection, persistence and analysis. This is not pure inference time or total project compute.

This experiment uses a fresh model instance and donor collection under the shared execution settings in Appendix A.

Reference/baseline and repeat/self comparisons require identical full generated token sequences, complete float32 log-probability tensors at positions 653 and 663, and all cache values at those boundaries. Repeat/self additionally require all 192 cut-618 values to equal the recipient baseline. Self replacements exercise the actual G, O and double-hook paths. All validation outputs must be single complete Edits for their own world, with exact applicable old, source invariants and preserved distractor. Hook-off references omit internal observations; shared outer bookkeeping and cache collection are outside that contrast.

Arm	Exact recipient layers at cut 618	Exact layers at 653 and 663
Baseline/repeat	0–47	0–47
G17 self	0–47	0–47
O18 self	0–47	0–47
Joint self	0–47	0–47
G17 only	0–16	0–16
O18 only	0–47	0–17
Joint	0–16	0–16

All slots preserve occupancy, shape, dtype, device and boundary identity; permitted changes concern values only. Reference/baseline uses common observed boundaries. For baseline rows, an exact recipient comparison denotes self-consistency rather than a second independent baseline. Donors are immutable value copies; calls use fresh cache owners. Joint pre-old cache values above layer 16 may differ after G17 replacement and are not incorrectly required to match the recipient baseline. Lower-layer residual observations are also compared to the independently retained recipient baseline.

The state checks in Appendix A.3 apply throughout the forced prefix. Scientific calls are checked individually for intervention placement and preservation of non-target states.

D.2.3 Results and Donor Equality

The joint primary endpoint requires all 10 checks to pass. The output must contain one complete Edit under the forced prefix, with exactly the registered fields and `calc.py`. It must retain exact recipient old, a unique applicable anchor and the distractor. The applied program must match target

$T_{1-s,1-r}$, the O donor source's variables and that source's structure. Its new string must differ from the actual G donor new and from the actual O donor new. Single-entry arms use their own registered target/source capability endpoint; joint success is N/A for them. All 12 scientific conditions remain in the denominator.

Arm	Recipient	G donor	O donor	Applied target	Equal G	Equal O
G17	X_{00}	X_{01}	–	T_{01}	no	N/A
G17	X_{01}	X_{00}	–	T_{00}	yes	N/A
G17	X_{10}	X_{11}	–	T_{11}	yes	N/A
G17	X_{11}	X_{10}	–	T_{10}	yes	N/A
O18	X_{00}	–	X_{10}	T_{10}	N/A	yes
O18	X_{01}	–	X_{11}	T_{11}	N/A	no
O18	X_{10}	–	X_{00}	T_{00}	N/A	yes
O18	X_{11}	–	X_{01}	T_{01}	N/A	yes
Joint	X_{00}	X_{01}	X_{10}	T_{11}	no	no
Joint	X_{01}	X_{00}	X_{11}	T_{10}	no	no
Joint	X_{10}	X_{11}	X_{00}	T_{01}	no	no
Joint	X_{11}	X_{10}	X_{01}	T_{00}	no	no

Every row passes the common Edit/old/applicability/distractor terms and its expected source-feature checks. All 4 joint rows pass the full 10-part endpoint. The 8 O18/joint rows change recipient variables and structure despite being applicable. Each target classification agrees on all 65 integers in $[-32, 32]$; these inputs are an evaluation domain, not independent intervention replications. The 4 constructed worlds per arm do not support population-level uncertainty estimates or broad generalization.

G17-only at X_{00} writes `offset - 3 - 7` and `offset + 4 - 7`, whereas its donor baseline X_{01} writes `offset - 10` and `offset - 3`. O18-only at X_{01} writes `shifted - 4` and `shifted - 11`, whereas its donor X_{11} writes `shifted + 3 - 7` and `shifted - 4 - 7`. These equivalent rewrites explain the 2 single-factor non-equality cases.

Joint outputs for recipients X_{01}, X_{10}, X_{11} also equal the complete new strings of the unselected target-world baselines X_{10}, X_{01}, X_{00}, respectively. The X_{00} joint output is an equivalent rewrite of target-world baseline X_{11}. Thus no selected-donor equality is observed, but neither novel strings nor unseen combinations are required by the endpoint. Literal equality alone does not diagnose a neural copy operation.

D.2.4 Observed Example: Separate and Joint Effects

Recipient X_{00} displays S_0, using `amount`/`offset`, and asks to add 7. G17 receives rule-position states from X_{01} (the same source, subtract 7). O18 receives old-position states from X_{10} (source S_1, add 7). The listings below are the actual decoded new strings of the baseline and the 3 scientific calls. All 4 continue from the same forced S_0 old argument. The joint output uses `count`/`shifted` and subtracts 7 from S_1's returns.

Baseline: source A, add 7

```
def adjust(amount):
    offset = amount + 2
    if offset < 0:
        return offset - 3 + 7
    return offset + 4 + 7
```

G17 only: source A, subtract 7

```
def adjust(amount):
    offset = amount + 2
    if offset < 0:
        return offset - 3 - 7
    return offset + 4 - 7
```

O18 only: source B, add 7

```
def adjust(count):
    shifted = count - 2
    if shifted < 0:
        return shifted + 3 + 7
    return shifted - 4 + 7
```

Joint: source B, subtract 7

```
def adjust(count):
    shifted = count - 2
    if shifted < 0:
        return shifted - 4
    return shifted - 11
```

D.3 Transfer to New Source Functions and Rules

D.3.1 Design and Evaluation Domain

This follow-up changes both source functions and both rules while retaining the G17/O18 sites, full-vector replacement and donor contrasts from Appendix D.2. The design follows the initial results, but these programs and rules were not used to select the sites. The 4 worlds X'_{sr} cross the functions below with multiplication by 3 ($r = 0$) or squaring ($r = 1$) of every returned value. The exact instructions are *Multiply every returned value by exactly three here* and *Replace every returned value by its square only*.

