Finding Interpretable Prompt-Specific Circuits in Language Models

Gabriel Franco [1] **Lucas M. Tassis** [1] **Azalea Rohr** [2] **Mark Crovella** [1 2]

Abstract

Understanding the internal circuits that language models use to solve tasks remains a central challenge in mechanistic interpretability. A crucial part of finding circuits is understanding why each attention head attends where it does. To this end, we introduce **ACC++**, an improved circuit-tracing method based on the principle of *attention-causal communication* (ACC) (Franco & Crovella, 2025), which identifies *signals*, i.e., contents of low dimensional subspaces that cause attention on a token pair. ACC++ extracts circuits from a *single forward pass*, without replacement models or patching. Circuits identified by ACC++ consist of components that are causal for the model's attention decisions, together with the low-dimensional signals used to communicate between them. Here, we first detail the conceptual advances that ACC++ makes over previous work. We then show that across multiple models, a substantial portion of ACC++ signals are *interpretable*: many signals admit a short natural-language description. We next present a number of new insights into model behavior obtained via ACC++. First, we use ACC++'s interpretable circuits to characterize the sensitivity of indirect object identification (IOI) circuits to prompt structure. We find that prompt-specific circuits form well-defined clusters, and across clusters, heads receive systematically different signals corresponding to distinct mechanisms for identifying the IO name. Next, in multilingual IOI, ACC++ circuits show that while model *components* are reused across languages, *signals* are often language-specific. In a four-language IOI case study, cross-language circuit distances are consistent with linguistic relatedness. Together, these results show that ACC++ can shed light on a broad spectrum of model behaviors.[1]

[1]Department of Computer Science, Boston University, Boston, USA [2]Faculty of Computing & Data Sciences, Boston University, Boston, USA. Correspondence to: Gabriel Franco <gvfranco@bu.edu>.

Mechanistic Interpretability Workshop at the 43rd International Conference on Machine Learning, Seoul, South Korea, 2026. *Copyright 2026 by the author(s).*

1. Introduction

Large language models (LLMs) have become widely deployed, yet their internal computations remain difficult to understand. This opacity is increasingly a practical problem: when models fail, behave inconsistently across prompt wordings, or exhibit unexpected generalization, we often lack a mechanistic account of *why*. Mechanistic interpretability aims to fill this gap by explaining model behavior in terms of human-understandable computations implemented by collections of model components—often called *circuits* (Elhage et al., 2021). In this view, a circuit is not merely a correlated set of neurons or features, but a set of causal pathways: specific components write information into the residual stream, and downstream components selectively read and transform it.

To trace such circuits, we introduce **ACC++**, a circuit-tracing method that builds on *attention-causal communication* (ACC) (Franco & Crovella, 2025). ACC leverages the observation that attention heads communicate through low-dimensional subspaces that can be identified via analysis of the head's query–key (QK) matrix (Franco et al., 2026; Pan et al., 2024; Merullo et al., 2024b; Franco & Crovella, 2025; Ahmad et al., 2025). For a given head, ACC uses this principle to trace which upstream components write into those low-dimensional subspaces – termed *signals* – that are causal for the head's attention decision.

Here, we make two sets of contributions. Our **first set of contributions** consists of a number of conceptual advances over prior work (Franco & Crovella, 2025). These have the effect of *dramatically reducing the noise* present in traces. ACC++ traces contain *much lower-dimensional signals*, which supports natural language interpretability, as we show below. Further, they contain *50% to 90% fewer components*. We show that many components included by ACC can be omitted without violating the ACC++ attention-level counterfactual criterion, suggesting that they are spurious for

[1]Code available at www.github.com/gaabrielfranco/finding-highly-interpretable-circuits

the traced attention decisions. We describe these advances and the benefits they provide, including natural-language interpretability, in §2.

A key benefit of the attention-causal communication approach is that it can be used to trace circuits on a *per-prompt* basis. The availability of interpretable prompt-specific circuits opens additional avenues for understanding model behavior.

Accordingly, our **second set of contributions** is to surface previously-unobserved model behaviors that are visible only when using per-prompt tracing. We use the Indirect Object Identification (IOI) task as our setting, studying its ACC++ circuits across GPT-2, Pythia 160M and Gemma-2 2B. We first show that *distinctly different circuits are associated with different IOI prompt formats*. For example, GPT-2 circuits are highly dependent on the order of the proper nouns; in contrast, Pythia circuits are instead sensitive to lower-level wording changes in prompts. We explain this result by examining and interpreting the signals involved in each trace type, showing that qualitatively different signals are used in each case. We next study the IOI task in a multilingual setting, and show that while common components are used in each circuit, that *signals used are language-specific*, resulting in consistent differences between per-language traces that correlate with linguistic similarity of those languages. We describe these contributions in §3.

Finally, we note that ACC++ uses only a single forward model pass and does not require replacement models, patching, or counterfactual inputs. As a result, ACC++ makes per-prompt circuits viable units of analysis at scale, and supports the study of how model circuits vary across prompts, templates, and languages.

2. Refining Attention Causal Communication

As mentioned above, ACC++ is an improved method for circuit tracing based on the principle of *attention-causal communication* (ACC) as introduced in (Franco & Crovella, 2025). ACC++ makes a number of conceptual advances over previous methods that together enable it to construct accurate and interpretable per-prompt circuits with no tuning involved beyond a single attention threshold.

Background. For exposition, we fix an attention head (ℓ, a) attending to a destination-source[2] token pair (d, s). Let $\mathbf{x}^j$ denote the residual stream at token position j on input to head (ℓ, a). The pre-softmax attention score on the pair (d, s) can then be written as $A'_{ds} = \mathbf{x}^{d\top}\Omega\mathbf{x}^s$, where $\Omega = W_Q W_K^\top$ is the head's *QK matrix*. We write A'_d for the row of pre-softmax scores at destination d and A_d for the corresponding post-softmax attention weights, so that A_{ds} is the attention placed on source token s. By *upstream components* we mean token embeddings, attention heads and feedforward (FFN) modules in layers before ℓ. Moreover, the residual stream at any layer and token position decomposes additively into individual upstream component outputs, $\mathbf{x}^d = \sum_i \mathbf{o}_i^d$ (Figure 1b).

Prior work has shown that the singular vectors of QK matrices often align with meaningful features in residual representations (Franco et al., 2026), exposing the communication channels used by attention heads (Pan et al., 2024; Merullo et al., 2024b; Franco & Crovella, 2025; Ahmad et al., 2025). If $\Omega = \sum_{k=1}^{R} \sigma_k \mathbf{u}_k \mathbf{v}_k^\top$ is its singular value decomposition (SVD), then $A'_{ds} = \sum_{k=1}^{R} (\mathbf{x}^{d\top}\mathbf{u}_k)\sigma_k(\mathbf{v}_k^\top \mathbf{x}^s)$. Intuitively, the SVD yields paired query-side and key-side directions, and each direction defines a *candidate signal* through which upstream components can influence attention on a token pair.

When an attention head attends to a particular token pair, that choice is determined by features written into the token representations by upstream components (Elhage et al., 2021). To trace a circuit, one can seek to isolate the small subset of upstream contributions that are in fact causal for that attention decision. The attention-causal communication (ACC) problem of (Franco & Crovella, 2025) formalized this goal for a fixed destination-source token pair. To pinpoint causality, methods in (Franco & Crovella, 2025) seek a small subspace and a small set of *signals*, i.e., upstream components writing into that subspace, such that removing those contributions from the residual stream would cause the head to not attend to the token pair. ACC builds this subspace from the singular vectors of Ω, leveraging their alignment with meaningful features in the residual stream.

Tracing Circuits with ACC++. ACC++ identifies the signals causal for a single attention firing: a head (ℓ, a) attending to a token pair (d, s). It proceeds in four steps (Figure 1): (a) fix the target head (ℓ, a) and token pair (d, s); (b) decompose the destination residual $\mathbf{x}^d$ as a sum of upstream component outputs; (c) form candidate signals by projecting each component output $\mathbf{o}_i^d$ onto the left singular vectors $\{\mathbf{u}_k\}$ of Ω, so that every (component, direction) pair yields one rank-1 candidate; and (d) score candidates by their contribution to A_{ds} using Integrated Gradients (IG) (Sundararajan et al., 2017) and greedily remove top-ranked candidates until $A_{ds} < \tau$. The source side is analogous, using right singular vectors. A complete and detailed description of ACC++ is provided in Appendices A through D; see Appendix C.6, Figure 11, for a more detailed summary of the method.

We define a *circuit* for a prompt as the directed graph traced by applying ACC++ recursively across the model. Its nodes are model components, and each edge is a signal returned by

[2]We call the query token the *destination* token and the key token the *source* token.

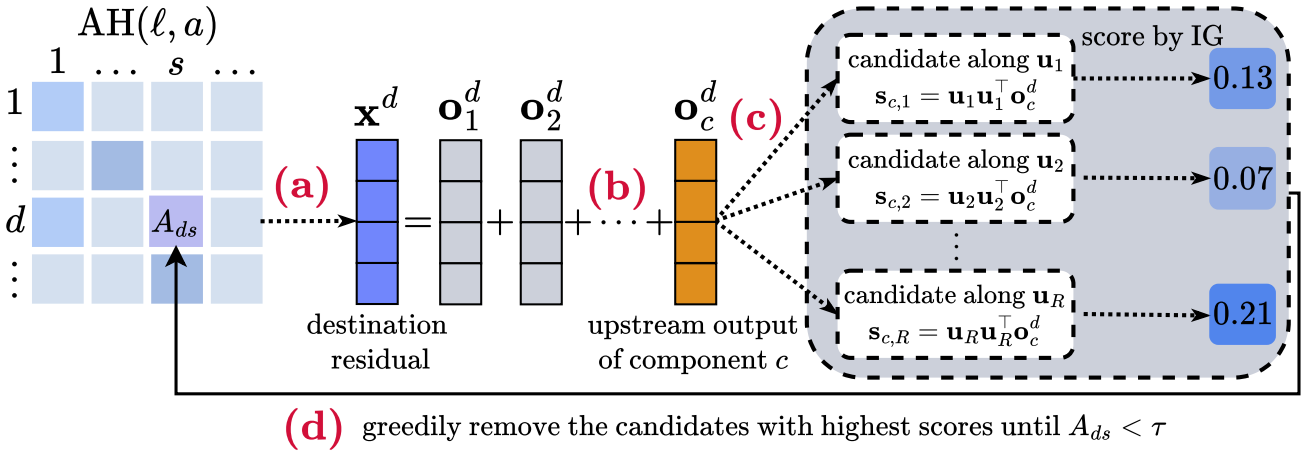


Figure 1. **ACC++ method (destination side; source side is analogous).** Given an attention head (ℓ, a) attending to a token pair (d, s), ACC++ identifies which upstream contributions drive the attention in four steps. **(a) Select head and token pair**, **(b) Decompose the destination residual**, **(c) Construct candidate signals from singular directions**, and **(d) Compute attribution by Integrated Gradients and greedily remove.**

ACC++, linking an upstream component to a downstream attention firing along a specific subspace. We build the circuit by seeding at the components with a direct effect on the logit direction of interest (e.g., the correct-answer direction), running ACC++ on each attention firing in the current trace, and recursing upstream until none remain.

ACC++ Advances. ACC++ makes two conceptual advances and one technical advance over previous work (Franco & Crovella, 2025). On the conceptual side, ACC++ (1) dramatically sharpens the identification of candidate signals by identifying them more precisely with subspace-component pairs; and (2) more accurately measures attention causality by defining its objective directly on the head's nonlinear Softmax-derived attention. In comparison, prior work (1) identified subspaces and upstream components independently, leading to higher-dimensional and potentially less-interpretable signals; and (2) used a linear surrogate for attention, which introduced spurious components in the trace. We find that these conceptual advances yield signals which frequently admit a short natural-language description (§ 2), and circuits that are much less noisy without sacrificing causality of attention (Table 1).

The technical advance contributed by ACC++ addresses heterogeneity in model architecture: we show that, under mild full-rank assumptions, it is always possible to construct a single QK matrix for each attention head, such that the matrix correctly computes attention regardless of whether bias or Rotary Positional Embeddings (RoPE) (Su et al., 2024) is used in the particular model. Among other benefits, this vastly simplifies the use of SVD-based analysis of QK matrices in real models.

Precise Signal Candidates. The first conceptual advance in ACC++ concerns how it turns the empirical observation

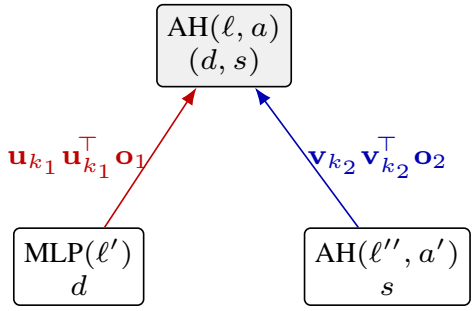


Figure 2. ACC++ edges feeding the head (ℓ, a) attending (d, s). A destination edge (red) comes from an upstream component writing at the destination token d and carries the signal $\mathbf{u}_k\mathbf{u}_k^\top\mathbf{o}_i$, using a left singular vector $\mathbf{u}_k$. A source edge (blue) comes from an upstream component writing at the source token s and carries $\mathbf{v}_k\mathbf{v}_k^\top\mathbf{o}_i$, using a right singular vector $\mathbf{v}_k$.

of SVD-derived communication channels (Pan et al., 2024; Merullo et al., 2024b; Franco & Crovella, 2025; Ahmad et al., 2025) into specific candidate signals. A candidate signal corresponds to one upstream component writing in one singular vector direction. Figure 2 shows the relationship between upstream components, signals, and the downstream head.

Each candidate signal $\mathbf{s}_i$ induces a vector $\mathbf{c}_i$ of pre-Softmax score contributions across source token positions. For example, in the destination view, consider a candidate signal $\mathbf{s}_i$ corresponding to upstream component m projected onto singular direction k, i.e., $\mathbf{s}_i = \mathbf{u}_k\mathbf{u}_k^\top\mathbf{o}_m$ (Figure 1c). The contribution of signal $\mathbf{s}_i$ to the pre-Softmax attention score of source position j is $[\mathbf{c}_i]_j = \mathbf{s}_i^\top\Omega\mathbf{x}^j$. Figure 2 illustrates both edge types: a destination edge from an upstream component writing at d carries the signal $\mathbf{u}_k\mathbf{u}_k^\top\mathbf{o}_1$, while a source edge from an upstream component writing at s

carries $\mathbf{v}_k \mathbf{v}_k^\top \mathbf{o}_2$. Thus $\mathbf{c}_i$ collects the contribution of that candidate signal across all source positions. The source-side view is analogous, with the candidate contribution entering on the key side. Note that this is a complete accounting for pre-Softmax attention. Namely, summing across all candidate-induced vectors recovers the exact pre-Softmax score vector for the destination token: $A'_d = \sum_i \mathbf{c}_i$.

A key strategy taken in ACC++ is to identify the set of candidate signals as the cross-product of *all* outputs of upstream components with *all* singular vector-derived channels of the head. This is in comparison to previous work (Franco & Crovella, 2025) which separately identified components and channels. By exhaustively and separately considering each possible upstream-channel combination, ACC++ generates a smaller set of more precise signal candidates for subsequent evaluation.

Accurate Attention Accounting. Once candidate signals have been defined, ACC++ identifies a small set whose removal, considered as a counterfactual, would cause the chosen attention weight A_{ds} to fall below a threshold τ (Figure 1d). (We discuss the rationale for our choice of τ in Appendix D.)

There are two cases of this counterfactual: *destination* signals, removed from the destination residual $\mathbf{x}^d$ and affecting the query, and *source* signals, removed from the source residuals $\{\mathbf{x}^j\}_{j \leq d}$ and affecting the competing keys (Figure 2). Because the entries of A_d are coupled by Softmax, the nonlinear impact of signal removal must be evaluated using the full row A_d, not just using the individual token s. Under Softmax, a candidate is important not because it increases the score of A'_{ds} in isolation, but because it increases that score relative to the competing source tokens, thereby increasing A_{ds}.

Since Softmax is nonlinear, exact identification of the minimal set of candidate signals to remove is challenging. To address this efficiently, ACC++ uses a greedy strategy based on an estimate of the nonlinear impact of each signal removal. ACC++ first ranks candidate signals by how much each affects the true attention weight A_{ds}, using IG. Given these attribution scores, ACC++ solves the counterfactual by greedy removal. Starting from the full set of candidates, it repeatedly removes the highest-scoring candidate, recomputes the attention weight, and stops once the chosen weight falls below τ. The selected candidates are signals in the resulting trace.

This is a key point where ACC++ diverges from prior work (Franco & Crovella, 2025) which only operated on pre-Softmax scores A' via a linear surrogate. In contrast, ACC++ evaluates the nonlinear Softmax explicitly, allowing it to identify much smaller circuits that are still causal for attention, as we show below.

Table 1. Comparison of ACC and ACC++ graph sizes across tasks and models. Entries are mean ± std. error.

Task	Model	Nodes ACC	Nodes ACC++	Edges ACC	Edges ACC++
IOI	Gemma-2 2B	688 ± 12	87 ± 3	7086 ± 200	215 ± 11
	GPT-2 Small	205 ± 2	85 ± 1	639 ± 9	154 ± 3
	Pythia-160M	123 ± 2	76 ± 1	354 ± 8	147 ± 4
GP	Gemma-2 2B	121 ± 20	8 ± 0	806 ± 159	7 ± 1
	GPT-2 Small	114 ± 3	27 ± 1	263 ± 10	28 ± 1
	Pythia-160M	63 ± 3	33 ± 2	132 ± 8	52 ± 4
GT	GPT-2 Small	176 ± 3	45 ± 1	594 ± 12	81 ± 2
	Pythia-160M	78 ± 4	43 ± 5	177 ± 13	88 ± 12

Unified QK Analysis. ACC++ also makes technical contributions for analyzing QK matrices. Expressing an attention score A'_{ds} as a bilinear form $\mathbf{x}^{d\top} W_Q W_K^\top \mathbf{x}^s$ overlooks details that vary across model architecture. In particular, some models add a bias term to the attention calculation, and some models use RoPE positional encoding. Both of these prevent using a single $\Omega = W_Q W_K^\top$ matrix to describe the attention computation on a per-attention head basis. ACC++ overcomes these challenges by folding the bias and RoPE rotations into the tokens $\mathbf{x}^d$ and $\mathbf{x}^s$. In the case of bias, ACC++ adds a function of the bias to the tokens, and in the case of RoPE it performs a suitable rotation on the tokens, *before computing attention*. We describe the specifics in Appendix B, which presents proofs that after these transformations on the tokens, the correct attention score is computed as a simple bilinear form of Ω. This vastly simplifies the analysis of Ω using SVD since there is only one Ω per head, independent of bias or token positions.

Evaluation. We use ACC++ to trace circuits for Indirect Object Identification (IOI) (Wang et al., 2023), Greater-Than (GT) (Hanna et al., 2023b), and Gender Pronoun (GP) (Athwin et al.) in GPT-2 Small (Radford et al., 2019), Pythia-160M (Biderman et al., 2023), and Gemma-2 2B (Team et al., 2024). We show that ACC++ recovers circuits with high overlap (precision, recall, and F1 against reference circuits, following (Franco & Crovella, 2025)) with ACC circuits and with patching-based methods (Path Patching (Wang et al., 2023), EP (Bhaskar et al., 2025), ACDC (Conmy et al., 2023), EAP (Syed et al., 2024)). Full results are in Appendix E.

The ACC++ signals present in these circuits are lower-dimensional than ACC's: across IOI, GP, and GT, the majority use a *single* singular direction of Ω (Appendix E, Figures 16 and 17). Moreover, we ablate subsets of the signals selected by ACC++ on IOI and show that these signals also often have a measurable causal effect on the model's IOI performance while only minimally perturbing the residual stream, with post-ablation cosine similarity close to 0.99 and norm ratio close to 1; full results are reported in the Appendix E.

Table 2. Autointerpretation scores for ACC++ signals on IOI. We report medians with interquartile range (25%–75%); the SAE column is from (Paulo et al., 2025).

Metric	SAE	GPT-2	Pythia	Gemma
Accuracy	0.76 (0.67–0.86)	0.70 (0.60–0.80)	0.65 (0.53–0.76)	0.60 (0.53–0.70)
Precision	–	0.68 (0.57–0.79)	0.62 (0.52–0.75)	0.58 (0.51–0.68)
Recall	–	0.85 (0.70–0.95)	0.85 (0.65–0.95)	0.80 (0.65–0.90)

Table 1 shows that **ACC++ circuits are from 2× to 10× smaller** than ACC traces while preserving the traced attention decisions under our counterfactual attention-weight criterion. Finally, we emphasize that ACC++ tracing is automatic. Apart from specifying a value for τ (for which ACC++ incorporates reasonable defaults, see Appendix D) there is no tuning involved, no replacement models to train, and no counterfactual prompts to formulate.

Automatic Interpretation of ACC++ Signals. Next, we interpret signals by constructing a short natural language description of each signal (i.e., edge in the trace). We use the fully automated autointerpretation framework of (Paulo et al., 2025) for SAE features and adapt it to ACC++ signals.

Starting from a signal identified by ACC++, we (i) retrieve high-activation contexts by scoring token pairs in cached activations from The Pile (Gao et al., 2020) (via `pile-10k` (Nanda, 2024)), (ii) prompt an LLM to propose a short natural-language description of the signal from these contexts, and (iii) evaluate the description with the a LLM-as-a-judge protocol of (Paulo et al., 2025) by mixing top contexts with random contexts. Full details are presented in Appendix F. We evaluate this pipeline on a large set of signals (13,406 from GPT-2 small, 15,316 from Pythia-160M, and 22,223 from Gemma-2 2B) generated by a random sample of 100 prompts from the IOI task.

A substantial fraction of signals are interpretable. Table 2 reports quantitative interpretability for the signals across the three models. The Table shows that ACC++ signals achieve autointerpretation performance that approaches SAE-feature baselines. To put these results in context, we note that SAE features are explicitly trained to separate superposed features, while ACC++ signals arise directly from causal attention interactions.

We also use the same evaluation to define when a signal should count as *interpretable* under a hypothesis-testing criterion. Given a candidate interpretation and judge scores, we ask whether the interpretation reliably explains the signal on meaningful contexts, rather than succeeding equally on random contexts. Concretely, we compute a one-sided p-value using Fisher's exact test (Fisher, 1922) to test whether the judge accepts the interpretation more often on top contexts than on random contexts. We then control false discoveries across signals by reporting the fraction of signals with False Discovery Rate (FDR) $\leq 5\%$; signals passing this criterion are labeled *interpretable*. This measure of interpretability is meant to distinguish signal from noise, rather than to fully quantify the intrinsic quality of the interpretation.

We then estimate the fraction of interpretable signals under our criterion. We find that the fraction of interpretable signals varies across models: 63% (95% CI: 62–64%) in GPT-2 Small, 50% (49–51%) in Pythia-160M, and 31% (30–32%) in Gemma-2 2B. This result suggests that many ACC++ signals correspond to consistent attention interactions rather than noise.

ACC++ Outputs. The output of ACC++ is a graph in which nodes are model components and edges are the signals causal for attention, annotated with their natural language interpretation. To illustrate typical outputs of ACC++, we provide traces of two representative tasks for Gemma-2 2B. First, to illustrate factual recall, we trace the prompt "Fact: capital of the state containing Dallas is," for which the model predicts "Austin," in Appendix J, Figure 44. Second, we illustrate in-context translation by tracing the prompt "Mary gave me a book and a card. Marie m'a donné un livre et une", for which the model predicts "carte," in Appendix J.

We show examples of automatically generated signal interpretations in Figure 5, with additional traces in Appendix H (Figures 38 and 39) and Appendix J (Figures 44 and 45). To aid in circuit inspection, we provide a user dashboard in which the interpretation of the signals in each circuit can be browsed and the evidence can be examined, as illustrated in Appendix F.3, Figures 27–31.

3. ACC++ Yields Mechanistic Insights into Model Behavior

ACC++ produces a circuit per prompt whose nodes are model components and whose edges carry the causal signals driving attention (§ 2). These circuits contain rich information about how the model produced its prediction, and much of it can be put into human-readable form. We use these interpretable signals to derive mechanistic insights in two settings: (1) the sensitivity of IOI circuits to prompt structure, including a natural-language account of how the model retrieves the indirect object (§ 3.1); (2) how mGPT solves multilingual IOI through a mix of language-independent and language-specific signals (§ 3.2).

3.1. Understanding IOI Circuit Sensitivity to Prompt Structure

We first use ACC++ to revisit IOI, a canonical benchmark in mechanistic interpretability (Wang et al., 2023; Conmy

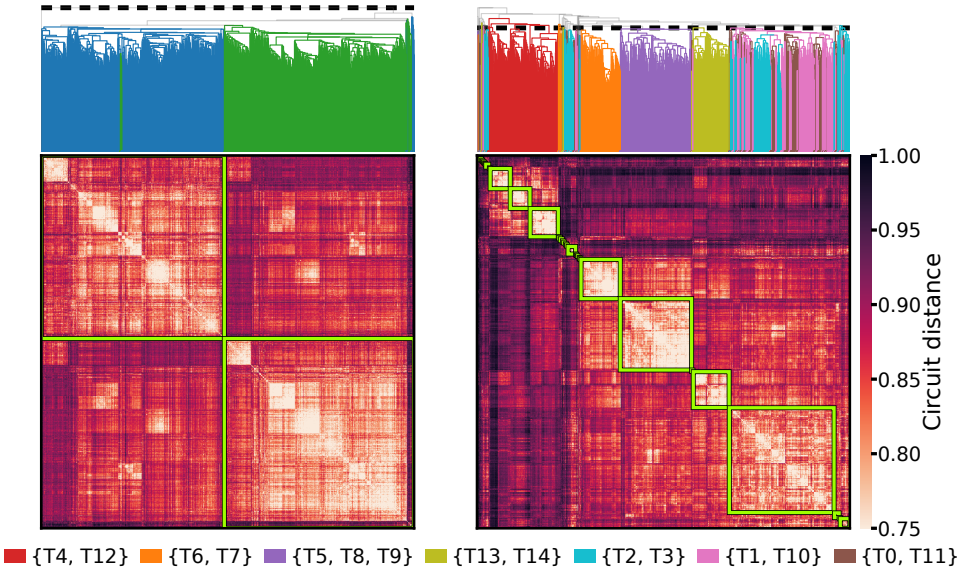


Figure 3. Average linkage clustering of prompt-level traces exposes the sensitivity of IOI circuits to prompt structure. Left: GPT-2 small. Right: Pythia 160M. Branches in the dendrograms are colored by high-level template (ABBA vs. BABA) or by low-level template group (see legend below).

et al., 2023; Makelov et al., 2023; Bhaskar et al., 2025; Syed et al., 2024; Merullo et al., 2024a; Tigges et al., 2024; Hanna et al., 2024; Haklay et al., 2025; Méloux et al., 2025; Ahmad et al., 2025; Franco & Crovella, 2025; Garriga-Alonso et al., 2024; Wu & Barez, 2025). The IOI task has sentences of the form "When Mary and John went to the store, John gave the bottle to", where the model should answer "Mary" instead of "John". Prior work has studied IOI using prompt templates and has highlighted the importance of positional structure in circuit analysis (Haklay et al., 2025). Here we ask a more specific question: *how much of the variation across prompts is driven by prompt structure, and how can signal-level traces explain the difference?*

We use a collection of IOI prompts balanced across two levels of templates: two high-level templates (reversing the role order of the two names, ie, ABBA vs BABA), and 15 low-level templates (changing surface wording, eg "While ..." vs "Then..."). Details are in Appendix H.1.

