FROM SIMILAR CODE TO NEURAL CORRESPONDENCE: CAUSAL INTERVENTIONS GUIDED BY PROGRAM STRUCTURE

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models can generate code that closely resembles existing implementations, but output similarity alone does not explain how similar code is supported by the model's internal computation. We study whether program structure can help identify internal states that play corresponding causal roles across similar programs. We introduce Program Structured Causal Alignment (PSCA), a framework that uses program structure to select corresponding positions across programs and evaluates the transferred internal states through causal intervention. PSCA evaluates this correspondence in two complementary ways. Intact causal compatibility asks whether transfers selected by program structure better support code prediction than comparison transfers in an otherwise intact computation. Causal recovery asks whether they restore part of the lost prediction after disruption and achieve greater recovery than comparison transfers. During intact computation, transfers selected by program structure generally support prediction better than comparison transfers. Across three model families and nine fine-tuned models, data-flow-aligned restoration consistently improves prediction over both the disrupted condition and mismatch restoration at the prespecified intervention layers, while control-flow recovery varies across models and intervention settings. Together, these results provide evidence that program structure can guide the identification of internal states with meaningful functional effects on code prediction.

1 INTRODUCTION

Large language models can generate code that closely resembles existing implementations (Ciniselli et al., 2022; Yu et al., 2023; Wu et al., 2025). Existing similarity analysis can identify such code after generation, but it does not explain how the model internally supports the generation of similar code. Understanding this process requires comparing internal computation across similar programs. For this comparison, we need an external hypothesis about which parts of the programs should correspond. Program representations can make data flow and control flow relationships explicit (Allamanis et al., 2018; Guo et al., 2021; Cummins et al., 2021). These relationships provide an external hypothesis for identifying corresponding parts of similar programs, even when their tokens and positions differ. Because these relationships are defined independently of the model, they allow us to examine whether program level correspondence is also reflected in neural computation. We therefore ask: when program structure identifies corresponding parts of two similar programs, do the neural states at those parts also play corresponding causal roles in code generation?

To investigate this question, we use causal interventions, such as activation patching, which modify internal model states and observe how prediction changes (Meng et al., 2022; Zhang & Nanda, 2024; Geiger et al., 2025; Mueller et al., 2026). When such interventions are applied across two programs, we need to specify which positions in one program correspond to which positions in the other. We refer to this mapping as an *alignment*. Existing work provides general methods for studying the causal roles of internal representations (Geiger et al., 2022; 2024; Sun et al., 2025), and recent work has applied similar interventions to code (Abdelsalam et al., 2026). These methods either start from a predefined alignment, learn an alignment from intervention results, or construct code examples with corresponding token positions specified in advance. However, they do not directly evaluate

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

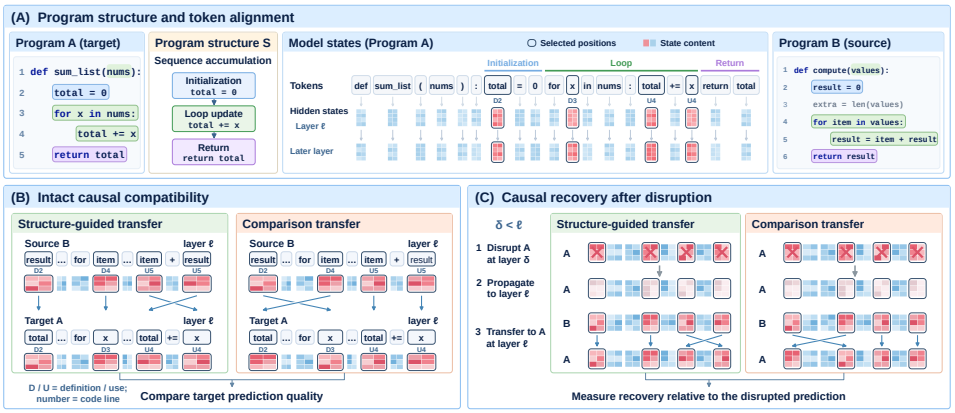


Figure 1: **An example of state transfer across similar programs.** Program structure identifies corresponding positions across two programs. We compare structure-guided transfers with comparison transfers during intact computation and after disruption.

whether an alignment defined by program structure predicts which transferred internal states better support code prediction.

Our key insight is that program structure tells us which positions across programs are meaningful to compare, while causal intervention allows us to assess whether states at those positions better support code prediction. For example, Figure 1 shows two accumulation programs with different variable names and token positions. Data-flow relations identify the correspondence between the definitions and uses of `total` and `result`, providing corresponding positions for state transfer.

Based on this insight, we introduce **Program Structured Causal Alignment (PSCA)**. PSCA uses program structure to define an alignment between positions in two programs and maps these positions to model tokens. It then transfers internal states according to this alignment and measures how the transfer affects code prediction. We compare these transfers with comparison transfers that do not follow the same program correspondence, allowing us to examine whether program structure provides useful guidance for selecting intervention sites.

We evaluate PSCA in two settings. First, we compare transfers selected by program structure with alternative transfers, without first disrupting the target computation. We ask which better supports code prediction, and refer to this as **intact causal compatibility**. Second, we first disrupt the target computation so that its prediction becomes worse, and then measure how much each transfer restores the lost prediction. We refer to this as **causal recovery**. In simple terms, the first setting asks which transfer works better during normal computation, while the second asks which helps more after the computation has been disrupted.

We evaluate PSCA across three model families and nine fine-tuned models using data-flow and control-flow correspondences. During intact computation, structure-aligned transfers generally support code prediction better than comparison transfers. At the prespecified intervention layers, data-flow-aligned restoration improves prediction over both the disrupted condition and mismatch restoration in all nine models. Control-flow recovery varies more across models and intervention settings.

Our contributions are:

1. **Causal correspondence across similar programs.** We study whether program structure predicts which transferred internal states better support code prediction.
2. **Program Structured Causal Alignment.** We introduce PSCA, which uses program structure to select positions across programs and evaluates the transferred states through causal intervention.
3. **Empirical evidence.** Across models and different forms of program structure, transfers selected by program structure generally support prediction better than alternative transfers

during normal computation. After disruption, data flow shows consistent partial recovery, while control flow results are more variable.

2 Program Structured Causal Alignment

PSCA evaluates whether an alignment defined by program structure predicts which transferred states better support code prediction. We construct this alignment by matching program relations and mapping the code locations they connect to model token positions. We then compare transfers that follow or break the program correspondence. Figure 2 summarizes the procedure.

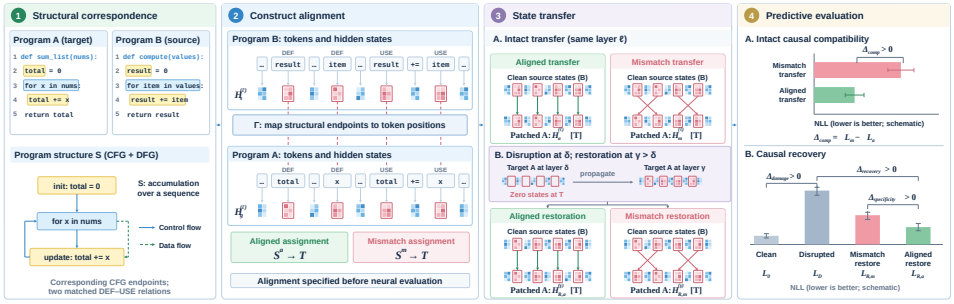


Figure 2: **Overview of PSCA.** PSCA consists of four steps: (1) identify structural correspondence; (2) map corresponding program locations to token positions; (3) perform aligned and mismatch state transfers; and (4) evaluate intact causal compatibility and recovery after disruption.

2.1 Preliminaries and Problem Formulation

PSCA uses activation patching to transfer hidden states between program computations. We call the computation supplying the states the *source* and the computation receiving them the *target*. Each *state transfer* copies complete hidden vectors from selected source token positions to selected target positions at the same decoder block (layer) output. Source states come from a *clean run*, a computation without activation intervention.

Let s be a source program and g a similar generated target program. Both computations use the same autoregressive language model with parameters θ, which remain unchanged during intervention. Let x_s and x_g denote their tokenized inputs, including the context used for each computation. At layer ℓ, the hidden states are $H_\theta^{(\ell)}(x) \in \mathbb{R}^{n_x \times D}$ for $x \in \{x_s, x_g\}$, where n_x is the input length and D is the hidden dimension. The vector $H_\theta^{(\ell)}(x)[i]$ is the hidden state, or activation, at token position i. We abbreviate the source and target state matrices as $H_s^{(\ell)}$ and $H_g^{(\ell)}$.

Let U be the current target state matrix at layer ℓ. For ordered position lists $S = (s_1, \ldots, s_k)$ and $T = (t_1, \ldots, t_k)$, with distinct target positions, define

$$\mathrm{Patch}(U, H_s^{(\ell)}; T, S)[j] = \begin{cases} H_s^{(\ell)}[s_i], & j = t_i \text{ for some } i, \\ U[j], & j \notin T. \end{cases} \tag{1}$$

Each pair $t_i \leftarrow s_i$ specifies which source state is written to which target position. Other positions remain unchanged at this layer; the model then continues from the modified target states.

Problem formulation. PSCA asks whether state transfers guided by program correspondence better support target prediction than comparison transfers using the same target positions. An *aligned transfer* follows this correspondence, using source positions S^a for target positions T. Our primary comparison, a *mismatch transfer*, uses source positions S^m that break the correspondence while preserving T and the number of transferred states. Both sets of source positions are selected without using neural activations or intervention outcomes. Section 2.2 specifies their construction; Section 2.3 defines the evaluation criteria.

2.2 From Program Structure to Token Alignment

Program relations connect code locations, whereas the state transfer defined in Section 2.1 operates on token positions. We therefore match relations across programs and map the connected code locations to tokens.

We consider two forms of program structure: control flow and data flow, indexed by $v \in \mathcal{V} = \{\mathrm{ctrl}, \mathrm{data}\}$. We write $Z_v(p) = \alpha_v(p)$ for the structure extracted from program p by the corresponding extraction procedure α_v. For $v = \mathrm{ctrl}$, a control flow graph (CFG) connects program points between which execution can proceed directly. For $v = \mathrm{data}$, a data flow graph (DFG) connects a value's definition (DEF), where it is introduced or assigned, to a use (USE), where it is read.

Let $\mathcal{R}_v(p)$ contain the relations in $Z_v(p)$. Each relation has two *endpoints*: the code locations it connects. A *program correspondence* pairs selected source and target relations, without requiring the entire programs to match:

$$\Pi_v(s, g) \subseteq \mathcal{R}_v(s) \times \mathcal{R}_v(g). \tag{2}$$

For control flow, matched expressions or statements must also form direct control flow edges with compatible endpoint roles and branch outcomes (e.g., true or false). For data flow, matching preserves definition and use roles and checks compatible bindings (which variable or attribute an occurrence refers to) and scopes (where those bindings apply).

An *intervention unit* is a set of matched relations whose endpoint states are transferred together. We index it by $b \in \mathcal{B}_v(s, g)$ and write its relation pairs as $\mathcal{C}_{v,b} \subseteq \Pi_v(s, g)$. A control flow unit contains one source–target relation pair; a data flow unit contains two matched definition–use relation pairs.

Each endpoint occupies a *code span*, a contiguous region of program text. PSCA maps the endpoints of corresponding relations to model token positions, giving the *alignment*

$$\Gamma_v(\mathcal{C}_{v,b}) = (S^a_{v,b}, T_{v,b}), \tag{3}$$

where $S^a_{v,b}$ and $T_{v,b}$ are the ordered source and target position lists used by the patch operator. We call the ordered source list the *source assignment*. We locate tokens using *tokenizer offsets*, the text boundaries associated with each token. Corresponding endpoint spans must contain the same number of tokens, which are paired in order. If endpoints share a target position, identical writes are combined; conflicting writes are excluded. The set $\mathcal{B}_v(s, g)$ contains only units that pass these matching and token mapping checks.

For mismatch transfer, we replace $S^a_{v,b}$ with $S^m_{v,b}$ while keeping $T_{v,b}$ and the number of transferred states unchanged. For control flow, another eligible source relation from the same program pair supplies the states, with matching endpoint labels and token counts. Each target relation is evaluated separately, and source relations are used in a balanced way across units in the pair. For data flow, we exchange the two source relations within a unit while preserving definition and use roles: definitions still supply definitions, and uses still supply uses. This breaks the cross-program pairing without changing the source relations themselves. Appendix B gives the complete extraction, matching, and mismatch construction rules.

2.3 Causal Evaluation

The previous section specifies the source and target positions for state transfer. We now evaluate whether aligned transfers better support target prediction than mismatch transfers, both during intact computation and after disruption. Using the positions constructed above and the patch operator in Equation 1, define

$$\mathcal{I}^{(\ell)}_{v,b,c}(U) = \mathrm{Patch}\left(U, H^{(\ell)}_s; T_{v,b}, S^c_{v,b}\right). \tag{4}$$

Here U is the current target state matrix at layer ℓ, and $c = a$ or $c = m$ selects the aligned or mismatch source positions. Within each comparison, the model, source and target inputs, target positions, transfer layer, and any earlier disruption are held fixed.

For each target program, all intervention conditions are scored on the same suffix z. Let L_X denote its mean negative log likelihood (NLL) (Gneiting & Raftery, 2007) under condition X. Lower NLL means that the model assigns greater probability to this suffix. Subscripts 0, a, and m denote the clean run, aligned transfer, and mismatch transfer, respectively. We additionally report the Kullback–Leibler (KL) divergence (Kullback & Leibler, 1951) $K_X = \mathrm{KL}(P_0 \| P_X)$, where P_0 and P_X are the clean and intervened next-token distributions. The divergence is computed over the full vocabulary at each scored position and then averaged over those positions. Lower KL means closer agreement with the clean prediction distribution. Section 3.1 specifies the scoring procedure.

Intact causal compatibility. We first apply transfers without an earlier disruption and compare their prediction losses using $\Delta_{\mathrm{comp}} = L_m - L_a$. A positive value, $\Delta_{\mathrm{comp}} > 0$, means that the aligned transfer supports target prediction better than the mismatch transfer. This relative advantage does not require the aligned transfer to improve on the clean prediction.

Causal recovery after disruption. Intact compatibility does not establish whether transferred states can restore prediction after disruption. We evaluate this by disrupting the target computation and transferring states at a later layer.

Writing $T = T_{v,b}$, we start from the clean target run, set the hidden vectors at positions T to zero at layer δ, and continue the disrupted computation to a later layer $\gamma > \delta$:

$$H_D^{(\delta)}[T] \leftarrow 0, \qquad H_D^{(\gamma)} = F_{\delta \to \gamma}\left(H_D^{(\delta)}\right), \tag{5}$$

where $F_{\delta \to \gamma}$ denotes decoder blocks $\delta + 1$ through γ. We then apply aligned and mismatch transfers separately to the same disrupted state, calling each a *restoration intervention*:

$$H_{R,a}^{(\gamma)} = \mathcal{I}_{v,b,a}^{(\gamma)}\left(H_D^{(\gamma)}\right), \qquad H_{R,m}^{(\gamma)} = \mathcal{I}_{v,b,m}^{(\gamma)}\left(H_D^{(\gamma)}\right). \tag{6}$$

Let L_D be the loss after disruption with no restoration, and let $L_{R,a}$ and $L_{R,m}$ be the losses after aligned and mismatch restoration. We evaluate three contrasts:

$$\boxed{\Delta_{\mathrm{damage}} = L_D - L_0} \quad \boxed{\Delta_{\mathrm{recovery}} = L_D - L_{R,a}} \quad \boxed{\Delta_{\mathrm{specificity}} = L_{R,m} - L_{R,a}} \tag{7}$$

Positive values respectively indicate that disruption worsens prediction, aligned transfer recovers part of that loss, and aligned transfer recovers more than mismatch transfer. We call the third contrast *restoration specificity*.

We assess causal recovery using the average effects of all three contrasts and their uncertainty, as specified in Section 3.1. Recovery may be partial: $L_{R,a}$ can remain above L_0 even when aligned transfer improves prediction over the disrupted run. Replacing L with K gives the corresponding contrasts for recovery toward the clean predictive distribution.

Compatibility does not imply recovery. Positive intact compatibility can coexist with zero recovery (Proposition 1, Appendix F.2). We therefore evaluate the two criteria separately.

3 Experiments

We first compare PSCA with alternative source-position alignments, then assess whether its aligned states support prediction during intact computation and recover prediction after disruption.

3.1 Experimental Setup

Models and data. We use Gemma-2-9B-it (Team et al., 2024), Llama-3.1-8B-Instruct (Grattafiori et al., 2024), and Qwen2.5-Coder-7B-Instruct (Hui et al., 2024), referred to as Gemma, Llama, and Qwen. Each model is fine-tuned on original code (Orig.), lexically transformed code (Lex.), or structurally transformed code (Struct.), yielding nine fine-tuned models. The three settings share source functions, prompts, and control examples, while varying the training targets. They elicit similar-code continuations for testing correspondence across different code forms. The code comes from Python GitHub repositories created in 2025 or later, after the release of all three model versions. We retrieve source candidates with JPlag (Prechelt et al., 2002) from a common library of original functions and retain pairs with valid CFG or DFG correspondences, yielding 1,566 CFG pairs and 3,813 DFG pairs across the nine models. Training, library construction, and retrieval details are given in Appendix A.

Interventions and comparisons. We use the aligned and mismatch transfers defined in Section 2.3. We additionally compare PSCA with three alternative source-selection rules on the same target positions. Same-offset selects source states at matching relative token offsets, constrained-random selects a feasible source assignment at random, and clone-span uses token correspondences from registered code-clone spans. These comparisons examine whether program structure provides useful guidance beyond token position, feasible random selection, and surface code similarity. The clone comparison includes only pairs with a complete clone-span mapping for the PSCA target positions. We evaluate compatibility, baseline comparisons, and recovery in full layer sweeps. For compact side-by-side presentation in the main text, we summarize compatibility and baseline comparisons at the midpoint of each architecture, providing a common reference at similar relative depths. For recovery, we show disruption at roughly one quarter of model depth and restoration at roughly three quarters, using a common early-to-late configuration for comparison across architectures. Exact layer indices and additional schedules are reported in Appendices B and C.