New source S'_0

```
def compute(amount):
    total = amount * 2
    if total > 5:
        return total - 1
    return total + 6
```

New source S'_1

```
def compute(index):
    result = index * 3
    if result <= 4:
        return result + 2
    return result - 5
```

The file also contains `untouched(z)`, returning `z * 11`, which must remain unchanged. Each prompt presents its source and supplies the same complete function, including its final newline, as old. Action choice and old localization remain fixed. Targets T'_{sr} apply the registered rule to every source return expression, preserving variable names, assignments and branch structure. Evaluation uses all 65 integers in $[-32, 32]$; these are semantic test inputs, not independent model trials.

For recipient X'_{sr}, G17 receives states from $X'_{s,1-r}$ and O18 from $X'_{1-s,r}$. Joint replacement targets $T'_{1-s,1-r}$. All 4 target combinations have same-run unmodified baselines, so the holdout concerns source and rule factors relative to site development, not an unseen combination. The visible context still contains both the original-source occurrence and serialized old.

D.3.2 Execution and Results

The 4 contexts have 663 tokens. Original source occupies $[440, 475)$, G17 the 8 instruction tokens $[512, 520)$ at layer 17 input, and O18 the 35 old tokens $[618, 653)$ at layer 18 input. Each call starts fresh with a 618-token prefill and 45 forced single-token forwards at positions 618–662. Free generation is greedy and unpatched, capped at 384 tokens. The final EOS is saved and not fed back; a call with n generated tokens uses $45 + n$ forwards, at most 429.

The model, 8-H20 device setup, bfloat16 computation and resolved grouped-mm/Torch DeltaNet backend match the initial composition study. A new manifest, model instance, donor collection and ledger bind this study separately. The 24 validation calls comprise 4 hook-off references, 4 instrumented baselines, 4 repeats and 4 selves for each of G17, O18 and joint. Observer, same-input self, cache-lineage and runtime identity requirements follow Appendix D.2. No prior instance's gate or donor is reused.

All 24 validation calls pass before the 12 science calls. Actual counts are 2,400 validation plus 1,200 science forwards, totaling 3,600 against the 15,444 cap. Each of the 36 calls produces 55 tokens including EOS and uses 100 forwards. The single formal run took 24,210.70 seconds (approximately 6.73 hours), including evidence collection, persistence and analysis. The ledger contains no incomplete or failed attempt.

Arm	Recipient	G donor	O donor	Applied target	Equal G	Equal O
G17	X'_{00}	X'_{01}	–	T'_{01}	yes	N/A
G17	X'_{01}	X'_{00}	–	T'_{00}	yes	N/A
G17	X'_{10}	X'_{11}	–	T'_{11}	yes	N/A
G17	X'_{11}	X'_{10}	–	T'_{10}	yes	N/A
O18	X'_{00}	–	X'_{10}	T'_{10}	N/A	yes
O18	X'_{01}	–	X'_{11}	T'_{11}	N/A	yes
O18	X'_{10}	–	X'_{00}	T'_{00}	N/A	yes
O18	X'_{11}	–	X'_{01}	T'_{01}	N/A	yes
Joint	X'_{00}	X'_{01}	X'_{10}	T'_{11}	no	no
Joint	X'_{01}	X'_{00}	X'_{11}	T'_{10}	no	no
Joint	X'_{10}	X'_{11}	X'_{00}	T'_{01}	no	no
Joint	X'_{11}	X'_{10}	X'_{01}	T'_{00}	no	no

Every scientific row passes the complete Edit envelope, exact forced old, unique application, distractor preservation and expected source-variable/structure checks. Each target matches all 65 test inputs. All 4 joint rows satisfy the 10-part endpoint, including inequality to both selected donors' complete new strings. Both single-entry arms equal their respective selected donor strings in all 4/4 cases. All 4 joint strings equal the corresponding unselected target-world baseline. Literal comparisons therefore distinguish the selected donor answers from the combined target, without identifying an internal copying or recomputation mechanism.

D.3.3 Actual Generated Example

For recipient X'_{00}, the rule donor is X'_{01} (source S'_0, square) and the source donor is X'_{10} (source S'_1, multiply by 3). These are actual decoded new strings; the 4 calls share the same visible recipient prefix and old argument.

Baseline: new A, multiply by 3

```
def compute(amount):
    total = amount * 2
    if total > 5:
        return (total - 1) * 3
    return (total + 6) * 3
```

G17 only: new A, square

```
def compute(amount):
    total = amount * 2
    if total > 5:
        return (total - 1) ** 2
    return (total + 6) ** 2
```

O18 only: new B, multiply by 3

```
def compute(index):
    result = index * 3
    if result <= 4:
        return (result + 2) * 3
    return (result - 5) * 3
```

Joint: new B, square

```
def compute(index):
    result = index * 3
    if result <= 4:
        return (result + 2) ** 2
    return (result - 5) ** 2
```

D.4 Replication Across Code Structures

D.4.1 Design

The extension freezes 4 structure families before model execution: straight-line arithmetic, multiple returns, nested branches and a dataflow chain. Each family crosses 2 sources with 2 rules, multiplying every returned value by 3 or squaring it. G17 receives the same-source/opposite-rule donor; O18 receives the other-source/same-rule donor. The 12 science calls comprise all 4 worlds in each of rule-only, source-only and joint arms. Each block has its own fresh model instance, 4 hook-off references, 4 instrumented baselines, 4 repeats and 12 self replacements, giving 24 validation calls before science. Model states and runtime gates do not cross block boundaries.

The 4 blocks use 144 calls and 14,328 forwards, within the frozen cap of 144 calls and 61,740 forwards. Independent review checks outputs and ledgers for every block; the results below include all 4 structures.