Circuits cluster by prompt format. For each model, we trace a circuit with ACC++ on every prompt where the model predicts the correct indirect object. To compare traces, we represent each circuit as a set of edge–singular-vector pairs, calculate pairwise Jaccard distances between circuits from different prompts (we will refer to as circuit distance), and group them using average-linkage hierarchical clustering. Details in Appendix G.

Figure 3 summarizes how prompt-level circuits organize. Each panel shows the matrix of pairwise circuit distances, with rows and columns reordered by the clustering. Dendrograms on top of the figure are colored by high-level template (ABBA vs. BABA) or by low-level template group, and the horizontal dashed line marks the cut height that defines the clusters outlined by green boxes on the diagonal. In GPT-2 Small, the dendrogram splits cleanly into one ABBA cluster (left) and one BABA cluster (right), with no clear substructure related to low-level template. On the other hand, Pythia-160M has a larger number of smaller clusters, each aligned with low-level template groups. This shows that IOI prompt sensitivity is strong in both models, but the relevant form of variation is model-dependent: *GPT-2 is strongly sensitive to role order, whereas Pythia is more sensitive to surface wording.*

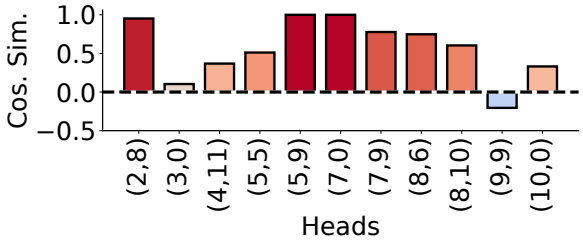


Figure 4. Signal similarity uncovers common and distinct functionality across ABBA vs. BABA circuits.

ACC++ explains prompt-level circuit variation. The substantial prompt-level variation of IOI circuits is highly structured: prompts form coherent clusters of circuits for solving the same task. To characterize what drives this sensitivity,

we summarize each cluster by a single *representative circuit*: the traced graph whose component set has the smallest average circuit distance to the other circuits in the cluster (Appendix G.3). This gives a stable unit of analysis for comparing different clusters.

To compare the signals used by each representative, we aggregate the incoming ACC++ signals of each attention head (see Figure 2) into a single destination signal summary and source summary, and then compute cosine similarities between these signal summaries across representatives of the clusters (Appendix G.4). Figure 4 shows the destination-signal comparison for the GPT-2 ABBA and BABA representatives. The figure shows that many signals are reused across templates. However, the name-mover head $(9, 9)$ receives markedly different input signals (negative cosine similarity) between ABBA and BABA, even though these signals are carried on the same token. Inspection shows that this is a critical difference that drives the separation of circuits for the two prompt types into distinct clusters. More details and other models comparisons are in Appendix H.

Interpreting the signals involved (§ 2) provides further insight on why the (9,9) head processes different inputs in the two cases. Figure 5 shows the circuit components upstream of head $(9, 9)$ in each template representative (details on how these qualitative trace figures were constructed from the annotated interactive traces are given in Appendix F.3). In the BABA case, the indirect object appears after the subject, and the model tags it with the feature "second item in a parallel pair." Four different heads each contribute this feature to the "Kelly" token. At the same time, head $(8, 6)$ suppresses the "Jack" token by writing the same feature into "to". Head $(9, 9)$ then attends to the pair (to, Kelly) as a result of this feature match, moving the indirect object to the output.

In contrast, in the ABBA circuit the indirect object comes before the subject and so cannot be identified as the "second item." The key features used to identify it are instead more generic structural cues, such as "structural or sequence-completing token" and "tokens in parallel structures that repeat or mirror a preceding element." Thus the two templates reuse a canonical IOI component, but feed it with different upstream mechanisms: one centered on identifying the second item in a pair, and one centered on broader structural matching.

We note the value of ACC++ here: by producing circuits with interpretable signals, ACC++ does not just recover that GPT-2 uses different circuits for ABBA and BABA; it shows that the same head (9,9) can implement its function using different incoming signals, and lets us describe what those signals are doing.

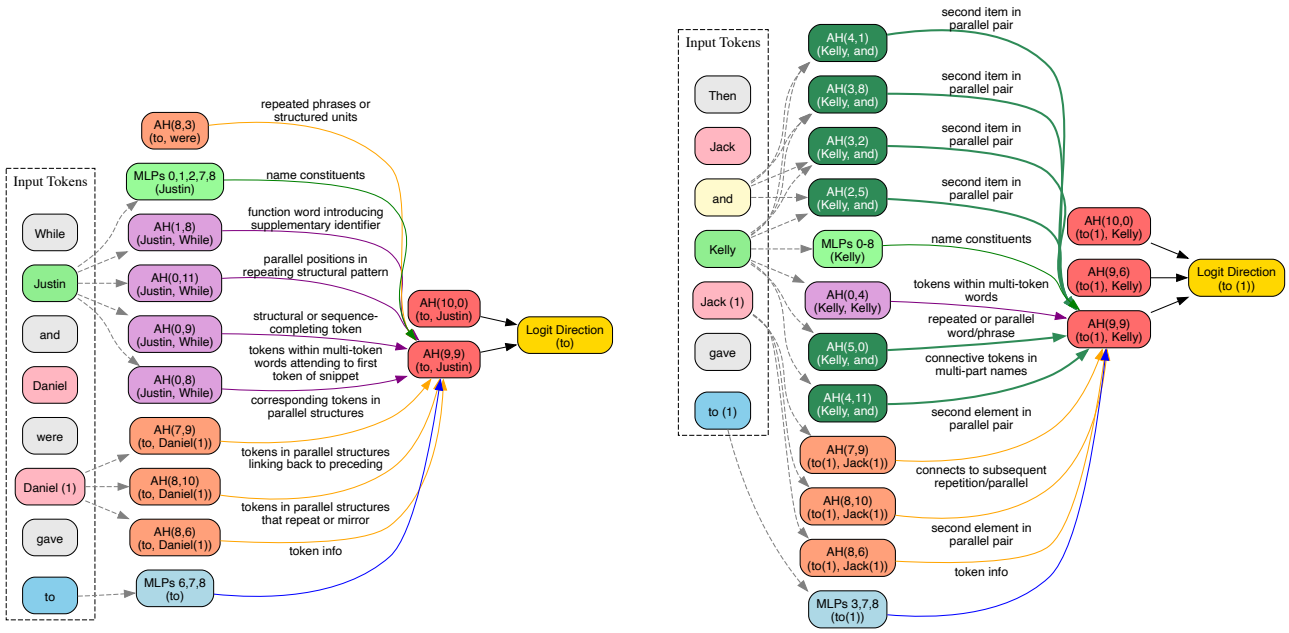
3.2. Multilingual IOI: Shared Mechanisms, Language-Specific Signals

The sensitivity of IOI circuits to prompt structure shows that even when components are reused across task variants, the signals involved may differ in important ways. We next explore this phenomenon in a setting where differences between prompt types are substantially greater: multilingual task variation. We extend the IOI task by constructing a multilingual IOI dataset spanning English, Spanish, French, and Portuguese prompts (Appendix I). The results in this section use mGPT (Shliazhko et al., 2024), but a similar analysis holds for Gemma-2 2B (Appendix I)

Components are reused; signals, not necessarily. Prior work has shown that models can use the same components when processing IOI prompts and their translation into another language (Zhang et al., 2025). We first confirm that ACC++ also recovers this component reuse (Appendix I.2, Figure 40).

However, analyzing signals via ACC++ shows *how* components are reused. Thus, we can analyze if components reuse the same signals across languages, or if these components carry language-specific signals. Figure 6 shows the clustering results using the same pipeline as § 3.1, which summarizes how multilingual traces cluster in mGPT. Importantly, clustering is based on signal use as well as component use, and so shows more than just component reuse. As with the monolingual case, the primary clustering feature is template structure: prompts sharing the same high-level template group together regardless of language. This indicates that models carry some language-agnostic signals. However, within each high-level template cluster there is a *secondary* grouping: prompts cluster by language with sub-clusters corresponding to English, Spanish, French, and Portuguese. This secondary clustering indicates that together with shared mechanisms, circuits also incorporate language-specific signals.

Circuit distance correlates with linguistic distance. In Figure 6, Spanish and Portuguese mix more than the other languages, suggesting that more closely related languages share more signals. To test whether the secondary language-specific clustering is meaningful, we correlate our cross-language circuit distances with typological distance between languages. For each language pair, we compute the average Jaccard distance between paired prompt traces (matched within high- and low-level templates) and correlate it against measures of language similarity as provided by URIEL+ (Khan et al., 2025). Figure 7 shows positive Pearson correlations with URIEL+ syntactic ($r = 0.83$) and genetic ($r = 0.88$) distance for mGPT, with the same pattern in Gemma-2 2B ($r = 0.80$ and $r = 0.84$, respectively; see Appendix I.2 for the corresponding figure and methodology). Although the panel of six language pairs



(a) ABBA: "While Justin and Daniel were working at the house, Daniel gave a snack to"

(b) BABA: "Then, Jack and Kelly went to the garden. Jack gave a basketball to"

Figure 5. Traces are interpretable and expose algorithmic differences in circuits between ABBA and BABA in GPT-2. AH node labels are (destination token, source token); edge labels are automatically generated. Red: feeds logits, purple: low-level, orange: provide inhibition signal. Dark green nodes show that, in BABA only, the circuit relies on identifying the "second item in a parallel pair", i.e. "Kelly", as the appropriate output.

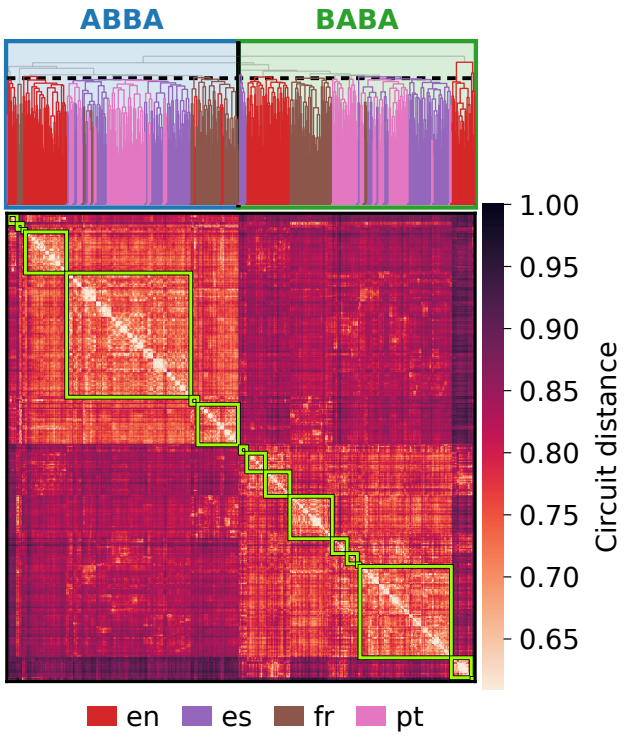


Figure 6. Average linkage clustering of prompt-level traces exposes grouping by high-level template first then by language. Branches in the dendrograms are colored by language.

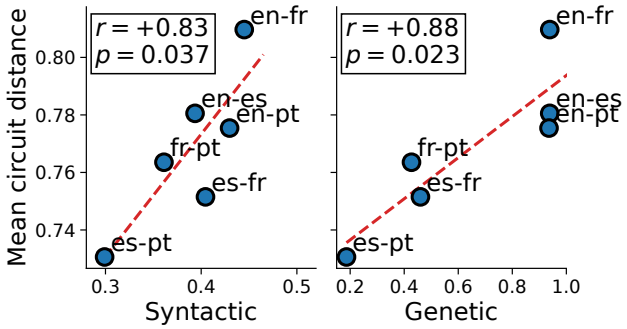


Figure 7. Mean circuit distances between mGPT language pairs strongly correlate with URIEL+ syntactic (left) and genetic (right) distances.

is small, the consistency of the trend across two models as well as both syntactic and genetic distances suggests that cross-language circuit distance is not arbitrary, and instead mirrors the relatedness of these languages.

4. Related Work

A large body of work suggests that many features are represented in low-dimensional directions (Elhage et al., 2022; Olah et al., 2020; Mikolov et al., 2013; Alain & Bengio, 2018; Park et al., 2023; Gurnee & Tegmark, 2024; Gurnee

et al., 2023; Marks et al., 2024; Levy & Geva, 2025; Engels et al., 2025; Kantamneni & Tegmark, 2025; Hernandez et al., 2024). Sparse Autoencoders (SAEs) are widely used to extract such features (Huben et al., 2024; Bricken et al.), cross-layer transcoders extend this idea to prompt-level attribution graphs over a learned replacement model (Lindsey et al., 2025), and Low-Rank Sparse Attention applies the same surrogate-model strategy to attention layers themselves (He et al., 2026); each having known drawbacks, including identifiability and stability issues (Leask et al., 2025; Gao et al., 2024; Bussmann et al.; Chanin et al., 2024) and a reliance on activations alone that does not use model parameters (bilalchughtai & Bushnaq). (Arora et al., 2026) avoids surrogates by showing that MLP neurons themselves form a sparse feature basis suitable for circuit tracing using gradient-based attribution. ACC++, similarly to (Franco & Crovella, 2025), considers *both* activations and weights (QK matrices) in identifying signals, and operates directly on the original model rather than on a replacement model.

Several recent studies also use the singular value decomposition (SVD) of attention interactions to study how heads communicate through low-dimensional subspaces. The work in (Merullo et al., 2024b) shows that inter-layer communication can be identified with subspaces aligned with singular vectors of attention heads, and (Franco et al., 2026) shows that singular vectors themselves can align with attention-causal features. The work in (Ahmad et al., 2025) uses an optimization procedure to identify task-relevant directions in the SVD basis of an attention head, and shows that a single head can simultaneously implement multiple independent computations. In a different setting, (Pan et al., 2024) studies token interactions in vision transformers and treats left/right singular vectors of the query–key interaction as paired interacting directions. ACC++ builds on this line by using SVD directions as *candidate communication channels* and pairing them with upstream components to form precise signal candidates (§ 2).

For circuit tracing, many methods rely on counterfactual interventions such as activation patching (Zhang & Nanda, 2024; Goldowsky-Dill et al., 2023; Wang et al., 2023; Conmy et al., 2023; Hanna et al., 2023a; Lieberum et al., 2023; Hanna et al., 2024). Patching is time-consuming, requiring many forward passes, often requires constructing a counterfactual dataset, and has multiple documented weaknesses (Makelov et al., 2023; Mueller, 2024; McGrath et al., 2023; Rushing & Nanda, 2024; Garriga-Alonso et al., 2024). The work in (Méloux et al., 2025) reports that EAP-IG traces (Hanna et al., 2024) can show high structural variance across models and tasks. Moreover, (Wu & Barez, 2025) pursues per-prompt circuit discovery, ensembling EAP-IG over query paraphrases to find faithful query-specific circuits. ACC++ is complementary: it traces a per-prompt circuit from a single forward pass and exposes the signals that causes each attention decision, rather than only the component-to-component edges that drive overall performance.

5. Conclusions

We present ACC++, a practical tool for circuit tracing and mechanistic analysis. From a single forward model pass and without replacement models, activation patching, or hand-designed counterfactual inputs, it produces circuits whose components are causal for attention and whose edges admit a natural-language interpretation in a substantial fraction of cases. By making per-prompt circuits a viable unit of analysis at scale, ACC++ circuits uncover insights regarding the sensitivity of circuits to prompt formats, and the role of signals across languages.

Limitations. ACC++ has some limitations that we leave as future work. First, the method depends on setting the threshold τ, and we use a heuristic based on the distribution of the $d \cdot A_{ds}$ per model/dataset. A principled, dataset-agnostic rule for τ is left to future work. Second, ACC++ circuits do not trace upstream components that caused FFNs to output an attention-causal signal. Third, because our signal interpretations rely on an LLM as a judge, the interpretations are suggestive but not conclusive. Finally, the multilingual analysis of § 3.2, including the correlation with URIEL+ distances, is based on a small number of language pairs and should also be read as suggestive rather than conclusive.

Acknowledgements

Carson Loughridge suggested the methodological improvement for models with bias described in Appendix B.1. This work also benefited from early feedback from Aaron Mueller, Micah Benson, Divya Appapogu, Freya Behrens, Najoung Kim, Yukyung Lee, and Micah Adler. This research was funded by a grant from Coefficient Giving and by NSF award CNS-2312711.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Ahmad, A., Joshi, A., and Modi, A. Beyond components: Singular vector-based interpretability of transformer circuits. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=7UbXEQNny7.

Alain, G. and Bengio, Y. Understanding intermediate layers using linear classifier probes, 2018. URL https://arxiv.org/abs/1610.01644.

Arora, A., Wu, Z., Steinhardt, J., and Schwettmann, S. Language model circuits are sparse in the neuron basis. *arXiv preprint arXiv:2601.22594*, 2026.

Athwin, C., Corlouer, G., Kran, E., Barez, F., and Nanda, N. Identifying a preliminary circuit for predicting gendered pronouns in gpt-2 small, 2023. *URL https://cmathw.itch. io/identifying-a-preliminary-circuit-for-predicting-gen dered-pronouns-in-gpt-2-smal*.

Benjamini, Y. and Hochberg, Y. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society. Series B (Methodological)*, 57(1):289–300, 1995. ISSN 00359246. URL http://www.jstor.org/stable/2346101.

Bhaskar, A., Wettig, A., Friedman, D., and Chen, D. Finding transformer circuits with edge pruning. *Advances in Neural Information Processing Systems*, 37:18506–18534, 2025.

Biderman, S., Schoelkopf, H., Anthony, Q. G., Bradley, H., O'Brien, K., Hallahan, E., Khan, M. A., Purohit, S., Prashanth, U. S., Raff, E., et al. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pp. 2397–2430. PMLR, 2023.

bilalchughtai and Bushnaq, L. Activation space interpretability may be doomed. https://www.lesswrong.com/posts/gYfpPbww3wQRaxAFD/activation-space-interpretability-may-be-doomed.

Bricken, T., Templeton, A., Batson, J., Chen, B., Jermyn, A., Conerly, T., Turner, N. L., Anil, C., Denison, C., Askell, A., Lasenby, R., Wu, Y., Kravec, S., Schiefer, N., Maxwell, T., Joseph, N., Tamkin, A., Nguyen, K., McLean, B., Burke, J. E., Hume, T., Carter, S., Henighan, T., and Olah, C. Towards monosemanticity: Decomposing language models with dictionary learning. https://transformer-circuits.pub/2023/monosemantic-features.

Bussmann, B., Pearce, M., Leask, P., Bloom, J., Sharkey, L., and Nanda, N. Showing SAE latents are not atomic using Meta-SAEs. https://www.lesswrong.com/posts/TMAmHh4DdMr4nCSr5/showing-sae-latents-are-not-atomic-using-meta-saes.

Chanin, D., Wilken-Smith, J., Dulka, T., Bhatnagar, H., and Bloom, J. A is for absorption: Studying feature splitting and absorption in sparse autoencoders, 2024. URL https://arxiv.org/abs/2409.14507.

Conmy, A., Mavor-Parker, A., Lynch, A., Heimersheim, S., and Garriga-Alonso, A. Towards automated circuit discovery for mechanistic interpretability. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S. (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 16318–16352. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/34e1dbe95d34d7ebaf99b9bcaeb5b2be-Paper-Conference.pdf.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Elhage, N., Nanda, N., Olsson, C., Henighan, T., Joseph, N., Mann, B., Askell, A., Bai, Y., Chen, A., Conerly, T., DasSarma, N., Drain, D., Ganguli, D., Hatfield-Dodds, Z., Hernandez, D., Jones, A., Kernion, J., Lovitt, L., Ndousse, K., Amodei, D., Brown, T., Clark, J., Kaplan, J., McCandlish, S., and Olah, C. A Mathematical Framework for Transformer Circuits. *Transformer Circuits Thread*, 2021. URL https://transformer-circuits.pub/2021/framework/index.html.

Elhage, N., Hume, T., Olsson, C., Schiefer, N., Henighan, T., Kravec, S., Hatfield-Dodds, Z., Lasenby, R., Drain, D., Chen, C., Grosse, R., McCandlish, S., Kaplan, J., Amodei, D., Wattenberg, M., and Olah, C. Toy Models of Superposition. *Transformer Circuits Thread*, 2022. URL https://transformer-circuits.pub/2022/toy_model/.

Engels, J., Michaud, E. J., Liao, I., Gurnee, W., and Tegmark, M. Not all language model features are one-dimensionally linear, 2025. URL https://arxiv.org/abs/2405.14860.

Fisher, R. A. On the interpretation of χ^2 from contingency tables, and the calculation of p. *Journal of the royal statistical society*, 85(1):87–94, 1922.

Franco, G. and Crovella, M. Pinpointing attention-causal communication in language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=wUoK24u4x7.

Franco, G., Loughridge, C., and Crovella, M. Singular vectors of attention heads align with features, 2026. URL https://arxiv.org/abs/2602.13524.

Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Gao, L., la Tour, T. D., Tillman, H., Goh, G., Troll, R., Radford, A., Sutskever, I., Leike, J., and Wu, J. Scaling and evaluating sparse autoencoders, 2024. URL https://arxiv.org/abs/2406.04093.

Garriga-Alonso, A. et al. Adversarial circuit evaluation. *arXiv preprint arXiv:2407.15166*, 2024.

Goldowsky-Dill, N., MacLeod, C., Sato, L., and Arora, A. Localizing model behavior with path patching, 2023. URL https://arxiv.org/abs/2304.05969.

Gurnee, W. and Tegmark, M. Language models represent space and time. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=jE8xbmvFin.

Gurnee, W., Nanda, N., Pauly, M., Harvey, K., Troitskii, D., and Bertsimas, D. Finding neurons in a haystack: Case studies with sparse probing. *arXiv*, 2023. doi: 10.48550/arxiv.2305.01610.

Haklay, T., Orgad, H., Bau, D., Mueller, A., and Belinkov, Y. Position-aware automatic circuit discovery. *arXiv preprint arXiv:2502.04577*, 2025.

Hanna, M., Liu, O., and Variengien, A. How does gpt-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. *arXiv*, 2023a. doi: 10.48550/arxiv.2305.00586.

Hanna, M., Liu, O., and Variengien, A. How does gpt-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. *Advances in Neural Information Processing Systems*, 36:76033–76060, 2023b.

Hanna, M., Pezzelle, S., and Belinkov, Y. Have faith in faithfulness: Going beyond circuit overlap when finding model mechanisms. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=TZ0CCGDcuT.

He, Z., Wang, J., Lin, R., Ge, X., Shu, W., Tang, Q., Zhang, J., and Qiu, X. Towards understanding the nature of attention with low-rank sparse decomposition. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=9A2etpDFIB.

Hernandez, E., Sharma, A. S., Haklay, T., Meng, K., Wattenberg, M., Andreas, J., Belinkov, Y., and Bau, D. Linearity of relation decoding in transformer language models, 2024. URL https://arxiv.org/abs/2308.09124.

Huben, R., Cunningham, H., Smith, L. R., Ewart, A., and Sharkey, L. Sparse autoencoders find highly interpretable features in language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=F76bwRSLeK.

Kantamneni, S. and Tegmark, M. Language models use trigonometry to do addition, 2025. URL https://arxiv.org/abs/2502.00873.

Khan, A., Shipton, M., Anugraha, D., Duan, K., Hoang, P. H., Khiu, E., Doğruöz, A. S., and Lee, E.-S. A. Uriel+: Enhancing linguistic inclusion and usability in a typological and multilingual knowledge base. In *Proceedings of the 31st International Conference on Computational Linguistics*, pp. 6937–6952, 2025.

Leask, P., Bussmann, B., Pearce, M., Bloom, J., Tigges, C., Moubayed, N. A., Sharkey, L., and Nanda, N. Sparse autoencoders do not find canonical units of analysis, 2025. URL https://arxiv.org/abs/2502.04878.

Levy, A. A. and Geva, M. Language models encode numbers using digit representations in base 10, 2025. URL https://arxiv.org/abs/2410.11781.

Lieberum, T., Rahtz, M., Kramár, J., Nanda, N., Irving, G., Shah, R., and Mikulik, V. Does circuit analysis interpretability scale? evidence from multiple choice capabilities in chinchilla. *arXiv*, 2023. doi: 10.48550/arxiv.2307.09458.

Lindsey, J., Gurnee, W., Ameisen, E., Chen, B., Pearce, A., Turner, N. L., Citro, C., Abrahams, D., Carter, S., Hosmer, B., Marcus, J., Sklar, M., Templeton, A., Bricken, T., McDougall, C., Cunningham, H., Henighan, T., Jermyn, A., Jones, A., Persic, A., Qi, Z., Thompson, T. B., Zimmerman, S., Rivoire, K., Conerly, T., Olah, C., and Batson, J. On the biology of a large language model. *Transformer Circuits Thread*, 2025. URL https://transformer-circuits.pub/2025/attribution-graphs/biology.html.

Makelov, A., Lange, G., and Nanda, N. Is this the subspace you are looking for? an interpretability illusion for subspace activation patching. *arXiv*, 2023. doi: 10.48550/arxiv.2311.17030.

Marks, S., Rager, C., Michaud, E. J., Belinkov, Y., Bau, D., and Mueller, A. Sparse feature circuits: Discovering and

editing interpretable causal graphs in language models. *arXiv*, 2024.

McGrath, T., Rahtz, M., Kramar, J., Mikulik, V., and Legg, S. The hydra effect: Emergent self-repair in language model computations. *arXiv*, 2023. doi: 10.48550/arxiv.2307.15771.

Méloux, M., Portet, F., and Peyrard, M. Mechanistic interpretability as statistical estimation: A variance analysis of EAP-IG. In *Mechanistic Interpretability Workshop at NeurIPS 2025*, 2025. URL https://openreview.net/forum?id=UbOAXViKsU.

Merullo, J., Eickhoff, C., and Pavlick, E. Circuit component reuse across tasks in transformer language models. In *The Twelfth International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=fpoAYV6Wsk.

Merullo, J., Eickhoff, C., and Pavlick, E. Talking heads: Understanding inter-layer communication in transformer language models, 2024b. URL https://arxiv.org/abs/2406.09519.