Evaluation and aggregation. We evaluate intervention effects using NLL and KL as defined in Section 2.3. NLL measures support for the observed target continuation, while KL measures departure from the clean predictive distribution. All conditions score the same target suffix under teacher forcing, so they are compared using the same preceding target tokens. We first average intervention contrasts within each program pair and then weight program pairs equally. We report 95% bootstrap confidence intervals over program pairs within each fine-tuned model. Further scoring and statistical details are provided in Appendices B.6 and C.

3.2 Comparison with Alternative Alignments

We compare PSCA with same-offset, constrained-random, and clone-span transfers at the same target positions, evaluating PSCA on the same program pairs as each baseline. Table 1 shows a consistent mean advantage over positional and random source selection: for both CFG and DFG, PSCA yields lower NLL and KL across all nine models. For DFG, the NLL reductions range from 0.428 to 0.776 nats/token across these comparisons, and all paired NLL and KL intervals exclude zero. These results show that program-defined correspondence provides useful source-state selection beyond token position and feasible random assignment.

For CFG, this advantage extends to clone-span alignment, which selects source states using surface code correspondence. PSCA yields lower mean NLL and KL in all nine models, with intervals excluding zero in five NLL and all nine KL comparisons. Control-flow correspondence therefore guides source-state selection beyond the correspondence supplied by clone spans.

For DFG, we focus on Gemma when interpreting the clone-span comparison. Requiring valid clone-span mappings for all PSCA target positions leaves few eligible pairs for Llama and Qwen. PSCA yields lower mean KL in all three Gemma models and lower mean NLL in Lex. and Struct. The paired confidence intervals include zero under both metrics. Full paired comparisons and coverage details for all models are provided in Appendices C.6 and C.7.

Table 1: **Comparison with alignment baselines.** Mean NLL and KL; lower is better. Colors mark differences whose confidence intervals exclude zero: blue (+) favors PSCA; orange (−) favors the baseline. Unshaded differences have intervals including zero. $\text{PSCA}_{s/r}$ and PSCA_c are evaluated on the same pairs as Same-offset/Random and Clone-span, respectively.

Metric	Method	Gemma			Llama			Qwen		
		Orig.	Lex.	Struct.	Orig.	Lex.	Struct.	Orig.	Lex.	Struct.
CFG / NLL ↓	**$\text{PSCA}_{s/r}$**	**0.0346**	**0.1400**	**0.0503**	**0.2923**	**0.6530**	**0.4650**	**0.2062**	**0.4137**	**0.2376**
	Same-offset	0.1058^{+}	0.2146^{+}	0.1173^{+}	0.3197^{+}	0.6819^{+}	0.4937	0.2216^{+}	0.4495^{+}	0.2647^{+}
	Random	0.1052^{+}	0.1811^{+}	0.1045^{+}	0.3241^{+}	0.6653	0.4851	0.2316^{+}	0.4515^{+}	0.2682^{+}
	PSCA_c	**0.0382**	**0.1436**	**0.0439**	**0.3319**	**0.6083**	**0.7056**	**0.3692**	**0.3744**	**0.2901**
	Clone	0.0881^{+}	0.1868^{+}	0.1123^{+}	0.3421	0.6259	0.7114	0.3883^{+}	0.4301^{+}	0.3019
CFG / KL ↓	**$\text{PSCA}_{s/r}$**	**0.0106**	**0.0116**	**0.0080**	**0.0027**	**0.0059**	**0.0018**	**0.0068**	**0.0030**	**0.0052**
	Same-offset	0.0857^{+}	0.1011^{+}	0.0833^{+}	0.0449^{+}	0.0196^{+}	0.0596^{+}	0.0334^{+}	0.0304^{+}	0.0413^{+}
	Random	0.0825^{+}	0.0532^{+}	0.0633^{+}	0.0492^{+}	0.0192^{+}	0.0512^{+}	0.0411^{+}	0.0292^{+}	0.0389^{+}
	PSCA_c	**0.0143**	**0.0240**	**0.0064**	**0.0024**	**0.0056**	**0.0014**	**0.0113**	**0.0037**	**0.0048**
	Clone	0.0627^{+}	0.0655^{+}	0.0757^{+}	0.0302^{+}	0.0155^{+}	0.0502^{+}	0.0407^{+}	0.0523^{+}	0.0320^{+}
DFG / NLL ↓	**$\text{PSCA}_{s/r}$**	**0.1566**	**0.2627**	**0.1220**	**0.4618**	**0.4721**	**0.7893**	**0.4506**	**0.3660**	**0.8087**
	Same-offset	0.8270^{+}	0.6907^{+}	0.7633^{+}	1.1205^{+}	1.0115^{+}	1.3399^{+}	1.0494^{+}	1.0649^{+}	1.5844^{+}
	Random	0.7675^{+}	0.6959^{+}	0.7478^{+}	1.1398^{+}	0.9755^{+}	1.3551^{+}	1.0111^{+}	1.0474^{+}	1.5307^{+}
	PSCA_c	**0.2669**	**0.0737**	**0.0480**	**3.0164**	**0.0004**	**4.7751**	**2.4458**	**0.0038**	**2.9497**
	Clone	0.2580	0.0899	0.0694	2.8485^{-}	0.0007^{+}	4.4412^{-}	2.2705^{-}	0.0054^{+}	2.7555^{-}
DFG / KL ↓	**$\text{PSCA}_{s/r}$**	**0.1141**	**0.2085**	**0.0716**	**0.0146**	**0.0300**	**0.0092**	**0.0366**	**0.0447**	**0.0546**
	Same-offset	0.7882^{+}	0.6419^{+}	0.7176^{+}	0.7293^{+}	0.6007^{+}	0.6707^{+}	0.7136^{+}	0.7526^{+}	0.9627^{+}
	Random	0.7261^{+}	0.6469^{+}	0.7016^{+}	0.7560^{+}	0.5791^{+}	0.6922^{+}	0.6831^{+}	0.7368^{+}	0.9164^{+}
	PSCA_c	**0.0752**	**0.0267**	**0.0305**	**0.0005**	**0.0000**	**0.0030**	**0.0494**	**0.0000**	**0.0621**
	Clone	0.0856	0.0515	0.0497	0.0013^{+}	0.0001^{+}	0.0064^{+}	0.0171	0.0007^{+}	0.0140^{-}

3.3 Intact Causal Compatibility

We next ask whether program-defined correspondence selects source states that better support target prediction than structural mismatch during intact computation. At the prespecified comparison layers, aligned transfers have lower mean NLL and KL for both CFG and DFG in all nine models. Figure 3(a–c) shows a larger advantage for DFG; CFG also benefits from aligned transfer across a range of model depths.

Data flow. Aligned DFG transfer lowers mean NLL by 0.120–0.310 nats/token relative to mismatch transfer across the nine models. Its mean KL is also lower by 0.121–0.374 nats/token. The confidence intervals exclude zero for every model under both metrics. Thus, states selected by data-flow correspondence support the observed continuation better than mismatched states and keep the predictive distribution closer to the clean computation. The agreement between the two metrics supports intact compatibility under DFG alignment.

Control flow. Aligned CFG transfer also lowers mean NLL and KL in all nine models at the comparison layers. The NLL reduction ranges from 0.00045 to 0.0293 nats/token, and the KL reduction from 0.0036 to 0.0334 nats/token. The confidence intervals exclude zero in four models for NLL and seven for KL. These comparisons provide more consistent evidence for preserving agreement with the clean predictive distribution than for increasing the probability of the observed continuation. The KL results and full estimates are reported in Appendix Figure 7 and Table 4.

Layer sensitivity. The CFG depth profiles show that the compatibility advantage extends beyond the prespecified comparison layer (Figure 3(d–f)). Mean NLL favors aligned transfer across a broad range of layers in all three model families, and mean KL favors it throughout the profiles. The magnitude changes with transfer depth: Gemma and Llama generally show larger NLL reductions at earlier layers and smaller reductions later, while Qwen shows smaller and less uniform changes. These profiles support CFG compatibility across multiple intervention layers and show how its strength depends on where states are transferred. Together with the DFG results, they support program-defined correspondence as a useful guide for selecting compatible source states during intact computation.

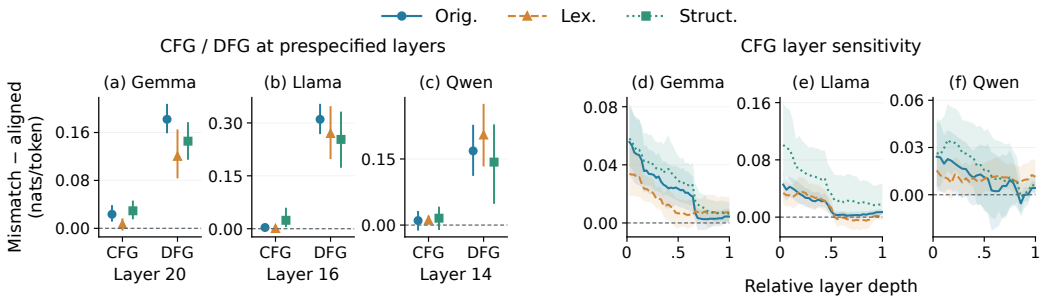


Figure 3: **Intact causal compatibility.** (a–c) CFG/DFG at prespecified layers; (d–f) CFG depth profiles. Values show NLL differences $L_m - L_a$. Shading and bars indicate confidence intervals.

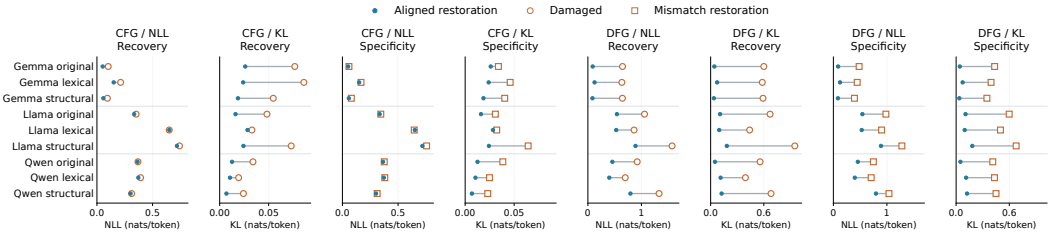


Figure 4: **Causal recovery after disruption.** Filled blue points show aligned restoration; open orange circles show disruption and squares show mismatch restoration. A blue point to the left of the circle indicates recovery; to the left of the square, an advantage over mismatch restoration. Each row represents one model.

3.4 Causal Recovery after Disruption

We next ask whether states selected by program correspondence restore prediction after disruption, and whether preserving the correspondence improves this restoration. Figure 4 shows consistent DFG recovery across all nine models: aligned restoration improves prediction over both the disrupted condition and mismatch restoration under NLL and KL. CFG also shows consistent KL recovery and mean NLL recovery in eight of the nine models.

Data flow. Disruption increases mean NLL by 0.41–0.83 nats/token across the nine models. Aligned restoration recovers 0.30–0.68 nats/token of this increase and yields NLL that is 0.24–0.44 nats/token lower than mismatch restoration. The confidence intervals for damage, recovery, and restoration specificity exclude zero in every model. After aligned restoration, NLL remains 0.03–0.15 nats/token above the clean condition. Thus, DFG-aligned states recover a large part of the prediction loss caused by disruption. Their advantage over mismatch restoration shows that preserving data-flow correspondence improves recovery beyond transferring source states alone. KL shows the same pattern. Aligned restoration reduces mean KL by 0.28–0.77 nats/token relative to disruption and by 0.31–0.49 nats/token relative to mismatch restoration. The confidence intervals for all three KL contrasts also exclude zero in every model. DFG restoration therefore both increases the probability assigned to the observed continuation and brings the full predictive distribution closer to the clean distribution.

Control flow. Aligned CFG restoration reduces mean KL relative to disruption by 0.0045–0.0620 nats/token across all nine models. Mean KL is also lower than after mismatch restoration in all nine models. The confidence intervals exclude zero in all nine models for recovery and in seven for the advantage over mismatch restoration. Under NLL, aligned restoration improves over both disruption and mismatch restoration in eight of nine models. Among these eight, the mean reduction relative to disruption ranges from 0.0036 to 0.0622 nats/token. The difference between the metrics separates two aspects of recovery. KL measures agreement with the full clean predictive distribution,

whereas NLL measures support for the observed continuation. CFG-aligned states therefore restore the broader predictive distribution more consistently than they improve the likelihood of that specific continuation.

Damage checks and confidence intervals for the paired differences are reported in Appendix Table 2 and Figure 8. Results for additional disruption and restoration layers are reported in Appendix C.

4 Related Work

Causal intervention and alignment. Causal tracing, activation patching, and causal mediation study the contribution of internal states to prediction (Meng et al., 2022; Zhang & Nanda, 2024; Mueller et al., 2026). Causal abstraction formalizes relationships between high-level and neural computation under interventions (Geiger et al., 2025). IIT trains neural models under specified causal alignments (Geiger et al., 2022), while DAS and HyperDAS learn neural alignments through interventional objectives (Geiger et al., 2024; Sun et al., 2025). Axiomatic approaches formalize criteria for validating mechanistic interpretations (Palumbo et al., 2025). Unlike approaches that assume a specified alignment or learn neural alignments through interventional objectives, PSCA derives candidate token alignments from program structure before neural evaluation.

Functional correspondence and intervention evidence. Prior work examines circuit component reuse across tasks (Merullo et al., 2024) and distinguishes causal efficacy from representational invariance (Opielka et al., 2026). Other work examines when successful interventions provide faithful evidence about the model's native computation (Makelov et al., 2024; Grant et al., 2026). These works motivate a distinction between the functional effects of transferred states and stronger claims about the model's native computation.

Program structure and neural code models. Structural and neurosymbolic approaches use program graphs for code representation, type inference, and data-flow analysis (Guo et al., 2021; Cummins et al., 2021; Li et al., 2025; Yu et al., 2025). Probing and semantic benchmarks assess program understanding (Jin & Rinard, 2024; Xu et al., 2026). Closely related, Abdelsalam et al. (2026) use residual-stream activation patching on controlled, token-aligned program variants to localize causally relevant states. Whereas their intervention setting specifies token correspondence by construction, PSCA derives token alignments from program structure across similar programs and evaluates their functional significance through state transfer.

5 Discussion and Conclusion

PSCA asks whether program structure can identify neural states with corresponding causal effects across similar programs. Across the evaluated models, aligned transfers generally support target prediction better than mismatch transfers. This result shows that program correspondence provides useful guidance for selecting states for causal intervention.

Intact causal compatibility and causal recovery capture different properties. The former evaluates whether aligned transfers better support prediction during intact computation, while the latter asks whether they restore prediction after an earlier disruption. Compatibility does not necessarily imply recovery: Proposition 1 establishes that positive intact compatibility can coexist with zero recovery. In our experiments, data flow shows consistent recovery, while control flow recovery varies more across models and intervention settings.

Our conclusions concern complete hidden states. These states may contain program role, lexical, positional, and contextual information, so the observed effects do not isolate which part of the state produces the advantage. Appendix F.3 formalizes this distinction. Our evaluation also focuses on teacher-forced next-token prediction rather than free generation or program execution. Overall, PSCA provides a way to evaluate whether program correspondence defined independently of neural activations is also reflected in the causal effects of neural states.

Reproducibility Statement

Appendices A–E describe the experimental units, mapping constraints for each abstraction, metrics, populations and uncertainty procedures. The accompanying result package preserves the backing tables and figures. The primary DFG evidence uses completed target function anchors; CFG generic comparisons retain their exact eligible populations and evaluated schedules.

Ethics Statement

The study investigates internal computation through controlled interventions during code prediction. Software ownership, training membership and provenance lie outside the evaluated intervention effects. Access and redistribution of source code must respect its licenses.

AI Use Statement

Language models assisted with manuscript restructuring, drafting, figure design, comparison of the implementation with the text and consistency checks. The present revision used the completed result summaries to align the manuscript narrative and retained the existing experimental values and statistical analyses.

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.

Miltiadis Allamanis, Marc Brockschmidt, and Mahmoud Khademi. Learning to represent programs with graphs. In *International Conference on Learning Representations*, 2018.

Matteo Ciniselli, Luca Pascarella, and Gabriele Bavota. To what extent do deep learning-based code recommenders generate predictions by cloning code from the training set? In *Proceedings of the 19th International Conference on Mining Software Repositories*, pp. 167–178, 2022.

Chris Cummins, Zacharias V Fisches, Tal Ben-Nun, Torsten Hoefler, Michael FP O'Boyle, and Hugh Leather. Programl: A graph-based program representation for data flow analysis and compiler optimizations. In *International Conference on Machine Learning*, pp. 2244–2253. PMLR, 2021.

Atticus Geiger, Zhengxuan Wu, Hanson Lu, Josh Rozner, Elisa Kreiss, Thomas Icard, Noah Goodman, and Christopher Potts. Inducing causal structure for interpretable neural networks. In *International Conference on Machine Learning*, pp. 7324–7338. PMLR, 2022.

Atticus Geiger, Zhengxuan Wu, Christopher Potts, Thomas Icard, and Noah Goodman. Finding alignments between interpretable causal variables and distributed neural representations. In Francesco Locatello and Vanessa Didelez (eds.), *Proceedings of the Third Conference on Causal Learning and Reasoning*, volume 236 of *Proceedings of Machine Learning Research*, pp. 160–187. PMLR, 01–03 Apr 2024. URL https://proceedings.mlr.press/v236/geiger24a.html.

Atticus Geiger, Duligur Ibeling, Amir Zur, Maheep Chaudhary, Sonakshi Chauhan, Jing Huang, Aryaman Arora, Zhengxuan Wu, Noah Goodman, Christopher Potts, and Thomas Icard. Causal abstraction: A theoretical foundation for mechanistic interpretability. *Journal of Machine Learning Research*, 26(83):1–64, 2025. URL https://jmlr.org/papers/v26/23-0058.html.

Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American statistical Association*, 102(477):359–378, 2007.

Satchel Grant, Simon Jerome Han, Alexa Tartaglini, and Christopher Potts. Addressing divergent representations from causal interventions on neural networks. In *International Conference on Learning Representations*, volume 2026, pp. 11330–11356, 2026.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Daya Guo, Shuo Ren, Shuai Lu, Zhangyin Feng, Duyu Tang, Shujie Liu, Long Zhou, Nan Duan, Alexey Svyatkovskiy, Shengyu Fu, Michele Tufano, Shao Kun Deng, Colin Clement, Dawn Drain, Neel Sundaresan, Jian Yin, Daxin Jiang, and Ming Zhou. GraphCodeBERT: Pre-training code representations with data flow. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=jLoC4ez43PZ.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*, 2024.