D.4.2 Sources and Semantic Evaluation

The 2 exact source functions in each reviewed structure are:

Straight-line source A

```
def compute(amount):
    total = amount * 4
    return total + 7
```

Straight-line source B

```
def compute(index):
    result = index * 5
    return result - 6
```

Multiple-return source A

```
def compute(amount):
    if amount < 2:
        return amount - 3
    if amount > 8:
        return amount + 5
    return amount * 4
```

Multiple-return source B

```
def compute(index):
    if index < 3:
        return index - 4
    if index > 9:
        return index + 6
    return index * 5
```

Nested source A

```
def compute(amount):
    if amount > 2:
        if amount < 8:
            return amount * 4
        return amount - 3
    return amount + 5
```

Nested source B

```
def compute(index):
    if index > 3:
        if index < 9:
            return index * 5
        return index - 4
    return index + 6
```

Dataflow source A

```
def compute(amount):
    total = amount * 4
    offset = total + 7
    scaled = offset * 2
    return scaled - total
```

Dataflow source B

```
def compute(index):
    result = index * 5
    shifted = result - 6
    adjusted = shifted * 3
    return adjusted - result
```

Every file also contains the unrelated function `untouched(z)`, which returns `z * 11`. The displayed source and supplied old are the same complete function, including its final newline. The other source and complete target answers are absent from each recipient prompt. All 4 target worlds nevertheless have separate unmodified baseline runs; this is not an unseen-combination test.

The oracle uses the 65 integers in $[-32, 32]$ plus $-7/2$, $-7/4$, $6/5$ and $9/5$, for 69 distinct inputs. Evaluation uses exact rational arithmetic and a bounded AST interpreter. The extra values include zeros of return expressions that multiplication and squaring treat differently nearby. Target behaviors are distinguishable on this domain. Source-variable and statement-structure checks are separate from return semantics, exact old, unique applicability, complete Edit format, preservation of the unrelated function and equality to selected donor new strings. Finite-domain agreement does not prove equivalence on every possible input.

D.4.3 Intervention Spans and Execution Counts

All G spans contain 8 tokens at layer 17 input. O18 covers the full old function at layer 18 input; lengths and absolute positions vary by structure according to the fixed span-selection procedure. The intervals below are half-open.

Structure	Context tokens	G span	OLD span	Forced steps
Straight-line	633	$[497, 505)$	$[603, 623)$	30
Multiple returns	677	$[519, 527)$	$[625, 667)$	52
Nested branches	677	$[519, 527)$	$[625, 667)$	52
Dataflow chain	663	$[512, 520)$	$[618, 653)$	45

Each call starts with an empty cache and prefills to just before old, then processes the forced steps individually. The last forced forward supplies the first generated token. Greedy generation is capped at 384 tokens; EOS is retained and not fed back. For k forced steps and n actual generated tokens including EOS, a call uses $k + n$ forwards.

Structure	Validation calls	Science calls	Val. forwards	Sci. forwards	Tokens/call
Straight-line	24	12	1,560	780	35
Multiple returns	24	12	2,856	1,428	67
Nested branches	24	12	2,856	1,428	67
Dataflow chain	24	12	2,280	1,140	50
Total	96	48	9,552	4,776	—

The 4 blocks use 2,340, 4,284, 4,284 and 3,420 forwards. All calls terminate with EOS; final ledgers contain no partial attempt or failure. Independent outcome reviews are separate records and preserve the original execution terminals.

D.4.4 Results

For each of the 4 structures, every world $00, 01, 10, 11$ passes the registered target/source checks in all 3 arms. Rule-only gives $T_{s,1-r}$; source-only gives $T_{1-s,r}$; joint gives $T_{1-s,1-r}$. Each single-arm output equals its selected donor new string in 4/4 cases. Joint outputs equal neither selected donor new in 4/4 cases. All 12 scientific outputs per block have exact old, a uniquely applicable anchor, expected variables/structure, complete Edit format and an unchanged unrelated function. The 8 source-only/joint outputs per block intentionally change recipient source features.

For each arm, we compute success over the 4 worlds within each family and then average the 4 family rates with equal weight. This descriptive mean is 100% for rule-only, source-only and joint, with 4/4 success in every constituent family. These are deterministic results for constructed worlds, not independent random samples of a program population. The arms share baselines within a block; the 69 oracle inputs evaluate each program and do not add model trials. Full call-level scores, decoded outputs and the fixed source/target definitions are retained in the reproduction records.

D.5 Reusing Fixed Donor States Across Recipients

D.5.1 Sources, Rules and Donor Selection

We now test whether the same captured states retain their effects when applied to different recipients. The experiment crosses 3 source functions, A, B and D, with multiplication by 3 and squaring:

Source A

```
def compute(x):
    a = x * 2
    b = a + 3
    c = b * 4
    return c - a
```

Source B

```
def compute(y):
    a = y - 40
    b = a * a
    c = b + 7
    return c + a
```

Source D

```
def compute(z):
    a = z * 30
    b = z + 2
    c = a * b
    return c - b
```

In A and B, the assignments form the dependency chain `input` $\rightarrow$ `a` $\rightarrow$ `b` $\rightarrow$ `c`; the return also reads `a`. In D, `a` and `b` both read the input, `c` reads both of them, and the return reads `c` and `b`. All functions have the same statement layout, so transferring D tests a change in data dependencies within that layout.

We capture the 8 G17 instruction vectors once from A/square and the 35 O18 old-code vectors once from D/multiply. These same payloads are applied separately and jointly to A/multiply and B/multiply, without recapture or adaptation. All donors are unmodified baselines in the same model instance, and each recipient keeps its own visible prompt and supplied Edit/old prefix. On B/multiply, a same-rule control uses G17 from A/multiply, while a same-source control uses O18 from B/square.

D.5.2 Execution and Controls

The 6 source/rule worlds each receive a hook-off reference, an instrumented baseline, a repeat, a G17 self replacement, an O18 self replacement and a joint self replacement. All 36 validation calls pass before the 8 scientific calls. Every target world has its own baseline, including D/square; the joint target is distinct from the selected donors but is a known baseline target.

All contexts contain 663 tokens through the new-string opener. The half-open G span is $[512, 520)$ and the old span is $[618, 653)$. Each call prefills 618 tokens with a fresh cache, then processes 45

forced single-token steps; the last forced forward supplies the first generated token. Greedy generation yields 50 tokens including EOS in every call, for 95 forwards per call. Validation uses 3,420 forwards and science uses 760, totaling 4,180, below the registered cap of 18,876. No call ends in a partial attempt or failure. The run and its in-run checks take approximately 11.6 hours on 8 H20 GPUs.