Mikolov, T., Yih, W.-t., and Zweig, G. Linguistic regularities in continuous space word representations. In Vanderwende, L., Daumé III, H., and Kirchhoff, K. (eds.), *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 746–751, Atlanta, Georgia, June 2013. Association for Computational Linguistics. URL https://aclanthology.org/N13-1090.

Mueller, A. Missed causes and ambiguous effects: Counterfactuals pose challenges for interpreting neural networks. *arXiv*, 2024. doi: 10.48550/arxiv.2407.04690.

Nanda, N. The first 10,000 elements of the pile. https://huggingface.co/datasets/NeelNanda/pile-10k, 2024.

Olah, C., Cammarata, N., Schubert, L., Goh, G., Petrov, M., and Carter, S. Zoom in: An introduction to circuits. *Distill*, 2020. doi: 10.23915/distill.00024.001. https://distill.pub/2020/circuits/zoom-in.

Pan, X., Philip, A., Xie, Z., and Schwartz, O. Dissecting query-key interaction in vision transformers. In Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., and Zhang, C. (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 54595–54631. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/6216515a5e0b3257c49dcb1647e497d1-Paper-Conference.pdf.

Park, K., Choe, Y. J., and Veitch, V. The linear representation hypothesis and the geometry of large language models. In *Causal Representation Learning Workshop at NeurIPS 2023*, 2023. URL https://openreview.net/forum?id=T0PoOJg8cK.

Paulo, G. S., Mallen, A. T., Juang, C., and Belrose, N. Automatically interpreting millions of features in large language models. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=EemtbhJOXc.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Rushing, C. and Nanda, N. Explorations of self-repair in language models. *arXiv*, 2024. doi: 10.48550/arxiv.2402.15390.

Shliazhko, O., Fenogenova, A., Tikhonova, M., Kozlova, A., Mikhailov, V., and Shavrina, T. mgpt: Few-shot learners go multilingual. *Transactions of the Association for Computational Linguistics*, 12:58–79, 2024.

Su, J., Ahmed, M., Lu, Y., Pan, S., Bo, W., and Liu, Y. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Sundararajan, M., Taly, A., and Yan, Q. Axiomatic attribution for deep networks. In *International conference on machine learning*, pp. 3319–3328. PMLR, 2017.

Syed, A., Rager, C., and Conmy, A. Attribution patching outperforms automated circuit discovery. In *The 7th BlackboxNLP Workshop*, 2024. URL https://openreview.net/forum?id=RysbaxAnc6.

Team, G., Riviere, M., Pathak, S., Sessa, P. G., Hardin, C., Bhupatiraju, S., Hussenot, L., Mesnard, T., Shahriari, B., Ramé, A., et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

Team, G., Kamath, A., Ferret, J., Pathak, S., Vieillard, N., Merhej, R., Perrin, S., Matejovicova, T., Ramé, A., Rivière, M., et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

Tigges, C., Hanna, M., Yu, Q., and Biderman, S. LLM circuit analyses are consistent across training and scale. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=3Ds5vNudIE.

Wang, K. R., Variengien, A., Conmy, A., Shlegeris, B., and Steinhardt, J. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. In *The Eleventh*

International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023. OpenReview.net, 2023. URL https://openreview.net/forum?id=NpsVSN6o4ul.

Wu, T.-Y. and Barez, F. Query circuits: Explaining how language models answer user prompts. *arXiv preprint arXiv:2509.24808*, 2025.

Zhang, F. and Nanda, N. Towards best practices of activation patching in language models: Metrics and methods, 2024. URL https://arxiv.org/abs/2309.16042.

Zhang, R., Yu, Q., Zang, M., Eickhoff, C., and Pavlick, E. The same but different: Structural similarities and differences in multilingual language modeling. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=NCrFA7dq8T.

A. Notation Used in the Appendices

We follow mostly the notation from (Franco & Crovella, 2025). In the model, token embeddings are D-dimensional, there are H attention heads in each layer, and there are L layers. We define $R = \frac{D}{H}$, which is the dimension of the spaces used for keys and queries in the attention mechanism. We use N to denote the number of tokens in a given prompt. Superscript indices will denote (layer, head, destination token, source token); reduced sets of indices will be used where there is no confusion, and subscripts generally denote matrix components unless otherwise stated.

Attention scores and weights. The attention mechanism operates on a set of N tokens in D-dimensional embeddings: $X \in \mathbb{R}^{N \times D}$. Each token $\mathbf{x} \in \mathbb{R}^D$ is passed through linear transforms given by $\mathbf{x}^\top W_K$, $\mathbf{x}^\top W_Q$, using weight matrices $W_K, W_Q \in \mathbb{R}^{D \times R}$. Then the inner product is taken for all pairs of transformed tokens to yield attention scores

$$A'_{ds} = \mathbf{x}^{d\top} \Omega \mathbf{x}^s, \tag{1}$$

in which $\Omega = W_Q W_K^\top$, $\mathbf{x}^d$ is the destination token, and $\mathbf{x}^s$ is the source token of the attention computation. To enforce masked self-attention, A'_{ds} is set to $-\infty$ for $d < s$. Attention scores are then normalized for each destination d, yielding attention weights

$$A_d = \mathrm{Softmax}(A'_d/\sqrt{R}), \qquad A_{ds} = [A_d]_s. \tag{2}$$

Layer/head indexing convention in this appendix. Throughout the appendix sections that follow, we fix an attention head (ℓ, a) and analyze its QK circuit at a fixed layer input. Accordingly, we write $\mathbf{x}^d$ to mean "the input residual embedding at token d at the input to layer ℓ". When we need to refer to other layers explicitly, we will add a layer superscript, e.g. X^ℓ for the layer-ℓ residual matrix.

SVD of the QK matrix. Our methods make use of the SVD of Ω. The matrix Ω has size $D \times D$, but due to its construction it has maximum rank R. We therefore work with the SVD of $\Omega = U\Sigma V^\top$ in which $U \in \mathbb{R}^{D \times R}$, $V \in \mathbb{R}^{D \times R}$ and $\Sigma \in \mathbb{R}^{R \times R}$. U and V are orthonormal matrices with $U^\top U = I$ and $V^\top V = I$, and $\Sigma = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_R)$ with $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_R \geq 0$. Important to our work is that the SVD of Ω can equivalently be written as

$$\Omega = \sum_{k=1}^{R} \mathbf{u}^k \sigma^k \mathbf{v}^{k\top}. \tag{3}$$

Residual decomposition into upstream component outputs. We will use the standard view that the residual stream at the input to a layer is a linear sum of outputs of upstream components (Elhage et al., 2021). Let $\mathcal{C}(\ell)$ denote the set of all components upstream of the start of layer ℓ (including the input embedding). For each token position d, let $\mathbf{o}_c^d \in \mathbb{R}^D$ denote the output written into token d by component $c \in \mathcal{C}(\ell)$. Then

$$\mathbf{x}^d = \sum_{c \in \mathcal{C}(\ell)} \mathbf{o}_c^d. \tag{4}$$

(Here the subscript c indexes components; it is not a matrix component.)

B. A Unified Bilinear Form Encompassing Bias and Rotational Positional Encoding

As discussed in Section 2, ACC++ makes a number of methodological improvements to ACC. One such improvement overcomes a significant source of complexity in the original ACC. To deal with bias in the attention computation, ACC used homogeneous coordinates (vectors with an additional component equal to 1), and to deal with rotary positional encoding, ACC used position-specific Ω matrices. Both of these techniques added significant complexity to the code. In ACC++ we eliminate these sources of complexity, and cast all cases (whether involving bias, rotary encoding, both, or neither) in terms of a single Ω matrix per head, which means that ACC++ only requires one SVD operation per head.

Core Assumptions

To achieve these improvements, we express the attention score as a bilinear form involving Ω *in all cases*. We rely on the fact that the query and key weight matrices, $W_Q \in \mathbb{R}^{D\times R}$ and $W_K \in \mathbb{R}^{D\times R}$ (where $D > R$), are **well-conditioned** (and thus have full column rank). For the models studied in this work, this assumption is **supported empirically:** Appendix B.4 shows that both W_Q and $W_K^\top$ are well-conditioned, in all cases having condition numbers below 1000, and usually much lower.

This assumption has a critical implication: for a full column rank matrix W, $W^\dagger W = I_R$, and when W is well conditioned then computing $W^\dagger W$ is very close to I_R numerically. Therefore, our derivations rely on the following identities, which allow our factorization of bias terms and simplification of the rotational matrix composition:

$$W_Q^\dagger W_Q = I_R \quad \text{and} \quad W_K^\top (W_K^\top)^\dagger = I_R$$

B.1. Models with Bias (e.g, GPT-2)

Let $\mathbf{x}^d, \mathbf{x}^s \in \mathbb{R}^{D\times 1}; W_Q, W_K \in \mathbb{R}^{D\times R}; \mathbf{b}_Q, \mathbf{b}_K \in \mathbb{R}^{R\times 1}$. The standard attention score for a model with bias is given by:

$$A'_{ds} = \left(\mathbf{x}^{d\top} W_Q + \mathbf{b}_Q^\top\right)\left(\mathbf{x}^{s\top} W_K + \mathbf{b}_K^\top\right)^\top$$

To factor out W_Q and W_K, we rely on the well-conditioning of the matrices, which means $W_Q^\dagger W_Q = I_R$ and $W_K^\dagger W_K = I_R$. This allows us to state $\mathbf{b}_Q^\top = \mathbf{b}_Q^\top (W_Q^\dagger W_Q)$ and $\mathbf{b}_K^\top = \mathbf{b}_K^\top (W_K^\dagger W_K)$, permitting the factorization:

$$\begin{aligned}
&= \left[\left(\mathbf{x}^{d\top} + \mathbf{b}_Q^\top W_Q^\dagger\right) W_Q\right]\left[\left(\mathbf{x}^{s\top} + \mathbf{b}_K^\top W_K^\dagger\right) W_K\right]^\top \\
&= \left(\mathbf{x}^{d\top} + \mathbf{b}_Q^\top W_Q^\dagger\right) W_Q W_K^\top \left(\mathbf{x}^s + W_K^{\dagger\top} \mathbf{b}_K\right) \\
&= \left(\mathbf{x}^{d\top} + \mathbf{c}^{d\top}\right) \Omega \left(\mathbf{x}^s + \mathbf{c}^s\right)
\end{aligned}$$

The final expression is a bilinear form with additive bias terms. The unified attention matrix is $\Omega = W_Q W_K^\top$, and the constant bias-derived vectors are $\mathbf{c}^{d\top} = \mathbf{b}_Q^\top W_Q^\dagger$ and $\mathbf{c}^s = W_K^{\dagger\top} \mathbf{b}_K$.

B.2. Models with RoPE (e.g., Gemma-2)

Let $\mathbf{x}^d, \mathbf{x}^s \in \mathbb{R}^{D\times 1}; W_Q, W_K \in \mathbb{R}^{D\times R}; \mathbf{b}_Q, \mathbf{b}_K \in \mathbb{R}^{R\times 1}; \mathcal{R}^d, \mathcal{R}^s \in \mathbb{R}^{R\times R}$. The attention score for token pair (d, s) in RoPE models is given by:

$$\begin{aligned}
A'_{ds} &= \left(\mathbf{x}^{d\top} W_Q \mathcal{R}^d\right)\left(\mathbf{x}^{s\top} W_K \mathcal{R}^s\right)^\top \\
&= \mathbf{x}^{d\top} W_Q \mathcal{R}^d \mathcal{R}^{s\top} W_K^\top \mathbf{x}^s \\
&= \mathbf{x}^{d\top} W_Q \mathcal{R}^{(d-s)} W_K^\top \mathbf{x}^s \\
&= \mathbf{x}^{d\top} \Omega^{(d-s)} \mathbf{x}^s
\end{aligned}$$

This form uses a position-dependent attention matrix $\Omega^{(d-s)}$. To reformulate this using a fixed Ω, we choose operators $M_d, M_s \in \mathbb{R}^{D\times D}$ that apply an appropriate rotation directly to the input vectors, such that $A'_{ds} = ({\mathbf{x}^d}^\top M_d)W_Q W_K^\top(M_s\mathbf{x}^s)$.

Let $M_d = W_Q\mathcal{R}^d W_Q^\dagger$ and $M_s = (W_K^\top)^\dagger {\mathcal{R}^s}^\top W_K^\top$. We verify this formulation:

$$\begin{aligned}
A'_{ds} &= ({\mathbf{x}^d}^\top M_d)\Omega(M_s\mathbf{x}^s) \\
&= ({\mathbf{x}^d}^\top M_d)W_Q W_K^\top(M_s\mathbf{x}^s) \\
&= {\mathbf{x}^d}^\top W_Q\mathcal{R}^d(W_Q^\dagger W_Q)(W_K^\top(W_K^\top)^\dagger){\mathcal{R}^s}^\top W_K^\top\mathbf{x}^s
\end{aligned}$$

Based on our full row/column rank assumptions, $W_Q^\dagger W_Q = I_R$ and $W_K^\top(W_K^\top)^\dagger = I_R$ (as $W_K^\top$ is full row rank), allowing the expression to simplify:

$$\begin{aligned}
&= {\mathbf{x}^d}^\top W_Q\mathcal{R}^d{\mathcal{R}^s}^\top W_K^\top\mathbf{x}^s \\
&= {\mathbf{x}^d}^\top W_Q\mathcal{R}^{(d-s)}W_K^\top\mathbf{x}^s \\
&= {\mathbf{x}^d}^\top\Omega^{(d-s)}\mathbf{x}^s
\end{aligned}$$

This confirms that using the position-dependent operators M_d and M_s to transform tokens on *input* to the attention computation, the core attention computation can use a fixed, position-independent matrix $\Omega = W_Q W_K^\top$.

B.3. Models with Bias and RoPE (e.g, Pythia)

For this mode, we combine the two previous derivations, showing that the attention score for token pair (d, s) in a RoPE model with bias is given by:

$$\begin{aligned}
A'_{ds} &= \left[\left({\mathbf{x}^d}^\top W_Q + \mathbf{b}_Q^\top\right)\mathcal{R}^d\right]\left[\left({\mathbf{x}^s}^\top W_K + \mathbf{b}_K^\top\right)\mathcal{R}^s\right]^\top \\
&= \left[\left({\mathbf{x}^d}^\top W_Q + \mathbf{b}_Q^\top\right)\mathcal{R}^d\right]\left[{\mathcal{R}^s}^\top\left(W_K^\top\mathbf{x}^s + \mathbf{b}_K\right)\right] \\
&= \left({\mathbf{x}^d}^\top W_Q\mathcal{R}^d + \mathbf{b}_Q^\top\mathcal{R}^d\right)\left({\mathcal{R}^s}^\top W_K^\top\mathbf{x}^s + {\mathcal{R}^s}^\top\mathbf{b}_K\right) \\
&= \left({\mathbf{x}^d}^\top W_Q\mathcal{R}^d{\mathcal{R}^s}^\top W_K^\top\mathbf{x}^s\right) + \left({\mathbf{x}^d}^\top W_Q\mathcal{R}^d{\mathcal{R}^s}^\top\mathbf{b}_K\right) + \left(\mathbf{b}_Q^\top\mathcal{R}^d{\mathcal{R}^s}^\top W_K^\top\mathbf{x}^s\right) + \left(\mathbf{b}_Q^\top\mathcal{R}^d{\mathcal{R}^s}^\top\mathbf{b}_K\right)
\end{aligned}$$

This is the expression we aim to reconstruct. Our goal is to show that this expression is equivalent to the combined form:

$$A'_{ds} = \left[\left({\mathbf{x}^d}^\top + {\mathbf{c}^d}^\top\right)M_d\right]W_Q W_K^\top\left[M_s\left(\mathbf{x}^s + \mathbf{c}^s\right)\right]$$

where M_d, M_s, $\mathbf{c}^d$, and $\mathbf{c}^s$ are defined as in the previous sections. We verify this by expanding the left and right sides of this new expression.

First, we expand the left-hand query term:

$$\begin{aligned}
\left[\left({\mathbf{x}^d}^\top + {\mathbf{c}^d}^\top\right)M_d\right]W_Q &= \left({\mathbf{x}^d}^\top + \mathbf{b}_Q^\top W_Q^\dagger\right)(W_Q\mathcal{R}^d W_Q^\dagger)W_Q \\
&= \left({\mathbf{x}^d}^\top + \mathbf{b}_Q^\top W_Q^\dagger\right)W_Q\mathcal{R}^d(W_Q^\dagger W_Q)
\end{aligned}$$

Using the fact that here, $W_Q^\dagger W_Q = I_R$:

$$\begin{aligned}
&= \left({\mathbf{x}^d}^\top W_Q + \mathbf{b}_Q^\top W_Q^\dagger W_Q\right)\mathcal{R}^d \\
&= \left({\mathbf{x}^d}^\top W_Q + \mathbf{b}_Q^\top\right)\mathcal{R}^d \\
&= {\mathbf{x}^d}^\top W_Q\mathcal{R}^d + \mathbf{b}_Q^\top\mathcal{R}^d
\end{aligned}$$

Second, we expand the right-hand key term:

$$\begin{aligned} W_K^\top \left[M_s \left(\mathbf{x}^s + \mathbf{c}^s\right)\right] &= W_K^\top M_s \mathbf{x}^s + W_K^\top M_s \mathbf{c}^s \\ &= W_K^\top (W_K^\top)^\dagger \mathcal{R}^{s\top} W_K^\top \mathbf{x}^s + W_K^\top (W_K^\top)^\dagger \mathcal{R}^{s\top} W_K^\top {W_K^\dagger}^\top \mathbf{b}_K \end{aligned}$$

Using $W_K^\top (W_K^\top)^\dagger = I_R$ and $W_K^\top ({W_K^\dagger}^\top) = W_K^\top (W_K^\top)^\dagger = I_R$:

$$\begin{aligned} &= (I_R)\mathcal{R}^{s\top} W_K^\top \mathbf{x}^s + (I_R)\mathcal{R}^{s\top}(I_R)\mathbf{b}_K \\ &= \mathcal{R}^{s\top} W_K^\top \mathbf{x}^s + \mathcal{R}^{s\top}\mathbf{b}_K \qquad (*) \end{aligned}$$

where for $(*)$ we use the fact that ${W_K^\dagger}^\top = (W_K^\top)^\dagger$.

Combining the left and right parts, we have:

$$\begin{aligned} A'_{ds} &= \left({\mathbf{x}^d}^\top W_Q \mathcal{R}^d + \mathbf{b}_Q^\top \mathcal{R}^d\right)\left(\mathcal{R}^{s\top} W_K^\top \mathbf{x}^s + \mathcal{R}^{s\top}\mathbf{b}_K\right) \\ &= \left({\mathbf{x}^d}^\top W_Q \mathcal{R}^d \mathcal{R}^{s\top} W_K^\top \mathbf{x}^s\right) + \left({\mathbf{x}^d}^\top W_Q \mathcal{R}^d \mathcal{R}^{s\top}\mathbf{b}_K\right) + \left(\mathbf{b}_Q^\top \mathcal{R}^d \mathcal{R}^{s\top} W_K^\top \mathbf{x}^s\right) + \left(\mathbf{b}_Q^\top \mathcal{R}^d \mathcal{R}^{s\top}\mathbf{b}_K\right) \end{aligned}$$

This result exactly matches the original expanded equation. Therefore, the combined formulation correctly reconstructs the full attention score for Pythia-style models.

B.4. Checking the condition number of the W_Q and W_K^T matrices

Figures 8, 9, and 10 report the condition numbers of W_Q (left) and $W_K^\top$ (right) for the models considered. In all cases, the matrices are well-conditioned, with condition number < 1000, and usually much less.

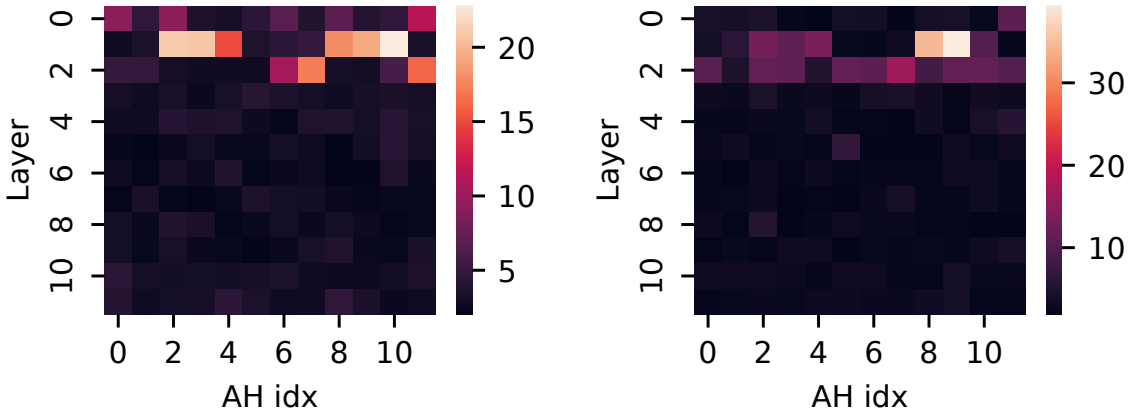


Figure 8. Condition numbers of W_Q (left) and $W_K^\top$ (right) from GPT-2 small.

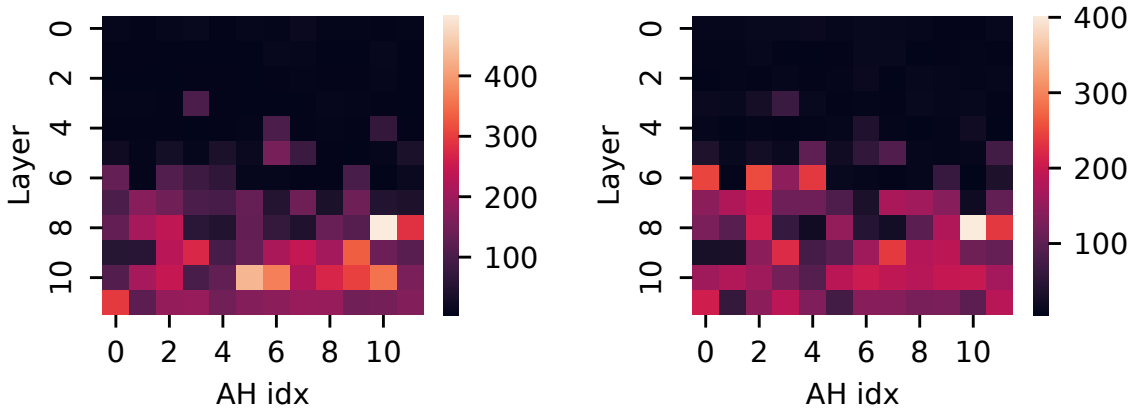


Figure 9. Condition numbers of W_Q (left) and $W_K^\top$ (right) from Pythia-160M.

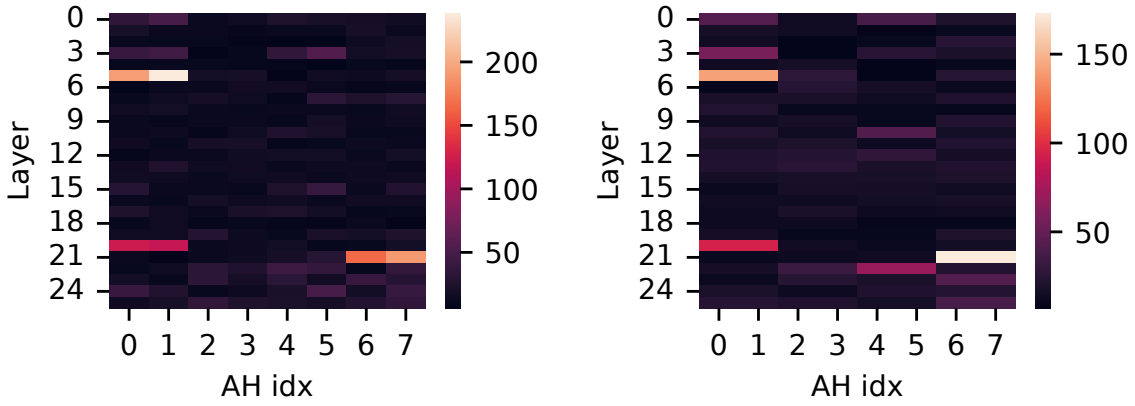


Figure 10. Condition numbers of W_Q (left) and $W_K^\top$ (right) from Gemma-2 2B.

C. ACC++: Refined ACC as a Counterfactual Search

This appendix provides the mathematical description of our refined attention-causal communication (ACC++) procedure. We follow the notation of Appendix A, and we assume throughout that we are analyzing a fixed attention head (ℓ, a) so that $\mathbf{x}^d$ denotes the residual at token d at the input to layer ℓ.

C.1. Problem definition

Fix a destination–source pair (d, s) such that the head places some attention on s, i.e. A_{ds} is high. As in (Franco & Crovella, 2025), we distinguish between *destination signals* (signals residing in $\mathbf{x}^d$) and *source signals* (signals residing in the source-token residuals $\{\mathbf{x}^j\}_{j\le d}$; these affect the weight A_{ds} through Softmax competition). Accordingly, we consider one of the following intervention targets, denoted generically by $\mathbf{x}$:

- **Destination case:** $\mathbf{x} = \mathbf{x}^d$ (identify destination-side signals that causally influence attention from d).
- **Source case:** $\mathbf{x} = \{\mathbf{x}^j\}_{j\le d}$ (identify source-side signals that causally influence the distribution over sources for destination d; this includes signals in $\mathbf{x}^s$ and is necessary to account for Softmax competition across all $j \le d$).

We distinguish two ACC++ counterfactuals, corresponding to *destination signals* (in $\mathbf{x}^d$) and *source signals* (in the source residuals $\{\mathbf{x}^j\}_{j\le d}$). Each counterfactual induces a different minimal signal set. A key improvement of ACC++ over ACC (Franco & Crovella, 2025) is that the counterfactual objective is defined directly in terms of the *post-Softmax* attention weight A_{ds}, rather than via a linear surrogate of attention; consequently, interventions are evaluated through their effect on the entire destination row A_d and its Softmax normalization. Finally, although the exposition in this appendix assumes a simplified setting with no QK bias and global positional encoding, the ACC++ procedure extends unchanged to models with attention bias terms and/or RoPE: these variants only modify the pre-Softmax scores by adding a constant term and/or applying a per-position rotation to token representations prior to multiplication by Ω. We refer to Appendix B for the precise forms and the corresponding minor bookkeeping changes.

To orient the reader, we now give a brief roadmap of the ACC++ procedure before introducing the formal definitions.