Charles Jin and Martin Rinard. Emergent representations of program semantics in language models trained on programs. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 22160–22184. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/jin24e.html.

S. Kullback and R. A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951. doi: 10.1214/aoms/1177729694.

Gen Li, Yao Wan, Hongyu Zhang, Zhou Zhao, Wenbin Jiang, Xuanhua Shi, Hai Jin, and Zheng Wang. Dataflow-guided neuro-symbolic language models for type inference. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 36000–36020. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/li25cj.html.

Aleksandar Makelov, Georg Lange, Atticus Geiger, and Neel Nanda. Is this the subspace you are looking for? an interpretability illusion for subspace activation patching. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/70b8505ac79e3e131756f793cd80eb8d-Abstract-Conference.html.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 17359–17372. Curran Associates, Inc., 2022. doi: 10.52202/068431-1262. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/6f1d43d5a82a37e89b0665b33bf3a182-Paper-Conference.pdf.

Jack Merullo, Carsten Eickhoff, and Ellie Pavlick. Circuit component reuse across tasks in transformer language models. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 18349–18377, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/4ffbfbcf4ad7304d57158b046525e46c-Paper-Conference.pdf.

Aaron Mueller, Jannik Brinkmann, Millicent Li, Samuel Marks, Koyena Pal, Nikhil Prakash, Can Rager, Aruna Sankaranarayanan, Arnab Sen Sharma, Jiuding Sun, Eric Todd, David Bau, and Yonatan Belinkov. The quest for the right mediator: Surveying mechanistic interpretability for NLP through the lens of causal mediation analysis. *Computational Linguistics*, 52(1):331–378, March 2026. doi: 10.1162/coli.a.572. URL https://aclanthology.org/2026.cl-1.10/.

Gustaw Opielka, Hannes Rosenbusch, and Claire Stevenson. Causality $\neq$ invariance: Function and concept vectors in llms. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 83393–83417, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/86b3697c4eb7792c951831636bfdacd5-Paper-Conference.pdf.

Nils Palumbo, Ravi Mangal, Zifan Wang, Saranya Vijayakumar, Corina S. Pasareanu, and Somesh Jha. Validating mechanistic interpretations: An axiomatic approach. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 47509–47544. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/palumbo25a.html.

Lutz Prechelt, Guido Malpohl, and Michael Philippsen. Finding plagiarisms among a set of programs with JPlag. *Journal of Universal Computer Science*, 8(11):1016–1038, 2002. doi: 10.3217/jucs-008-11-1016.

Jiuding Sun, Jing Huang, Sidharth Baskaran, Karel D'Oosterlinck, Christopher Potts, Michael Sklar, and Atticus Geiger. Hyperdas: Towards automating mechanistic interpretability with hypernetworks. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 8341–8358, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/180d2acb13633fe78688d0d2347c731f-Paper-Conference.pdf.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

Weibin Wu, Haoxuan Hu, Zhaoji Fan, Yitong Qiao, Yizhan Huang, Yichen Li, Zibin Zheng, and Michael Lyu. An empirical study of code clones from commercial ai code generators. *Proceedings of the ACM on Software Engineering*, 2(FSE):2874–2896, 2025.

Jade Xu, Renliang Sun, Zijian Ding, Weikai Li, Kaiqiao Han, Han Zhang, Chengyue Wang, Haixin Wang, Wei Wang, Yizhou Sun, et al. How well do llms understand code? *Communications AI & Computing*, 1(1):13, 2026.

Qingchen Yu, Xin Liu, Qingguo Zhou, and Chunming Wu. Bridging the gaps between graph neural networks and data-flow analysis: The closer, the better. *Proceedings of the ACM on Software Engineering*, 2(ISSTA):686–706, 2025.

Zhiyuan Yu, Yuhao Wu, Ning Zhang, Chenguang Wang, Yevgeniy Vorobeychik, and Chaowei Xiao. CodeIPPrompt: Intellectual property infringement assessment of code language models. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 40373–40389. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/yu23g.html.

Fred Zhang and Neel Nanda. Towards best practices of activation patching in language models: Metrics and methods. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 1651–1678, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/06a52a54c8ee03cd86771136bc91eb1f-Paper-Conference.pdf.

APPENDIX

CONTENTS

A Experimental Environments and Population Accounting **14**

B PSCA Implementation for Each Abstraction **15**

B.1 CFG structural correspondence and source grounding 15

B.2 Projection and physical assignment 16

B.3 Frozen CFG Structural Mismatch 16

B.4 Target Function DFG correspondence 16

B.5 Historical DFG implementation restricted to components 16

B.6 Outcome Definitions and Matched Controls 17

B.7 Sequential Disruption and Restoration Specification 18

B.8 Damage checks before restoration 19

C Statistical Specification and CFG Tables **19**

C.1 Full intact compatibility estimates 19

C.2 Complementary compatibility and recovery figures 20

C.3 DFG recovery at the prespecified layers 23

C.4 CFG recovery and matched generic comparisons 23

C.5 Illustration of a program pair and selection 23

C.6 Alignment Comparisons at Selected Layers 29

C.7 Clone-Span Comparator 32

C.8 Neural Comparator Informed by Clean States 34

D Similarity Associations and Coverage **36**

E Historical DFG Scope and Serialization Sensitivity **36**

F Interventional Correspondence: Proofs and Measurement **39**

F.1 General Risk Formulation of the Causal Criteria 39

F.2 Separation of Intact Compatibility and Recovery 40

F.3 Role isolation assumption and counterexample 41

F.4 Measurement connection: logarithmic risk 41

F.5 Estimators with equal pair weights and generic comparisons 42

F.6 Relation to causal abstraction 43

A Experimental Environments and Population Accounting

We evaluate nine models and track the program pairs and intervention units retained at each stage of retrieval, matching, and evaluation. The architectures are Gemma 2 9B Instruct (Team et al., 2024), Llama 3.1 8B Instruct (Grattafiori et al., 2024), and Qwen 2.5 Coder 7B Instruct (Hui et al., 2024), each under orig, lexical, and structural training settings.

Training data and code variants. The source and shared control training records cover 164 Python GitHub repositories. Repository metadata place all creation dates on or after January 1, 2025, following the release of the three model versions. Each model uses 6,060 training examples: 3,030 source-function examples and 3,030 shared control examples. Orig., Lex., and Struct. use the same source function identities and identical prompts, while their target code is original, identifier-transformed, or refactored, respectively. The prompts provide the original function signature and specification; training targets contain complete functions. The construction retains source functions with valid targets in all three settings and applies English-text and Python syntax filters. This design varies the code forms used for training while keeping the functions, prompts, and control examples shared. It produces similar-code continuations on which we test program-defined correspondence through activation transfer. Training records report three epochs, with 570 steps for Gemma/Llama and 567 for Qwen. Lex. contains 68 targets from the L1 transformation pool and 2,962 from the identifier-renaming pool; Struct. contains 54 LLM-refactored and 2,976 programmatically refactored targets. Each setting therefore represents a mixture of the listed transformations.

Each model contributes 3,030 selected greedy query records, for 27,270 samples. The training settings share source-function families and control examples, while the target code versions vary by setting. Generation logs record the model associated with each sample. The intervention runs use models with frozen identities.

Source-library construction. The retrieval library contains original Python functions and methods from 120 GPL/AGPL GitHub repositories. We extract module-level functions and class methods using Python ASTs, retaining nontrivial definitions with 5–200 lines and at least 24 tokens under the extraction script's token-count approximation. The extractor excludes test functions and configured test, documentation, example, and environment directories. The preprocessing removes normalized exact duplicates within and across the source and control pools. It also filters near-duplicate candidates found by MinHash/LSH over token 5-grams and verified by Jaccard similarity at a threshold of 0.85. Each source is exported as a Python file with a repository, path, and function identifier. After repeated source identifiers are removed from the scan manifest, the library contains 51,191 distinct source functions. It includes the original versions of all 3,030 functions used for variant training; the shared control functions are outside this library. The same original-code library is used for all nine models.

Candidate retrieval and selection. We extract code from each saved greedy generation and compare it with the source library using JPlag 6.3.0. The recorded configuration uses the Python 3 frontend, a minimum match length of nine tokens, and no optional match merging. Source files are divided into 64 shards; JPlag's old/new mode compares generated queries against each source shard. We merge the results and rank candidates by decreasing average similarity, breaking ties by source identifier. The similarity threshold is 0.178423, taken from the calibration targeting a 1% false-positive rate on 19,404 mismatched source–source pairs. Candidate selection precedes neural intervention and does not use activation states or intervention outcomes.

The saved candidate lists include queries whose registered original source passes the similarity threshold. For each such query, the export records that source and up to five other candidates, retaining their global ranks. The subsequent Top-5 selection keeps ranks 1–5 and requires all five candidates to be present; the registered source need not be among those five. This yields 16,548 query instances and 82,740 candidate pairs. The queries span 2,813 original-function families; the retrieved candidates span 16,248 distinct source functions. These are subsets of the full 51,191-function library.

Structural extraction, relation matching, token projection, and intervention validity checks then determine the CFG and DFG analysis populations, as specified in Appendix B. The downstream graph population contains 37,340 pair rows. Direct CFG correspondence covers 12,127 pairs and 65,802

edges; the primary transfer assignments cover 7,297 pairs and 27,603 correspondences. The current population for the compatibility contrast spans 1,862 pairs, 977 queries, 363 source families, and 17,956 memberships across five overlapping strata. GENERAL uses 1,566 pairs, 819 queries and 324 families. We aggregate memberships within program pairs to account for their dependence. There are 643,224 membership–layer cells per metric and 682,116 distinct completed physical tasks for each full layer NLL/KL export. These counts record computational work; statistical estimates use program pairs as the sampling units.

The primary target function DFG Population A contains 42,227 joint two relation cases from 3,813 program pairs across nine models. Each scientific case is evaluated at its frozen compatibility anchor and early to late restoration anchor. All six required condition records per case are complete in the anchor export. We average cases within each pair before weighting pairs equally.

B PSCA Implementation for Each Abstraction

We match program relations, project their endpoints to token positions, and construct aligned and mismatch assignments for state transfer. In Section 2.2, $Z_v(p) = \alpha_v(p)$ denotes the high level structural representation. The graph $\mathcal{G}_v(p)$ implements that representation with program nodes, directed relations and endpoint spans. Matching yields $\Pi_v(s, g)$; grouping its relation pairs gives the local paired intervention configurations $\mathcal{C}_{v,b}$. Token projection implements Γ_v, producing $(S^a_{v,b}, T_{v,b})$; the control rule produces $S^m_{v,b}$ with the same target positions. These assignments for each condition are used by $\mathcal{I}^{(\ell)}_{v,b,c}$. Algorithm 1 summarizes this construction.

Algorithm 1 Constructing PSCA correspondence hypotheses

Inputs: programs s, g, abstraction v, and frozen program-analysis, grounding, serialization and tokenization contracts.

1. Extract structures $\mathcal{G}_v(s)$ and $\mathcal{G}_v(g)$ implementing the structural representations $Z_v(s)$ and $Z_v(g)$.
2. Construct admissible relation correspondences $\Pi_v(s, g)$ from program structures before neural evaluation.
3. Group relation pairs into intervention configurations $\mathcal{C}_{v,b}$, indexed by $b \in \mathcal{B}_v(s, g)$, under the abstraction-specific rule.
4. Apply Γ_v by grounding endpoint spans to ordered token positions consistent with the proposed roles, yielding $(S^a_{v,b}, T_{v,b})$; construct $S^m_{v,b}$ with the same target positions and write budget; reject invalid projections and conflicting writes.
5. Return $\{(S^c_{v,b}, T_{v,b})\}_{b \in \mathcal{B}_v(s,g),\, c \in \{a,m\}}$ for eligible units.

Output: aligned and mismatch token assignments for the compatibility and recovery interventions in Section 2.

B.1 CFG structural correspondence and source grounding

We construct CFG correspondence by matching syntax nodes and verifying the control flow edges and code spans that connect them. For each frozen component pair, code-diff 0.1.3 applies GumTree isomapping followed by edit mapping with parameters 1000 and 0.5. Accepted named proposals for AST nodes require equal kinds, unique forward/inverse matches, component containment and unambiguous anchor identity. Lexical scopes are bound to unique graphs that own the Joern exports before grounding nodes by exact code span, compatible CPG label and exact CODE. The V2P1 scope contract converts half open Java UTF-16 offsets to source characters, allowing a trailing whitespace suffix. The unique lexical-scope binding determines graph ownership.

Proposals grounded on both sides with compatible scopes define a partial injective node map. A source program edge is retained only when its mapped endpoints lie in one generated program graph and form a direct edge with unique occurrences and matching attributes. We exclude edges with ambiguous occurrences. The contract admits general directed edges, including labeled true/false branches. All four endpoint supports must agree with source hashes, lexical observations, span text, scope, anchor and occurrence identity; synthetic/nonlexical endpoints and conflicting evidence are excluded.

B.2 Projection and physical assignment

We validate token coverage and endpoint widths to convert each structural match into an executable state-transfer assignment. CFG source program offsets come from the registered serialization to preserve the coordinates of the frozen input. Input reconstruction for generated programs is checked against frozen input hashes and lengths. Tokenizer offsets project code support with exact coverage or admitted whitespace slack; nonwhitespace contamination, incomplete coverage and ambiguous mapping are rejected. Each endpoint's source and generated program token widths must match, and support order determines a direct token-to-token transfer.

Logical writes to the same generated program position are coalesced only when source program serialization, position and layer coincide. Different sources at that position are a conflict. Runtime capture and write checks preserve model, tokenizer, input and token identities at decoder block output. Entire hidden vectors are transferred. Each CFG edge row is a separate intervention even when the assignment construction uses cycles to balance use of source states.

B.3 Frozen CFG Structural Mismatch

The CFG control evaluates the effect of reassigning source relations while preserving the target positions and compatible write geometry. It constructs a frozen reassignment among eligible source relations within each pair. Alternative source program relations come from the same pair and compatible input context. Identical raw edges, exact aliases of source positions, width incompatibilities and invalid physical plans are excluded. GENERAL matches endpoint labels and token geometry. Semantic annotations group eligible directed edges into the other named strata.

For each pair/stratum, exact integer bipartite assignment of minimum cost first minimizes exclusion self loops, then seeded SHA-256 weight sum, then a unique binary set code. Removing self loops yields a maximum-cardinality balanced subset in which every selected target receives a different source relation. The seed is `CFG_MISMATCH_ASSIGNMENT_V1_SELECTION_SEED_0`. Each selected generated program target retains its own aligned and mismatch plans and is evaluated separately. Selection is independent of layer outcomes and intervention effects.

B.4 Target Function DFG correspondence

We select definition–use relations within complete target functions and validate two matched relations for each joint intervention. Population A uses the complete, uniquely identified target function AST extent and the intended source function from the frozen retrieval pair. Both DEF and USE must lie inside the target function extent. Relations inside nested scopes remain eligible under the existing containment convention, with scope and binding metadata retained as matching constraints. Function identity and relation inclusion are determined from program structure before neural evaluation. The existing DEF–USE semantics and hard matching gates are retained, with deterministic disambiguation respecting order. Projection uses the frozen native input serialization and permits whitespace slack at span boundaries. Spans containing additional nonwhitespace characters are excluded.

Two complete matched relations form each joint case. DEF widths and USE widths are checked separately across source and target and for the relation exchange control. Exactly identical writes may be coalesced; conflicting writes and degenerate mismatch assignments are excluded. Population A admits each case when its two relations pass projection, width, and physical-validity checks. This case-level eligibility determines the reported population.

B.5 Historical DFG implementation restricted to components

We use a historical component-level analysis to assess sensitivity to relation scope and input serialization. The rebuilt adapter lexicalizes REACHING_DEF edges local to each method into DEF–USE relations within the endpoints' local AST contexts using direct edges. It requires compatible bindings, resolved common lexical scope and verified definition binding kind, rejecting synthetic sinks and invalid assignment target to identical assignment RHS relations. Within paired components, hard gates match relation type, NAME/ATTRIBUTE symbol family, endpoint roles, AST kinds, semantic events, lexical scope kinds, program roles and definition binding kind.

Relations are sorted by DEF span, USE span and ID. The matcher accepts a unique perfect match respecting order, or a monotone assignment satisfying the hard constraints when admitted by the frozen rule. Unequal cardinalities and unresolved matches fail at the component level. The resulting correspondence depends on the specified source order used by the algorithm.

Historical DFG cases contain two distinct, complete relations with equal widths across both source/generated program units for each role separately. The aligned assignment maps each relation to its counterpart; STRUCTURAL MISMATCH swaps the whole relations while preserving DEF/USE roles and token order. The two relations are patched jointly within a case. Native input bindings reuse accepted endpoint evidence. SAME_PROMPT and FUNCTION_ONLY have separate projections: the former uses its inspection wrapper and prefix cutoff, while the latter uses source text. This builder uses text location plus tokenizer offset overlap. CFG uses offsets from the serialization registry. The historical DFG runner consumes the frozen V2 scoring serialization and positions.

Factorial192 uses the immutable implementation labels `BASELINE`, `ZERO DAMAGE`, `CORRECT`, and `PERMUTED`, denoting CLEAN, same layer damage, ALIGNED, and STRUCTURAL MISMATCH, respectively. These are evaluated as separate branches, each starting from the clean target computation and applying its intervention at the same layer. Historical generic results apply to this component population. The target-function sequential evaluation uses the validated coordinates and schedules in Appendix B.7.

B.6 Outcome Definitions and Matched Controls

We evaluate next-token predictions along previously generated code using teacher forcing. Within each comparison, all conditions score the same target suffix and use the same preceding token context. We form paired contrasts within each control's eligible population. The main text uses s, g for source/generated programs and v, b for abstraction/intervention unit. Historical result labels remain in preserved figures and tables: C, M, N map to conditions a, m, n, and CR, MR, NR to restoration branches $(R, a), (R, m), (R, n)$. Thus $L_M - L_C$ is the same contrast for each unit as $\Delta_{\text{comp}} = L_m - L_a$; reported S aggregates that contrast using the specified pair weighting. The structural configuration symbol $\mathcal{C}_{v,b}$ is distinct from the historical condition subscript C.

Input contexts and scoring suffix. Source states are captured from code-token positions in a clean run on the source fragment. The target input combines its recorded completion prompt with the saved target code. The user messages are shown below; both are rendered with the model's native chat template using `add_generation_prompt=True`. The target prompt and code are tokenized separately with `add_special_tokens=False` and concatenated, after removing outer Markdown fences and surrounding whitespace from the saved code.