Control checks. The hook-off, repeat and self-replacement controls pass for all 6 source/rule worlds. During execution, an independent process rechecks each call's saved cache and residual states and recomputes its score; all 44 calls pass. A subsequent review reproduces the scores and token sequences and verifies the recorded intervention schedule. Detailed verification records accompany the reproducibility materials.

D.5.3 RESULTS

Table 4 reports all 8 conditions. Both joint outputs are exactly:

```
def compute(z):
    a = z * 30
    b = z + 2
    c = a * b
    return (c - b) ** 2
```

Recipient	G17 donor	O18 donor	Generated target	Equals selected donor new
A/multiply	A/square	None	A/square	G: yes
A/multiply	None	D/multiply	D/multiply	O: yes
A/multiply	A/square	D/multiply	D/square	G: no; O: no
B/multiply	A/square	None	B/square	G: no
B/multiply	None	D/multiply	D/multiply	O: yes
B/multiply	A/square	D/multiply	D/square	G: no; O: no
B/multiply	A/multiply	None	B/multiply	G: no
B/multiply	None	B/square	B/multiply	O: no

Table 4: **Fixed donor states retain their effects across recipients.** All rows pass the registered target and source checks. The last 2 rows are controls; G and O denote the selected rule and source donors.

Every condition passes target semantics on the 65 integers in $[-32, 32]$ plus $-7/2$, $-1/2$, $1/2$ and $7/2$, evaluated with exact rational arithmetic in a bounded AST interpreter. We separately check variable names, original arithmetic, definition/use dependencies and statement layout after removing the registered return-value transformation. Recognized rule wrappers are `e*3`, `3*e`, `e**2` and `e*e`. Exact parsed old, unique applicability, complete Edit format and preservation of `untouched(z)` also pass in all 8 conditions. These checks evaluate 8 generated programs; the 69 input values do not add model trials.

Rule-only, source-only and joint interventions each succeed in 2/2 recipients. Both controls retain B/multiply. Source-only outputs equal the complete D/multiply donor replacement; both joint outputs instead apply squaring to D. These results support reuse of the fixed states across the 2 recipients, including transfer of D's dependency graph within the shared statement layout. Edit and old remain supplied.

D.6 SELECTIVITY ACROSS DONOR BACKGROUNDS

D.6.1 DESIGN AND SOURCE PROGRAMS

Finally, we vary each donor's unselected factor while keeping recipient A/multiply fixed. Rule donors differ in source, and source donors differ in rule. The selected worlds use 3 straight-line sources and 3 rules: multiplication by 3, squaring and addition of 7. The sources are:

Source A

```
def compute(amount):
    total = amount * 4
    return total + 7
```

Source B

```
def compute(index):
    result = index * 5
    return result - 6
```

Source C

```
def compute(score):
    scaled = score * 6
    return scaled + 2
```

G17 takes states from either A/square or C/square: both donors specify squaring but differ in source. O18 takes states from either B/multiply or B/add-7: both donors contain source B but differ in rule. We test each of the 4 single interventions and all 4 pairings. The visible recipient prompt and supplied Edit/old prefix stay fixed. Layers, full-vector replacement and span-selection rules are unchanged from the straight-line composition experiment.

D.6.2 Execution and Controls

The 6 selected worlds are A/multiply, A/square, C/square, B/multiply, B/add-7 and B/square. Each receives a hook-off reference, an instrumented baseline, a repeat and G17, O18 and joint self replacements, giving 36 validation calls before the 8 scientific calls. These 6 worlds include an unmodified baseline for every scientific target, including B/square.

Contexts contain 633 tokens through the new-string opener. The half-open G17 span is $[497, 505)$ and the O18 span is $[603, 623)$. Each call uses a fresh 603-token prefill followed by 30 forced single-token steps. The last forced forward supplies the first generated token. All calls terminate with EOS. The experiment uses 2,848 forwards: 2,328 for validation and 520 for the scientific conditions.

Control checks. We compare instrumented baselines with 6 hook-off references to check the effect of observation hooks, and use 24 repeat and self-replacement comparisons to check repeatability and intervention identity. All 30 comparisons pass when evaluated on the saved trajectories after generation. Independent reanalysis also confirms the scores, token sequences and cache/state checks for all 44 calls. Implementation and verification records accompany the reproducibility materials.

D.6.3 Results

G17 alone produces A/square for both rule donors, and O18 alone produces B/multiply for both source donors (Table 5). In particular, C/square states do not import C's source features through G17, and B/add-7 states do not import addition through O18. All 4 joint conditions yield exactly:

```
def compute(index):
    result = index * 5
    return (result - 6) ** 2
```

G17 donor	O18 donor	Generated target	Equals selected donor new
A/square	None	A/square	G: yes
C/square	None	A/square	G: no
None	B/multiply	B/multiply	O: yes
None	B/add-7	B/multiply	O: no
A/square	B/multiply	B/square	G: no; O: no
A/square	B/add-7	B/square	G: no; O: no
C/square	B/multiply	B/square	G: no; O: no
C/square	B/add-7	B/square	G: no; O: no

Table 5: **Rule and source effects persist across donor backgrounds.** The recipient is A/multiply in every row. All 8 conditions pass target semantics and expected source features. Joint outputs differ from both selected donors' complete replacements.

Scoring uses the 65 integers in $[-32, 32]$ and the rational inputs $-7/2$, $-7/4$, $6/5$ and $9/5$. A bounded AST interpreter evaluates exact rational outputs against the source-specific rule oracle. Variable names and statement structure are scored separately. All 8 generated edits also have exact parsed old, a unique applicable anchor and an unchanged distractor function. The 69 inputs evaluate each of the 8 generated programs. Across these donor choices, G17 transfers the square rule without changing recipient source features, and O18 transfers source B while preserving the multiplication rule.