1. **Set up the counterfactual.** Choose an attention head and a destination–source pair, then decide whether we are searching for *destination* signals (in the destination token) or *source* signals (distributed across all source tokens that compete under Softmax).
2. **Enumerate candidate signals.** Decompose the residual stream into outputs of upstream components, and project each component's contribution onto the head's singular-vector directions to form a set of candidate signals.
3. **Build a contribution table.** Convert candidates into a contribution matrix whose rows correspond to candidate signals and whose columns correspond to source positions in the destination row of attention scores.
4. **Score candidates with attribution.** Use Integrated Gradients to assign each candidate a fixed importance score for the attention weight on the chosen source token, while accounting for Softmax competition across all sources.
5. **Solve the counterfactual by greedy removal.** Starting from the full set of candidates, iteratively remove the highest-scoring candidates and recompute the attention weight until it drops below a chosen threshold; the removed candidates form the ACC++ explanation set.

Destination intervention (find destination signals). Let M_{dst} denote a set of destination-side signal candidates. We define the intervention at the residual level as

$$\text{do}_{\text{dst}}(M_{\text{dst}}): \qquad \tilde{\mathbf{x}}^d = \mathbf{x}^d - \Delta^d(M_{\text{dst}}) \tag{5}$$

and define the induced scores and weights by

$$A'_{dj}\big(\text{do}_{\text{dst}}(M_{\text{dst}})\big) = \tilde{\mathbf{x}}^{d\top}\Omega\mathbf{x}^j, \qquad A_d\big(\text{do}_{\text{dst}}(M_{\text{dst}})\big) = \text{Softmax}\Big(A'_d\big(\text{do}_{\text{dst}}(M_{\text{dst}})\big)/\sqrt{R}\Big). \tag{6}$$

Here $\Delta^d(M_{\text{dst}})$ is the total vector removed from token d; see Eqs. (16)–(17).

The destination ACC++ objective is to find a small M_{dst} such that

$$\Big[A_d\big(\text{do}_{\text{dst}}(M_{\text{dst}})\big)\Big]_s < \tau. \tag{7}$$

Appendix D shows how we find τ in this work.

Source intervention (find source signals). Let M_{src} denote a set of source-side signal candidates. We intervene on the source-side (key-side) residuals (for all $j \le d$) while keeping the destination residual fixed on the query side as

$$\text{do}_{\text{src}}(M_{\text{src}}): \qquad \tilde{\mathbf{x}}^j = \mathbf{x}^j - \Delta^j(M_{\text{src}}) \ \text{ for all } j \le d, \tag{8}$$

and define the induced scores and weights by

$$A'_{dj}\big(\text{do}_{\text{src}}(M_{\text{src}})\big) = \mathbf{x}^{d\top}\Omega\tilde{\mathbf{x}}^j, \qquad A_d\big(\text{do}_{\text{src}}(M_{\text{src}})\big) = \text{Softmax}\Big(A'_d\big(\text{do}_{\text{src}}(M_{\text{src}})\big)/\sqrt{R}\Big). \tag{9}$$

Similarly $\Delta^j(M_{\text{src}})$ is the total vector removed at token j; see Eqs. (16)–(17).

The source ACC++ objective is to find a minimal M_{src} such that

$$\Big[A_d\big(\text{do}_{\text{src}}(M_{\text{src}})\big)\Big]_s < \tau. \tag{10}$$

Because $A_d = \text{Softmax}(A'_d/\sqrt{R})$ couples all sources $j \le d$ through Softmax normalization, both objectives must be evaluated with respect to their effect on the full score vector A'_d, not only the entry A'_{ds}.

C.2. Signal candidates from component outputs and the SVD basis

We construct candidates by decomposing residuals into upstream component outputs and projecting onto singular-vector directions of Ω. Recall (Appendix A) that for each token position j,

$$\mathbf{x}^j \;=\; \sum_{c\in\mathcal{C}(\ell)} \mathbf{o}_c^j. \tag{11}$$

Destination candidates (use U). Using the left singular vectors $\{\mathbf{u}^k\}_{k=1}^R$ of Ω (Eq. (3)), define the rank-1 projector

$$P_U^k \;=\; \mathbf{u}^k\mathbf{u}^{k\top}. \tag{12}$$

A destination-side candidate for component c and direction k is

$$\mathbf{s}_c^{dk} \;=\; P_U^k\mathbf{o}_c^d \in \mathbb{R}^D, \tag{13}$$

so that $\mathbf{x}^d = \sum_c\sum_k \mathbf{s}_c^{dk}$.

Note that we are saying that $\mathbf{x}^d = \sum_c\sum_k \mathbf{s}_c^{dk}$ for the sake of notation simplicity, but this is not strictly true. This is the projection of $\mathbf{x}^d$ into the attention head space, which is the only part of the residual that's used to compute attention.

Source candidates (use V). Using the right singular vectors $\{\mathbf{v}^k\}_{k=1}^R$ of Ω (Eq. (3)), define

$$P_V^k \;=\; \mathbf{v}^k\mathbf{v}^{k\top}. \tag{14}$$

For each source position $j \le d$, a source-side candidate is

$$\mathbf{s}_c^{jk} \;=\; P_V^k\mathbf{o}_c^j \in \mathbb{R}^D, \tag{15}$$

so that $\mathbf{x}^j = \sum_c\sum_k \mathbf{s}_c^{jk}$ for each $j \le d$.

Again, note that we are saying that $\mathbf{x}^j = \sum_c\sum_k \mathbf{s}_c^{jk}$ for the sake of notation simplicity, but this is not strictly true. This is the projection of $\mathbf{x}^s$ into the attention head space, which is the only part of the residual that's used to compute attention.

Intervention vectors. Given a set of candidate indices $M \subseteq \mathcal{C}(\ell) \times \{1, \ldots, R\}$, we define the residual-space vector removed at token position j as the sum of the corresponding candidate signal vectors. For the destination objective,

$$\Delta^d(M_{\text{dst}}) \;=\; \sum_{(c,k)\in M_{\text{dst}}} \mathbf{s}_c^{dk} \;=\; \sum_{(c,k)\in M_{\text{dst}}} P_U^k\, \mathbf{o}_c^d. \tag{16}$$

For the source objective, the same candidate set M_{src} is applied across all source positions $j \le d$:

$$\Delta^j(M_{\text{src}}) \;=\; \sum_{(c,k)\in M_{\text{src}}} \mathbf{s}_c^{jk} \;=\; \sum_{(c,k)\in M_{\text{src}}} P_V^k\, \mathbf{o}_c^j, \qquad \forall j \le d. \tag{17}$$

C.3. Contribution matrix (destination or source view)

Our solver operates on a contribution matrix whose rows correspond to candidates and whose columns correspond to source positions $j \le d$. Let $Q = |\mathcal{C}(\ell)| \cdot R$ be the total number of signal candidates, ie, the number of upstream components times the number of singular vectors R (rank of Ω).

Destination view. If we intervene on $\mathbf{x}^d$, substitute $\mathbf{x}^d = \sum_{c,k} \mathbf{s}_c^{dk}$ into $A'_{dj} = \mathbf{x}^{d\top}\Omega\mathbf{x}^j$ to obtain

$$A'_{dj} \;=\; \sum_c \sum_k \left(\mathbf{s}_c^{dk}\right)^\top \Omega \mathbf{x}^j. \tag{18}$$

We form a matrix $C \in \mathbb{R}^{Q\times d}$ with one row per pair (c,k) and define

$$C_{ij} \;=\; \left(\mathbf{s}_c^{dk}\right)^\top \Omega \mathbf{x}^j, \qquad j \le d, \tag{19}$$

where row i corresponds to candidate (c,k).

Source view. If we intervene on source residuals $\{\mathbf{x}^j\}_{j\le d}$, we use the source-side candidates $\mathbf{s}_c^{jk} = P_V^k \mathbf{o}_c^j$ (Eq. (15)) and, $\forall j \le d$, substitute $\mathbf{x}^j = \sum_c \sum_k \mathbf{s}_c^{jk}$ into $A'_{dj} = \mathbf{x}^{d\top}\Omega\mathbf{x}^j$ to obtain

$$A'_{dj} \;=\; \sum_c \sum_k \mathbf{x}^{d\top}\Omega\, \mathbf{s}_c^{jk}. \tag{20}$$

We form a contribution matrix $C \in \mathbb{R}^{Q\times d}$ with one row per pair (c,k), and define

$$C_{ij} \;=\; \mathbf{x}^{d\top}\Omega\, \mathbf{s}_c^{jk}, \qquad j \le d, \tag{21}$$

where row i corresponds to candidate (c,k). Equivalently, row i is the vector $\mathbf{c}_i \in \mathbb{R}^d$ whose j-th entry is the contribution of (c,k) to the score A'_{dj}. With this definition, removing a row $i = (c,k)$ corresponds to removing what component c writes in direction $\mathbf{v}^k$ *in every source token* $j \le d$.

Row-sum decomposition of scores. In either case, the candidate construction implies an exact decomposition of the full score vector:

$$A'_d \;=\; \sum_{i=1}^{Q} \mathbf{c}_i, \qquad A_d \;=\; \mathrm{Softmax}(A'_d). \tag{22}$$

Here A'_d denotes the normalized score vector $A'_d/\sqrt{R}$ from Eq. (2); we absorb the $1/\sqrt{R}$ factor into C.

C.4. Attribution via integrated gradients (probability at index s)

We seek to attribute the impact of each candidate on the single attention weight $A_{ds} = [A_d]_s$, rather than to the entire vector A_d. Let $C \in \mathbb{R}^{Q\times d}$ be the contribution matrix (destination or source view), let $\mathbf{c}_i \in \mathbb{R}^d$ denote row i, and note that

$$A'_d \;=\; \sum_{i=1}^{Q} \mathbf{c}_i. \tag{23}$$

Define the path

$$\mathbf{z}(t) \;=\; t\,A'_d, \qquad \mathbf{p}(t) \;=\; \mathrm{Softmax}(\mathbf{z}(t)), \qquad t \in [0,1], \tag{24}$$

so that $\mathbf{p}(1) = A_d$ and $A_{ds} = p_s(1)$. Let $g(\mathbf{z}) = [\mathrm{Softmax}(\mathbf{z})]_s$ denote the scalar output of interest. Integrated gradients (Sundararajan et al., 2017) assigns candidate i the attribution

$$\mathrm{IG}_i \;=\; \int_0^1 \langle \nabla g(\mathbf{z}(t)),\; \mathbf{c}_i \rangle\, dt. \tag{25}$$

Using the identity

$$\frac{\partial\, p_s}{\partial z_j} \;=\; p_s\,(\mathbf{1}\{j = s\} - p_j), \tag{26}$$

we obtain an explicit form that matches our implementation:

$$\mathrm{IG}_i \;=\; \int_0^1 p_s(t)\left(C_{is} \;-\; \sum_{j \le d} C_{ij}\, p_j(t)\right) dt \;=\; \int_0^1 p_s(t)\Big(C_{is} - \langle \mathbf{c}_i, \mathbf{p}(t)\rangle\Big)\, dt. \tag{27}$$

This formulation makes the role of Softmax competition explicit: the term $\langle \mathbf{c}_i, \mathbf{p}(t)\rangle$ couples candidate i to the full distribution over sources at each point along the path.

In practice we approximate the integral in Eq. (27) using a trapezoidal rule with $T = 64$ steps.

C.5. Greedy counterfactual solver and the induced intervention

We solve the destination objective (Eq. (7)) and the source objective (Eq. (10)) using the same greedy procedure, applied to the appropriate contribution matrix C. In both cases, rows of C are indexed by candidates (c, k) (so $C \in \mathbb{R}^{Q \times d}$), and we write $\mathbf{c}_i \in \mathbb{R}^d$ for row i.

Greedy removal in contribution space. We compute integrated-gradients attributions $\{\mathrm{IG}_i\}_{i=1}^{Q}$ *once*, using the original score vector $A'_d = \sum_{i=1}^{Q} \mathbf{c}_i$, and treat these attributions as fixed throughout the greedy procedure.Let $S \subseteq \{1, \ldots, Q\}$ denote the set of active candidates (initialized to all candidates), and define the partial score vector

$$A'_d(S) \;=\; \sum_{i \in S} \mathbf{c}_i, \qquad A_{ds}(S) \;=\; \Big[\mathrm{Softmax}\big(A'_d(S)\big)\Big]_s. \tag{28}$$

While $A_{ds}(S) > \tau$, remove the highest-attribution remaining candidate according to the *fixed* scores IG_i:

$$i^\star \;=\; \arg\max_{i \in S} \mathrm{IG}_i, \qquad S \leftarrow S \setminus \{i^\star\}. \tag{29}$$

Let $M = \{1, \ldots, Q\} \setminus S$ denote the removed set. We write M_{dst} or M_{src} depending on whether we are solving the destination or source objective.

Induced intervention for the destination objective. In the destination view, candidate i corresponds to a pair (c, k) and a destination-side signal vector $\mathbf{s}_c^{dk} = P_U^k \mathbf{o}_c^d$. Thus the greedy-selected set M_{dst} induces the residual-level intervention

$$\mathrm{do}_{\mathrm{dst}}(M_{\mathrm{dst}}): \qquad \tilde{\mathbf{x}}^d \;=\; \mathbf{x}^d - \sum_{(c,k) \in M_{\mathrm{dst}}} \mathbf{s}_c^{dk}, \qquad \tilde{\mathbf{x}}^j = \mathbf{x}^j \;\text{ for } j \le d,\; j \ne d. \tag{30}$$

Induced intervention for the source objective. In the source view, candidate i corresponds to a pair (c, k) and (for each source position $j \le d$) a source-side signal vector $\mathbf{s}_c^{jk} = P_V^k \mathbf{o}_c^j$. Removing candidate (c, k) means removing what component c writes in direction $\mathbf{v}^k$ *from every source token* $j \le d$. Accordingly, the greedy-selected set M_{src} induces

$$\mathrm{do}_{\mathrm{src}}(M_{\mathrm{src}}): \qquad \tilde{\mathbf{x}}^j \;=\; \mathbf{x}^j - \sum_{(c,k) \in M_{\mathrm{src}}} \mathbf{s}_c^{jk} \;\text{ for all } j \le d, \qquad \tilde{\mathbf{x}}^d = \mathbf{x}^d \text{ on the query side.} \tag{31}$$

Equivalently, the intervened scores for the source objective are computed as $A'_{dj}(\mathrm{do}_{\mathrm{src}}) = \mathbf{x}^{d\top} \Omega \tilde{\mathbf{x}}^j$, so that only the source-side (right) argument is modified.

C.6. ACC++ Recipe (high-level)

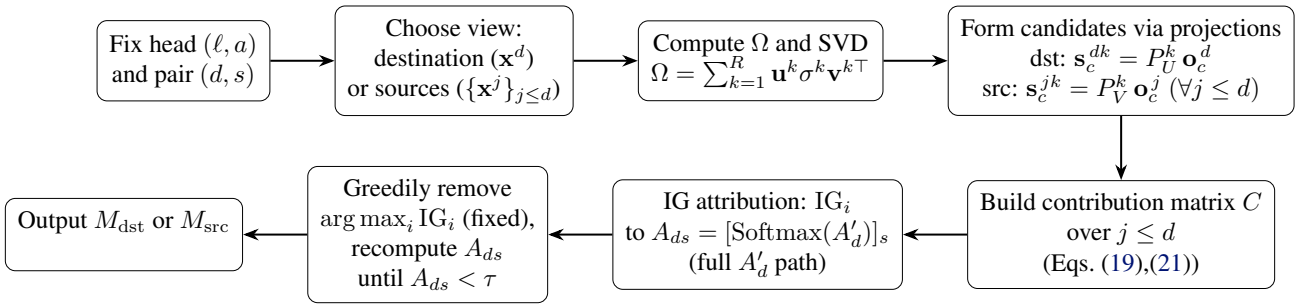


Figure 11. Recipe for the ACC++ solver, explicitly distinguishing destination candidates (P_U^k) and source candidates (P_V^k), with source-view candidates applied across all $j \le d$.

D. Finding τ

ACC requires a threshold τ that flags unusually large attention weights A_{ds} for a given destination token d. Because each attention row is a probability distribution over the available sources $j \leq d$, the typical scale of an entry depends on the row context size d. To make thresholds comparable across different d, models, and datasets, we study the scaled statistic

$$d \cdot A_{ds},$$

where the context size for row d is exactly the number of attendable tokens in that row, i.e. $|\{1, \ldots, d\}| = d$.

To understand appropriate values for τ, we proceed as follows. For each model–dataset setting, we run a forward pass over the evaluation set, collect all attention weights A_{ds} across layers/heads/tokens, and plot the empirical CDF (ECDF) of $d \cdot A_{ds}$. Across GPT-2 small (Fig. 12), Pythia-160M (Fig. 13), and Gemma-2 2B (Fig. 14) on our datasets, the ECDFs exhibit a consistent knee in the range $1.5 \leq d \cdot A_{ds} \leq 2.5$, separating the bulk of near-uniform attention from the heavy tail of unusually concentrated attention.

Motivated by this consistent knee, we use a *dynamic* threshold that adapts to the row context size:

$$\tau(d) = \frac{k}{d}.$$

where $1.5 \leq k \leq 2.5$.

We choose a dynamic global τ (depending on the context size) for simplicity. However, the impact of this strategy is unclear in other scenarios. We leave further study of better threshold-setting methods for ACC++ as future work. Lower values of k produce larger graphs, while higher values produce smaller graphs. In the experiments presented in § 3, we use $k = 2.5$ for the IOI task, and $k = 1.5$ for the mIOI task (see Figure 15 for ECDF for Gemma-2 2B and mGPT).

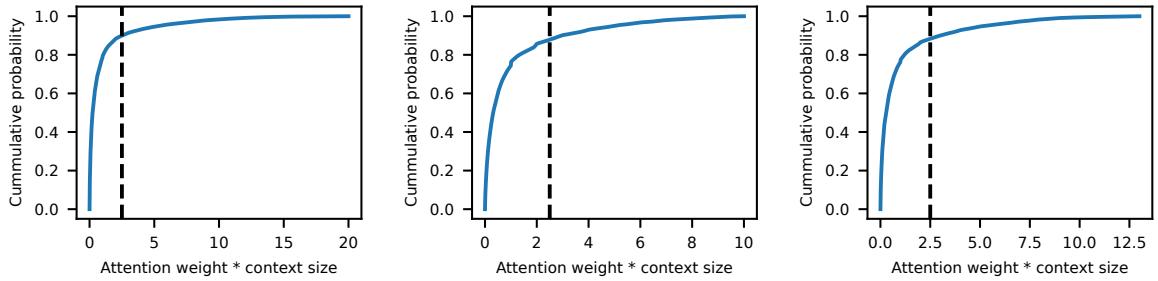


Figure 12. ECDF of the attention weight times the context size for GPT-2 small in IOI (a), GP (b), and GT (c). The vertical line shows a $\tau = 2.5$.

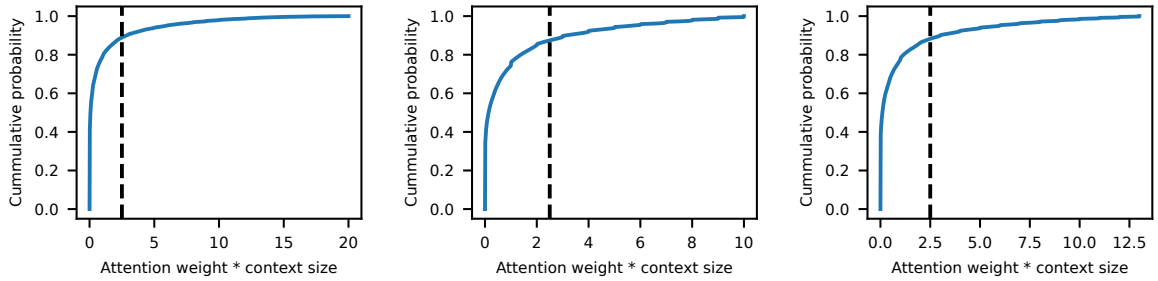


Figure 13. ECDF of the attention weight times the context size for Pythia-160M in IOI (a), GP (b), and GT (c). The vertical line shows a $\tau = 2.5$.

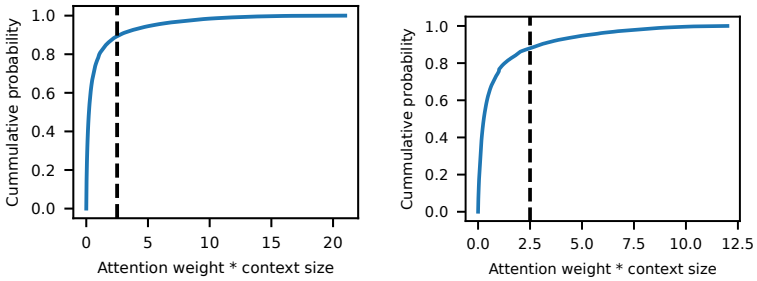


Figure 14. ECDF of the attention weight times the context size for Gemma-2 2B in IOI (a) and GP (b). The vertical line shows a $\tau = 2.5$.

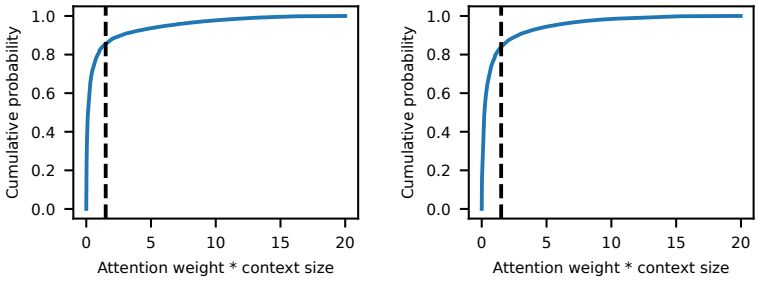


Figure 15. ECDF of the attention weight times the context size for mGPT (left) and Gemma-2 2B (right) on Multilingual IOI. The vertical line shows a $\tau = 1.5$.

E. Reproducing Prior ACC Results with ACC++

We repeat the circuit-tracing and intervention experiments of (Franco & Crovella, 2025), replacing the original ACC solver (Relative Attention; RA) with ACC++. Following (Franco & Crovella, 2025), we use GPT-2 Small, Pythia-160M, and Gemma-2 2B (Radford et al., 2019; Biderman et al., 2023; Team et al., 2024) to solve three tasks: Indirect Object Identification (IOI) (Wang et al., 2023), Greater-Than (GT) (Hanna et al., 2023b), and Gender Pronoun (GP) (Athwin et al.). In this appendix we follow the *exact* experimental setup of (Franco & Crovella, 2025), including the same datasets and evaluation protocol; this setup differs from the IOI dataset used in the main body of this paper. Across models and tasks, ACC++ identifies substantially more *low-rank* causal signals: when we measure a signal's "dimensionality" as the number of singular vectors of Ω used by the selected signal, the majority of signals are rank-1 (Figure 16). Figure 17 shows the corresponding distributions for ACC (RA) versus ACC++ across the three tasks.

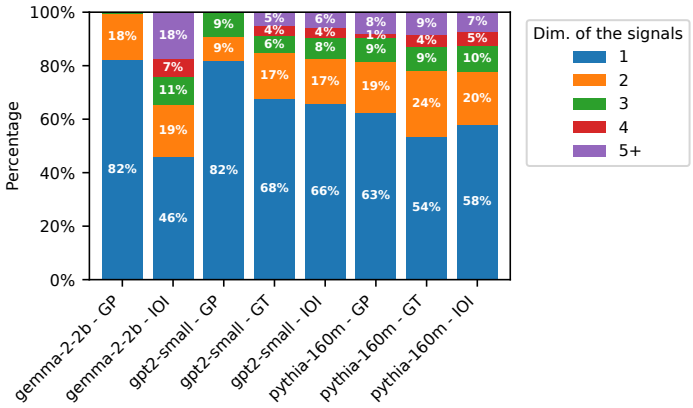
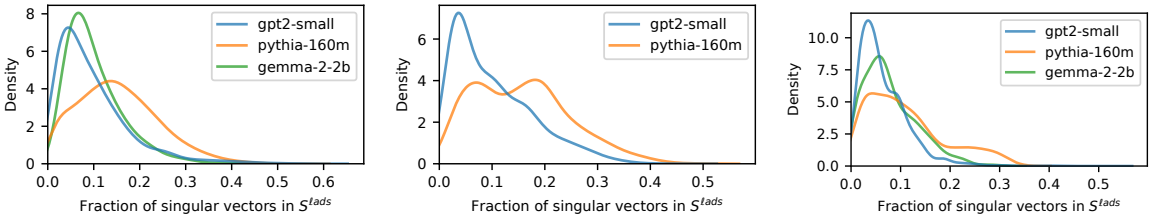
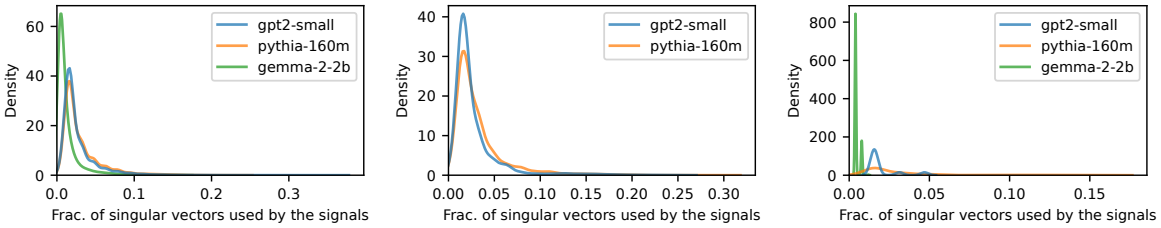


Figure 16. Across all tasks and models, most ACC++ signals are rank-1 (use one singular-vector direction).



(a) ACC. Source: (Franco & Crovella, 2025).

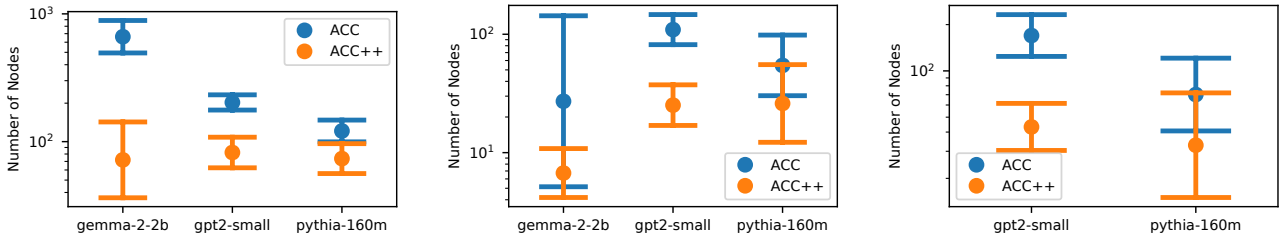


(b) ACC++.

Figure 17. ACC++ selects lower-dimensional signals than ACC across models and tasks, shifting mass toward rank-1 signals. From left to right: IOI, GT, GP.