The matched target code regions saved during pair construction determine the scoring boundary. The suffix z begins at the first target token whose nonempty character span starts at or after the largest end offset of those regions, and extends to the end of the saved code. The complete-function DFG evaluation retains this input and scoring suffix while selecting its intervention positions separately. Every condition uses the same scored tokens and preceding target tokens. For a scored token at input position p, NLL and KL use the next-token distribution from logits at position $p - 1$.

Source user message.

````
Inspect the following Python source fragment as code context.

Return a code-grounded response enclosed in a markdown code block.

```python
{source_fragment}
```
````

Target user message.

```
You are completing a Python function.
Output only the completed function.
Do not explain anything.
```

```
Do not output markdown headings or step-by-step text.
If you use markdown, output exactly one ```python block and nothing else.

Function signature to complete:
```python
{signature}
```
```

Here, `{signature}` is the stripped content of the first nonempty Python code block in the recorded instruction. If no such content is extracted, the stripped original instruction replaces the *Function signature to complete* section.

Let z be the scored suffix of length m and c_g its preceding context. Both are the same across conditions. For condition X, define

$$L_X(\ell) = -\frac{1}{m}\sum_{t=1}^{m} \log p_\theta^{X,\ell}(z_t \mid c_g, z_{<t}), \tag{8}$$

$$K_X(\ell) = \frac{1}{m}\sum_{t=1}^{m} D_{\mathrm{KL}}\Big(p_\theta^{0}(\cdot \mid c_g, z_{<t}) \,\|\, p_\theta^{X,\ell}(\cdot \mid c_g, z_{<t})\Big). \tag{9}$$

CLEAN is denoted by 0. KL uses the full vocabulary at the same scored positions. NLL measures how well the model predicts the target tokens given their preceding context; KL measures departure from the clean predictive distribution.

ALIGNED (C in historical results) preserves the proposed structural correspondence. STRUCTURAL MISMATCH (M) changes the source assignment: CFG uses a frozen balanced relation reassignment within the same program pair; DFG exchanges two whole relations with endpoint roles preserved. The main CFG GENERAL control matches endpoint labels, token widths, and support geometry. The transferred vectors also carry lexical and contextual information, so the contrast measures their combined effect under each assignment.

MATCHED GENERIC (N) selects alternative contiguous code token supports from the same source program and serialization, excluding registered structural supports. It preserves target positions, endpoint widths, write budget, model, and scoring positions. A frozen ladder of position bins (0/1/2) constructs nonoverlapping assignments; eligibility excludes alias incompatibility and missing legal spans. Positions outside the support may retain structural information. We evaluate CFG generic transfers at the frozen midpoint and early to late schedule. Across the five overlapping CFG strata, 13,830 of 17,956 memberships are eligible. Each overlapping stratum is analyzed separately to preserve its population definition. Every generic contrast uses aligned and generic records from the same eligible slice. CLEAN (0) is the unmodified reference.

$$\boxed{S(\ell) = L_M(\ell) - L_C(\ell)}, \qquad \boxed{A(\ell) = L_N(\ell) - L_C(\ell)}. \tag{10}$$

$S > 0$ means aligned transfer yields lower continuation NLL than structural mismatch. $A > 0$ means it yields lower NLL than matched generic transfer on the eligible population.

B.7 Sequential Disruption and Restoration Specification

We disrupt the target computation at an earlier layer and transfer source states at a later layer to measure recovery from the propagated damage. Both evaluated sequential operators zero the registered target endpoint support at a decoder block output. The perturbed trajectory propagates normally to the later restoration layer, where it supplies the common starting state for the restoration branches. Source activations are captured from a separate clean source trajectory at the restoration layer. Aligned and mismatch restoration use their frozen assignment plans, common target positions, and identical scored continuations. CFG also has matched generic restoration on its eligible early to late subset.

CFG has completed the four frozen schedules: offsets $+1$, $+5$, $+10$, and early to late (approximately one quarter to three quarters depth). Offset profiles use all registered admissible layer pairs. The $+1$ schedule includes one subsequent block. The main DFG results use the completed early to late anchor, with zero based (δ, γ) equal to $(9, 31)$ for Gemma, $(7, 23)$ for Llama, and $(6, 20)$ for Qwen. The target-function DFG comparison uses aligned and structural mismatch restoration.

Table 2: **Predictive effects of disruption before restoration.** Damage contrasts for the same configurations and schedules as Figure 4. Entries give means with equal pair weights and individual 95% intervals from the pair bootstrap. Positive $L_D - L_0$ indicates worse continuation prediction; positive $K_D - K_0$ indicates departure from the clean distribution ($K_0 = 0$). The populations contain 1,566 Z_{ctrl} pairs and 3,813 Z_{data} pairs.

		Z_{ctrl} (CFG)		Z_{data} (DFG)	
Model	Setting	$L_D - L_0$	$K_D - K_0$	$L_D - L_0$	$K_D - K_0$
Gemma	Original	0.07529 [0.05195, 0.10445]	0.07670 [0.05252, 0.10569]	0.59694 [0.55368, 0.64611]	0.60009 [0.55645, 0.65030]
	Lexical	0.09212 [0.06100, 0.12882]	0.08594 [0.05043, 0.12835]	0.58178 [0.49421, 0.67216]	0.58304 [0.49460, 0.67291]
	Structural	0.05008 [0.02976, 0.07314]	0.05473 [0.03196, 0.08111]	0.59386 [0.52283, 0.66965]	0.59090 [0.51990, 0.66633]
Llama	Original	0.01821 [−0.00318, 0.03915]	0.04827 [0.03492, 0.06428]	0.60962 [0.53344, 0.69270]	0.67020 [0.59418, 0.75247]
	Lexical	0.03889 [0.01403, 0.06290]	0.03292 [0.01835, 0.05362]	0.43728 [0.28672, 0.62844]	0.44000 [0.29520, 0.62562]
	Structural	0.03385 [−0.00954, 0.08197]	0.07304 [0.04211, 0.11568]	0.83275 [0.63670, 1.04106]	0.94964 [0.75928, 1.14558]
Qwen	Original	0.01154 [−0.00331, 0.02571]	0.03396 [0.02546, 0.04373]	0.49325 [0.36944, 0.63656]	0.55771 [0.43988, 0.69103]
	Lexical	0.02066 [0.00859, 0.03658]	0.01934 [0.00921, 0.03380]	0.40653 [0.26607, 0.60918]	0.39215 [0.25760, 0.59621]
	Structural	0.02034 [0.00873, 0.03444]	0.02426 [0.01516, 0.03460]	0.60592 [0.39219, 0.84434]	0.68097 [0.48562, 0.90533]

B.8 Damage checks before restoration

We measure the predictive deficit caused by disruption to establish the loss available for recovery. Disruption increases mean NLL and KL in all nine models for both CFG and DFG (Table 2). The mean NLL increase is 0.41–0.83 nats/token for DFG and 0.012–0.092 for CFG. The confidence intervals exclude zero for DFG under both metrics and for CFG under KL; some CFG NLL intervals include zero.

C Statistical Specification and CFG Tables

Multiple intervention units and layers share the same program pair. To account for this dependence, we aggregate within pairs and use resampling protocols that retain the relevant pair or family structure. For each stratum, model, pair and layer, membership contrasts are averaged before equal pair weighting. Completeness is assessed separately within each pair's model. The five strata remain separate. We summarize completed runs across all measured layers. Pointwise 95% bands use 2,000 percentile resamples of pair trajectories, base seed 20260917 with deterministic group offsets and sorted pair IDs. Each interval applies to its individual layer estimate. We retain all observations, including outliers. To assess sensitivity to shared source families, we resample whole families while retaining pair weighting. These intervals condition on the nine evaluated models.

CFG uses $\rho = (\ell + 1)/B$ with zero based block index. Frozen primary layers are 20, 16 and 14 for Gemma, Llama and Qwen, respectively. KL computes float32 log softmax over the full vocabulary followed by float64 reduction over the same scoring positions as NLL. NLL/KL concordance jointly resamples source families across all nine environments and depths to retain their dependence. Depth-bin summaries average measured layers within each pair/bin, then weight pairs equally. Mean/median differences in Table 3 document skew.

C.1 Full intact compatibility estimates

Table 3: Frozen midpoint GENERAL compatibility contrast; scientific units are source/generated program pairs. Layer indices are zero based and intervals are pointwise intervals from the pair bootstrap. Family counts are within each environment.

Environment	Layer	Depth	Pairs	Source families	Mean	Median	95% pair bootstrap CI	$S > 0$ fraction
Gemma / lexical	20	0.50000	144	70	0.00656	0.00211	[-0.00371, 0.01670]	0.77083
Gemma / orig	20	0.50000	381	214	0.02351	0.00087	[0.01134, 0.03869]	0.81627
Gemma / structural	20	0.50000	208	111	0.02928	0.00078	[0.01498, 0.04630]	0.75000
Llama / lexical	16	0.53125	81	40	0.00045	0.00038	[-0.00531, 0.00630]	0.56790
Llama / orig	16	0.53125	265	150	0.00339	0.00040	[-0.00112, 0.00816]	0.67170
Llama / structural	16	0.53125	133	75	0.02380	0.00114	[0.00441, 0.05934]	0.69173
Qwen / lexical	14	0.53571	89	39	0.01108	0.00206	[0.00212, 0.02154]	0.66292
Qwen / orig	14	0.53571	163	70	0.01023	0.00227	[-0.01246, 0.03200]	0.73620
Qwen / structural	14	0.53571	102	50	0.01562	0.00229	[-0.01122, 0.04175]	0.77451

Table 4 gives the complete midpoint estimates for CFG and DFG. Aligned DFG transfer lowers mean NLL by 0.120–0.310 nats/token and mean KL by 0.121–0.374 relative to mismatch transfer across the nine models. All confidence intervals exclude zero under both metrics. Aligned CFG transfer also lowers both means in all nine models, by 0.00045–0.0293 nats/token for NLL and 0.0036–0.0334 for KL. Its confidence intervals exclude zero in four models for NLL and seven for KL. Thus, the selected-layer comparisons show a more consistent advantage for DFG, with CFG providing clearer evidence under KL than NLL.

The complementary KL profiles and fixed-layer comparisons appear in Figure 7; the main NLL profiles are in Figure 3. Both metrics use the pointwise intervals over pair trajectories described above. The CFG profiles show that the advantage extends beyond the prespecified comparison layer. Gemma and Llama generally show larger NLL reductions at earlier depths and smaller reductions later, while Qwen shows smaller and less uniform changes. The normalized-depth summaries in Figure 5 describe this sensitivity to intervention depth; they do not locate control-flow information in a particular layer.

NLL and KL compatibility estimates also vary together across models and layers (Figure 6). Their Spearman correlation is 0.91440, with a source-family bootstrap confidence interval of $[0.69672, 0.97381]$. Both metrics summarize the same interventions and share depth dependence; the bootstrap retains this dependence across models and layers.

C.2 Complementary compatibility and recovery figures

Table 4: **Intact compatibility for CFG and DFG.** NLL and KL contrasts at layers 20, 16 and 14 for Gemma, Llama and Qwen, respectively. Entries give means with equal pair weights and individual 95% intervals from the pair bootstrap; positive values favor aligned transfer. Both metrics use the same pairs within each abstraction: 1,566 for CFG and 3,813 for DFG. CFG intervals match the pointwise intervals in the depth profiles in Figure 3.

		Z_{ctrl} (CFG)			Z_{data} (DFG)		
Model	Setting	Pairs	$L_m - L_a$	$K_m - K_a$	Pairs	$L_m - L_a$	$K_m - K_a$
Gemma	Original	381	0.02351 [0.01134, 0.03869]	0.02330 [0.01067, 0.03947]	1160	0.18206 [0.15881, 0.20787]	0.18181 [0.15895, 0.20765]
	Lexical	144	0.00656 [−0.00371, 0.01670]	0.01522 [−0.00214, 0.04108]	586	0.12016 [0.08343, 0.16517]	0.12080 [0.08455, 0.16597]
	Structural	208	0.02928 [0.01498, 0.04630]	0.02762 [0.01403, 0.04375]	600	0.14555 [0.11454, 0.17726]	0.14539 [0.11527, 0.17759]
Llama	Original	265	0.00339 [−0.00112, 0.00816]	0.00690 [0.00345, 0.01087]	590	0.31025 [0.26839, 0.35418]	0.36564 [0.32783, 0.40639]
	Lexical	81	0.00045 [−0.00531, 0.00630]	0.00360 [−0.00022, 0.00839]	136	0.27009 [0.19747, 0.34760]	0.30414 [0.22585, 0.38984]
	Structural	133	0.02380 [0.00441, 0.05934]	0.02572 [0.00708, 0.06034]	204	0.25326 [0.17280, 0.33204]	0.37371 [0.30406, 0.45140]
Qwen	Original	163	0.01023 [−0.01246, 0.03200]	0.03344 [0.01959, 0.05114]	290	0.16828 [0.11161, 0.22754]	0.23878 [0.19913, 0.28631]
	Lexical	89	0.01108 [0.00212, 0.02154]	0.01580 [0.00804, 0.02648]	103	0.20434 [0.13306, 0.27511]	0.21110 [0.14868, 0.26997]
	Structural	102	0.01562 [−0.01122, 0.04175]	0.02535 [0.01259, 0.04246]	144	0.14289 [0.04826, 0.22853]	0.24217 [0.18347, 0.30451]

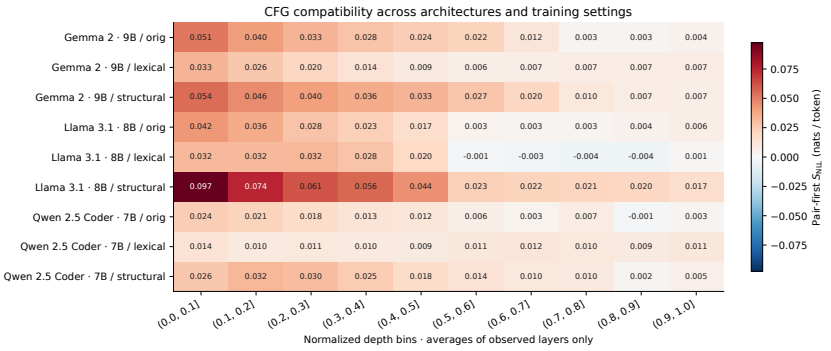


Figure 5: Compatibility contrast across nine environments. Ten common bins of normalized depth summarize measured layers within pairs and then weight pairs equally. The bins describe how signed effects vary across depth, architecture, and training setting.

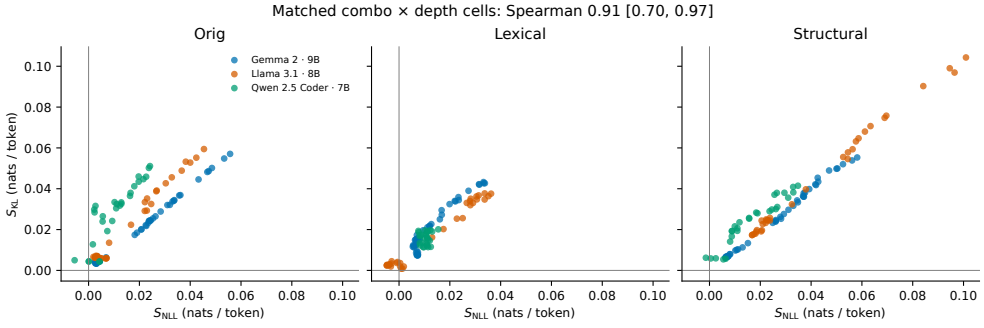


Figure 6: Association between NLL and KL compatibility profiles. Each point is a model–layer mean on the same CFG GENERAL intervention population. The correlation interval resamples source-family trajectories jointly across models and layers.

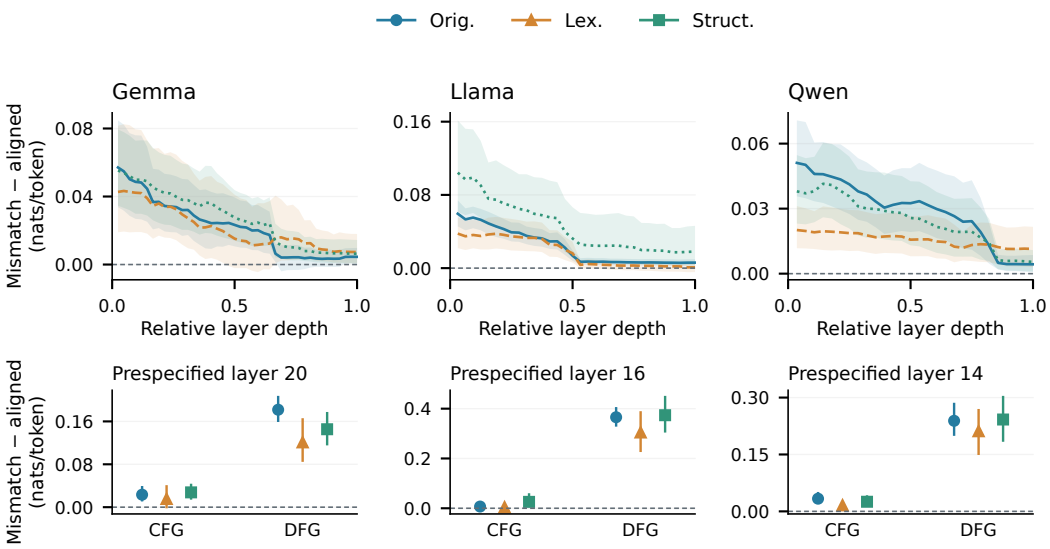


Figure 7: **Intact causal compatibility under KL.** The populations and comparisons match Figure 3; depth profiles are shown above the fixed-layer comparisons here. Positive $K_m - K_a$ favors aligned transfer. Shading and bars show individual 95% pair-bootstrap intervals, pointwise for the depth profiles. Vertical scales differ across panels.