E Intervention Sites and Restoration of the Source Effect

We compare intervention sites and test whether downstream restoration interrupts their source effects. First, we contrast visible-source and serialized `old`-code positions, then compare residual and cache restoration to motivate O18. The final studies extend Section 4. Full residual restoration tests whether complete programs recover recipient source features while retaining the changed rule. Old-code key/value substitution at one layer-19 attention read measures next-token preferences.

E.1 Contrasting Source-Code and Old-Code Intervention Sites

E.1.1 Sources, Targets, and Prompt Contexts

This study uses Qwen3.5-122B-A10B at the input to zero-based decoder layer 17 (B17), with its own task population, donor captures and validation records. The instruction is *Increase the returned value by exactly seven units*. Each file contains one function from Appendix D.1, a blank line and the distractor `def untouched(z): return z * 7`. This study uses a multiply-by-7 distractor; the later composition studies use multiply-by-11. The target edit changes only return expressions. It preserves argument names, local variables, assignments, branches and statement order.

An AST transformation maps each return expression e to $e + 7$. Target T_0, for example, returns `offset` $-\,3+7$ in its negative branch and `offset` $+\,4+7$ otherwise, retaining parameter `amount` and variable `offset`. Target T_1 retains `count` and `shifted`. We check both targets against an independent original-function oracle plus 7 on all 65 integers in $[-32, 32]$; their outputs differ at every tested input. These execution tests complement the structural checks and establish agreement on the specified domain.

Context W_{so} displays source S_s in the file and supplies S_o in the forced old-string argument. The assistant prefix specifies `Edit`, file `calc.py`, old and the new-string opener; the model generates the replacement. Consistent worlds W_{00} and W_{11} serve as recipients. Conflicting worlds W_{01} and W_{10} serve as donor baselines: their generated replacements follow serialized old, although that old cannot be applied to the displayed file. This conflict lets us vary the two source occurrences independently.

For the W_{00} OLD intervention, W_{01} provides the B17 inputs at the old-string positions. Recipient tokens remain S_0, so the resulting exact old anchor still applies to the S_0 file. The generated replacement instead matches T_1 on all 65 inputs and violates the recipient variable/structure invariants. The reciprocal intervention behaves symmetrically.

E.1.2 Registered Spans and Donors

All 4 complete forced contexts have 662 tokens. Spans are exact half-open token intervals in the frozen tokenizer, including the registered framing newline where tokenization joins it to source text. No padding or interpolation is used. Donors are collected from unmodified baselines in the same model instance and same execution path.

Entry	Token interval	Length	Donor for recipient W_{ss}
SRC	$[440, 475)$	35	$W_{1-s,s}$
UP	$[610, 617)$	7	$W_{1-s,s}$
OLD	$[617, 652)$	35	$W_{s,1-s}$
MID (source)	$[652, 662)$	10	$W_{1-s,s}$
MID (old)	$[652, 662)$	10	$W_{s,1-s}$

We replace full B17 input vectors at the registered positions. SRC and UP are patched during prefill; OLD and MID are patched as their forced tokens are processed individually. Generation then proceeds without replacement. The source donor changes the visible source while preserving old; the old donor changes old while preserving the visible source. MID tests both donor definitions at the same positions. Because the entries differ in position, span length and subsequent computation, this comparison tests which interventions suffice to redirect the output.

E.1.3 Execution, Validation, and Readouts

Every call starts from an empty cache with one 617-token prefill followed by 45 single-token forwards at positions 617 through 661. The final forced forward supplies the first free token. Greedy argmax then continues with a 384-token cap and frozen EOS IDs 248044/248046; EOS is retained and not fed back. Thus a call requires at most $46 + 383 = 429$ forwards. The complete 38-call budget was at most 16,302 forwards. All actual calls produced 51 tokens in 96 forwards: 2,688 validation forwards followed by 960 science forwards, 3,648 total.

Validation consists of 4 hook-disabled references, 4 instrumented baselines, 4 independent repeats, and 16 self interventions (4 worlds crossed with SRC/UP/OLD/MID). All pass before science. Reference-to-baseline comparison checks complete generated suffixes, greedy identity and distribution TV at most 10^{-5}, and exact complete cache values at boundaries 617 and 662. References omit internal captures, so unobserved internal quantities are not claimed identical. The outer ledger and boundary copying shared by both arms are outside this non-perturbation comparison. Repeat/self checks additionally compare captured states; self interventions execute the actual replacement path.

Scoring parses the complete action and distinguishes exact old, unique application, target behavior on 65 inputs, recipient variables/structure, distractor preservation, source switch, literal donor-new equality, refusal, correction, partial and mixed outcomes. All 10 science calls retain their original denominator. Exact old and applicability here validate the forced-prefix construction; they are not evidence of natural old generation. The 2 reciprocal observations per entry are a case study, not a sample supporting population-level confidence intervals.

E.1.4 Complete Outcomes

Entry	Recipient	Donor	Applied target	Recipient invariants	Donor-new equal
SRC	W_{00}	W_{10}	T_0	pass	yes
SRC	W_{11}	W_{01}	T_1	pass	yes
UP	W_{00}	W_{10}	T_0	pass	yes
UP	W_{11}	W_{01}	T_1	pass	yes
OLD	W_{00}	W_{01}	T_1	fail	yes
OLD	W_{11}	W_{10}	T_0	fail	yes
MID (source)	W_{00}	W_{10}	T_0	pass	yes
MID (source)	W_{11}	W_{01}	T_1	pass	yes
MID (old)	W_{00}	W_{01}	T_0	pass	no
MID (old)	W_{11}	W_{10}	T_1	pass	no

Each generated program matches its listed target on all 65 inputs. OLD switches to the donor-source target in both directions; SRC, UP and both MID conditions retain the recipient target. All 10 calls retain exact, uniquely applicable old anchors and the distractor, with no observed corrections, refusals, mixed target labels or scoring failures. Donor-new equality occurs in 8 calls, including 6 whose outputs also equal the recipient target. Thus the source-switch evidence comes from the changed program features and semantics, rather than string equality alone.