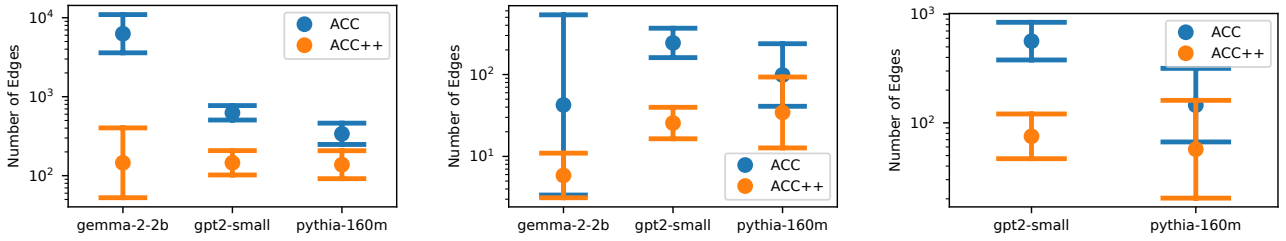
Despite this compression, ACC++ preserves downstream circuit results under the (Franco & Crovella, 2025) setup. Circuit quality remains comparable to the original ACC solver across tasks and models (Figure 23). At the same time, the traced circuits are more compact: ACC++ produces graphs with fewer nodes and fewer edges on average (Figures 18 and 19), and their in-degree distributions shift downward across IOI, GP, and GT, indicating that ACC++ typically establishes the required causal explanations with fewer incoming signals per node (Figures 20, 21, and 22).

Finally, these lower-rank signals remain causal for *model behavior* (not just attention weights): intervening on ACC++ signals yields substantial changes in IOI performance, whereas random-signal controls have much weaker effects (Figure 24). Consistent with the low-rank nature of these edits, interventions minimally perturb the residual stream (cosine similarity near 1 and norm ratios concentrated near 1), indicating that the behavioral effects are not driven by large out-of-distribution residual changes (Figure 25).



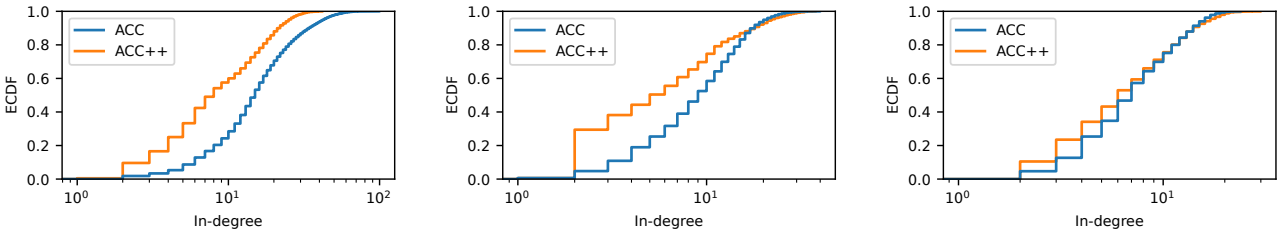
(a) IOI *(b)* GP *(c)* GT

Figure 18. ACC++ traces use fewer nodes than ACC across tasks (IOI, GP, GT), indicating more compact circuits under the same tracing setup. Error bars show standard deviation.



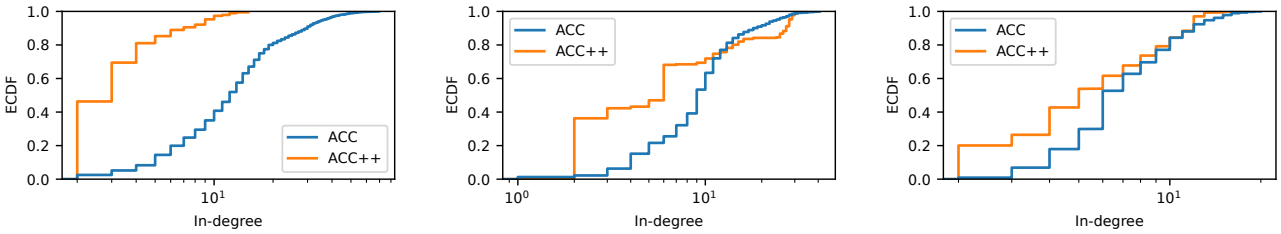
(a) IOI *(b)* GP *(c)* GT

Figure 19. ACC++ traces use fewer edges than ACC across tasks (IOI, GP, GT), reflecting a smaller set of causal communications needed to explain the same behaviors. Error bars show standard deviation.



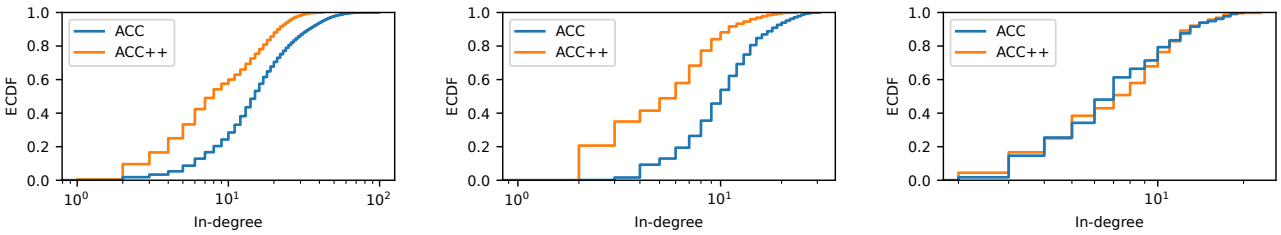
(a) Gemma-2 2B *(b)* GPT-2 Small *(c)* Pythia-160M

Figure 20. ACC++ requires fewer incoming causal signals per node than ACC on IOI (pooled in-degrees across traced circuits), indicating that ACC++ typically establishes attention causality with fewer signals.



(a) Gemma-2 2B *(b)* GPT-2 Small *(c)* Pythia-160M

Figure 21. ACC++ requires fewer incoming causal signals per node than ACC (RA) on GP (pooled in-degrees across traced circuits), indicating that ACC++ typically establishes attention causality with fewer signals.



(a) Gemma-2 2B *(b)* GPT-2 Small *(c)* Pythia-160M

Figure 22. ACC++ requires fewer incoming causal signals per node than ACC (RA) on GT (pooled in-degrees across traced circuits), indicating that ACC++ typically establishes attention causality with fewer signals.

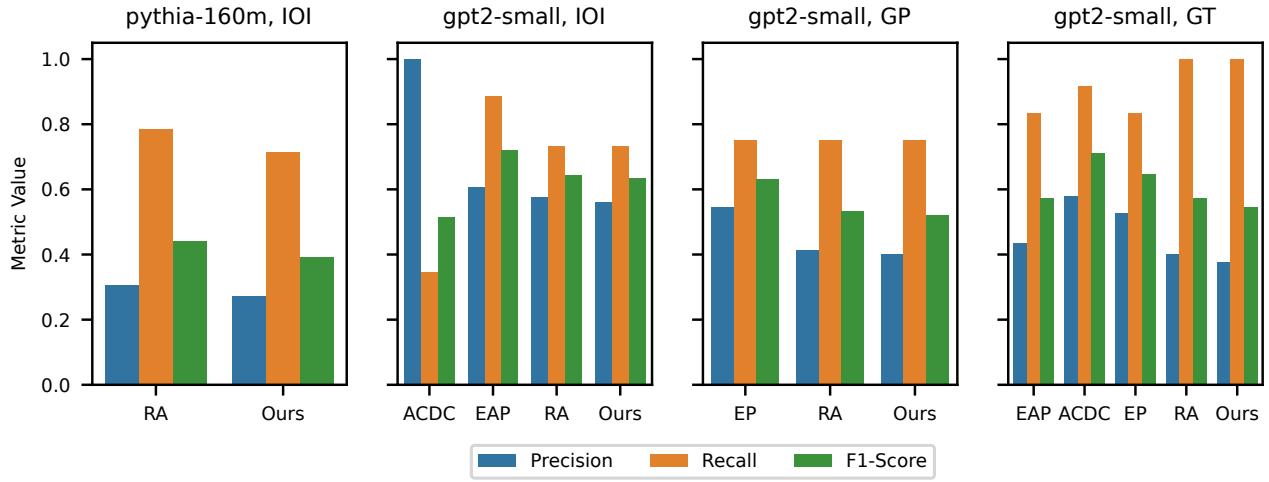
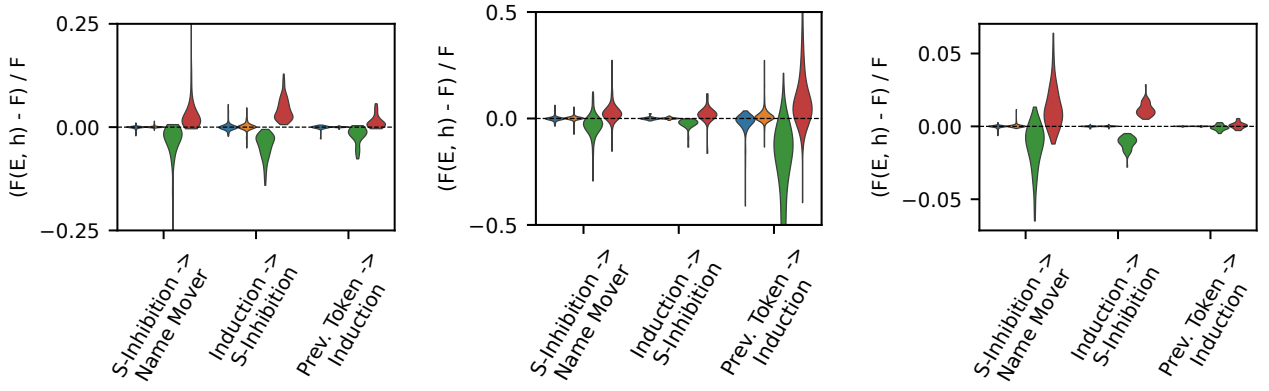
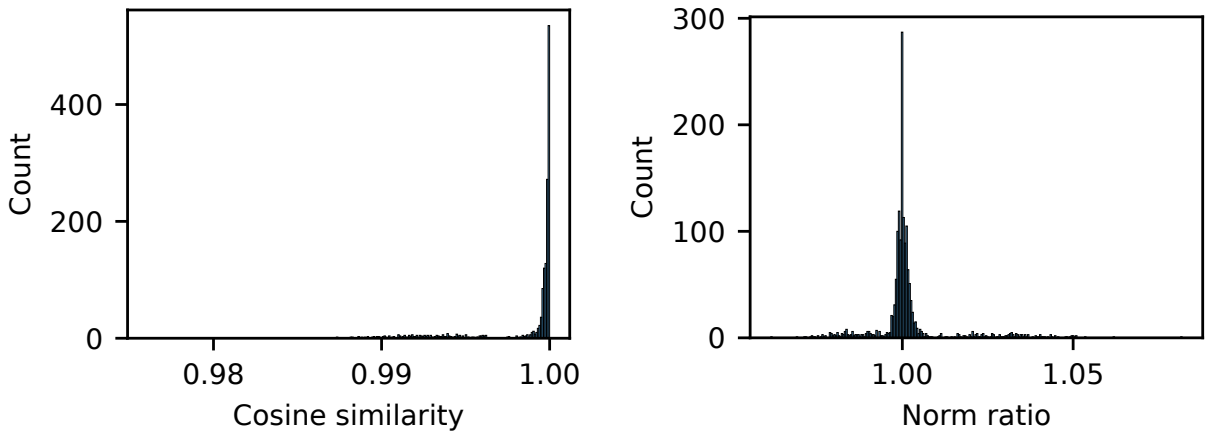


Figure 23. Circuit performance under the experimental setup of (Franco & Crovella, 2025): precision, recall, and F1 against canonical reference circuits. ACC++ (Ours) is shown alongside the original ACC solver (Relative Attention; RA) and the patching-based methods reported by (Franco & Crovella, 2025) (ACDC, EAP, EP) as additional reference points. ACC++ matches RA on this protocol; the agreement with the patching-based methods indicates that ACC++ recovers circuits structurally consistent with those baselines.



(a) GPT2-Small *(b)* Pythia-160M *(c)* Gemma-2 2B

Figure 24. Causal effect of ACC++ signal interventions on Indirect Object Identification (IOI) performance across models. Green/red bars show ACC++ signal ablation/boosting, while blue/orange bars show random signal ablation/boosting controls.



(a) Cosine similarity

(b) Norm ratio

Figure 25. Residual-stream perturbation induced by ACC++ interventions. We report cosine similarity and norm ratio between residual vectors before vs. after intervention. Despite minimal residual change (consistent with low-rank signals), interventions can have substantial causal effects on performance (cf. Figure 24).

F. Autointerpretation Pipeline Details

This appendix provides implementation details for the autointerpretation results in Section 2. We first explain how we convert each one-sided ACC++ signal (destination or source) into a *signal pair* by selecting the direction on the other side of the attention interaction that maximizes the pair's influence on the head's attention score (Appendix F.1). The reason of using signal pairs is that attention is a bilinear interaction between destination and source tokens, so a one-sided signal (destination-only or source-only) does not specify what it is matching against; pairing identifies the corresponding direction, allowing us to retrieve contexts and interpret the signal as a concrete destination token attending to a source token.

We then describe our fully automated pipeline, adapted from (Paulo et al., 2025), which (i) retrieves high-activation token-pair contexts for each signal from `pile-10k`, (ii) prompts an LLM to produce a short natural-language interpretation from the top contexts, and (iii) evaluates the interpretation with a standardized fuzzing-style procedure that compares top contexts to random controls and labels signals as interpretable using our statistical criterion (Appendix F.2).

F.1. Finding Signal Pairs

Let's consider models with bias such as GPT-2 to derive the signal interpretability. From Section B.1, we have that

$$\begin{aligned} A'_{ds} &= \left(\mathbf{x}^{d\top} + \mathbf{c}^{d\top}\right) \Omega \left(\mathbf{x}^{s} + \mathbf{c}^{s}\right) \\ &= \tilde{\mathbf{x}}^{d\top} \Omega \tilde{\mathbf{x}}^{s} \\ &= \sum_{k=1}^{R} \tilde{\mathbf{x}}^{d\top} \mathbf{u}^{k} \sigma^{k} \mathbf{v}^{k\top} \tilde{\mathbf{x}}^{s}, \end{aligned}$$

where $\mathbf{c}^{d\top} = \mathbf{b}_Q^\top W_Q^\dagger$ and $\mathbf{c}^s = W_K^{\dagger\top} \mathbf{b}_K$.

We now fix a subset of singular directions $I = \{i_1, \ldots, i_k\}$ and consider a *destination signal* $\mathbf{p}$ that lives in the corresponding left-singular subspace,

$$\mathbf{p} \in \mathrm{Span}\{\mathbf{u}^{i_1}, \ldots, \mathbf{u}^{i_k}\}, \qquad \|\mathbf{p}\| = 1.$$

Intuitively, we would like to find a *corresponding source signal* $\mathbf{q}$ (also unit norm) such that the pair $(\mathbf{p}, \mathbf{q})$ has maximal influence on the attention bilinear form A'_{ds}. For fixed $\tilde{\mathbf{x}}^d$, $\tilde{\mathbf{x}}^s$, and fixed $\mathbf{p}$, the dependence on $\mathbf{q}$ enters only through the inner term

$$\mathbf{p}^\top \Omega\, \mathbf{q},$$

up to multiplicative factors that do not depend on $\mathbf{q}$. Therefore, it suffices to solve the following optimization problem:

$$\mathbf{q}^* \;=\; \arg\max_{\|\mathbf{q}\|=1} \; \mathbf{p}^\top \Omega\, \mathbf{q}.$$

The next lemma shows that the optimal corresponding signal $\mathbf{q}$ is always the normalized vector $\Omega^\top \mathbf{p}$, and hence lies in the right-singular subspace aligned with the singular directions that span $\mathbf{p}$.

Lemma. Let $\Omega \in \mathbb{R}^{D\times D}$ have SVD

$$\Omega = \sum_{m=1}^{R} \sigma^m\, \mathbf{u}^m \mathbf{v}^{m\top},$$

and let $I = \{i_1, \ldots, i_k\}$ be a subset of indices. Define

$$U_I = [\,\mathbf{u}^{i_1} \;\ldots\; \mathbf{u}^{i_k}\,], \qquad V_I = [\,\mathbf{v}^{i_1} \;\ldots\; \mathbf{v}^{i_k}\,], \qquad \Sigma_I = \mathrm{diag}(\sigma^{i_1}, \ldots, \sigma^{i_k}).$$

Let $\mathbf{p} \in \mathrm{Span}\{\mathbf{u}^{i_1}, \ldots, \mathbf{u}^{i_k}\}$ with $\|\mathbf{p}\| = 1$. Consider

$$\mathbf{q}^* \;=\; \arg\max_{\|\mathbf{q}\|=1} \; \mathbf{p}^\top \Omega\, \mathbf{q}.$$

Then, if $\Omega^\top \mathbf{p} \neq \mathbf{0}$,

$$\mathbf{q}^* \;=\; \frac{\Omega^\top \mathbf{p}}{\|\Omega^\top \mathbf{p}\|},$$

and in particular $\mathbf{q}^* \in \mathrm{Span}\{\mathbf{v}^{i_1}, \ldots, \mathbf{v}^{i_k}\}$. If $\Omega^\top \mathbf{p} = \mathbf{0}$, then $\mathbf{p}^\top \Omega \mathbf{q} = 0$ for all $\|\mathbf{q}\| = 1$ and every unit $\mathbf{q}$ is optimal.

Proof. Since $\mathbf{p} \in \mathrm{Span}\{\mathbf{u}^{i_1}, \ldots, \mathbf{u}^{i_k}\}$ and $\|\mathbf{p}\| = 1$, there exists $\boldsymbol{\alpha} \in \mathbb{R}^k$ with

$$\mathbf{p} = U_I \boldsymbol{\alpha}, \qquad \|\boldsymbol{\alpha}\| = 1,$$

because the columns of U_I are orthonormal.

We want to solve

$$\mathbf{q}^* \;=\; \arg\max_{\|\mathbf{q}\|=1} \; \mathbf{p}^\top \Omega\, \mathbf{q}.$$

First, note that we can, without loss of generality, restrict $\mathbf{q}$ to the span of the right singular vectors $\{\mathbf{v}^{i_1}, \ldots, \mathbf{v}^{i_k}\}$. Indeed, decompose any $\mathbf{q}$ as

$$\mathbf{q} = V_I \boldsymbol{\beta} + \mathbf{w},$$

where $\boldsymbol{\beta} \in \mathbb{R}^k$ and $\mathbf{w} \perp \mathrm{Span}\{\mathbf{v}^{i_1}, \ldots, \mathbf{v}^{i_k}\}$. Using the SVD,

$$\mathbf{p}^\top \Omega \mathbf{q} = \sum_{m=1}^{R} \sigma^m (\mathbf{u}^{m\top} \mathbf{p})(\mathbf{v}^{m\top} \mathbf{q}).$$

Since $\mathbf{p} \in \mathrm{Span}\{\mathbf{u}^{i_1}, \ldots, \mathbf{u}^{i_k}\}$, we have $\mathbf{u}^{m\top}\mathbf{p} = 0$ for all $m \notin I$, so

$$\mathbf{p}^\top \Omega \mathbf{q} = \sum_{m \in I} \sigma^m (\mathbf{u}^{m\top} \mathbf{p})(\mathbf{v}^{m\top} \mathbf{q}) = \sum_{m \in I} \sigma^m (\mathbf{u}^{m\top} \mathbf{p})(\mathbf{v}^{m\top} V_I \boldsymbol{\beta}),$$

because $\mathbf{v}^{m\top}\mathbf{w} = 0$ for all $m \in I$. Thus the component $\mathbf{w}$ does not affect the objective, but does contribute to the norm $\|\mathbf{q}\|^2 = \|V_I \boldsymbol{\beta}\|^2 + \|\mathbf{w}\|^2$. Under the constraint $\|\mathbf{q}\| = 1$, any nonzero $\mathbf{w}$ would reduce the available norm for $V_I \boldsymbol{\beta}$ without changing $\mathbf{p}^\top \Omega \mathbf{q}$, and therefore cannot be optimal. Hence we may assume $\mathbf{w} = 0$ and restrict to

$$\mathbf{q} = V_I \boldsymbol{\beta}, \qquad \|\boldsymbol{\beta}\| = 1.$$

Because V_I has orthonormal columns, we have

$$\|\mathbf{q}\|^2 = \|V_I \boldsymbol{\beta}\|^2 = \boldsymbol{\beta}^\top V_I^\top V_I \boldsymbol{\beta} = \|\boldsymbol{\beta}\|^2,$$

so the constraint $\|\mathbf{q}\| = 1$ is equivalent to $\|\boldsymbol{\beta}\| = 1$.

Now rewrite the objective in the reduced coordinates:

$$\mathbf{p}^\top \Omega \mathbf{q} = \boldsymbol{\alpha}^\top U_I^\top \Omega V_I \boldsymbol{\beta}.$$

But $U_I^\top \Omega V_I = \Sigma_I$ (by the definition of the SVD block on indices I), so

$$\mathbf{p}^\top \Omega \mathbf{q} = \boldsymbol{\alpha}^\top \Sigma_I \boldsymbol{\beta}.$$

Thus the problem reduces to

$$\boldsymbol{\beta}^* \;=\; \arg\max_{\|\boldsymbol{\beta}\|=1} \; \boldsymbol{\alpha}^\top \Sigma_I \boldsymbol{\beta}.$$

Define $\boldsymbol{\gamma} := \Sigma_I \boldsymbol{\alpha} \in \mathbb{R}^k$. Then

$$\boldsymbol{\alpha}^\top \Sigma_I \boldsymbol{\beta} = \boldsymbol{\gamma}^\top \boldsymbol{\beta}.$$

If $\boldsymbol{\gamma} = \mathbf{0}$, then $\mathbf{p}^\top \Omega \mathbf{q} = \boldsymbol{\gamma}^\top \boldsymbol{\beta} = 0$ for all $\boldsymbol{\beta}$ with $\|\boldsymbol{\beta}\| = 1$, so $\mathbf{p}^\top \Omega \mathbf{q} = 0$ for all unit $\mathbf{q}$ and every unit $\mathbf{q}$ is optimal. This is exactly the degenerate case $\Omega^\top \mathbf{p} = \mathbf{0}$.

Assume now $\boldsymbol{\gamma} \neq \mathbf{0}$. By the Cauchy–Schwarz inequality,

$$\boldsymbol{\gamma}^\top \boldsymbol{\beta} \;\leq\; \|\boldsymbol{\gamma}\|\, \|\boldsymbol{\beta}\| = \|\boldsymbol{\gamma}\|, \qquad \|\boldsymbol{\beta}\| = 1.$$

Equality holds if and only if $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ are linearly dependent. Thus to maximize $\boldsymbol{\gamma}^\top \boldsymbol{\beta}$ under $\|\boldsymbol{\beta}\| = 1$, we must have

$$\boldsymbol{\beta}^* = \lambda\, \boldsymbol{\gamma}$$

for some scalar λ. Enforcing $\|\boldsymbol{\beta}^*\| = 1$ gives

$$1 = \|\boldsymbol{\beta}^*\| = |\lambda|\,\|\boldsymbol{\gamma}\| \quad \Rightarrow \quad |\lambda| = \frac{1}{\|\boldsymbol{\gamma}\|}.$$

To maximize (rather than minimize) $\boldsymbol{\gamma}^\top \boldsymbol{\beta}$, we take the positive sign, so

$$\boldsymbol{\beta}^* = \frac{\boldsymbol{\gamma}}{\|\boldsymbol{\gamma}\|}.$$

Mapping back to the original space,

$$\mathbf{q}^* = V_I \boldsymbol{\beta}^* = V_I \frac{\boldsymbol{\gamma}}{\|\boldsymbol{\gamma}\|} = \frac{V_I \boldsymbol{\gamma}}{\|\boldsymbol{\gamma}\|}.$$

We now express $\Omega^\top \mathbf{p}$ in terms of this basis:

$$\Omega^\top \mathbf{p} = \sum_{m=1}^{R} \sigma^m \mathbf{v}^m \mathbf{u}^{m\top} \mathbf{p} = \sum_{m \in I} \sigma^m \mathbf{v}^m (\mathbf{u}^{m\top} \mathbf{p}) = V_I(\Sigma_I \boldsymbol{\alpha}) = V_I \boldsymbol{\gamma}.$$

Therefore

$$\|\Omega^\top \mathbf{p}\| = \|V_I \boldsymbol{\gamma}\| = \|\boldsymbol{\gamma}\|$$

(because V_I has orthonormal columns), and we obtain

$$\mathbf{q}^* = \frac{V_I \boldsymbol{\gamma}}{\|\boldsymbol{\gamma}\|} = \frac{\Omega^\top \mathbf{p}}{\|\Omega^\top \mathbf{p}\|}.$$

This proves that, in the non-degenerate case, the unique maximizer is

$$\mathbf{q}^* = \frac{\Omega^\top \mathbf{p}}{\|\Omega^\top \mathbf{p}\|},$$

and it lies in $\mathrm{Span}\{\mathbf{v}^{i_1}, \ldots, \mathbf{v}^{i_k}\}$, since $\Omega^\top \mathbf{p} = V_I \boldsymbol{\gamma}$.

□

Corollary (source-to-destination correspondence). An entirely analogous result holds if we fix a *source* signal and optimize over the destination. Let $\mathbf{q} \in \mathrm{Span}\{\mathbf{v}^{i_1}, \ldots, \mathbf{v}^{i_k}\}$ with $\|\mathbf{q}\| = 1$, and consider

$$\mathbf{p}^* \;=\; \arg\max_{\|\mathbf{p}\|=1} \; \mathbf{p}^\top \Omega\, \mathbf{q}.$$

Writing $\mathbf{p}^\top \Omega\, \mathbf{q} = \langle \mathbf{p},\, \Omega \mathbf{q} \rangle$ and applying the Cauchy–Schwarz inequality exactly as in the lemma, we obtain

$$\mathbf{p}^* \;=\; \frac{\Omega\, \mathbf{q}}{\|\Omega\, \mathbf{q}\|},$$

whenever $\Omega\, \mathbf{q} \neq \mathbf{0}$. Moreover, since $\mathbf{q}$ has support only on $\{\mathbf{v}^{i_1}, \ldots, \mathbf{v}^{i_k}\}$, we have $\Omega\, \mathbf{q} \in \mathrm{Span}\{\mathbf{u}^{i_1}, \ldots, \mathbf{u}^{i_k}\}$, so the optimal destination signal $\mathbf{p}^*$ lies in the corresponding left-singular subspace.