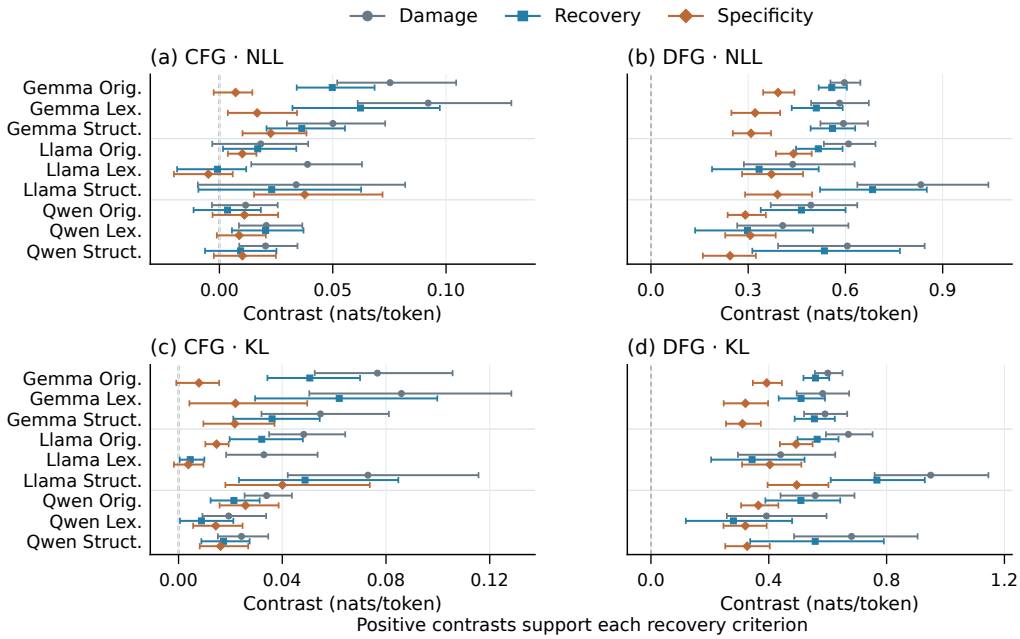


Figure 8: **Absolute paired contrasts underlying causal recovery.** Damage, recovery, and restoration specificity for both NLL and KL, with individual 95% pair-bootstrap intervals. Positive values support the corresponding criterion in Equation 7. Horizontal scales differ across panels.

C.3 DFG RECOVERY AT THE PRESPECIFIED LAYERS

After disruption raises mean NLL by 0.41–0.83 nats/token, aligned DFG restoration recovers 0.30–0.68 nats/token and yields NLL 0.24–0.44 lower than mismatch restoration across the nine models (Table 5). The remaining mean gap to clean prediction is 0.03–0.15 nats/token: restoration recovers much of the loss, but not all of it. Under KL, aligned restoration reduces divergence from the clean distribution by 0.28–0.77 nats/token relative to disruption and by 0.31–0.49 relative to mismatch restoration (Table 6). The confidence intervals for damage, recovery, and restoration specificity exclude zero for every model under both metrics. These comparisons show that preserving data-flow correspondence improves recovery of both continuation likelihood and agreement with the clean predictive distribution.

We estimate compatibility and recovery within each model, using pair resampling to quantify uncertainty. The frozen compatibility layers are 20, 16, and 14 for Gemma, Llama, and Qwen, respectively. Sequential anchors are given in Appendix B.7. The anchor export contains every required condition for every intended case. We average these cases within each program pair, followed by equal pair weighting separately within models. The reported percentile intervals use the imported frozen protocol: 2,000 replacement resamples, NumPy default_rng seed 20260914, pair order by first appearance, and linear 2.5/97.5 percentile interpolation. We report an individual interval for each contrast within each model. Compatibility KL uses float32 reduction per token; sequential KL uses float64 accumulation.

C.4 CFG RECOVERY AND MATCHED GENERIC COMPARISONS

Aligned CFG restoration moves the predictive distribution toward the clean distribution in all nine models, reducing mean KL by 0.0045–0.0620 nats/token relative to disruption. It also yields lower mean KL than mismatch restoration in all nine models. The confidence intervals exclude zero in all nine models for recovery and seven for restoration specificity (Table 8). Under NLL, aligned restoration improves over both disruption and mismatch restoration in eight of nine models, with several confidence intervals including zero (Table 7). Thus, CFG recovery is more consistent for agreement with the clean distribution than for the likelihood of the observed continuation.

We report paired estimates on each eligible population. The original population CFG early to late contrasts retain their separately completed intervals from the pair bootstrap, using the same seed-20260914 protocol as the matched generic analysis. These are distinct from the seed-20260917 trajectory intervals in the full layer intact figures. The family summaries in Table 9 pool program pairs within each model family; the model tables provide individual uncertainty estimates. GENERAL has 1,566 pairs in the original population. Pair counts eligible for generic comparisons are reported separately in Table 10.

Matched generic controls. Matched generic transfer uses alternative contiguous token supports in source code, with the same target positions and write geometry, on its own eligible CFG subset. Aligned transfer yields lower mean NLL than generic transfer in eight of nine models during intact computation and all nine after disruption. The confidence intervals exclude zero in five models for each comparison. Under KL, aligned transfer yields lower means in all nine models during intact computation and after disruption, with every confidence interval excluding zero (Table 10). These comparisons show that the advantage over generic transfer is more consistent under KL than NLL. We assess restoration jointly with the damage and recovery contrasts. Target Function DFG uses structural mismatch as its comparison.

C.5 ILLUSTRATION OF A PROGRAM PAIR AND SELECTION

Figure 10 uses target function Population A pair `CP-5c4de0041d94d07c2712` in Gemma orig, concerning `FaceLandmarkDetectors.unload_models`. The source contains a docstring omitted from the generated program; the displayed code pattern is shared. Retrieval ranks this source first with JPlag similarity 1.0. We use this highly similar pair to illustrate the predictive effects of the intervention conditions.

The pair was selected after observing outcomes. Among the 1,160 Gemma orig pairs, we ranked mean NLL compatibility within pairs, damage, recovery, and restoration specificity, then minimized

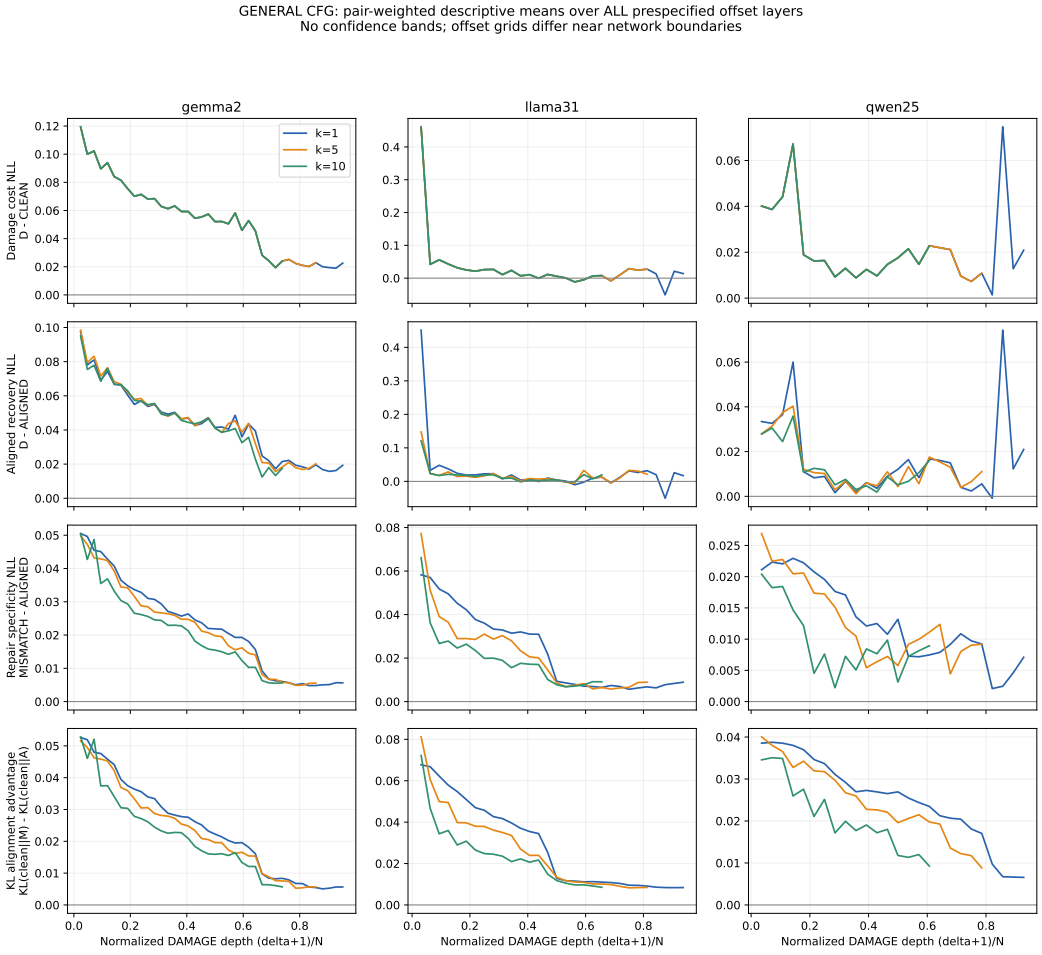


Figure 9: CFG GENERAL offset profiles for $+1$, $+5$, and $+10$. The source plot uses "correct" for aligned restoration and "specificity" for RepairSpecificity. Descriptive family means pool CFG pairs across training settings and compare aligned with structural mismatch restoration.

Table 5: Target Function DFG Population A: NLL means and individual 95% intervals from the pair bootstrap. Compatibility uses the intact anchor; the other contrasts use the early to late schedule. Cases are averaged within pairs, then pairs weighted equally.

Model	Pairs	C	DamageEffect	Recovery	RepairSpecificity
Gemma orig	1160	0.18206 [0.15881, 0.20787]	0.59694 [0.55368, 0.64611]	0.55790 [0.51748, 0.60455]	0.39216 [0.34620, 0.44256]
Gemma lexical	586	0.12016 [0.08343, 0.16517]	0.58178 [0.49421, 0.67216]	0.51005 [0.43426, 0.59142]	0.32137 [0.24823, 0.39860]
Gemma structural	600	0.14555 [0.11454, 0.17726]	0.59386 [0.52283, 0.66965]	0.56004 [0.49258, 0.63006]	0.30871 [0.25250, 0.37055]
Llama orig	590	0.31025 [0.26839, 0.35418]	0.60962 [0.53344, 0.69270]	0.51628 [0.44737, 0.59114]	0.43973 [0.38551, 0.49589]
Llama lexical	136	0.27009 [0.19747, 0.34760]	0.43728 [0.28672, 0.62844]	0.33362 [0.18845, 0.51748]	0.37169 [0.28097, 0.46922]
Llama structural	204	0.25326 [0.17280, 0.33204]	0.83275 [0.63670, 1.04106]	0.68383 [0.52178, 0.85106]	0.39042 [0.29047, 0.49713]
Qwen orig	290	0.16828 [0.11161, 0.22754]	0.49325 [0.36944, 0.63656]	0.46418 [0.33871, 0.60040]	0.29106 [0.23667, 0.35440]
Qwen lexical	103	0.20434 [0.13306, 0.27511]	0.40653 [0.26607, 0.60918]	0.29787 [0.13626, 0.49967]	0.30646 [0.22916, 0.38497]
Qwen structural	144	0.14289 [0.04826, 0.22853]	0.60592 [0.39219, 0.84434]	0.53499 [0.31273, 0.76813]	0.24416 [0.16017, 0.32370]

Table 6: Target Function DFG Population A: KL contrasts and individual 95% intervals from the pair bootstrap. Conditions and populations match the NLL table; KL is $\mathrm{KL}(P_0 \| P_X)$ over the full vocabulary.

Model	Pairs	C	DamageEffect	Recovery	RepairSpecificity
Gemma orig	1160	0.18181 [0.15895, 0.20765]	0.60009 [0.55645, 0.65030]	0.55867 [0.51776, 0.60537]	0.39306 [0.34592, 0.44445]
Gemma lexical	586	0.12080 [0.08455, 0.16597]	0.58304 [0.49460, 0.67291]	0.50958 [0.43364, 0.59082]	0.32069 [0.24695, 0.39770]
Gemma structural	600	0.14539 [0.11527, 0.17759]	0.59090 [0.51990, 0.66633]	0.55461 [0.48783, 0.62433]	0.31087 [0.25471, 0.37305]
Llama orig	590	0.36564 [0.32783, 0.40639]	0.67020 [0.59418, 0.75247]	0.56411 [0.49764, 0.63673]	0.49257 [0.43771, 0.54871]
Llama lexical	136	0.30414 [0.22585, 0.38984]	0.44000 [0.29520, 0.62562]	0.34336 [0.20438, 0.52142]	0.40375 [0.30943, 0.51060]
Llama structural	204	0.37371 [0.30406, 0.45140]	0.94964 [0.75928, 1.14558]	0.76716 [0.61022, 0.92968]	0.49473 [0.39591, 0.60250]
Qwen orig	290	0.23878 [0.19913, 0.28631]	0.55771 [0.43988, 0.69103]	0.50911 [0.38843, 0.64262]	0.36473 [0.30593, 0.43272]
Qwen lexical	103	0.21110 [0.14868, 0.26997]	0.39215 [0.25760, 0.59621]	0.27975 [0.11843, 0.47912]	0.32044 [0.24621, 0.39290]
Qwen structural	144	0.24217 [0.18347, 0.30451]	0.68097 [0.48562, 0.90533]	0.55753 [0.33687, 0.79058]	0.32657 [0.25226, 0.40353]

the sum of absolute deviations of their percentile ranks from 0.5, breaking ties by pair ID. Its four percentile ranks are 0.5091, 0.4823, 0.5065, and 0.5160. Population estimates use all eligible pairs at the prespecified layer schedules.

Each displayed condition is a descriptive mean over all 16 eligible joint two-relation interventions in the pair. The code panel shows the grounding of two relations in one unit, with both relations sharing the definition of `keep_essential`; equal definition writes are coalesced. The three token positions per endpoint are shown in the frozen native inputs. The scoring suffix has 38 tokens, beginning with `(model_name)` in the preceding removal call and continuing through the final `if not keep_essential` block. The displayed excerpt omits intervening code and comments and normalizes indentation; intervention and scoring coordinates remain those of the complete serialized inputs.

Table 7: CFG GENERAL early to late NLL: original population means and 95% intervals from the pair bootstrap for all nine models.

Model	Pairs	DamageEffect	Recovery	RepairSpecificity
Gemma orig	381	0.07529 [0.05195, 0.10445]	0.04977 [0.03409, 0.06846]	0.00713 [−0.00253, 0.01451]
Gemma lexical	144	0.09212 [0.06100, 0.12882]	0.06224 [0.03222, 0.09724]	0.01665 [0.00370, 0.03424]
Gemma structural	208	0.05008 [0.02976, 0.07314]	0.03636 [0.02078, 0.05539]	0.02262 [0.01018, 0.03842]
Llama orig	265	0.01821 [−0.00318, 0.03915]	0.01690 [0.00156, 0.03390]	0.01009 [0.00374, 0.01632]
Llama lexical	81	0.03889 [0.01403, 0.06290]	−0.00090 [−0.01869, 0.01182]	−0.00489 [−0.02009, 0.00586]
Llama structural	133	0.03385 [−0.00954, 0.08197]	0.02308 [−0.00927, 0.06250]	0.03761 [0.01528, 0.07197]
Qwen orig	163	0.01154 [−0.00331, 0.02571]	0.00359 [−0.01142, 0.01828]	0.01101 [−0.00305, 0.02591]
Qwen lexical	89	0.02066 [0.00859, 0.03658]	0.02040 [0.00552, 0.03710]	0.00873 [−0.00115, 0.02049]
Qwen structural	102	0.02034 [0.00873, 0.03444]	0.00936 [−0.00637, 0.02516]	0.01013 [−0.00248, 0.02491]

Table 8: CFG GENERAL early to late KL: original population means and individual 95% intervals from the pair bootstrap, on the same pairs as the NLL table.

Model	Pairs	DamageEffect	Recovery	RepairSpecificity
Gemma orig	381	0.07670 [0.05252, 0.10569]	0.05063 [0.03427, 0.07000]	0.00788 [−0.00087, 0.01566]
Gemma lexical	144	0.08594 [0.05043, 0.12835]	0.06196 [0.02952, 0.09979]	0.02196 [0.00417, 0.04964]
Gemma structural	208	0.05473 [0.03196, 0.08111]	0.03606 [0.02112, 0.05445]	0.02171 [0.00958, 0.03700]
Llama orig	265	0.04827 [0.03492, 0.06428]	0.03211 [0.01970, 0.04795]	0.01466 [0.01033, 0.01931]
Llama lexical	81	0.03292 [0.01835, 0.05362]	0.00451 [0.00045, 0.00996]	0.00377 [−0.00178, 0.00959]
Llama structural	133	0.07304 [0.04211, 0.11568]	0.04878 [0.02330, 0.08477]	0.04006 [0.01806, 0.07375]
Qwen orig	163	0.03396 [0.02546, 0.04373]	0.02135 [0.01236, 0.03129]	0.02583 [0.01583, 0.03860]
Qwen lexical	89	0.01934 [0.00921, 0.03380]	0.00882 [0.00055, 0.02113]	0.01435 [0.00563, 0.02469]
Qwen structural	102	0.02426 [0.01516, 0.03460]	0.01737 [0.00880, 0.02740]	0.01617 [0.00817, 0.02685]

In this pair, aligned intact transfer keeps mean NLL near the clean value (0.00388 versus 0.00371 nats/token), whereas mismatch transfer raises it to 0.07916. Disruption raises mean NLL to 0.34746; aligned restoration reduces it to 0.00474, compared with 0.15218 after mismatch restoration. The remaining mean gap to clean prediction is 0.00103 nats/token. Thus, aligned restoration brings prediction close to the clean condition and recovers more than mismatch restoration in this illustrative pair. The intact write is at layer 20; damage and restoration occur at layers 9 and 31 (zero based).

Table 9: CFG GENERAL early to late NLL: descriptive family summaries pooling program pairs across the three training settings. Means give pairs equal weight, in nats/token. Table 7 reports uncertainty for each setting.

Family	Pairs	DamageEffect	Recovery	RepairSpecificity
Gemma	733	0.07144	0.04842	0.01340
Llama	479	0.02605	0.01561	0.01520
Qwen	354	0.01637	0.00947	0.01018

Table 10: CFG GENERAL matched generic specificity. A uses the frozen intact midpoint and A_R the early to late restoration schedule. Means and 95% intervals from the pair bootstrap are copied from the completed eligible population analysis. The listed pair counts apply to each reported contrast, with condition differences formed within the corresponding eligible slice.