E.2 Comparing Residual and Cache Restoration

We start from B17 OLD replacement and keep the fixed-rule worlds and 617+45 schedule. We compare 3 restoration conditions: recipient B18 inputs at each old token, recipient B17 cache states after old, and both. This study motivated O18 without searching layers on the later source-by-rule outcomes. Cache restoration follows positions 0–651 and precedes position 652. Only B17's 4 containers are restored; the other 188 slots, global position history and higher-layer writes made during old processing remain intact.

After 20 validation calls, 8 scientific calls cover both directions of all 4 conditions. OLD alone and OLD with cache restoration each produce the donor-source target in 2/2 directions. B18 input restoration, with or without cache restoration, recovers the recipient target, variables and structure in 2/2 directions per condition. All 8 retain exact applicable old anchors and the distractor. The 28 calls consume 2,688 forwards; offline replay verifies evidence, scores, schedules, gate and terminal bindings. Thus B18 input restoration interrupts the induced source shift, whereas B17 cache restoration at the specified time does not.

E.3 Restoring Downstream Residuals While Preserving the Patched Rule

E.3.1 Intervention and Reference Trajectory

Joint G17+O18 changes both rule and source. This restoration experiment tests whether the source shift can be reversed while retaining the changed rule. The 4 worlds X'_{sr} and targets T'_{sr} from Appendix D.3 cross 2 functions with multiplication by 3 and squaring. The reference is a matched G17-only trajectory, which applies the changed rule to the recipient source. The visible prompt, supplied Edit and old argument remain fixed.

G17 and O18 keep their baseline donors: $X'_{s,1-r}$ at instruction positions $[512, 520)$, and $X'_{1-s,r}$ at old positions $[618, 653)$. All layers are zero-based. Let $z^{G}_{18,t}$ be the complete decoder-layer-18 output at old position t in the same-world G17-only trajectory. Restoration F performs

$$z^{G+O+F}_{18,t} \leftarrow z^{G}_{18,t}, \qquad t \in [618, 653),$$

after layer 18's attention and MoE residual calculations and before layer 19 consumes the output. We apply F at each of the 35 old positions as its token is processed. The positions form one fixed intervention span; no layer or onset search is performed. The reference is the same-instance G17-only scientific trajectory.

F changes the outgoing residual while leaving layer 18's convolutional and recurrent cache states intact. Layer 19 and higher layers then compute from the restored residual and their existing caches. Lower-layer comparisons cover the 46 forced forwards with identical inputs. All hidden-state replacement stops before free generation.

E.3.2 Execution, Validation and Complete Outcomes

Each call uses a fresh 618-token prefill plus single-token inputs at positions 618–662, followed by greedy generation capped at 384 tokens including EOS. The model, bfloat16 execution and resolved backend follow Appendix D.3; the instance, manifest, ledger and donor collection are new. The 28 validation calls comprise 4 hook-off references, 4 baselines, 4 repeats, and 4 selves for each of G17, O18, joint and joint+F. F selves exercise the actual 3-replacement path with same-world baseline payloads. All 28 pass before science.

Science comprises 4 G17-only, 4 joint and 4 joint+F trajectories. Every call generates 55 tokens including EOS and consumes 100 forwards. The actual total is 2,800 validation plus 1,200 science forwards, or 4,000 against a frozen cap of 17,160. The formal run took 28,093.53 seconds (approximately 7.80 hours), including evidence collection and analysis. The ledger has no partial or failed attempt. Each experiment retains its own denominator; the G17-only and joint rows here are new same-instance measurements, not reused results from the transfer study.

Recipient	G17-only output	Joint output	Joint+F output	F equals G17-only
X'_{00}	T'_{01}	T'_{11}	T'_{01}	yes
X'_{01}	T'_{00}	T'_{10}	T'_{00}	yes
X'_{10}	T'_{11}	T'_{01}	T'_{11}	yes
X'_{11}	T'_{10}	T'_{00}	T'_{10}	yes

Each of these 12 outputs matches its listed target on all 65 integers in $[-32, 32]$, with the expected variables and source structure. The 65 inputs test program behavior; they are not independent neural trials. All 12 are complete single Edits with exact supplied old, a unique applicable anchor and a preserved distractor. No partial-source, mixed-source or wrong-rule scientific output is observed. All 4 joint+F new strings and entire generated token sequences equal their matched G17-only continuations; they differ from the joint continuations. F therefore restores $T'_{s,1-r}$, not the original-rule baseline T'_{sr}.

G17-only strings equal their selected G baseline donor strings in 4/4 cases. Joint strings equal neither selected baseline donor, but equal the unselected target-world baseline in 4/4. F strings equal the matched G17-only and G baseline strings. These are literal output equalities, not direct observations of a copying operation inside the model.

E.3.3 Generated Programs and State Verification

For recipient X'_{00}, the visible task asks to multiply source A's return values by 3. G17-only produces the left program; joint replacement produces the right. Joint+F produces exactly the left program again, including the full generated token sequence.

G17-only and joint+F: A, square

```
def compute(amount):
    total = amount * 2
    if total > 5:
        return (total - 1) ** 2
    return (total + 6) ** 2
```

Joint: B, square

```
def compute(index):
    result = index * 3
    if result <= 4:
        return (result + 2) ** 2
    return (result - 5) ** 2
```

The exported records contain all 140 scientific F residual replacements. Every before-vector equals the matched joint vector, every after-vector equals the matched G17-only vector, and every layer-19-consumed vector equals the after-vector. All 140 replacements are nonzero. The recorded event order is native layer-18 exit, F replacement, consumption, then layer-19 entry. Cache descriptors agree across all $140 \times 192 = 26{,}880$ external-transaction comparisons. Within the identical forced-input window, the joint/F lower-layer comparisons cover 13,984 slots, including 6,992 with nonempty value hashes.