Extending this methodology to RoPE models. This pairing methodology continues to hold for RoPE models (with or without bias) because it is fundamentally a statement about maximizing a bilinear form, and RoPE only applies linear maps to the vectors that enter that bilinear form. In the non-RoPE case, the attention score has the form $\tilde{\mathbf{x}}^{d\top} \Omega \tilde{\mathbf{x}}^s$, and the pairing step asks: for a fixed destination direction $\mathbf{p}$, which unit-norm source direction $\mathbf{q}$ maximizes $\mathbf{p}^\top \Omega \mathbf{q}$? The solution is $\mathbf{q} \propto \Omega^\top \mathbf{p}$, i.e. $\mathbf{q} = \Omega^\top \mathbf{p} / \|\Omega^\top \mathbf{p}\|$. With RoPE (and bias), the score can be rewritten as $((\mathbf{x}^{d\top} + \mathbf{c}^{d\top}) M_d)\, \Omega\, (M_s(\mathbf{x}^s + \mathbf{c}^s))$, where M_d and M_s are the position-dependent linear maps induced by pushing the RoPE rotations through W_Q and W_K. Defining the effective vectors $\tilde{\mathbf{x}}^{d\top} := (\mathbf{x}^{d\top} + \mathbf{c}^{d\top}) M_d$ and $\tilde{\mathbf{x}}^s := M_s(\mathbf{x}^s + \mathbf{c}^s)$ yields the same bilinear structure $\tilde{\mathbf{x}}^{d\top} \Omega \tilde{\mathbf{x}}^s$. Therefore, the maximization argument applies verbatim in the effective (RoPE-adjusted) space: for a fixed effective destination signal direction $\mathbf{p}$, the corresponding paired effective source direction is still given by $\mathbf{q} = \Omega^\top \mathbf{p} / \|\Omega^\top \mathbf{p}\|$ (and symmetrically $\mathbf{p} = \Omega \mathbf{q} / \|\Omega \mathbf{q}\|$). In other words, RoPE changes how residual components are mapped into the space where Ω acts, but it does not change the algebraic form of the pairing rule once those linear maps are accounted for.

F.2. Automated Interpretation Pipeline Description

We adopt an automated interpretability pipeline modeled after the methodology in (Paulo et al., 2025), and adapted to the setting of ACC++ signals. In our setting, signals are paired with a destination and source signals patterns within individual attention heads, instead of individual neurons or sparse autoencoder features. The pipeline consists of three stages: identification of high-activation signal examples, generation of a natural-language interpretation for each signal, and quantitative evaluation of interpretation quality. All stages of interpretation and evaluation are fully automated. No human annotations, labeling, or judgments are used at any point in the pipeline.

Top-Activation Discovery (context collection) All experiments are conducted using three pretrained transformer language models: GPT-2 Small, Pythia-160M, and Gemma-2 2B. We use the `NeelNanda/pile-10k` split of The Pile, consisting of the first 10,000 sequences of tokenized natural-language text (Gao et al., 2020). Signals are not explicit annotations in the dataset; they are ACC++-discovered activation patterns tied to a specific attention head and a directed destination–source interaction (Appendix C). Throughout this appendix, the object we interpret is a *signal pair* (Appendix F.1).

To find contexts where a given signal pair is active, we cache residual-stream activations over `pile-10k` in 32-token chunks. Fix a normalized signal pair $(\mathbf{p}, \mathbf{q})$ for a head (ℓ, a), and let $\mathbf{x}^d, \mathbf{x}^s$ denote the residual-stream inputs to layer ℓ at destination token d and source token s. Recall that the (unnormalized) attention logit is a bilinear form $A'_{ds} = \mathbf{x}^{d\top} \Omega \, \mathbf{x}^s$ (up to scaling), where $\Omega = W_Q W_K^\top$. We measure how strongly the attention interaction is driven *along the paired directions* by scoring token pairs via

$$\mathrm{score}(d, s; \mathbf{p}, \mathbf{q}) \;=\; \mathbf{x}^{d\top} \, \mathbf{p} \, \mathbf{q}^\top \, \mathbf{x}^s,$$

which is a proxy for the contribution of the paired signals to the head's attention logit for (d, s).

Construction of Signal Examples Each selected context is rendered as a 32-token text span that contains both the destination and source tokens. Within each span, the destination token is marked using `<< >>`, and the source token is marked using `[[ ]]`; if the same token serves as both destination and source, both markers are applied (`<<[[token]]>>`).

These examples are used in two downstream stages.

1. **Interpretation Stage:** we prompt an LLM with the top-40 examples for a signal to generate a single short interpretation.
2. **Evaluation Stage:** to score an interpretation, we take the top-20 examples (highest-scoring among the top-40) and mix them with 20 random control examples *sampled independently for that same signal*, then ask an LLM judge to decide whether the interpretation explains each example (Paulo et al., 2025).

Interpretation stage Given the top-40 activating examples for a signal, we generate a single natural-language interpretation that summarizes the shared pattern associated with the marked destination–source token pairs. We use `neuralmagic/DeepSeek-R1-Distill-Llama-70B-FP8-dynamic` as the interpreter model, run via vLLM (DeepSeek-AI, 2025). The interpreter is shown only the top-activation examples and is instructed to produce one concise, high-level description, explicitly taking into account both token identity and token position within the 32-token span. The full interpretation prompt is included verbatim below and is used without modification

System Prompt for the Signal Interpretation

```
You are a meticulous AI researcher conducting an important investigation into how
    hidden signals in language models correspond to patterns in text. Your task is to
    analyze the given text snippets and provide an interpretation that clearly
    summarizes the linguistic or semantic pattern they reveal.

Guidelines:
- You will be shown several short text examples.
- In each example, one or two important tokens will be surrounded by << >> and [[ ]] (
    e.g. "the [[cat]] sat on the << mat>>").
- These brackets indicate where the model's internal signal was most active. Signals
    are found in the context of attention heads, meaning that the token highlighted
    with << >> acts as a destination token, while the token highlighted with [[ ]] acts
     as a source token. If only one token is highlighted with both << >> and [[ ]], it
    means that this token acts as both destination and source tokens.
```

```
- Each example may include additional context words around the highlighted token(s).
- Your goal is to produce a concise, high-level description of what these highlighted
    tokens have in common (their meaning, grammatical role, or thematic pattern
    possibility).
- Focus on the underlying pattern, not on quoting or restating the examples.
- Remember to look at the position of the tokens in the text chunk. This also matters.
- Ignore the brackets and any tokenization artifacts when describing your
    interpretation.
- If the examples appear random or uninformative, say so briefly.
- Do not list multiple hypotheses; choose the single best interpretation.
- Keep your answer short and clear.
- If you are not more than 90\% certain about an interpretation, say "no valid
    interpretation found".
- The final line of your response must provide your conclusion in this exact format:
[interpretation]: your concise description here
```

The interpreter produces exactly one interpretation per signal, or returns `no valid interpretation found` when the top examples appear incoherent or uninformative. Signals with no returned interpretation are flagged and excluded from downstream scoring.

Gemini for qualitative, manual-case studies. For a small number of representative circuits used in our qualitative analysis—most notably the ABBA/BABA comparison in Figure 5 of the main paper and the additional traced examples in Appendix F.3—we instead used Gemini 3 Flash to generate natural-language interpretations for the signals. We chose Gemini for these case studies because we wanted the strongest possible interpretations for a handful of prompts that we analyze manually, and this setting does not require running the full fuzzing score pipeline at scale. We did not use Gemini for the full automated pipeline primarily due to its high cost.

Interpretation Scoring Setup We evaluate each proposed interpretation using a separate judge model, Gemma-3-27B-IT, also run via vLLM (Team et al., 2025). The judge is distinct from the interpreter and is used only to assess whether the interpretation reliably separates top-activation contexts from random controls. For each signal with an associated interpretation, we score 40 mixed examples: 20 top-activation examples (the highest-scoring subset of the top-40) and 20 random examples sampled for that signal. We evaluate the 40 examples in batches of 10, with each batch containing exactly 5 top-activation and 5 random examples; batches are scored independently and concatenated to yield 40 binary judgments per signal.

Each scoring prompt contains the candidate interpretation and 10 mixed examples with explicit source/destination markings. The judge returns a binary label for each example, where an example is accepted only if all marked tokens in that example are consistent with the interpretation. We use this criterion to avoid ambiguity from partial matches and to ensure that high scores reflect coherent signal-level regularities rather than occasional coincidental matches. The full scoring prompt is included verbatim below and is used without modification.

Fuzzing Scoring Prompt

```
You are an intelligent and meticulous linguistics researcher.

You will be given a specific linguistic feature of interest, such as  "male pronouns,"
     "negative sentiment," or "surname tokens."

You will then be given several text examples that are claimed to contain this feature.
     Portions of the text that supposedly represent the feature have been marked using
    << >> and [[ ]].
These brackets indicate where the model's internal signal was most active.
Features are found in the context of attention heads, meaning that the token
    highlighted with << >> acts as a destination token, while the token highlighted
    with [[ ]] acts as a source token. If only one token is highlighted with both << >>
     and [[ ]], it means that this token acts as both destination and source tokens.

Your task is to determine whether EVERY token inside each << >> and [[ ]] span is
```

```
    correctly labeled as an instance of the feature.

Important:
- An example is correct ONLY if every marked tokens are representative of the feature.
- There are exactly 10 examples below.

For each example in turn:
- Return 1 if ALL marked spans correctly represent the feature.
- Return 0 if ANY marked token is mislabeled.

Important Output Format Rules:
1. Return the results as a Python Dictionary.
2. The keys must be the example numbers (1 to 10), and the values must be the binary
    label (0 or 1).
3. Do not assume the order; explicitly check the number I assigned to each example.
4. Ignore any numbers or formatting artifacts INSIDE the text strings (e.g., if a text
     contains "4).", ignore it).
5. Output format:
   {
    1: 0,
    2: 1,
    ...
    10: 0
   }

Here are the examples:

<user_prompt>
Feature interpretation: Words related to American football positions, specifically the
     tight end position.
Text examples:
1. Getty Images [[ Patriots]]<< tight>> end Rob Gronkowski had his boss
2. posted You should know this[[ about]] offensive line coaches: they are large,
    demanding<< men>>
3. Media Day 2015 LSU [[ defensive]] end Isaiah Washington (94) speaks to<< the>>
4. running [[ backs]],'' he said. .. Defensive << end>> Carroll Phillips is improving
    and his injury is
5. [[ line]], with the left side namely << tackle>> Byron Bell at tackle and guard
    Amini
<assistant_response>
{
1: 1,
2: 0,
3: 0,
4: 1,
5: 1,
}

Now evaluate the following examples:
```

Signals for which the interpreter returns `no valid interpretation found` are treated as missing interpretations and are excluded from scoring-based analyses. Across models, the fraction of missing interpretations is small (GPT-2 Small 0.850%, Pythia-160M 0.359%, Gemma-2 2B 0.576%). We run the full pipeline over 22223 (Gemma-2 2B), 15316 (Pythia-160M), and 13406 (GPT-2 Small) signals.

Example top-activation contexts. Before reporting aggregate fuzzing metrics, we show a few retrieved top-activation contexts for a representative signal. As above, `<< >>` marks the destination token and `[[ ]]` marks the source token.

Prompt: "Then, Jack and Kelly went to the garden. Jack gave a basketball to" (BABA representative).
Signal: AH(8,6) (to (1), Jack (1)) → AH(9,9) (to (1), Kelly).
Interpretation: the second element in a parallel pair or sequence.
Explanation: The highlighted tokens consistently identify the second element in a parallel pair, list, or sequence (such

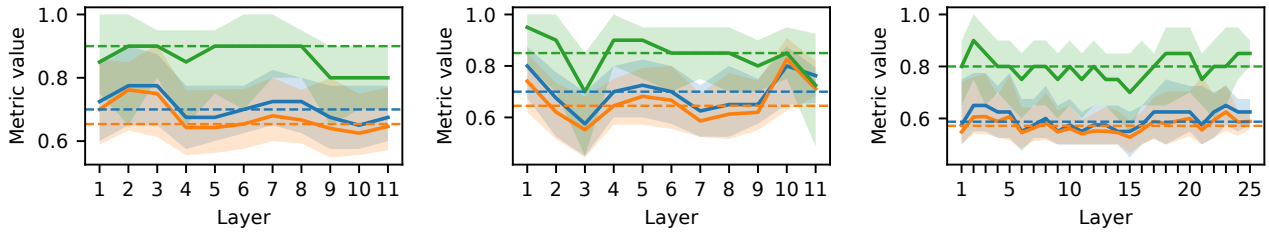
as “y” following “x”, “b” following “a”, or an item labeled “2” following one labeled “1”). The signal typically links this second element either to its own identifier or to the corresponding first element in the same or a preceding parallel structure.

Ten top-activation contexts (Interpretation: second element in a parallel pair or sequence)

```
- boost::arg<I> (*) () ): storage3<A1, A[[2]], A3>( a1,<< a>>2, a3 ) {}
- the function that initiates Ball(float x, float y, int Width, color Color) { ballX=
    x; ball[[Y]]<<=>> y; ballWidth
- speedX; ballY+=speedY; } void move(int X, int Y) { ballX=X; ball[[Y]]<<=>> Y; speedY
- ). \nfunction cartesianJoin(arr1, arr[[2]], col1,<< col>>2) {\n  var output = [];\n
     for (var
- that initiates Block(float x, float y, float Width, float Height, color Color) {
    blockX= x; block[[Y]]<<=>> y; block
- $, where $C_1$ and $C_2$ are cycles of lengths $d_1$ and $d<<[[_]]>>2$ respectively,
     $d
- abnormal Ser phosphorylation of IRS-1 and IRS-2, along with degradation of IRS-1 and
     IRS<<[[-]]>>2. Moreover, these alterations were atten
- the function that initiates Ball(float x, float[[ y]], int Width, color Color) {
    ballX= x; ballY<<=>> y; ballWidth
- $\\{1,2,\\cdots,\n2m\\}$ into two element blocks $\\{(\\pi_1,\\pi<<[[_]]>>2),
- m)}(x_1,\\dots,x_m;\\tilde{k}_1,\\tilde{k}<<[[_]]>>2
```

Quantitative evaluation of interpretation quality For each signal, the judge returns a binary decision for each of the 40 scored examples (20 top-activation, 20 random). We treat this as a binary classification problem, comparing judge-predicted labels against the known top-versus-random labels. We report standard per-signal metrics (accuracy, precision, and recall) and summarize them across signals; the main text reports medians and interquartile ranges (Table 2). Additionally, Figure 26 breaks down the median metrics by layer, alongside a median line across all layers.

Along with these classification metrics, we compute a one-sided p-value using Fisher’s exact test (Fisher, 1922) to measure whether the judge accepts the interpretation more often on top-activation contexts than on random contexts. We then control false discoveries across signals using False Discovery Rate (FDR) (Benjamini & Hochberg, 1995). We report the fraction of signals with FDR $\leq 5\%$ (computed within each layer); signals passing this criterion are labeled *interpretable*. This notion of interpretability is meant to distinguish signal from noise under the fuzzing protocol, rather than to fully quantify the intrinsic quality of the interpretation. In the main text, we summarize this criterion as the fraction of interpretable signals per model.



(a) GPT2-Small *(b)* Pythia-160M *(c)* Gemma-2 2B

Figure 26. Layerwise automated-interpretability metrics under the fuzzing protocol for three models. Solid lines show the median per layer for accuracy (blue), precision (orange), and recall (green); shaded bands show the interquartile range (25%–75%) across signals within each layer. Dashed horizontal lines denote the corresponding median metric computed across all layers (same color as the metric).

F.3. Interactive Circuit Visualizer

For all qualitative trace figures (Figures 5, 38, 39, 44, and 45), we additionally provide an interactive HTML visualization in the supplementary material. The viewer takes a traced GraphML file produced by ACC++ together with the autointerpretation outputs from § 2 and renders the circuit as a directed graph in which nodes correspond to model components (embeddings, attention heads, MLPs, and the logit direction) and edges correspond to signal flow between them. Users can prune weak edges based on a threshold, hide nodes that do not connect to the output, and click on any node or edge to see its quantitative attributes (raw weight, attention weight, and the singular vectors that define the signal) together with the natural-language

interpretation and a panel of top-activating contexts that ground that interpretation in real text. This tool allows users to inspect the circuit structure and its interpretation in an easy and intuitive way. The interface applies to any prompt traced with ACC++ and autointerpreted; the example panels below are drawn from a single Gemma-2 2B trace of `` `Fact: The capital of the state containing Dallas is'' `` (predicting `` `Austin'' ``), but the viewer itself is not specific to that prompt.

The static qualitative figures in the paper are constructed by analyzing these interactive visualizations and then identifying the most salient signals and components; this analysis step was assisted by Claude, which was given the traced graph together with the autointerpreted edge labels and used to produce a reduced, readable summary of the most salient paths. The interactive versions are provided to make it easier to audit and explore the underlying traced circuits.

Figures 27–31 show representative panels from the viewer, illustrating the range of signals it surfaces: token-level lexical signals (Figures 27 and 28, two singular-vector decompositions of the "Dallas" input embedding), a relational signal capturing inclusion patterns such as "X containing Y" (Figure 29), a higher-level association linking time and location across tokens (Figure 30), and a structured-completion signal that fires on predictable phrase patterns (Figure 31). Each panel pairs the autointerpretation of a single edge with its top-40 activating contexts.

strongest incoming raw: Embedding 0 Dallas (0.582061)

strongest incoming normalized: Embedding 0 Dallas (0.662414)

strongest outgoing raw: AH(5,4) (Dallas, Dallas) (0.030395)

strongest outgoing normalized: AH(5,4) (Dallas, Dallas) (0.031025)

Full Interpretation

Embedding 0 Dallas → AH(3,2) (Dallas, Dallas)

raw 0.582061 | attention 0.878697 | normalized 0.662414

The provided text examples consistently highlight the names of major cities that begin with the letter "D," most frequently "Dallas," but also "Denver" and "Detroit." These city names appear in various grammatical roles—as objects of prepositions, proper adjectives, or locations in legal citations—and the signal is active regardless of whether the name refers to the city itself, a county, a sports team, or even a person sharing the name. Other cities mentioned in the text that do not start with "D" (such as Selma, Frisco, or Cedar Park) are not highlighted, indicating that the pattern is specific to major city names starting with this letter.

[interpretation]: names of major cities starting with the letter D

Incoming edges (7 of 7 remaining)

Embedding 0 Dallas → AH(3,2) (Dallas, Dallas)

type d | raw 0.582061 | attention 0.878697 | normalized 0.662414

Singular vectors used: [82, 70, 12, 35, 136, 17, 76, 55, 62, 125, 95, 28]

interpretation:
names of major cities starting with the letter D

Open full interpretation below graph

Open top-40 activations below graph

Top-40 Activations

Embedding 0 Dallas → AH(3,2) (Dallas, Dallas)

2 matching contexts shown below. Examples are split into four columns.

token index: 0 (40 examples)

#1
' government had required that the name of an organization be placed on the permit application. In Dallas, the issue has been with the city of Dallas (regarding) some'

#2
' your MasterCard in the restaurant again yesterday." "That\'s the third time this month." "The hotel in Dallas called and said you left your suit in the'

#3
' Pre-register for $10 at www.pubruns.com\n\nRun through Architecture Icons\n\nThe iconic Dallas skyline voted into first place in the USA TODAY'

#4
' problems mixing and mastering… \nView this post\n\nLocated outside Dallas, Cool Pony Media is a record label and artist development company that works with various music genres,'

#5
'! Conrats to Marta for the amazing design.\n\nIf you have a true love for running, walking, and the great city of Dallas, then it"

#6
'1978.\n\nCharles F. Brega, Denver, Colo., for Carroll.\nEmmett Colvin, Dallas, Tex., for Wilson.'

#7
" Colorado state court on an unrelated matter the same day appellant's trial was to have begun in Dallas.\n\n\n14\nThis court has established several factors to"

#8
' After our workout, we will celebrate with two free beverages and hot appetizers all brought to you by L. A. Fuess Partners, the Dallas Center for Architecture'

#9
'1978.\n\nCharles F. Brega, Denver, Colo., for Carroll.\nEmmett Colvin, Dallas, Tex., for Wilson.'

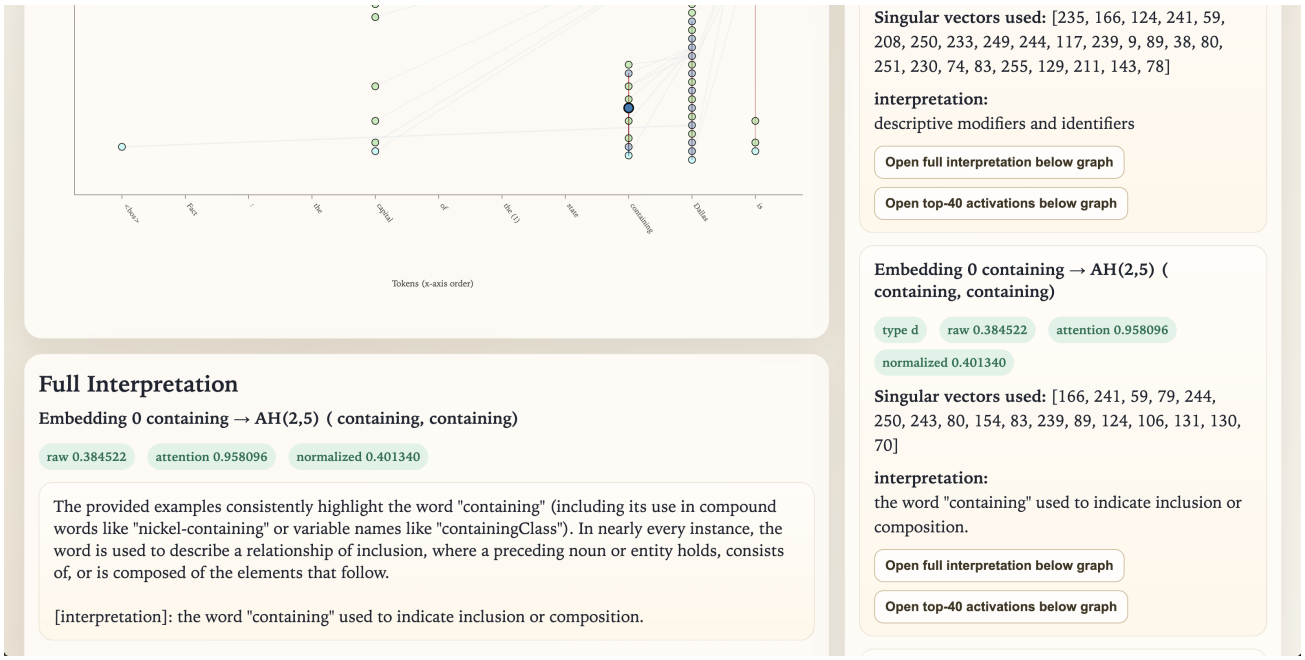
#10
'985.\n\nMark A. Nacol, John G. Tatum, Dallas, Tex., for defendant-appellant.\nLocke, Purnell, B'

#11
' Council of 13".\nStew Webb Whistle blower Grand Jury Demand against Hillary Clinton\nThe Denver Illuminati Zionist Organized Crime Chart\nLeonard Millman and'

#12
' Co and benefiting the educational programs of the Dallas Center for Architecture will hit the streets with the starting line at Klyde Warren Park on Saturday, February 20'

Figure 27. Autointerpretation and top-40 activations for the "Dallas" signal: a tightly localized lexical signal that fires on occurrences of the proper noun.

Figure 28. An alternative singular-vector decomposition of the same "Dallas" input, showing that a single token-level signal can split into multiple decompositions capturing slightly different contextual usages.



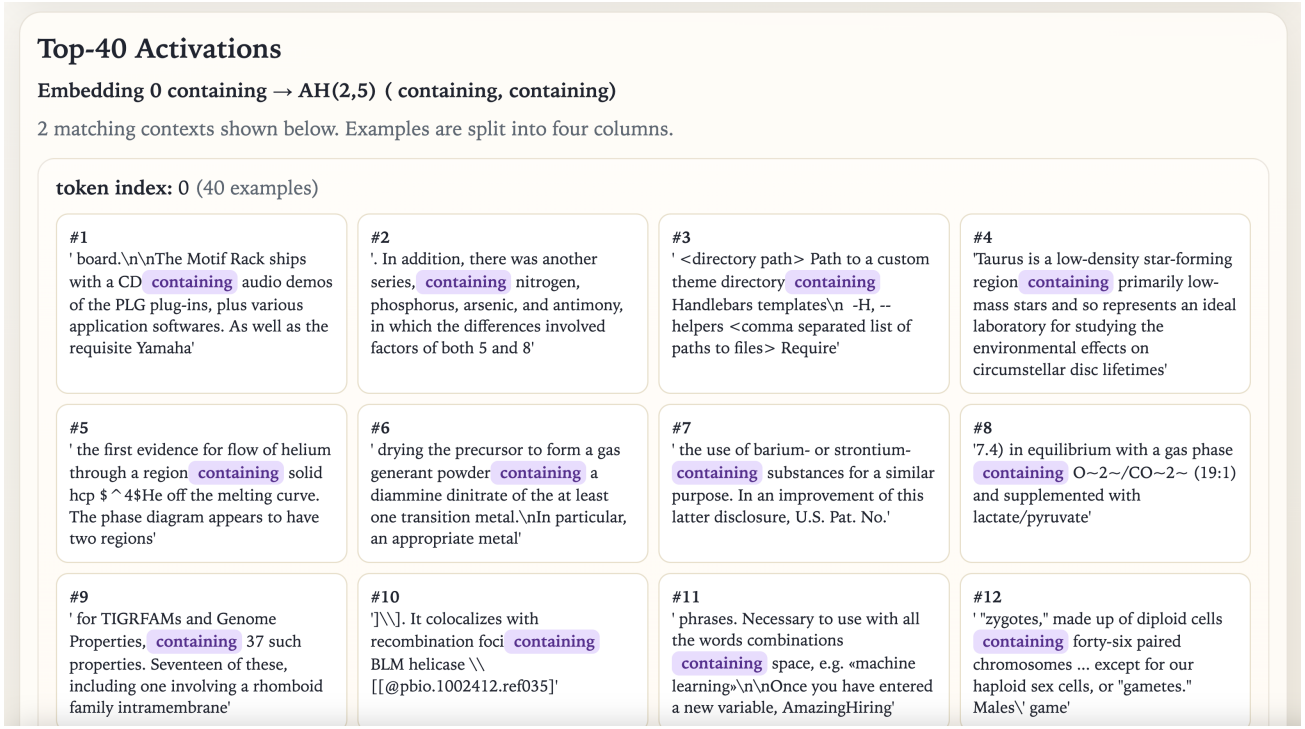


Figure 29. Autointerpretation and top-40 activations for the "containing" signal: a relational signal that captures inclusion patterns of the form "X containing Y" rather than a specific token.

Figure 30. A higher-level association between time expressions and entities, integrating information across tokens.

Figure 31. A structured-completion signal: the model identifies predictable phrase patterns, illustrating that circuits can encode reusable templates.

G. Clustering Pipeline

We now describe our pipeline for per-prompt circuit analysis. The pipeline takes a dataset of prompts as input, traces a circuit for each prompt using ACC++, and produces a clustering of these circuits into families. Finally, we provide a way of summarizing a family with a representative circuit.

G.1. Components

Given a circuit G, we can define the set of components present in G in three different levels of granularity: nodes, edges, and edge–singular-vector pairs.