Model	Pairs	A_{NLL}	$A_{R,\mathrm{NLL}}$	A_{KL}	$A_{R,\mathrm{KL}}$
Gemma orig	367	0.08307 [0.05028, 0.12070]	0.07355 [0.04573, 0.10580]	0.08102 [0.04881, 0.11759]	0.07297 [0.04485, 0.10547]
Gemma lexical	140	0.10164 [0.04649, 0.16565]	0.10067 [0.03656, 0.17570]	0.09874 [0.04482, 0.16231]	0.09906 [0.03565, 0.17424]
Gemma structural	198	0.05727 [0.02727, 0.09483]	0.03879 [0.02049, 0.05960]	0.06070 [0.03050, 0.09801]	0.04025 [0.02196, 0.06086]
Llama orig	245	0.03832 [0.00743, 0.07736]	0.05391 [0.02167, 0.09364]	0.05478 [0.02730, 0.09148]	0.06727 [0.03792, 0.10628]
Llama lexical	79	0.02051 [−0.00712, 0.04530]	0.00899 [−0.01383, 0.02893]	0.01616 [0.00767, 0.02676]	0.01322 [0.00412, 0.02523]
Llama structural	128	−0.00133 [−0.05128, 0.04952]	0.03435 [−0.01042, 0.08423]	0.05302 [0.02195, 0.09608]	0.06625 [0.03205, 0.11135]
Qwen orig	161	0.01001 [−0.00460, 0.02512]	0.01013 [−0.00506, 0.02781]	0.02872 [0.01744, 0.04174]	0.02286 [0.01048, 0.03927]
Qwen lexical	88	0.04439 [0.01582, 0.08119]	0.04163 [0.01188, 0.08081]	0.04146 [0.02013, 0.07557]	0.03316 [0.01188, 0.06871]
Qwen structural	101	0.00927 [−0.02369, 0.03525]	0.01244 [−0.00600, 0.02990]	0.03236 [0.01913, 0.04779]	0.02325 [0.01168, 0.03680]

Full-precision condition records, grounding, code, and selection metadata are preserved with the figure evidence.

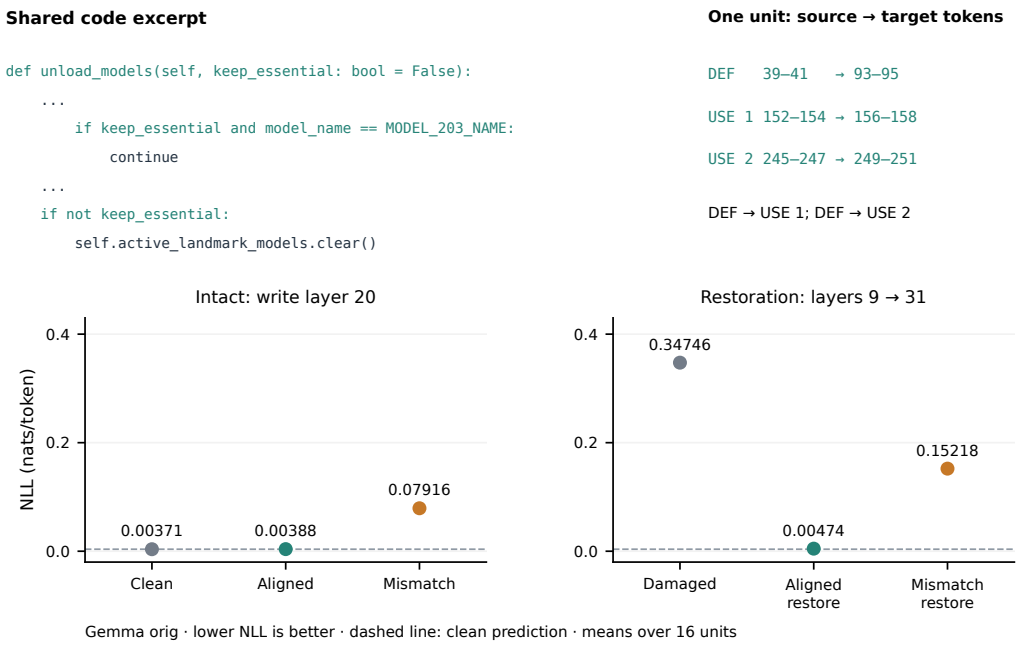


Figure 10: **One program pair under intact transfer and restoration.** Top: a shared code excerpt and the token grounding of two definition–use relations from one intervention unit; omitted code is marked by ellipses. Bottom: descriptive condition NLL averaged over all 16 eligible joint units in this pair. The dashed lines mark clean predictions. Layers are zero based: intact write 20; damage 9 and restoration 31. The Gemma orig pair was selected post hoc near the median ranks of four pair-level contrasts to illustrate a central observed case.

C.6 Alignment Comparisons at Selected Layers

We compare source-state selection methods on the target positions and write constraints specified by PSCA. Source–target pairs, native input serializations, model identities, scored tokens and write layers are the same across methods. We use Gemma layer 20, Llama layer 16 and Qwen layer 14 (zero based decoder block outputs). This design measures the predictive effect of source selection while using the same target positions across methods.

PSCA yields lower mean NLL than both same-offset and constrained-random transfer in all nine models for CFG and DFG (Tables 11 and 12). For DFG, the reduction ranges from 0.428 to 0.776 nats/token across models and the two alternatives. The paired confidence intervals exclude zero for every DFG comparison and most CFG comparisons. These results show that program correspondence guides source selection beyond relative token position and feasible random assignment.

Assignment rules and common populations. Same offset selection translates target offsets from the start of the registered code domain to the source domain and accepts exact valid positions. Constrained random selection uses seeded shuffled candidate depth first search and takes the first feasible joint assignment. The seed is 20260914; this procedure defines a seeded feasible assignment over the admissible joint assignments. Both methods preserve ordered endpoint widths, internal offsets, shared source slots and the constraints preventing distinct source slots from collapsing. An assignment must remain in the allowed serialized code domain. CFG uses the selected source code components; DFG uses the complete target function domains. The neural selector informed by clean states below uses the same constraints.

Every included pair retains all its intended units and complete results for all three baseline methods. The common sets comprise 1,279 CFG GENERAL pairs and 3,621 DFG pairs, drawn from the original 1,566 and 3,813 pairs. Same offset eligibility restricts these comparison populations. Each pair contributes its complete intended unit set. All estimates average within pairs and then give pairs equal weight. The existing NLL and DFG KL intervals are retained. CFG KL intervals use the same pair bootstrap with 2,000 resamples, seed 20260914, and the frozen first-appearance pair order. All intervals are individual 95% intervals. Table entries give raw condition means and paired contrasts $\Delta = L_{\text{baseline}} - L_a$, where L_a is the NLL under PSCA's aligned intervention as in Section 2.3; Each interval quantifies uncertainty in its paired contrast.

For CFG KL, we evaluate PSCA and the baselines with the same GPU scorer and physical GPU within each comparison. We combine completed results at the selected layers with additional anchor evaluations using that scorer and the frozen assignments. The resulting cohort contains 1,279 pairs and all 5,996 intended memberships. Identical assignments produce identical scores, and the NLL means and intervals agree with the original estimates to numerical precision.

KL comparison. PSCA also yields lower mean KL than same-offset and constrained-random transfer in all nine models for both CFG and DFG (Tables 13 and 14). The confidence intervals exclude zero for every comparison. Together with NLL, these results show that program-defined transfers better support the observed continuation and preserve closer agreement with the clean predictive distribution than the two alternatives. For DFG, NLL and KL scores for identical assignments reproduce exactly across the original PSCA and baseline executions. The baseline KL comparison uses GPU-scored values for all methods. The CFG aligned–mismatch analyses retain their original scoring path.

Table 11: CFG anchor comparison on pairs complete across methods: raw NLL and paired contrasts. Positive differences favor PSCA. Individual 95% intervals resample pairs; parentheses give pair counts.

Model (pairs)	PSCA	Same-offset	Δ Same [95% CI]	Random	Δ Random [95% CI]
Gemma orig (329)	0.034615	0.105754	+0.071139 [+0.044642, +0.102439]	0.105194	+0.070580 [+0.043792, +0.104447]
Gemma lexical (128)	0.140034	0.214630	+0.074596 [+0.032588, +0.125156]	0.181098	+0.041064 [+0.018053, +0.067586]
Gemma structural (149)	0.050297	0.117255	+0.066958 [+0.030425, +0.111717]	0.104521	+0.054224 [+0.028237, +0.084940]
Llama orig (224)	0.292294	0.319743	+0.027449 [+0.003466, +0.061028]	0.324121	+0.031827 [+0.002863, +0.077680]
Llama lexical (70)	0.652981	0.681872	+0.028890 [+0.007557, +0.051290]	0.665282	+0.012300 [−0.016127, +0.035246]
Llama structural (96)	0.465030	0.493710	+0.028681 [−0.025350, +0.093945]	0.485083	+0.020053 [−0.027033, +0.074059]
Qwen orig (131)	0.206163	0.221644	+0.015481 [+0.004578, +0.027859]	0.231602	+0.025439 [+0.014628, +0.037889]
Qwen lexical (74)	0.413651	0.449473	+0.035822 [+0.012918, +0.057791]	0.451477	+0.037826 [+0.018394, +0.057837]
Qwen structural (78)	0.237576	0.264747	+0.027171 [+0.008884, +0.048712]	0.268244	+0.030668 [+0.015470, +0.048694]

Table 12: DFG anchor comparison on pairs complete across methods: raw NLL and paired contrasts. The estimators and interval definitions match Table 11.

Model (pairs)	PSCA	Same-offset	Δ Same [95% CI]	Random	Δ Random [95% CI]
Gemma orig (1135)	0.156581	0.827050	+0.670469 [+0.591649, +0.758190]	0.767526	+0.610946 [+0.547083, +0.678833]
Gemma lexical (570)	0.262706	0.690691	+0.427985 [+0.311875, +0.557785]	0.695886	+0.433180 [+0.328521, +0.546955]
Gemma structural (584)	0.122021	0.763297	+0.641275 [+0.535514, +0.755943]	0.747776	+0.625755 [+0.516842, +0.745019]
Llama orig (555)	0.461752	1.120510	+0.658758 [+0.582886, +0.739330]	1.139815	+0.678062 [+0.590209, +0.773033]
Llama lexical (130)	0.472071	1.011508	+0.539438 [+0.350774, +0.823101]	0.975530	+0.503459 [+0.348104, +0.730530]
Llama structural (176)	0.789338	1.339892	+0.550554 [+0.408744, +0.701505]	1.355058	+0.565720 [+0.435199, +0.704564]
Qwen orig (259)	0.450616	1.049385	+0.598769 [+0.472933, +0.737218]	1.011083	+0.560467 [+0.428994, +0.721783]
Qwen lexical (89)	0.366021	1.064882	+0.698861 [+0.446808, +1.090993]	1.047436	+0.681415 [+0.434521, +1.060385]
Qwen structural (123)	0.808677	1.584394	+0.775717 [+0.517356, +1.066015]	1.530710	+0.722033 [+0.456917, +1.071659]

Table 13: CFG anchor comparison under $\mathrm{KL}(P_0 \| P_X)$ on the same 1,279 complete pairs used for NLL. All three assignments use the same GPU scorer; positive paired differences favor PSCA. Intervals are individual 95% pair-bootstrap intervals.

Model (pairs)	PSCA	Same-offset	Δ Same [95% CI]	Random	Δ Random [95% CI]
Gemma orig (329)	0.010608	0.085723	+0.075114 [+0.048285, +0.106808]	0.082549	+0.071940 [+0.044761, +0.106169]
Gemma lexical (128)	0.011642	0.101122	+0.089479 [+0.045597, +0.142438]	0.053196	+0.041553 [+0.020486, +0.067760]
Gemma structural (149)	0.008048	0.083326	+0.075278 [+0.038424, +0.119850]	0.063289	+0.055241 [+0.030821, +0.084622]
Llama orig (224)	0.002659	0.044872	+0.042213 [+0.020107, +0.073479]	0.049194	+0.046535 [+0.019138, +0.090427]
Llama lexical (70)	0.005887	0.019596	+0.013709 [+0.006176, +0.023804]	0.019179	+0.013292 [+0.006532, +0.022659]
Llama structural (96)	0.001767	0.059588	+0.057821 [+0.018715, +0.122035]	0.051225	+0.049457 [+0.017716, +0.099410]
Qwen orig (131)	0.006848	0.033395	+0.026547 [+0.016039, +0.039694]	0.041055	+0.034207 [+0.023475, +0.048225]
Qwen lexical (74)	0.002975	0.030378	+0.027403 [+0.013187, +0.046529]	0.029238	+0.026263 [+0.016136, +0.039244]
Qwen structural (78)	0.005155	0.041334	+0.036178 [+0.020145, +0.055645]	0.038917	+0.033762 [+0.020644, +0.049607]

Table 14: DFG anchor comparison on pairs complete across methods under $\mathrm{KL}(P_0 \| P_X)$. Raw means and paired differences use the same pairs as the NLL comparison; intervals are individual 95% intervals from the pair bootstrap.

Model (pairs)	PSCA	Same-offset	Δ Same [95% CI]	Random	Δ Random [95% CI]
Gemma orig (1135)	0.114108	0.788190	+0.674082 [+0.595159, +0.762937]	0.726140	+0.612032 [+0.547435, +0.679851]
Gemma lexical (570)	0.208482	0.641893	+0.433411 [+0.316548, +0.563561]	0.646910	+0.438428 [+0.333282, +0.552403]
Gemma structural (584)	0.071623	0.717575	+0.645952 [+0.539990, +0.761893]	0.701619	+0.629996 [+0.520785, +0.748808]
Llama orig (555)	0.014585	0.729263	+0.714677 [+0.639492, +0.791556]	0.756005	+0.741420 [+0.655273, +0.837997]
Llama lexical (130)	0.030045	0.600667	+0.570622 [+0.378704, +0.858837]	0.579062	+0.549017 [+0.384842, +0.783262]
Llama structural (176)	0.009170	0.670680	+0.661510 [+0.522322, +0.815213]	0.692168	+0.682998 [+0.553350, +0.821203]
Qwen orig (259)	0.036631	0.713593	+0.676962 [+0.558043, +0.811763]	0.683148	+0.646518 [+0.528879, +0.803685]
Qwen lexical (89)	0.044657	0.752585	+0.707928 [+0.462468, +1.092312]	0.736815	+0.692158 [+0.460882, +1.056055]
Qwen structural (123)	0.054645	0.962652	+0.908007 [+0.668349, +1.182209]	0.916441	+0.861796 [+0.617188, +1.202704]

C.7 CLONE-SPAN COMPARATOR

The clone-span comparator reuses the registered clone fragments and the endpoint-preserving monotone nearest-neighbor mapping between source and target span tokens. We restrict this mapping to the same target positions and number of written states used by PSCA. The model, target input, scoring tokens, intervention layer, and use of full-vector writes are the same across methods. CFG uses the registered component serialization. For DFG, a mapped source token is accepted in the complete function serialization only when its source-character interval and token identity match uniquely. Uncovered target positions or unprojectable source tokens make the unit ineligible. The inherited clone mapping permits source-token reuse and omission. This comparator measures span-based source selection at the same target positions as PSCA using full-vector state transfer.

A pair enters the comparison only if every intended unit has a valid clone mapping. This yields all 1,566 CFG GENERAL pairs and 394 of the 3,813 DFG pairs. The DFG counts for Gemma original, lexical, and structural are 127, 124, and 106; the corresponding Llama counts are 10, 3, and 6, and Qwen counts are 9, 2, and 7. Most exclusions arise because DFG target positions extend beyond the registered clone anchor. These are distinct populations from the same-offset-complete sets above. PSCA and clone are evaluated contemporaneously on the same included units; units are averaged within each pair before giving pairs equal weight. Tables 15 and 16 report the prespecified midpoint layers from the completed full-grid run, with 2,000 pair-bootstrap resamples, seed 20260914, and frozen first-appearance pair order. Each 95% interval applies to its individual contrast.

For CFG, PSCA yields lower mean NLL and KL than clone-span transfer in all nine models. The confidence intervals exclude zero in five models for NLL and all nine for KL. Program structure therefore provides useful source selection beyond the correspondence supplied by clone spans.

For DFG, PSCA yields lower mean NLL in four models and lower mean KL in seven. Clone-span transfer lowers NLL but increases KL relative to PSCA for Llama original and structural, showing that greater continuation likelihood need not imply closer agreement with the clean distribution. Under NLL, confidence intervals support PSCA in two models and clone-span transfer in four; under KL, they support PSCA in four and clone-span transfer in one. Thus, clone-span transfer can be competitive on the eligible DFG pairs, with its relative benefit depending on the model and metric.

Table 15: CFG clone-span comparison at the prespecified midpoint on 1,566 complete pairs. PSCA and clone means use the same pairs; $\Delta = \text{Clone} - \text{PSCA}$ is reported with individual 95% pair-bootstrap intervals for NLL and KL. Positive differences favor PSCA. Parentheses give model pair counts.

	NLL			KL		
Model (pairs)	PSCA	Clone	Δ [95% CI]	PSCA	Clone	Δ [95% CI]
Gemma orig (381)	0.038247	0.088053	0.049806 [0.034549, 0.065931]	0.014350	0.062749	0.048400 [0.033038, 0.064386]
Gemma lexical (144)	0.143597	0.186757	0.043160 [0.020646, 0.066066]	0.023995	0.065547	0.041551 [0.020719, 0.062852]
Gemma structural (208)	0.043937	0.112318	0.068381 [0.035569, 0.108704]	0.006398	0.075673	0.069276 [0.036677, 0.109687]
Llama orig (265)	0.331864	0.342131	0.010267 [−0.004806, 0.027932]	0.002409	0.030184	0.027774 [0.015877, 0.042618]
Llama lexical (81)	0.608328	0.625924	0.017596 [−0.004866, 0.037796]	0.005559	0.015469	0.009910 [0.005659, 0.015275]
Llama structural (133)	0.705587	0.711356	0.005769 [−0.036892, 0.056539]	0.001426	0.050188	0.048762 [0.018487, 0.090976]
Qwen orig (163)	0.369182	0.388342	0.019160 [0.003296, 0.040540]	0.011267	0.040737	0.029470 [0.015267, 0.049201]
Qwen lexical (89)	0.374403	0.430138	0.055735 [0.023591, 0.099137]	0.003687	0.052320	0.048633 [0.019837, 0.090797]
Qwen structural (102)	0.290085	0.301853	0.011768 [−0.011721, 0.034794]	0.004834	0.032024	0.027191 [0.014709, 0.043738]

Table 16: DFG clone-span comparison at the prespecified midpoint on 394 complete pairs. The definitions match Table 15. The Llama and Qwen populations contain 2–10 pairs per model.