Interpretation. Layer-18 output restoration recovers the recipient source while retaining the changed rule in all 4 conditions, with the layer-18 cache unchanged. This establishes a sufficient downstream restoration. Since F replaces full contextualized vectors from a trajectory carrying the changed rule, it does not isolate a source-only pathway.

E.4 Probing Source Recovery at One Attention Read

E.4.1 Attention Intervention and Reference Operands

We now test a narrower intervention than full residual restoration: substituting old-code keys and values at one attention read during replacement decoding. The same 2 `compute` functions and multiply-by-3/square rules use a new model instance, fresh references and fixed landmark prefixes. G17 replaces opposite-rule instruction states at positions $[512, 520)$ during the 618-token prefill; O18 replaces other-source old states at $[618, 653)$. Donors come from the 4 source/rule worlds.

For a joint trajectory, only the final registered attention invocation at zero-based layer 19 is modified. Let Q_J, K_J, V_J denote its native operands after rotary position encoding and cache update, and let K_G, V_G come from its matched G17-only trajectory with identical visible tokens. The read uses

$$Q = Q_J, \qquad (K,V)_{618:653} = (K_G, V_G)_{618:653}, \qquad (K,V)_{\neg[618:653)} = (K_J, V_J)_{\neg[618:653)}.$$

We splice temporary key/value clones before grouped-query expansion and native SDPA, preserving the query, mask, scaling, dropout, other native arguments and persistent cache. The intervention acts on one layer-19 attention invocation, not one head, and can change attention normalization. The remaining computation produces logits; no continuation is generated or predicted token fed back.

E.4.2 Landmarks, Controls and Execution

Each of 4 worlds has 3 independently initialized landmark calls. The source landmark consumes position 664 and scores position 665, contrasting `(amount` with `(index`. The rule landmark consumes 691 and scores 692, contrasting `**` with `*`; the preferred candidate follows the opposite rule. The scope landmark consumes 685 and scores 686, contrasting `(` with `total` or `result`. Rule and scope prefixes already specify recipient-source information. Scope is a lexical first-return-expression cue, not a full program-scope endpoint. These registered prefixes were fixed before the run, not selected from its outcomes.

The 24 seed controls comprise 4 hook-off references, 4 baselines, 4 repeats and 4 self controls for each of G17, O18 and joint replacement. Each of 12 landmark cases then has 7 validation calls: G17-only hook-off and capture; joint hook-off, capture and repeat; joint clone-only and self-read. This gives 108 validation calls before the 12 scientific joint-read-G calls. Controls compare complete distributions and terminal cache values; read checks additionally bind native operands, full prestates,

temporary-cache invariance, arguments and donor lineage. Source, rule and scope calls use 48, 75 and 69 forwards respectively. All controls and science calls belong to one model-load identity.

The completed run used 6,480 validation and 768 science forwards, exactly its registered 7,248 total. All 120 calls completed. Wall time was 125,797.09 seconds (34.94 hours), including evidence collection, persistence and validation.

E.4.3 Next-Token Margins and Control Results

For each landmark, m is the preferred candidate's log probability minus the alternative's. Source prefers the recipient variable; rule and scope use the registered candidates described above. G-only, joint and read use the same visible prefix. We report $m_{\mathrm{read}} - m_{\mathrm{joint}}$; a toward-G direction is defined only when $|m_G - m_{\mathrm{joint}}| \geq 10^{-4}$ nats. No case falls below that baseline-gap floor. No new outcome filter or recovery threshold is introduced.

World	Landmark	G-only	Joint	Read	Read − joint
00	source	20.187499	-3.750000	-3.625000	+0.125000
00	rule	5.625000	5.000000	4.875000	-0.125000
00	scope	9.750000	6.875000	6.875000	-1.716e-07
01	source	19.999999	-2.625000	-2.500000	+0.125000
01	rule	15.937500	13.375000	13.312500	-0.062500
01	scope	7.750000	6.125000	6.125000	+2.207e-07
10	source	23.562499	-2.000000	-1.875000	+0.125000
10	rule	6.000000	5.125000	5.375000	+0.250000
10	scope	9.250000	6.000000	6.000000	+1.364e-07
11	source	22.187501	-2.375000	-2.250000	+0.125000
11	rule	15.875000	13.500000	13.500000	+1.179e-07
11	scope	7.750000	6.000000	6.000000	-9.593e-08

Table 6: **Complete next-token margin measurements for the attention-read experiment.** Positive source margins favor the recipient-source token. Read replaces old-position K/V with the G-only reference at one layer-19 attention read. All 4 source margins remain negative. Rule and scope use their respective registered token pairs; no generated continuation is evaluated.

All 4 source margins increase by approximately 0.125 nats while remaining negative. The recipient-token probabilities change from 2.277% to 2.573%, 6.715% to 7.539%, 11.801% to 13.161%, and 8.454% to 9.470% in world order 00, 01, 10, 11. No source-pair preference is restored. The 4 worlds are crossed conditions, not independent population samples. The near-equal increments reflect these BF16 measurements. They do not estimate a universal effect size.

Rule changes are mixed. The scope changes are approximately 10^{-7} nats; their signed direction flags should not be read as substantive recovery. At every landmark, joint-repeat, clone-only and self-read log-probability bytes equal the joint reference exactly, yielding zero full-vocabulary TV and zero pair-margin difference in all 36 comparisons. All 12 scientific OLD substitutions have nonzero K and V value differences from the joint reference. All 12 greedy next tokens remain unchanged. Source-landmark full-vocabulary TV ranges from 0.003314 to 0.014084. Scope-landmark TV ranges from 2.21×10^{-6} to 2.25×10^{-5}: a near-zero pair margin does not mean the complete distribution is unchanged.

Interpretation. At this layer-19 read, key/value substitution shifts source preference but leaves all 4 source margins negative and all 12 greedy predictions unchanged. Full residual restoration recovers complete G17-only programs in all 4 generation conditions. These different endpoints establish the effects of the two interventions; the one-read result does not make broader restoration a necessary condition for recovery.