Nodes We represent the set of components in a circuit G as a binary vector $\mathbf{c} \in \{0,1\}^M$, where $\mathbf{c}_i = 1$ if the i-th node is included in G, and 0 otherwise. The possible nodes consist of Attention Heads and MLP blocks. For example, for the GPT-2 Small architecture, this results in a vector of dimension $M = 156$ (12 layers $\times$ 12 heads $+$ 12 MLPs).

Edges We represent the set of components in a circuit G as a binary vector $\mathbf{c} \in \{0,1\}^M$, where $\mathbf{c}_i = 1$ if the i-th edge is included in G, and 0 otherwise. For example, for the GPT-2 Small architecture, we have a total of 10,296 possible edges[3]. Since the circuits only use a few of the possible edges, we restrict M to the number of unique edges that appear across the dataset. This reduces the size of the binary vector and speeds up computation.

Edge–Singular-Vector Pairs We represent the set of components in a circuit G as a binary vector $\mathbf{c} \in \{0,1\}^M$, where $\mathbf{c}_i = 1$ if the i-th edge–singular-vector pair is included in G, and 0 otherwise. For example, for the GPT-2 Small architecture, we have a total of 658,944 possible pairs of edges and singular vectors ($10,296 \times 64$). Similarly to the edges as components, since the circuits only use a few of the possible pairs, we restrict M to the number of unique pairs that appear across the dataset.

Figure 32 provides an illustration of these three component definitions using a simplified subgraph. Notice that we increase the granularity of information when we go from nodes to edges to edge–singular-vector pairs: nodes represents which components of the language model are active, edges record the connectivity between components, and edge–singular-vector pairs not only record the connectivity, but the "channels" used for communication inside the model. We notice that the more fine-grained information is crucial for better clustering.

G.2. Clustering

Having established how to represent a circuit as a set of components, we now outline the pipeline used for clustering the circuits. Given a dataset $P = \{p_1, \ldots, p_K\}$ of K circuits, we construct a binary component matrix $C \in \{0,1\}^{K \times M}$, where the i-th row of C corresponds to the binary vector of circuit p_i. We compute pairwise distances between all circuits using the Jaccard distance, defined for two binary vectors $\mathbf{a}$ and $\mathbf{b}$ as:

$$d_J(\mathbf{a}, \mathbf{b}) = 1 - \frac{|\mathbf{a} \cap \mathbf{b}|}{|\mathbf{a} \cup \mathbf{b}|}$$

We then group the circuits via average linkage hierarchical clustering, which iteratively merges the pair of clusters with the smallest average inter-cluster distance.

G.3. Representative

Assuming a set of objects $C = \{c_1, \ldots, c_K\}$, and some distance function $d : C \times C \to \mathbb{R}^+$. We define the sample representative of C with respect to d as the point $\hat{c} \in C$ such that:

$$\hat{c} = \arg\min_{c \in C} \sum_{i=1}^{K} d(c_i, c) \tag{32}$$

[3] $(L \times (H+1) \times (L-1) \times H)/2 = (12 \times 13 \times 11 \times 12)/2 = 10,296$

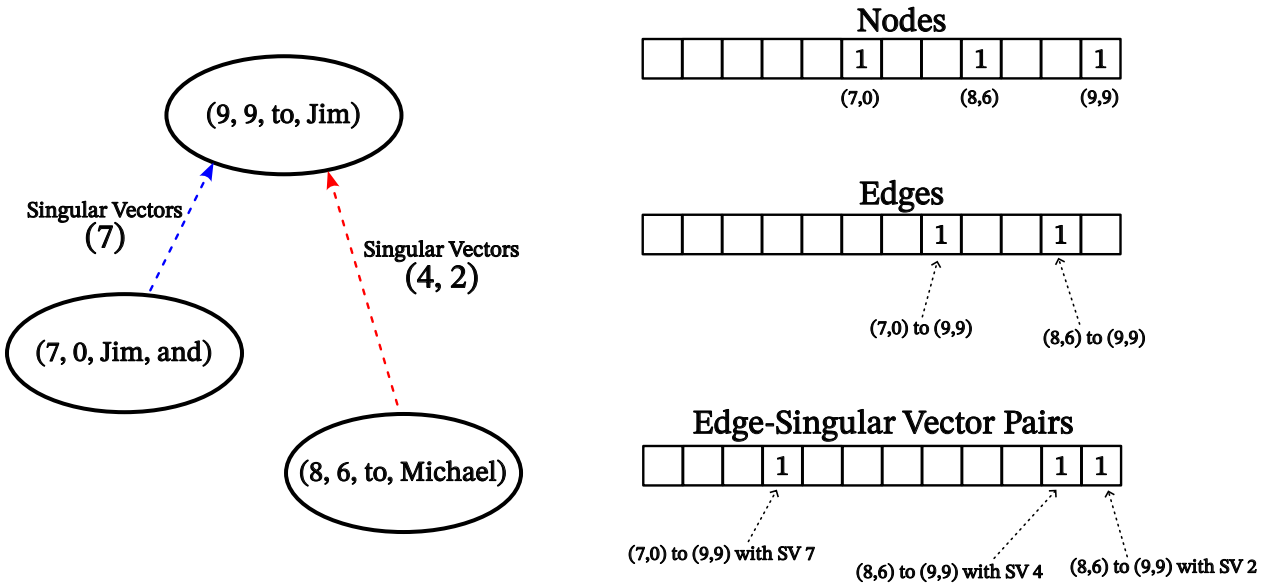


Figure 32. Illustration of circuit components at three levels of granularity: nodes, edges, and edge–singular-vector pairs.

For a given cluster, the representative is the object whose binary component vector minimizes the average Jaccard distance to all other members of the cluster. We consider the notion of sample representative as the representative circuit of a circuit family. This concept summarizes each cluster with a single object, which can be used for downstream comparison and interpretation. The sample representative is a 2-approximation to the representative.

G.4. Signals

Given a circuit G, we define the set of attention-head nodes that have incoming signals as $\mathcal{A} = \{a_1, \ldots, a_M\}$. For an attention-head node $a \in \mathcal{A}$, we define the *destination signal summary* $\mathbf{s}^a_{\text{dst}}$ as the normalized sum of all destination signals targeting a. Similarly, we define the *source signal summary* $\mathbf{s}^a_{\text{src}}$ as the normalized sum of all source signals targeting a.

For a circuit G, we stack the summaries of all nodes $a \in \mathcal{A}$ with incoming signals into two signal summary matrices:

- $S_{\text{dst}} \in \mathbb{R}^{M \times D}$, where the i-th row corresponds to the destination signal summary $\mathbf{s}^{a_i}_{\text{dst}}$.
- $S_{\text{src}} \in \mathbb{R}^{M \times D}$, where the i-th row corresponds to the source signal summary $\mathbf{s}^{a_i}_{\text{src}}$.

Given two circuits G_1 and G_2, we compute the similarity of their signals using the matrix product:

$$\text{Sim}_{\text{dst}} = S^{G_1}_{\text{dst}} \left(S^{G_2}_{\text{dst}} \right)^T \quad \text{and} \quad \text{Sim}_{\text{src}} = S^{G_1}_{\text{src}} \left(S^{G_2}_{\text{src}} \right)^T$$

The similarity matrices allow us to analyze whether nodes in distinct graphs use similar information.

H. Indirect Object Identification (IOI)

H.1. Dataset

We construct an IOI dataset balanced across high-level templates and low-level templates. The high-level template represents the order in which the indirect object (A) and subject (B) appear in the initial clause. We distinguish between two configurations: ABBA and BABA.

For example, consider the sentence: "Then, Michael and Jim went to the office. Michael gave a computer to Jim". Here, the initial clause introduces the subject "Michael" before the indirect object "Jim", thus it constitutes a BABA example. Reversing the order of the IO and subject in the initial clause would result in an ABBA example.

Low-level templates are variations on the wording of IOI sentences (e.g., different introductory clauses and connectives). There are 15 low-level templates described in Table 3. To construct the final dataset we sampled 100 examples for each unique combination of low-level template and high-level template, totaling 3000 examples ($100 \times 15 \times 2$). Finally, we used the ACC++ tracing to generate the circuits used in the following experiments.

Table 3. Low-level templates considering BABA as high-level template. The ABBA low-level templates are the same, but reversing A and B in the initial clause of the prompt.

ID	Color	Sentence
1		Then, [B] and [A] went to the [PLACE]. [B] gave a [OBJECT] to [A]
2		Then, [B] and [A] had a lot of fun at the [PLACE]. [B] gave a [OBJECT] to [A]
3		Then, [B] and [A] were working at the [PLACE]. [B] decided to give a [OBJECT] to [A]
4		Then, [B] and [A] were thinking about going to the [PLACE]. [B] wanted to give a [OBJECT] to [A]
5		Then, [B] and [A] had a long argument, and afterwards [B] said to [A]
6		After [B] and [A] went to the [PLACE], [B] gave a [OBJECT] to [A]
7		When [B] and [A] got a [OBJECT] at the [PLACE], [B] decided to give it to [A]
8		When [B] and [A] got a [OBJECT] at the [PLACE], [B] decided to give the [OBJECT] to [A]
9		While [B] and [A] were working at the [PLACE], [B] gave a [OBJECT] to [A]
10		While [B] and [A] were commuting to the [PLACE], [B] gave a [OBJECT] to [A]
11		After the lunch, [B] and [A] went to the [PLACE]. [B] gave a [OBJECT] to [A]
12		Afterwards, [B] and [A] went to the [PLACE]. [B] gave a [OBJECT] to [A]
13		Then, [B] and [A] had a long argument. Afterwards [B] said to [A]
14		The [PLACE] [B] and [A] went to had a [OBJECT]. [B] gave it to [A]
15		Friends [B] and [A] found a [OBJECT] at the [PLACE]. [B] gave it to [A]

H.2. Additional Clustering Results

We expand the results from Section 3.1 across all models and at three levels of component granularity (Appendix G.1): nodes (Figure 33), edges (Figure 34), and edge–singular-vector pairs (Figure 35). The clustermaps presented here can be read similarly to the ones in the main text: the two top bars annotate the high-level and low-level template of each leaf in the dendrogram, so contiguous colored blocks indicate prompts with the same template clustering together.

Clustering glanularity As we increase the granularity of the component definition: from nodes to edges to edge–singular-vector pairs, we observe a trend where the separability of high-level and/or low-level templates improve.

Model-dependent clustering Across all three models, the block structure is strongly aligned with prompt templates, but the relevant type of template differs by model. For GPT-2 Small, the dominant split is between high-level templates: ABBA and BABA form two clearly separated clusters. In contrast, Pythia-160M clusters primarily by low-level template, with little evidence of a consistent ABBA/BABA partition. For Gemma-2 2B, we find neither the clean high-level split of GPT-2 Small nor the dominant low-level grouping of Pythia-160M. While there is a tendency for separation between ABBA and BABA (as indicated by contiguous regions in the top annotation), the boundary is not sharp, and several clusters cut across both high- and low-level labels. The results suggest that Gemma does not rely on a single, consistent template-specific circuit, and is instead more sensitive to finer-grained prompt details beyond the two template types tracked here

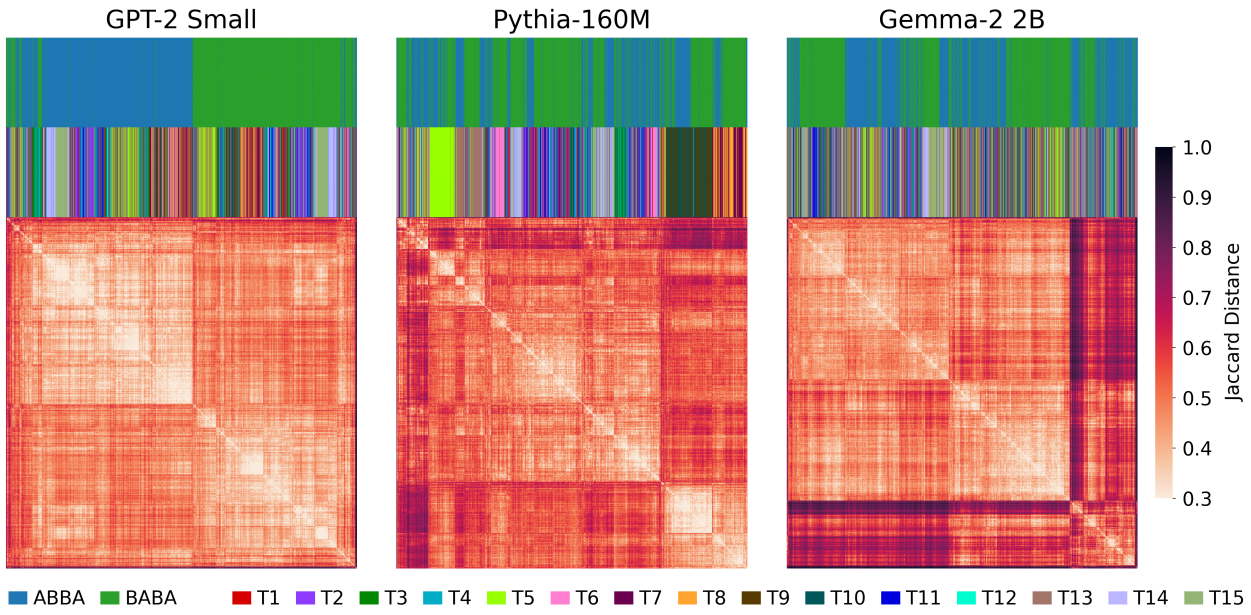


Figure 33. Average linkage clustering of prompt-level traces exposes distinct circuit families rather than a single universal IOI circuit. The top annotation bar indicates high-level templates (ABBA vs. BABA), while the bottom bar indicates low-level templates (see Appendix H.1 for color code). Prompts are represented as heads as components.

H.3. Additional Signal Similarity Results

We provide the source-side signal similarity in Figure 36, complementing the destination-side analysis in Figure 4 of the main text. Head $(9, 9)$ shows source-side cosine similarity near zero between the two representatives. This differs from the negative cosine similarity observed on the destination side, but it also shows that signals at $(9, 9)$ are different between the ABBA and BABA representatives.

Next, we present the complete signal matrix (without filtering only heads that appear in both prompts,;Appendix G.4). Figure 37 analyzes signal similarity (Appendix G.4) to the GPT-2 Small ABBA and BABA representatives; the Pythia-160M Templates 9 and 10 representatives; and Gemma-2 2B Templates 14 and 15 representatives. Because two representatives will not typically contain exactly the same set of nodes, the heatmaps compare all nodes in one representative against all nodes in the other, and strong matches can appear off-diagonal.

GPT-2 Small For GPT-2 Small (Figure 37; left-side), we observe three patterns. First, some attention heads are highly stable across templates: for example, head $(5, 9)$ shows an almost one-to-one match in both destination and source signals, suggesting an invariant computation that is reused in both ABBA and BABA. Second, some nodes align strongly despite being different heads, which suggests functional redundancy: for instance, the destination signals of $(3, 2)$ in the BABA representative closely match those of $(4, 7)$ in the ABBA representative, indicating that similar information can be routed by different heads in different templates. Third, some canonical IOI components are present in both representatives but carry different signals: notably, the name-mover head $(9, 9)$ shows completely different input signals, with negative cosine similarity, between ABBA and BABA, even though its high-level role is the same. This indicates that template changes can preserve *which* component is used while changing *how* that component implements its role, motivating a signal-level analysis of representatives as a way to study mechanistic variation across clusters.

Pythia-160M For Pythia-160M (Figure 37; middle), we compare the representatives for Template 9 (the clause "…working *at* the [PLACE] …") and Template 10 ("…commuting *to* the [PLACE] …"); see Appendix H.1, Table 3 for the full template definitions. Here, the node labels in the heatmaps (layer/head plus the associated token) let us distinguish two kinds of alignment: (i) the *same* attention head carrying the same role-level signal but with different associated tokens, and (ii) different heads carrying similar signals. Both destination and source heatmaps show strong near one-to-one matches: most high-similarity pairs preserve head identity while swapping the expected tokens, e.g. $(3, 2, \texttt{Marco})$ aligns with $(3, 2, \texttt{Jack})$

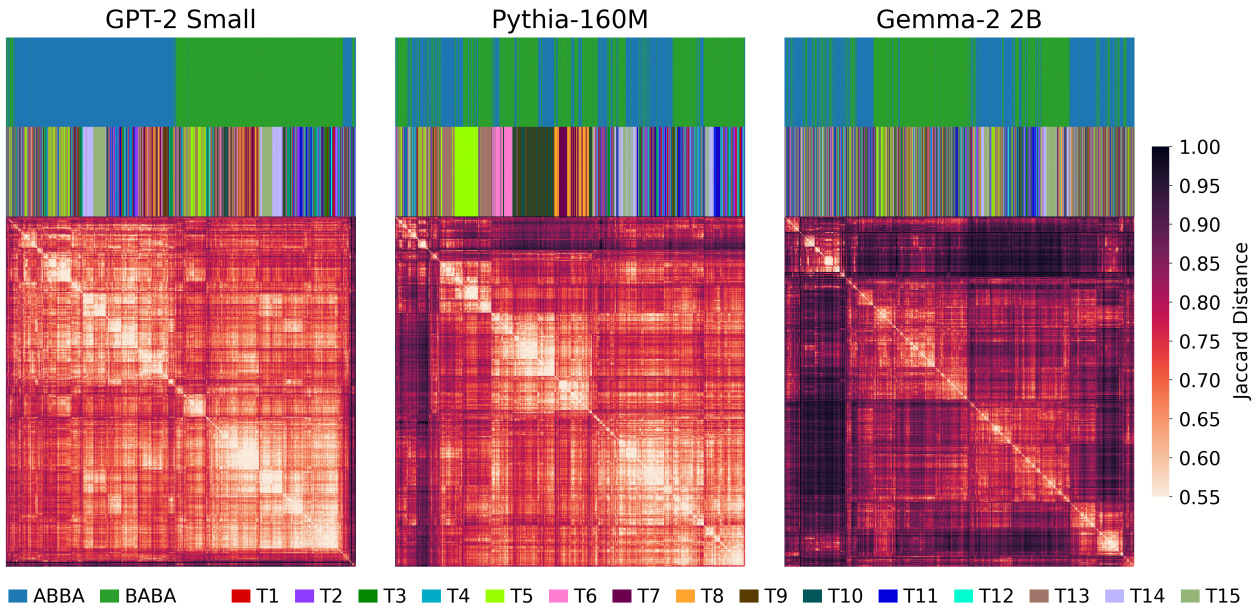


Figure 34. Average linkage clustering of prompt-level traces exposes distinct circuit families rather than a single universal IOI circuit. The top annotation bar indicates high-level templates (ABBA vs. BABA), while the bottom bar indicates low-level templates (see Appendix H.1 for color code). Prompts are represented as edges as components.

and $(2, 6, \texttt{computer})$ aligns with $(2, 6, \texttt{snack})$ in the source view, while destination-side matches concentrate on the same heads attending to structural tokens such as `to` and `and`. This suggests that, despite clustering by low-level template, many Pythia representatives reuse the same head-level routing mechanism, independent of the specific tokens being processed.

Gemma-2 2B For Gemma-2 2B, representative signal comparisons are consistent with the weaker template conditioning suggested by Figure 35. Even for two low-level templates whose clusters overlap in the clustermap (Template 14 vs. Template 15), the representatives share only a *partial* signal backbone. In Figure 37 (right-side), several heads align strongly on structural routing (e.g., head $(5, 0)$ matches almost perfectly on the destination token `and`, and also aligns on the corresponding name token in the source view). However, other heads that appear in both representatives have weak or near-zero alignment to their counterpart, including heads $(1, 4)$ and $(8, 1)$, indicating that Gemma can change head-level mechanisms even when templates are close.

H.4. Additional Interpretation Results

GPT-2: Tracing proper-noun features. Here we show that the model knows that the IO token is a proper noun, and how circuits use these features in computing model output.

Figure 38a shows a portion of the GPT-2 circuit for the ABBA representative. Proper noun features are shown in dark green. The figure shows that the "Justin" token is annotated with a "proper noun" feature by 5 MLPs (MLPs 0, 1, 2, 7, and 8) and by attention head (0, 8). It also shows that the "to" token is annotated with the same feature, indicating that it can match with a proper noun. This matching of proper-noun features between "to" and "Justin" increases the attention paid to the (to, Justin) pair by attention heads (10, 0) and (9, 9), and thereby supports the movement of the "Justin" token to the output.

Figure 38b shows that the "proper noun" feature is used in a similar fashion in the BABA representative, but that the "second item" pathway also makes use of the "proper noun" feature. Here, the same MLPs (0, 1, 2, 7, and 8) provide the "proper noun" annotation as before. However, here, MLP 0's annotation is also used by heads that are outputting the "second item in a parallel pair" signal. This shows that the fact that "Kelly" is a proper noun contributes to attention paid by the heads outputting the "second item" signal.

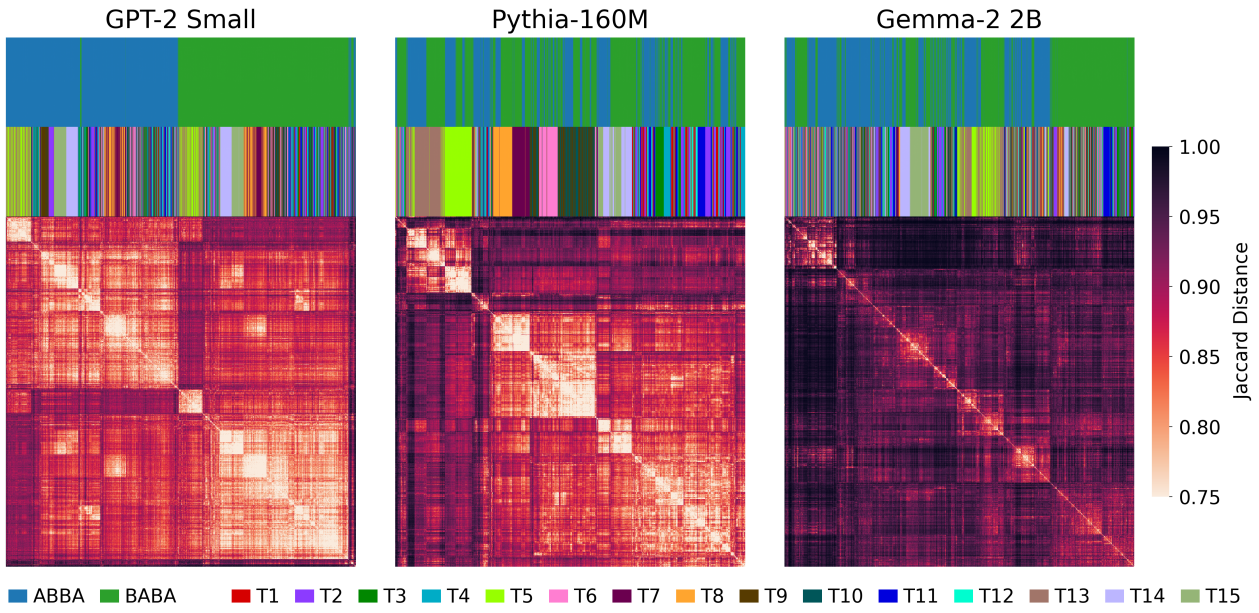


Figure 35. Average linkage clustering of prompt-level traces exposes distinct circuit families rather than a single universal IOI circuit. The top annotation bar indicates high-level templates (ABBA vs. BABA), while the bottom bar indicates low-level templates (see Appendix H.1 for color code). Prompts are represented as edge–singular-vector pairs as components.

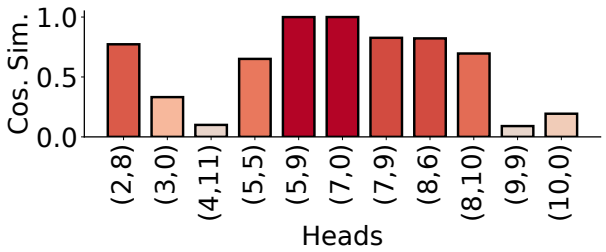


Figure 36. Comparison of source signals similarity for attention heads shared by the GPT-2 Small ABBA and BABA representative IOI circuits.

Pythia: Comparing Templates 9 and 10. For Pythia, we compare circuits from representatives of Templates 9 and 10 (see Figure 37, middle). The template 9 representative is "When Marco and William were working at the store, Marco gave a computer to," while the template 10 representative is "While Jack and Lewis were commuting to the station, Jack gave a stack to." The circuits are highly similar, but show some differences. One difference is the presence of head (4, 5) in template 10, attending to the token pair (to(1), to(1)). To understand the reason for this difference, we show the two circuits and their signal interpretations in Figure 39. Comparing these two circuits, we see that the difference arises because in template 10, there are two instances of the work "to," which requires additional disambiguation by the model. We see that head (4, 5) is recruited to annotate the second "to" so that it signifies the predecessor of a noun phrase. This makes it clear that the second "to" is introducing the indirect object and that the IO should be moved to it.