	NLL			KL		
Model (pairs)	PSCA	Clone	Δ [95% CI]	PSCA	Clone	Δ [95% CI]
Gemma orig (127)	0.266914	0.257972	-0.008941 $[-0.077062, 0.059382]$	0.075209	0.085611	0.010402 $[-0.052389, 0.076492]$
Gemma lexical (124)	0.073694	0.089942	0.016248 $[-0.042279, 0.065122]$	0.026721	0.051540	0.024819 $[-0.030097, 0.071126]$
Gemma structural (106)	0.048012	0.069405	0.021392 $[-0.035338, 0.067150]$	0.030497	0.049723	0.019226 $[-0.038413, 0.064756]$
Llama orig (10)	3.016436	2.848467	-0.167969 $[-0.264146, -0.081238]$	0.000458	0.001331	0.000873 $[0.000576, 0.001211]$
Llama lexical (3)	0.000372	0.000713	0.000340 $[5.94 \times 10^{-5}, 0.000775]$	4.56×10^{-7}	0.000104	0.000103 $[1.94 \times 10^{-5}, 0.000219]$
Llama structural (6)	4.775135	4.441246	-0.333889 $[-0.443950, -0.204773]$	0.003050	0.006445	0.003396 $[0.001167, 0.006332]$
Qwen orig (9)	2.445801	2.270503	-0.175298 $[-0.284300, -0.071073]$	0.049426	0.017060	-0.032366 $[-0.069327, 0.001074]$
Qwen lexical (2)	0.003779	0.005428	0.001649 $[0.001239, 0.002059]$	1.96×10^{-5}	0.000717	0.000698 $[0.000353, 0.001043]$
Qwen structural (7)	2.949699	2.755493	-0.194206 $[-0.286410, -0.094125]$	0.062128	0.013998	-0.048130 $[-0.088683, -0.011574]$

Table 17: CFG: source selection by cosine similarity to clean states versus PSCA, on the common complete anchor subset. $\Delta = L_{\text{cosine}} - L_a$; negative differences favor cosine. Raw means and original individual 95% intervals from the pair bootstrap are retained.

Model (pairs)	PSCA	Comparator	Δ [95% CI]
Gemma orig (329)	0.034615	0.027873	-0.006742 $[-0.011684, -0.002243]$
Gemma lexical (128)	0.140034	0.137264	-0.002770 $[-0.008361, +0.000767]$
Gemma structural (149)	0.050297	0.049727	-0.000570 $[-0.001473, -0.000002]$
Llama orig (224)	0.292294	0.292849	$+0.000555$ $[-0.000254, +0.001923]$
Llama lexical (70)	0.652981	0.653020	$+0.000039$ $[-0.000043, +0.000129]$
Llama structural (96)	0.465030	0.466483	$+0.001453$ $[-0.000123, +0.004338]$
Qwen orig (131)	0.206163	0.203359	-0.002805 $[-0.005269, -0.000731]$
Qwen lexical (74)	0.413651	0.413643	-0.000008 $[-0.000147, +0.000126]$
Qwen structural (78)	0.237576	0.236980	-0.000596 $[-0.002850, +0.001225]$

C.8 Neural Comparator Informed by Clean States

We select source positions by cosine similarity to clean target states to evaluate a comparator that uses neural information. It inherits PSCA's receiver supports, endpoint widths, ordered offsets and alias constraints, then enumerates legal source supports within the same code domains. An exact constrained search maximizes the sum of cosine similarities between tokens in corresponding order over logical endpoint occurrences. The PSCA assignment is an admissible candidate. Candidate source positions may carry different semantic DEF/USE roles. The mapping is saved before intervention and selected from the cosine scores before intervention NLL or KL are measured. The comparator uses clean target states at the intervention layer, including the later write layer in sequential experiments.

Tables 17–19 retain all nine models on the same populations complete across methods as the comparisons using simple rules. Cosine-based selection yields lower mean NLL than PSCA, with confidence intervals excluding zero, for CFG Gemma original, Gemma structural and Qwen original, and for DFG Gemma original and Qwen original. The remaining NLL comparisons have confidence intervals that include zero.

An assignment audit on the original scientific memberships found exact agreement with PSCA in 90.40% of CFG GENERAL intact cases and 72.34% of DFG Population A intact cases; corresponding restoration agreement was 90.83% and 88.18%. These rates summarize assignment agreement on the original populations; pair win rates on the subsets complete across methods are reported separately. Identical assignments produce identical NLL; the observed comparison therefore combines many identical interventions with a smaller set of mapping differences. Cosine selection uses the model's layer-specific clean states; PSCA selection uses program relations.

Table 18: DFG: source selection by cosine similarity to clean states versus PSCA under NLL. The pairs complete across methods match Table 12.

Model (pairs)	PSCA	Comparator	Δ [95% CI]
Gemma orig (1135)	0.156581	0.137147	-0.019434 $[-0.038957, -0.003789]$
Gemma lexical (570)	0.262706	0.252967	-0.009740 $[-0.027783, +0.003178]$
Gemma structural (584)	0.122021	0.123243	$+0.001222$ $[-0.009728, +0.011478]$
Llama orig (555)	0.461752	0.460650	-0.001102 $[-0.003563, +0.000603]$
Llama lexical (130)	0.472071	0.470368	-0.001702 $[-0.004928, +0.001186]$
Llama structural (176)	0.789338	0.789585	$+0.000247$ $[-0.000013, +0.000576]$
Qwen orig (259)	0.450616	0.435109	-0.015507 $[-0.041039, -0.001514]$
Qwen lexical (89)	0.366021	0.352945	-0.013076 $[-0.039569, +0.000462]$
Qwen structural (123)	0.808677	0.798803	-0.009874 $[-0.029040, +0.004790]$

Table 19: DFG: donor selection by cosine similarity to clean states versus PSCA under $\mathrm{KL}(P_0 \| P_X)$. The comparison uses the same pairs complete across methods and original individual 95% intervals from the pair bootstrap.

Model (pairs)	PSCA	Comparator	Δ [95% CI]
Gemma orig (1135)	0.114108	0.094702	-0.019405 $[-0.038816, -0.003724]$
Gemma lexical (570)	0.208482	0.198714	-0.009767 $[-0.027844, +0.003177]$
Gemma structural (584)	0.071623	0.072960	$+0.001337$ $[-0.009761, +0.011942]$
Llama orig (555)	0.014585	0.013052	-0.001533 $[-0.004256, -0.000023]$
Llama lexical (130)	0.030045	0.029837	-0.000207 $[-0.002652, +0.002362]$
Llama structural (176)	0.009170	0.009154	-0.000017 $[-0.000042, +0.000003]$
Qwen orig (259)	0.036631	0.021855	-0.014776 $[-0.039780, -0.001051]$
Qwen lexical (89)	0.044657	0.031871	-0.012786 $[-0.039013, +0.000678]$
Qwen structural (123)	0.054645	0.040567	-0.014078 $[-0.030758, -0.001058]$

Table 20: Retrieval associations with GENERAL midpoint compatibility contrast. Entries include denominators for each environment and intervals from the source-family bootstrap. Values pooled across environments are descriptive associations.

Environment	Predictor	Pairs	Queries	Estimate	95% CI lower	95% CI upper
All environments (descriptive)	rank	1566	819	-0.13274	-0.19171	-0.06752
All environments (descriptive)	similarity	1566	819	-0.00044	-0.09163	0.10684
Gemma / lexical	rank	144	70	-0.09901	-0.23806	0.03029
Gemma / lexical	similarity	144	70	-0.12791	-0.27957	0.06605
Gemma / orig	rank	381	214	-0.21040	-0.32539	-0.07570
Gemma / orig	similarity	381	214	0.07473	-0.06775	0.20708
Gemma / structural	rank	208	111	-0.27784	-0.38771	-0.15556
Gemma / structural	similarity	208	111	-0.08199	-0.27934	0.11678
Llama / lexical	rank	81	40	-0.16928	-0.35090	0.00916
Llama / lexical	similarity	81	40	0.09586	-0.14809	0.35708
Llama / orig	rank	265	150	-0.08072	-0.19412	0.03370
Llama / orig	similarity	265	150	-0.08881	-0.20809	0.03646
Llama / structural	rank	133	75	-0.04366	-0.20309	0.12019
Llama / structural	similarity	133	75	-0.12900	-0.32402	0.09309
Qwen / lexical	rank	89	39	-0.16325	-0.35158	0.03701
Qwen / lexical	similarity	89	39	-0.00242	-0.31435	0.29940
Qwen / orig	rank	163	70	-0.09480	-0.21639	0.03448
Qwen / orig	similarity	163	70	0.17360	-0.01991	0.35717
Qwen / structural	rank	102	50	-0.06619	-0.24786	0.11161
Qwen / structural	similarity	102	50	0.05305	-0.21677	0.34083

D Similarity Associations and Coverage

We join retrieval scores to measured intervention effects to assess how retrieval order relates to causal compatibility. The analysis uses the frozen midpoint and exact JPlag scores and ranks, joined by model and pair. For the 1,566 GENERAL pairs, pooled Spearman association between NLL compatibility and retrieval similarity is -0.00044 (95% bootstrap interval over source families $[-0.09163, 0.10684]$); the association with retrieval rank is -0.13274 ($[-0.19171, -0.06752]$). These are descriptive associations across measured candidates, with shared queries and source families. Associations within queries and comparisons of candidate order follow below. Rankings derived from similarity partially agree with measured compatibility in this population.

We compute every association from pairs with measured intervention effects. Correlations across candidates use bootstrap over source families preserving source programs and settings. Pooled values are descriptive and can reflect differences between queries. Spearman correlation within queries requires at least three measured source programs and nonconstant variables: the mean is -0.09850 over 189 eligible queries for rank and 0.19153 over 128 for similarity. These descriptive means use their respective eligible query subsets. Rank profiles average the available pairs separately at each rank. A comparison restricted to complete queries is reported separately below.

A lower rank exceeds rank 1 only when its estimate is greater by more than 10^{-12} nats/token; ties remain explicit. The population with five complete candidates has 93 wins by candidates at lower ranks, 25 unique rank-1 maxima and one top tie (119 queries). The population with at least two candidates has 184 wins by candidates at lower ranks, 102 unique rank-1 maxima and one tie (287 queries). These counts summarize observed candidate maxima; their sampling variation depends on the number of available candidates and the uncertainty of each estimate.

E Historical DFG Scope and Serialization Sensitivity

We evaluate three input serializations on the historical Factorial192 component population to measure their effect on DFG compatibility. The preserved plot subscript P denotes implementation label `PERMUTED`, which is STRUCTURAL MISMATCH (M); C denotes ALIGNED. Thus the plotted $\mathrm{NLL}_P - \mathrm{NLL}_C$ is the same compatibility contrast $L_M - L_C$. We report Factorial192 separately from target-function Population A (Section 3) because they use different relation scopes and share retrieval lineage. Factorial192 contains 192 cases, 116 pairs, 74 queries and 51 source families. The

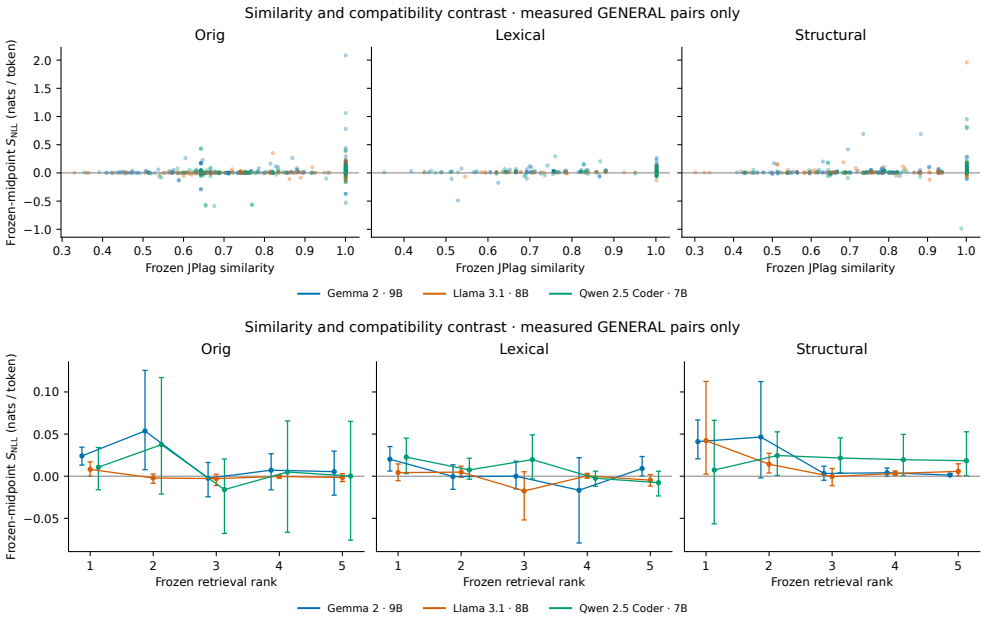


Figure 11: External similarity versus measured CFG compatibility contrast. Top: all measured GENERAL midpoint pairs, including outliers. Bottom: means of available pairs by rank with pointwise uncertainty. Each rank uses its eligible measured pairs. Complete comparisons of five source programs use 119 queries; 819 queries contribute at least one measured pair.

Table 21: DFG compatibility contrast AUC over normalized depth with aggregation within pairs first (192 cases, 116 pairs). Intervals resample pairs from the historical component population.

Regime	Mean	Median	95% CI
FUNCTION_ONLY	-0.01184	0.00012	$[-0.03764, 0.01044]$
NATIVE_PROMPT	0.00028	0.00007	$[-0.01685, 0.01780]$
SAME_PROMPT	-0.05204	-0.00815	$[-0.07791, -0.03102]$

three regimes yield 576 case–regime trajectories and 19,890 completed case–regime–layer bundles. Case counts vary by environment; Llama lexical has only one case/pair, so its uncertainty band is omitted. Full curves are in Figure 12. For AUC, integrate each case over normalized depth, average cases within pairs, then summarize pairs. The original $\ell/(B-1)$ integration coordinate is retained; plots display $(\ell+1)/B$. Normalizing by the integration interval preserves AUC under this affine coordinate change.

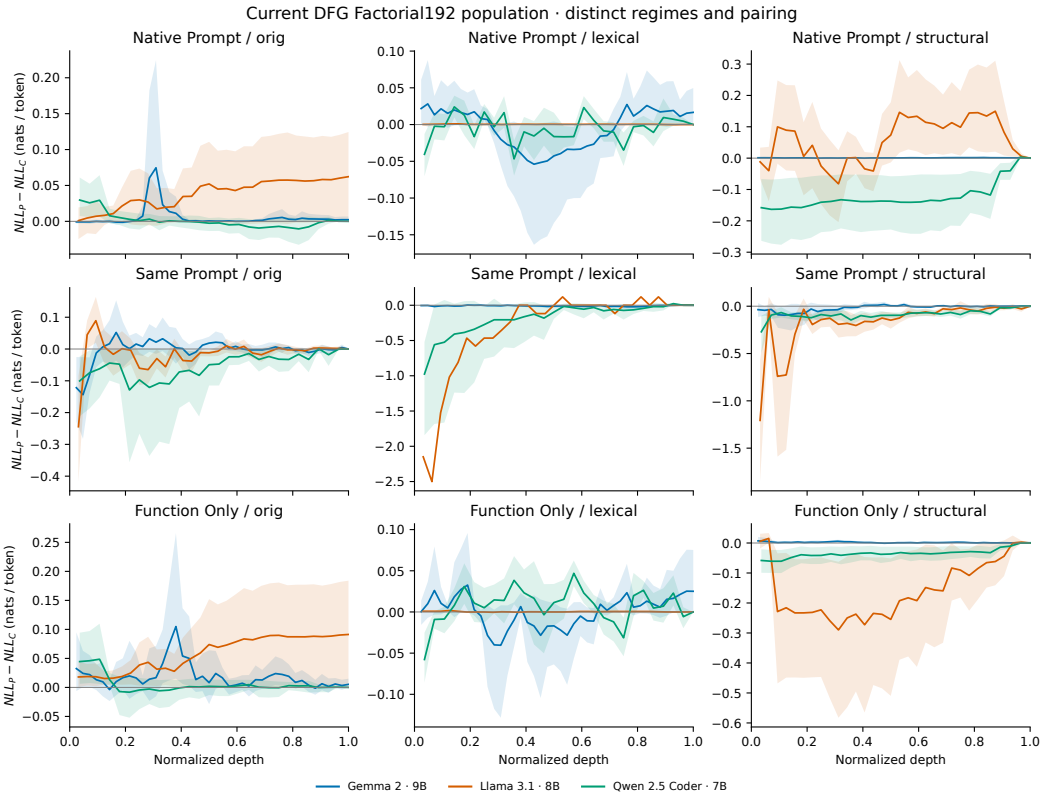


Figure 12: DFG compatibility contrast, $L_M - L_C$, by architecture, setting and serialization regime. Cases are averaged within pairs before equal pair weighting; bands are pointwise intervals over pair trajectories. Panel ranges retain all values. Each contrast compares aligned and structural mismatch transfers applied to separate clean branches at the same layer.

F INTERVENTIONAL CORRESPONDENCE: PROOFS AND MEASUREMENT

We formalize the two intervention criteria, construct an example with positive compatibility and zero recovery, and state an assumption for isolating program role information. We then derive the NLL and KL identities and estimators used to measure these contrasts.

F.1 GENERAL RISK FORMULATION OF THE CAUSAL CRITERIA

We define compatibility and recovery as predictive advantages over specified controls in intact and disrupted computations. For each eligible unit $u = (s, g, v, b)$, PSCA constructs an aligned assignment $\mathcal{A}_u = (S^a_u, T_u)$, where $(S^a_u, T_u) = \Gamma_v(\mathcal{C}_{v,b})$, independently of activations and outcomes. A control policy $\mathcal{M}$, specified before scoring, produces an assignment $\mathcal{M}_u = (S^m_u, T_u)$ that breaks correspondence under the same prescribed target positions and intervention constraints. This follows the broader principle that proposed mechanistic correspondences should predict behavioral consequences that can be assessed under intervention (Geiger et al., 2025; Palumbo et al., 2025).