F Reproducibility, Study Records, and Compute

This appendix describes the study records and released reproduction materials, accounts for completed content experiments, and summarizes verification coverage. A final diagnostic examines how the resolved DeltaNet backend affects one historical generation.

F.1 Recorded Protocols and Reproduction Materials

The Plan records specify the populations, selection rules and endpoints below. Discovery and confirmation use separate populations; the 4-condition serialization extension retains the original serialization population. Early records do not establish every freeze timestamp relative to outcome access, so we describe these as recorded protocols rather than comprehensive prospective preregistrations. Each content experiment has its own manifest, donor collection and validation records.

Study	Recorded protocol choices
Boundary discovery	population and selection criterion
Held-out confirmation	population, B17, and endpoints
Specificity controls	population, B17, controls, and specificity criterion
Serialization study	population, B17, landmarks, cue constructions, factorial conditions, and endpoints
Repository Cohort 1	cohort, B17, horizon rule, intervention protocol, and endpoints
Repository Cohort 2	cohort, B17, intervention protocol, and replication criterion

Study archives retain source and target definitions, prompts, token spans, intervention schedules, outputs, scoring code and verification records. Source code and donor mappings are versioned by study; debugging and startup histories are kept separate. The released `supplementary_code.zip` provides offline response rescoring, attention-distribution recomputation and Plan aggregation. Its `docs/EXPERIMENT_MAP.md` maps results to included evidence, and `docs/REPRODUCTION.md` gives execution instructions and coverage. The package excludes full state archives, private repository tasks and their verifier, and GPU generation code.

F.2 Completed Content Experiments and Compute

The table counts model calls and actual forward passes for the completed content experiments. Validation includes hook-off references, instrumented baselines, repeats and self replacements. Scientific calls implement the reported intervention contrasts. Attention-read calls score one next token; the other rows generate complete replacements. Totals exclude the Plan experiments, preliminary runs, stopped attempts and backend diagnostics.

Study	Validation calls	Scientific calls	Forwards
Separate rule/source effects (D.1)	20	16	3,400
Joint composition (D.2)	24	12	3,406
New functions and rules (D.3)	24	12	3,600
4 code structures (D.4)	96	48	14,328
Fixed-state reuse (D.5)	36	8	4,180
Donor-background selectivity (D.6)	36	8	2,848
Source-entry comparison (E.1)	28	10	3,648
Residual/cache restoration (E.2)	20	8	2,688
Output-residual restoration (E.3)	28	12	4,000
One attention read (E.4)	108	12	7,248
Total	**420**	**146**	**49,346**

A forward is one model invocation, whether processing a prefill sequence or a single token; these counts are not FLOPs. Wall times include state collection, persistence and validation as well as inference. Separate archives retain 2 side studies: recombination produced no executable hybrid, and visible-source manipulation showed a preference asymmetry without reliable executable reconstruction.

F.3 VERIFICATION COVERAGE

Runtime controls check intervention placement and numerical consistency. Independent reviews then reanalyze saved evidence at the levels listed below. These checks operate on recorded outputs or states; they do not rerun model inference. Their coverage is distinct from the smaller set of evidence redistributed in the supplementary package.

Study	Completed verification
Source-entry comparison	All 38 recorded-state analyses, scores and token schedules replayed.
Residual/cache restoration	Recorded evidence, scores and schedules checked for all 28 calls.
Separate effects, joint composition, new factors and 4 structures	All outputs rescored and token schedules reconstructed; saved runtime checks reviewed. Full neural tensors were not independently replayed.
Fixed-state reuse	All 44 full-state analyses repeated during execution; post-run review checks outputs, schedules and those replay records.
Donor-background selectivity	All 44 full-state analyses replayed. The 6 observer and 24 repeat/self cross-call comparisons were evaluated after generation.
Output-residual restoration	All outputs rescored; complete exported restoration residuals replayed. Cache review uses exported metadata and hashes.
One attention read	Complete exported distributions and old-position K/V values checked; cache review uses metadata and hashes.

For the final 2 rows, exported projections were checked against declared source hashes without rereading the complete original state archives. The source-entry run completed all calls and terminal verification before a reporting I/O error; no later model call occurred. Its scientific artifacts are complete, but the launcher did not exit cleanly. Per-study verification descriptions and inventories remain in the companion records.

F.4 SENSITIVITY TO THE RESOLVED DELTANET BACKEND

Historical native generation yields 5/96 semantic successes, compared with 4/96 at the later prefix-forcing study's unforced $k = 0$ baseline. Both use the same semantic oracle and yield 0/96 byte-exact registered targets; the later successes are a strict subset of the earlier ones. One double-complete J_{AP} cell in the quorum-vote-basis family accounts for the difference.

In the historical evaluation, generation selects `quorumWeight(value)` and produces a compilable semantic success. Under the later Transformers Torch fallback, it instead selects `value`, continues into an additional function, and terminates in a Go parse failure.

Two later independent model loads, approximately 46 minutes apart, produce identical tokens, bytes and token-level log-probability arrays on all 96 baseline cells. Under the Torch fallback, direct generation and a reconstruction of the historical pre-generation forward history are each repeated twice. All 4 attempts reproduce the same `value` parse-failure trajectory.

With the checkpoint, device map, architecture and cell fixed, switching from the Transformers Torch fallback to the FLA fast path recovers the historical `quorumWeight(value)` continuation in 2 repeated runs. Token IDs, suffix bytes and `new_string` match the historical output exactly; token-level log probabilities differ by at most 0.00348.

Backend choice is the only tested factor that reproduces the historical discrepancy. The historical log does not explicitly record the resolved backend, so an FLA-versus-Torch mismatch is the best-supported explanation, rather than directly recorded provenance.

The diagnostic is separate from the primary prefix-forcing comparison, whose conditions use one backend. The supplementary package includes `results/backend_audit_excerpt.json`; full environment-resolution logs and package paths remain in the study artifact `analysis/reproducibility_backend_audit_v1/backend_audit_public_v1.json`. The diagnostic shows why run records should name the resolved DeltaNet backend rather than label it only `native DeltaNet`.