For each evaluation, consider a given model, intervention schedule, population measure μ, and behavioral reference Q. Let $Y_e(\mathcal{A}_u)$ and $Y_e(\mathcal{M}_u)$ denote the downstream predictions produced by the aligned and control assignments in environment e, where e_{int} denotes an intact target trajectory and e_{dis} denotes a trajectory reached after an earlier disruption has propagated to the write layer. For a behavioral risk $\mathcal{R}_Q$, with smaller values indicating better prediction, define the correspondence advantage relative to the control

$$\Delta_Q(\mathcal{A}, \mathcal{M}; e) = \mathbb{E}_{u\sim\mu}[\mathcal{R}_Q(Y_e(\mathcal{M}_u)) - \mathcal{R}_Q(Y_e(\mathcal{A}_u))]\,. \tag{11}$$

Positive Δ_Q means the proposed assignment lowers risk relative to $\mathcal{M}$ for the specified population, environment, and reference. The contrast measures the effect of the complete transferred vectors. Appendix F.3 states the additional assumption needed to attribute this effect to program role information.

Definition 1 (Intact causal compatibility). A proposed correspondence exhibits relative causal compatibility when

$$\Delta_Q(\mathcal{A}, \mathcal{M}; e_{\text{int}}) > 0. \tag{12}$$

Compatibility therefore requires aligned transfer to outperform the specified control that breaks correspondence in an intact target computation. Comparison with CLEAN quantifies the effect relative to the unmodified computation.

Definition 2 (Causal recovery). We assess causal recovery by measuring the predictive deficit from disruption, the improvement after aligned restoration, and its advantage over mismatch. Let $\rho^Q_X = \mathbb{E}_\mu[\mathcal{R}_Q(Y_X)]$ denote population risk under condition X. After an earlier intervention produces a damaged trajectory, aligned states exhibit causal recovery with an advantage over $\mathcal{M}$ when

$$\rho^Q_D > \rho^Q_0, \qquad \rho^Q_{R,a} < \rho^Q_D, \qquad \rho^Q_{R,a} < \rho^Q_{R,m}. \tag{13}$$

The three inequalities require, respectively, a predictive deficit, recovery from that deficit, and an advantage for aligned over mismatch restoration. Restoration is partial when $\rho^Q_0 < \rho^Q_{R,a} < \rho^Q_D$. The final inequality is equivalently $\Delta_Q(\mathcal{A}, \mathcal{M}; e_{\text{dis}}) > 0$; the first two additionally establish damage and recovery.

Proposition 1 (Positive compatibility can coexist with zero recovery) *There exist a computation and admissible interventions such that*

$$\big(L_a = L_0 < L_m\big) \;\wedge\; \big(L_0 < L_D = L_{R,a} = L_{R,m}\big). \tag{14}$$

The same construction satisfies these relations with K in place of L.

Appendix F.2 constructs this computation by allowing earlier disruption to remove information outside the later restoration support.

Compatibility and recovery are evaluated separately in Section 3. NLL and KL from the clean distribution instantiate $\mathcal{R}_Q$; their measurement identities and estimators with equal pair weights are given in Appendices F.4–F.5.

F.2 Separation of Intact Compatibility and Recovery

We construct a computation in which earlier disruption removes information outside the later restoration support. This yields positive intact compatibility and zero recovery. Consider a given model, abstraction, layer schedule, eligible population μ, and two policies independent of outcomes. The aligned policy $\mathcal{A}$ supplies $\mathcal{A}_u = (S_u^a, T_u)$ for $u = (s, g, v, b)$, where $(S_u^a, T_u) = \Gamma_v(\mathcal{C}_{v,b})$. Together with the clean source states at the write layer, this specifies the values and locations of the aligned write. The control policy $\mathcal{M}$ supplies a frozen assignment $\mathcal{M}_u$ that breaks correspondence, with the same target positions and write geometry. As in Section 2.3, CFG uses a reassignment within pairs; DFG swaps two complete source relations while preserving DEF/USE roles. Each control policy is specified alongside $\mathcal{A}$ and forms part of the comparison being evaluated. Any random choice in constructing them is frozen before scoring; the estimand conditions on that realized choice.

Each evaluated environment is determined by its unit, model, and intervention schedule. In e_{int}, the target trajectory is intact at the write layer. In e_{dis}, damage at δ propagates to the write layer $\gamma > \delta$, retaining the propagated deviation from the clean trajectory. Aligned and mismatch branches share the same trajectory before the write within each environment and use clean source states captured at their write layer. Target positions outside the write support retain their current values; subsequent computation proceeds normally with later states unclamped. Under teacher forcing, all conditions use the same preceding token context at each scored position within a comparison. Each criterion is evaluated at its reported schedule; intact and restoration write layers follow their respective schedules and may differ.

The predictions $Y_e(\mathcal{A}_u)$ and $Y_e(\mathcal{M}_u)$ are collections of predictive distributions at those histories. The condition notation is $Y_a = Y_{e_{\text{int}}}(\mathcal{A}_u)$, $Y_m = Y_{e_{\text{int}}}(\mathcal{M}_u)$, $Y_{R,a} = Y_{e_{\text{dis}}}(\mathcal{A}_u)$, and $Y_{R,m} = Y_{e_{\text{dis}}}(\mathcal{M}_u)$. Y_D denotes the damage-only run, and Y_0 denotes CLEAN. For finite risks, $\rho_X^Q = \mathbb{E}_{u\sim\mu}[\mathcal{R}_Q(Y_X)]$. The behavioral reference Q may vary by unit and scored history but is the same across conditions. The measure μ weights pairs equally and eligible units within each pair equally. Population contrasts summarize these weighted averages and may vary across units.

Proof of Proposition 1. For any threshold $\tau > 0$, choose a finite scaling parameter $\lambda > \max\{2\tau, \log 3\}$. Consider a deterministic feedforward causal computation with early values $(t_1, t_2) = (1, -1)$. A later layer copies $h_1 = t_1$, $h_2 = t_2$, and $w = t_1$, and predicts the scored token 1 with probability

$$p(1) = \sigma(\lambda w(h_1 - h_2)/2)\,, \qquad \sigma(z) = (1 + e^{-z})^{-1}. \tag{15}$$

The write support is (h_1, h_2); w is outside it. A clean source supplies $(1, -1)$. The aligned policy writes this order; the control exchanges the values. Both policies use the same source multiset, target positions, and write budget. Take μ concentrated on this unit and set $q = \sigma(\lambda) > 3/4$.

With intact $w = 1$, aligned and clean predictions both equal q, whereas the control predicts $1-q$. For the point-mass reference Q^z on the scored token and the clean reference distribution $Q^0 = \text{Ber}(q)$, respectively,

$$\Delta_{Q^z}(\mathcal{A}, \mathcal{M}; e_{\text{int}}) = \log \frac{q}{1-q} = \lambda > \tau, \tag{16}$$

$$\begin{aligned}\Delta_{Q^0}(\mathcal{A}, \mathcal{M}; e_{\text{int}}) &= \text{KL}(\text{Ber}(q) \| \text{Ber}(1-q)) \\ &= (2q-1)\lambda > \lambda/2 > \tau.\end{aligned} \tag{17}$$

Now zero (t_1, t_2) at the earlier layer and propagate. This makes $h_1 = h_2 = w = 0$. Restoring (h_1, h_2) under either policy leaves $w = 0$, so damaged and both restored branches predict $1/2$. Damage is strictly positive: its excess log loss is $\log(2q) > 0$, and its clean distribution KL is $\text{KL}(\text{Ber}(q) \| \text{Ber}(1/2)) > 0$. For both metrics, the recovery contrasts are

$$\rho_D^Q - \rho_{R,a}^Q = 0, \qquad \Delta_Q(\mathcal{A}, \mathcal{M}; e_{\text{dis}}) = 0. \tag{18}$$

This deterministic construction proves that an arbitrarily large intact advantage under log loss or KL can coexist with zero recovery. Earlier damage sets $w = 0$, and restoring the selected support leaves the predictive gate closed.

Experimental implication. We evaluate intact compatibility and causal recovery separately because they measure transfers in different computational states. The recovery analysis in Section 3.4 measures damage, aligned recovery, and its advantage over mismatch restoration.

F.3 Role isolation assumption and counterexample

We derive a sufficient condition for attributing an assignment effect to program role information: all other influences are the same in the compared transfers. A counterexample then shows how lexical information can produce an advantage under matched write geometry. For a given unit u, the unrolled decoder with a prescribed write policy is a well defined causal system. Running both policies computes the paired potential responses $\ell_m(u)$ and $\ell_a(u)$, where ℓ denotes the logarithmic score. Their difference is a causal contrast between the implemented assignments. The following decomposition specifies which part of that contrast can be attributed to role information.

Lemma 2 (Role isolation under an exclusion condition) *Assume a role/nuisance decomposition exists such that, for a given environment and common target write geometry, the response factors as $\ell_X(u) = f_u(Z_X(u), N_X(u))$. Here Z_X is the content hypothesized to carry role information, N_X contains all other transferred influences relevant to the response, and f_u is the same downstream response function in both branches. If the interventions satisfy the exclusion assumption $N_a(u) = N_m(u)$ for μ-almost every u, then*

$$\mathbb{E}_\mu[\ell_m - \ell_a] = \mathbb{E}_\mu\left[f_u(Z_m, N_a) - f_u(Z_a, N_a)\right]. \tag{19}$$

Under this condition, the contrast measures a change in role content with all other influences unchanged. The result applies to intact and restoration comparisons with their respective common starting trajectories.

Proof and scope. Substitute $\ell_X = f_u(Z_X, N_X)$ into the left hand side, replace N_m by N_a using the assumption, and take expectations. For general N_a and N_m, the contrast decomposes as

$$\begin{aligned}\mathbb{E}_\mu[\ell_m - \ell_a] = \mathbb{E}_\mu[f_u(Z_m, N_a) - f_u(Z_a, N_a)] \\ + \mathbb{E}_\mu[f_u(Z_m, N_m) - f_u(Z_m, N_a)].\end{aligned} \tag{20}$$

The second term captures changes in other transferred influences. For example, if $f_u(z, \cdot)$ were uniformly L-Lipschitz, its absolute value would be at most $L\mathbb{E}_\mu\|N_m - N_a\|$. Our experiments estimate the combined response of complete vectors, with Z and N unobserved. Applying the role-specific interpretation requires evidence for the stated decomposition and exclusion condition.

Counterexample: lexical information produces a transfer advantage. Consider two written vectors $(0, 1)$ and $(0, -1)$, whose first coordinate is a putative role variable with identical value in both. Their second coordinates carry only a lexical feature n. Let the model's probability of the scored token be $\sigma(n_1 - n_2)$, with clean target values $n_1 = n_2 = 0$. The aligned assignment writes $(n_1, n_2) = (1, -1)$; mismatch exchanges the same vectors. Both policies use the same target positions, vector multiset, write budget, and squared displacement from the clean state. The resulting contrast is

$$L_m - L_a = -\log\sigma(-2) + \log\sigma(2) = 2 > 0. \tag{21}$$

The prediction depends on the lexical coordinate n, so the positive contrast is entirely lexical despite matched displacement, write budget, and source vectors. This example shows why role attribution requires the exclusion condition above. Related subspace-patching work examines how predictive effects can arise through dormant pathways (Makelov et al., 2024).

F.4 Measurement connection: logarithmic risk

We apply the logarithmic scoring identity with two reference distributions to express the population risk contrasts as NLL and KL differences. Consider a given model, abstraction, layer schedule, and eligible population. A unit u comprises a program pair and a legal intervention unit. Let η_t be its preceding token context at scored position t, shared across conditions under teacher forcing, and m_u its number of scored tokens. Write $P_X^u(\cdot \mid \eta_t)$ for the predictive distribution in Y_X at that history. Given reference distributions Q_t^u shared across conditions, excess logarithmic risk is

$$\mathcal{E}_Q(X) = \mathbb{E}_{u\sim\mu}\frac{1}{m_u}\sum_{t=1}^{m_u}\mathbb{E}_{Y\sim Q_t^u}\left[\log\frac{P_0^u(Y \mid \eta_t)}{P_X^u(Y \mid \eta_t)}\right]. \tag{22}$$

The expectation over Y averages token scores under the reference distribution without changing the preceding token context. Q_t^z is a point mass on the observed target token, while $Q_t^0 = P_0^u(\cdot \mid \eta_t)$. Assume finite vocabulary, strictly positive predictive probabilities, and finite expected losses.

For a given unit and preceding token context, let Q be a reference distribution and P_X a strictly positive predictive distribution on the finite vocabulary. The logarithmic scoring identity is

$$\mathbb{E}_{Y\sim Q}[-\log P_X(Y)] = H(Q) + \mathrm{KL}(Q\|P_X). \tag{23}$$

Subtracting the score of P_0 cancels $H(Q)$, yielding

$$\mathbb{E}_{Y\sim Q} \log \frac{P_0(Y)}{P_X(Y)} = \mathrm{KL}(Q\|P_X) - \mathrm{KL}(Q\|P_0). \tag{24}$$

For Q concentrated on the observed token z_t, averaging over scored tokens and units gives $\mathbb{E}_\mu[L_X - L_0]$ exactly. For $Q = P_0$, the second KL vanishes and the same averages give $\mathbb{E}_\mu[K_X]$. KL is nonnegative and equals zero exactly for equal distributions. An expectation of a nonnegative quantity is zero only if it is zero almost surely. Since each unit has finitely many scored positions with positive weight, $\mathcal{E}_{Q^0}(X) = 0$ holds exactly when patched and clean predictive distributions coincide at all scored positions, μ-almost surely. In terms of the general risk in Appendix F.1, choose $\mathcal{R}_Q(P_X) = m_u^{-1} \sum_t \mathbb{E}_{Y\sim Q_t^u}[-\log P_X(Y \mid \eta_t)]$. Then $\mathcal{E}_Q(X) = \rho_X^Q - \rho_0^Q$, and

$$\mathcal{E}_{Q^z}(X) = \mathbb{E}_\mu[L_X - L_0], \qquad \mathcal{E}_{Q^0}(X) = \mathbb{E}_\mu[K_X]. \tag{25}$$

For Q^0, subtracting the reference entropy, which is the same across conditions, is equivalent to using KL itself as $\mathcal{R}_{Q^0}$: every risk difference and both definitions are unchanged. Zero KL from the clean distribution means equality at the scored histories; distributional equality therefore requires zero KL across those histories, whereas matching clean NLL supplies a weaker condition. These identities follow from proper logarithmic scoring (Gneiting & Raftery, 2007).

Applying the same subtraction to aligned and mismatch transfers gives

$$\Delta_Q(\mathcal{A}, \mathcal{M}; e_{\mathrm{int}}) = \mathbb{E}_{u\sim\mu} \frac{1}{m_u} \sum_t \left[\mathrm{KL}(Q_t^u\|P_m^u) - \mathrm{KL}(Q_t^u\|P_a^u)\right], \tag{26}$$

with conditioning on the shared token contexts omitted from the notation. Thus a positive contrast is a reduction in logarithmic risk. The NLL contrast uses Q^z, concentrated on the observed target tokens; the KL contrast uses the clean distribution Q^0. Teacher forcing supplies the same preceding token contexts across intervention conditions, making the predictive contrasts directly comparable.

F.5 Estimators with equal pair weights and generic comparisons

To estimate the population contrasts while giving each pair equal weight, we compute differences within intervention units, average within each pair, and then average across pairs. Generic comparisons apply the same procedure to their own eligible subset. The NLL observations for each unit are

$$C(u) = L_m(u) - L_a(u), \tag{27}$$

$$\mathrm{DamageEffect}(u) = L_D(u) - L_0(u), \tag{28}$$

$$\mathrm{Recovery}(u) = L_D(u) - L_{R,a}(u), \tag{29}$$

$$\mathrm{RepairSpecificity}(u) = L_{R,m}(u) - L_{R,a}(u). \tag{30}$$

These observations supply the estimates used to evaluate the population criteria in Appendix F.1. With N pairs and B_j eligible units in pair j, the compatibility estimator is

$$\widehat{\Delta}_{Q^z}(\mathcal{A}, \mathcal{M}; e_{\mathrm{int}}) = \frac{1}{N} \sum_{j=1}^{N} \frac{1}{B_j} \sum_{b=1}^{B_j} C(j, b). \tag{31}$$

Replacing L with K yields the estimator for $\Delta_{Q^0}(\mathcal{A}, \mathcal{M}; e_{\mathrm{int}})$. Let $\widehat{\mu}$ place mass $1/N$ on each observed pair and $1/B_j$ on each eligible unit conditional on pair j. Equation 31 is exactly $\Delta_{Q^z}(\mathcal{A}, \mathcal{M}; e_{\mathrm{int}})$ evaluated under $\widehat{\mu}$. For sequential conditions the population contrasts are

$$d_Q = \mathcal{E}_Q(D), \qquad r_Q = \mathcal{E}_Q(D) - \mathcal{E}_Q(R, a), \qquad s_Q = \mathcal{E}_Q(R, m) - \mathcal{E}_Q(R, a). \tag{32}$$

Here $s_Q = \Delta_Q(\mathcal{A}, \mathcal{M}; e_{\text{dis}})$, whereas intact compatibility uses environment e_{int}. Substituting $\widehat{\mu}$ gives the existing DamageEffect, Recovery, and RepairSpecificity estimates obtained by averaging within pairs first. Definition 2 requires $d_Q > 0, r_Q > 0, s_Q > 0$. Partial restoration additionally requires $d_Q - r_Q > 0$. These identities estimate the specified empirical measure; uncertainty and any broader population interpretation depend on the sampling and clustering assumptions specified in Appendix C. Matched generic transfer uses a separate control policy and empirical subset, with its own reported risk differences. For eligible units, the additional generic contrasts are $A(u) = L_n(u) - L_a(u)$ in intact computation and

$$A_R(u) = L_{R,n}(u) - L_{R,a}(u) \tag{33}$$

after disruption. Averaging these contrasts within pairs and then equally across pairs estimates the corresponding risk differences using a separate empirical measure on the exact eligible subset. Each comparison uses the aligned mean from that same subset. Each reported 95% confidence interval quantifies uncertainty in its individual contrast.

F.6 Relation to causal abstraction

PSCA uses the principle from causal abstraction that a proposed correspondence should predict behavior under corresponding interventions (Geiger et al., 2025). The present criteria evaluate predictive correspondence within a specified population and intervention protocol.