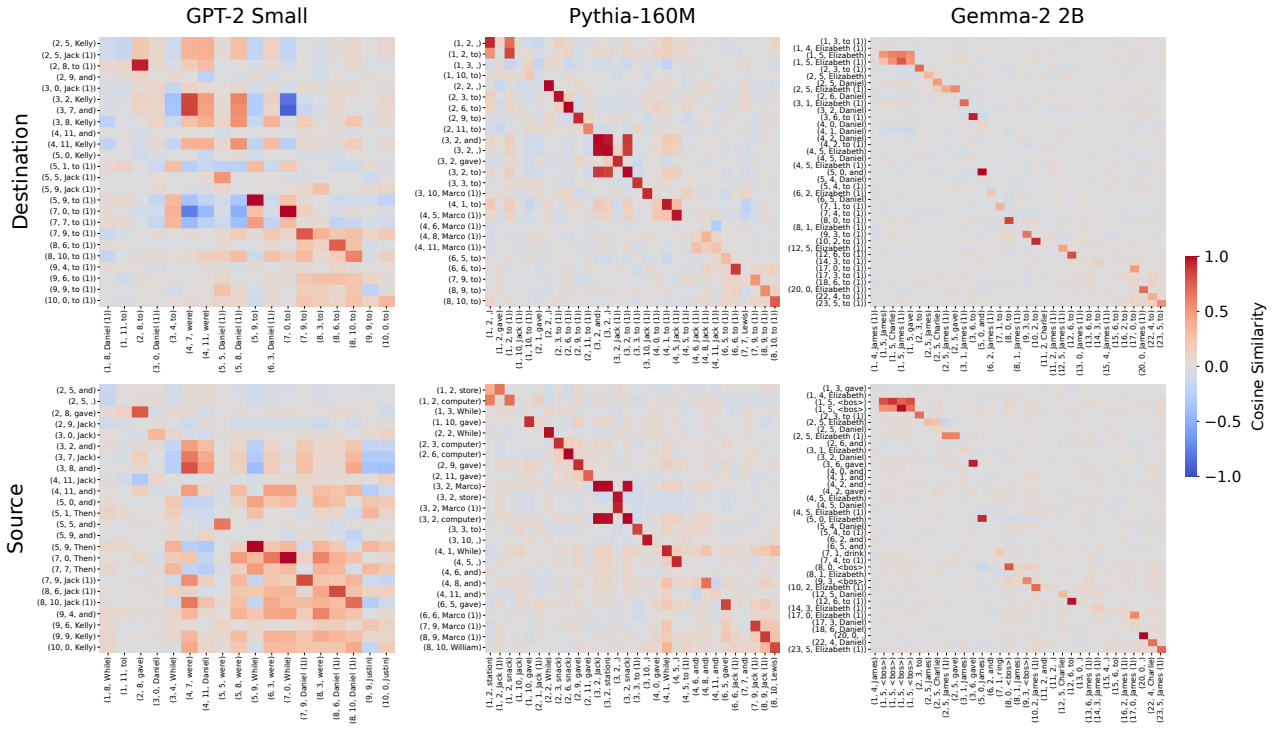
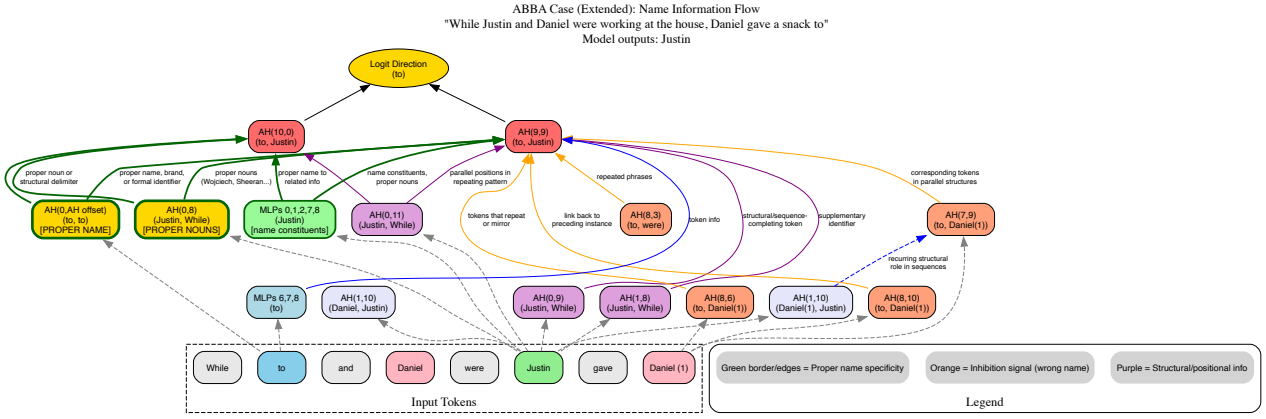
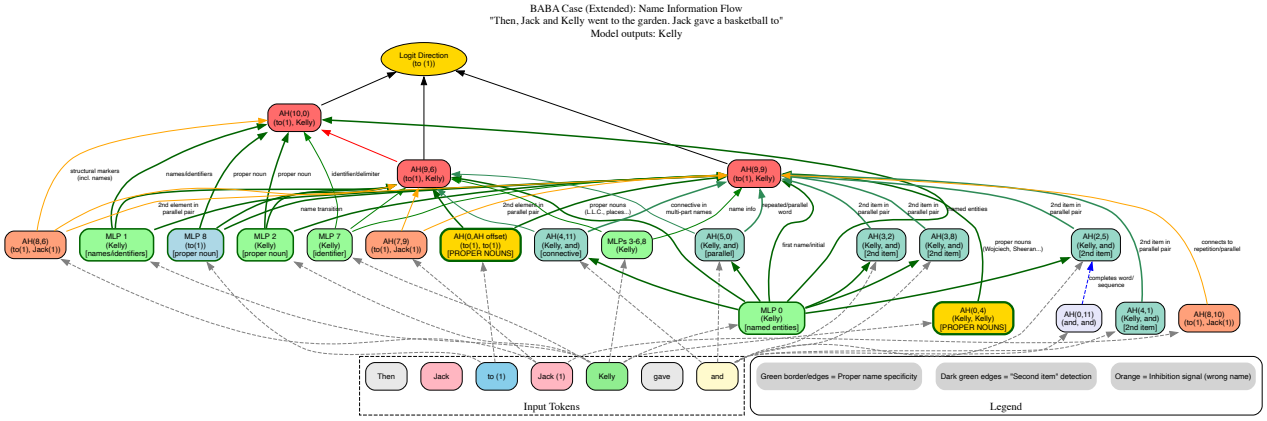


Figure 37. Signal similarity uncovers common and distinct functionality across prompts. Comparison of signal similarity matrices between representative circuits across three different models, organized by column: GPT-2 Small (left), Pythia-160M (middle), and Gemma-2 2B (right). Representatives for GPT-2 Small are from ABBA (x-axis) and BABA (y-axis). Representatives for Pythia-160M are from Template 10 (■, x-axis) and Template 9 (■, y-axis). Representatives for Gemma-2 2B are from Template 15 (■, x-axis) and Template 14 (■, y-axis). Numbers in parentheses (e.g. (1), (2)...) indicate the n-th occurrence of the token in the prompt.

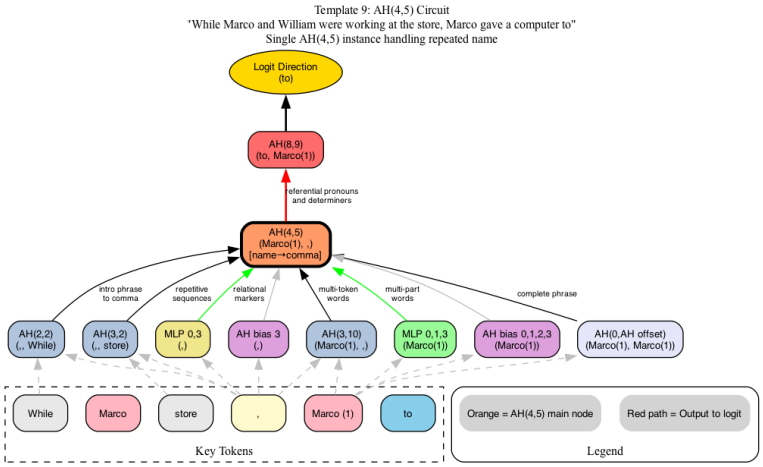


(a) ABBA

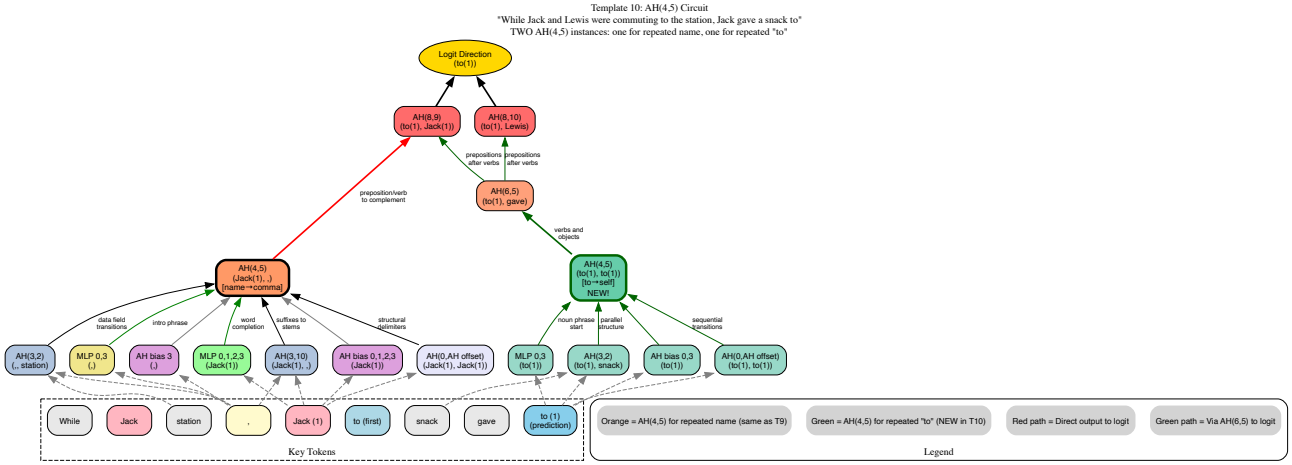


(b) BABA

Figure 38. Traces expose how model annotates and tracks the fact that the IO token is a proper noun. Same prompts as Figure 5.



(a) Template 9



(b) Template 10

Figure 39. Traces expose key differences in Pythia circuits for Templates 9 and 10: both include a shared repeated-name circuit instantiated by head $(4, 5)$, while Template 10 additionally instantiates head $(4, 5)$ on the repeated `to` token and routes this second instance through head $(6, 5)$ before reaching later-layer heads and logits.

I. Multilingual Indirect Object Identification (MIOI)

I.1. Dataset

We construct a multilingual version of the IOI dataset (Appendix H) spanning English, Spanish, French, and Portuguese. Each language uses the same set of template structures, with names adapted to be natural in each language. This produces a balanced dataset across both template types and language. In total there are 800 prompts across the four languages, four low-level templates and the two high-level templates (ABBA/BABA). The low-level templates are described in Table 4.

Table 4. Language-specific low-level templates considering BABA as high-level template. The ABBA low-level templates are the same, but reversing A and B in the initial clause of the prompt.

ID	Language	Sentence
0	English	When [B] and [A] went to the restaurant, [B] gave a drink to [A].
	Portuguese	Quando [B] e [A] foram ao restaurante, [B] deu uma bebida para [A].
	French	Quand [B] et [A] sont allés au restaurant, [B] a donné une boisson à [A].
	Spanish	Cuando [B] y [A] fueron al restaurante, [B] le dio una bebida a [A].
1	English	After [B] and [A] arrived at the library, [B] handed a book to [A].
	Portuguese	Depois que [B] e [A] chegaram à biblioteca, [B] entregou um livro para [A].
	French	Après que [B] et [A] sont arrivés à la bibliothèque, [B] a remis un livre à [A].
	Spanish	Después de que [B] y [A] llegaron a la biblioteca, [B] le entregó un libro a [A].
2	English	While [B] and [A] were at the school, [B] handed a pen to [A].
	Portuguese	Enquanto [B] e [A] estavam na escola, [B] entregou uma caneta para [A].
	French	Pendant que [B] et [A] étaient à l'école, [B] a remis un stylo à [A].
	Spanish	Mientras [B] y [A] estaban en la escuela, [B] le entregó un bolígrafo a [A].
3	English	When [B] and [A] met at the station, [B] handed a ticket to [A].
	Portuguese	Quando [B] e [A] se encontraram na estação, [B] entregou um bilhete para [A].
	French	Quand [B] et [A] se sont retrouvés à la gare, [B] a remis un billet à [A].
	Spanish	Cuando [B] y [A] se encontraron en la estación, [B] le entregó un boleto a [A].

I.2. Additional Results

We apply the same tracing and clustering pipeline (Appendix G) to mGPT and Gemma-2 2B: we trace each prompt with ACC++, represent each circuit as a set of discrete components, compute pairwise Jaccard distances between prompt-level traces, and cluster them via average linkage hierarchical clustering. Here, we report the results for Gemma-2 2B (similar to § 3.2 for mGPT). We used $\tau = \frac{1.5}{d}$ in these experiments (Appendix D).

Component reuse Figure 40 and Figure 41 presents the frequency of components for mGPT and Gemma-2 2B, respectively. We notice that the majority of the same heads and MLPs activate for all languages.

Language-specific signals Figure 42 shows that traces cluster by language in Gemma-2 2B. This result suggests the presence of language-specific signals being routed through the same components. Differently from mGPT, there is no high-level template clustering. This also mirrors the result we found in English-only IOI (§ 3.1, Appendix H), where Gemma-2 2B did not cluster by high-level or low-level templates.

Cross-language typological distance via URIEL+ We compare the cross-language structure of our circuits to typological distances between languages, drawn from URIEL+ (Khan et al., 2025). URIEL+ represents each language as a vector of binary and numeric features (e.g. word order, agreement, phoneme inventory) sourced from linguistic resources such as WALS and Glottolog, organized into six feature families: *syntactic*, *phonological*, *inventory*, *genetic*, *geographic*, and *featural* (a concatenation of the first three). The pairwise distance between two languages within a family is the cosine distance between their feature vectors in that family; we fill missing entries with the URIEL+ KNN imputer ($k = 10$) before computing distances. On the circuit side, we restrict to prompt pairs aligned by high-level template, low-level template, and position within bucket, giving the closest one-to-one matches available in the dataset (paired sample size 142–193 per language pair), and summarize each language pair by the mean Jaccard distance over its paired prompts. For each URIEL+ family we then correlate the six cross-language pair distances against the six URIEL+ distances using Pearson r, and report

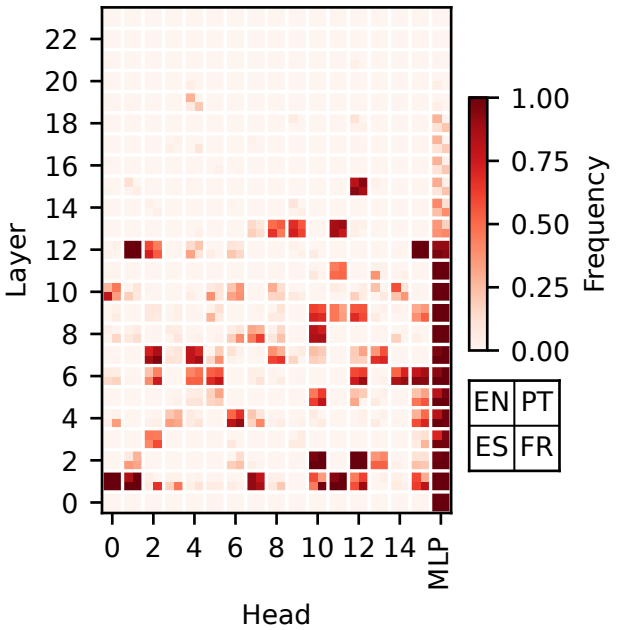


Figure 40. Frequency of components across traces for mGPT shows that multilingual models use similar set of components across languages.

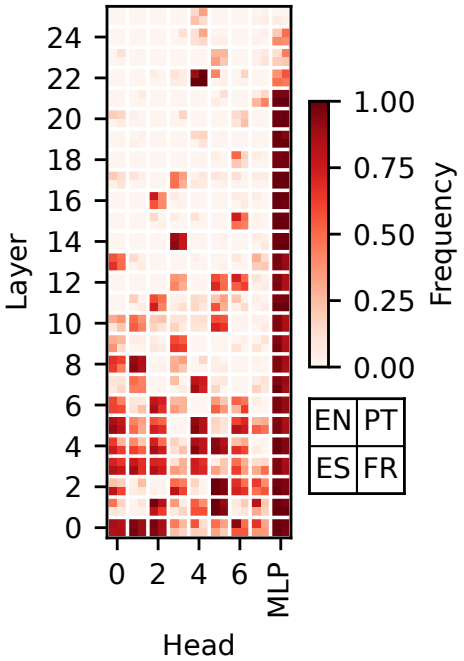


Figure 41. Frequency of components across traces for Gemma-2 2B shows that multilingual models use similar set of components across languages.

two-sided p-values from an exhaustive permutation test.

Circuit distance and typological distance (Gemma-2 2B) Figure 43 shows that the mean circuit distances strongly correlate to URIEL+ syntactic and genetic typological distances. No significant correlation was observed for the remaining feature families (*phonological*, *inventory*, *geographic*, and *featural*).

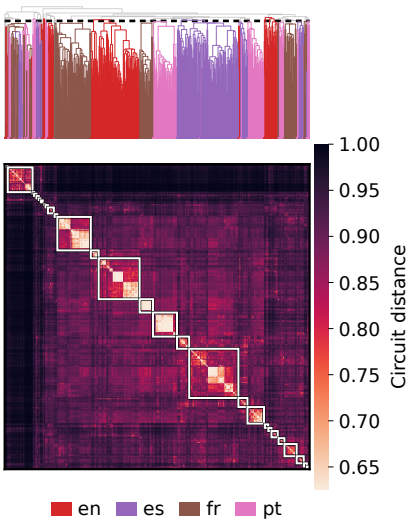


Figure 42. Average linkage clustering of prompt-level traces exposes grouping by language in Gemma-2 2B. Circuits are represented as sets of edge–singular-vector pairs.

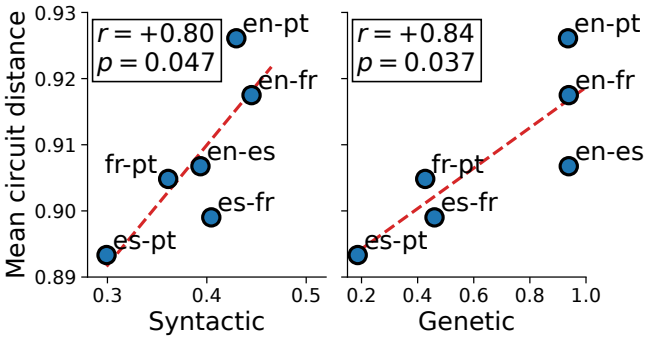


Figure 43. Mean circuit distances between Gemma-2 2B language pairs correlate with URIEL+ synthatic (left) and genetic (right) distances.

J. Factual knowledge, in-context learning, and translation examples

Gemma: Examples of factual knowledge and in-context learning. First, we take an example from (Lindsey et al., 2025): "Fact: The capital of the state containing Dallas is," for which Gemma correctly answers "Austin." We show in Figure 44 the interpretable circuit for this prompt. The circuit provides a number of insights.

First, it shows that the model immediate recognizes *the city of Dallas,* based on features present in the input embedding of the token "Dallas." Next, it shows a path (in blue) matching "state" with "containing" based on features that promote completion of a phrase particularly involving proper nouns. In this path, information about the notion of state is moved to "containing", and then "containing" is matched with "Dallas", causing "Dallas" to be annotated with the feature "predictable continuations."

The query itself is understood by integrating "capital" with "is" in head (16, 3). Here too, the features emphasize "predictable completions." Finally and crucially, MLP 17 adds a feature to "is" causing head (18, 5) to look for *geographic associations* in its source token ("Dallas"). Head (18, 5) then moves "Austin" to the end token ("is") and the "Austin" annotation is reinforced by inputs from MLPs 18 and 21 and head (22, 0).

Comparing to the methodology of (Lindsey et al., 2025), we note that Figure 44 adds considerable detail to the picture of how the model computes its output; unlike (Lindsey et al., 2025), ACC++ identifies the *specific heads and MLPs* that implement the circuit, and pinpoints the information flow between those components to illustrate how the computation progresses.

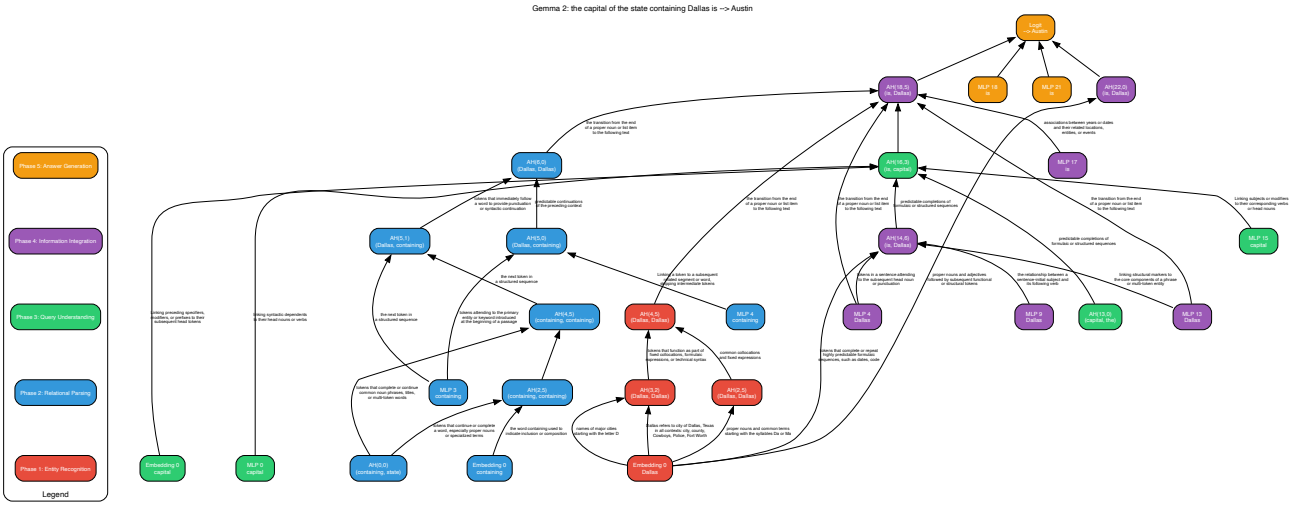


Figure 44. "Fact: The capital of the state containing Dallas is" (Austin)

Next, we turn to an example illustrating the use of ACC++ to elucidate a circuit that performs in-context learning (ICL) as well as factual recall. The prompt is "What sport Jordan played? A: Basketball. What sport Tom Brady played? A:" and the model correctly answers "football." Figure 45 shows the interpretable circuit for this prompt. Nodes are arranged vertically according to layer in the model.

The figure shows broadly how ICL is used in this setting by the model. The path in blue analyzes the one-shot phrase at the start of the prompt, moving information from "Jordan" into "basketball" in layer 1, and "sport" into "basketball" in layer 10. The path in red starts by recognizing the feature "Tom" that is present in the input embedding of "Brady," and head (3, 2) attends to "Brady" due to the feature "mentions of the primary entity or subject." The paths unite in layers 13-20, with a key step being head (16, 5) attending to the end token (":") due to its feature "completion of words or phrases." Heads 16, 17, 20, and 24 then move "football" to the end token, and this is reinforced by MLPs 21 and 25.

Gemma: Translation. We also apply ACC++ to a bilingual in-context translation prompt, `"Mary gave me a book and a card. Marie m'a donné un livre et une"`, for which Gemma-2-2b's top-2 next-token predictions are `carte` ($P = 0.9895$) and `lettre` ($P = 0.0041$). Removing the English demonstration (i.e. tracing the bare French prefix `"Marie m'a donné un livre et une"`) reduces the probability assigned to `carte` to $\approx$0.077, a factor of

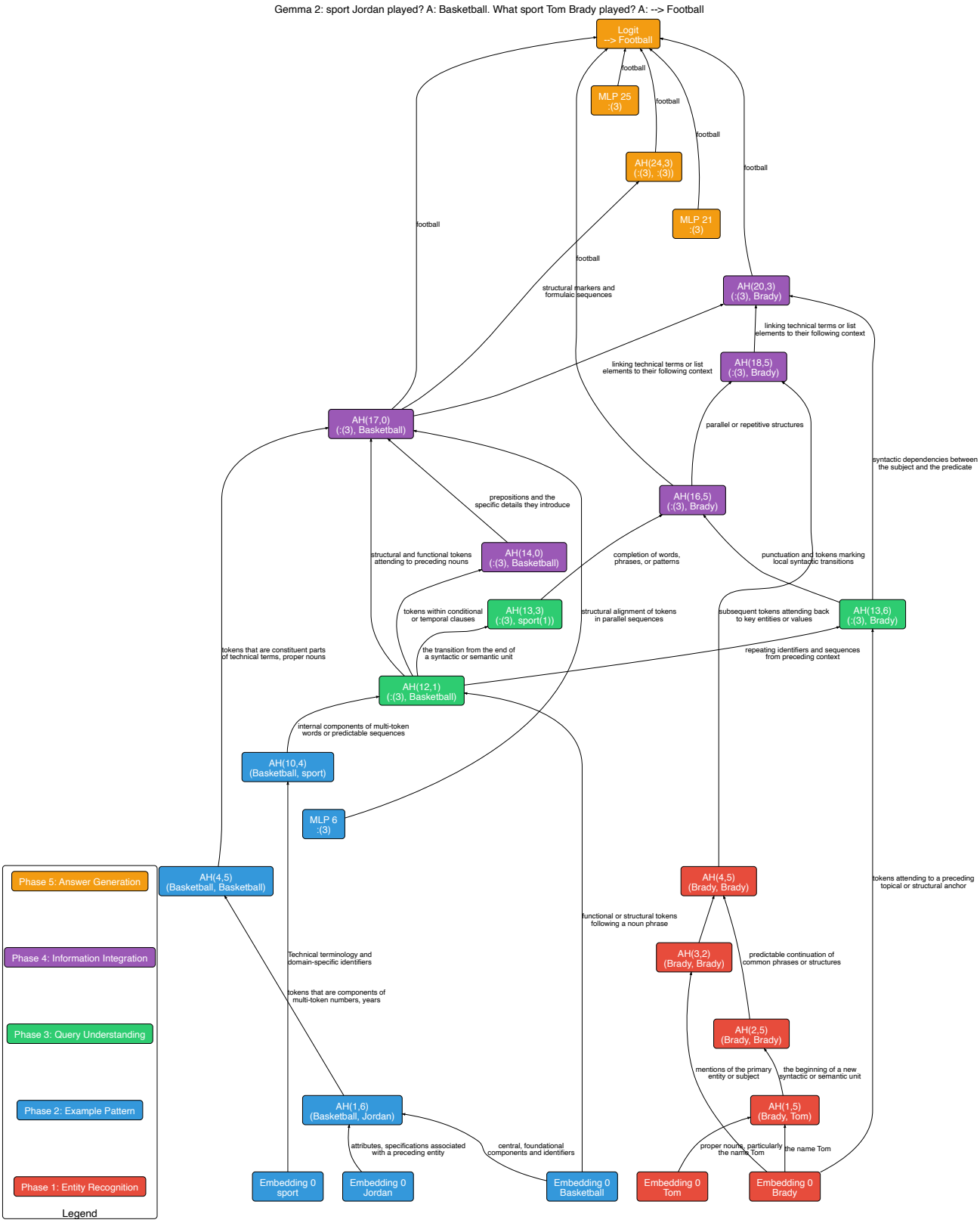


Figure 45. “What sport Jordan played? A: Basketball. What sport Tom Brady played? A:” (football).

$\sim 13\times$.

Table 5 reports the full set of root-incoming edges produced by ACC++ for both answer tokens; weights are signed logit-unit contributions. The two cross-lingual heads AH(18, 6) and AH(19, 1), which attend from the final French determiner `une` to the English demonstration noun `card`, appear only in the `carte` trace (ie, tracing from the `carte` logit direction); the `lettre` trace contains only MLP 24 and AH(25, 7). Figure 46 compares the bilingual `carte` trace (right) to the trace of the English-only prompt "`Mary gave me a book and a card`" (left, predicting `card`). Two nodes, one in each circuit, are taken to be matched when they share the same attention head (or MLP) and the cosine similarity between their aggregated incoming-signal vectors exceeds 0.25; in that case both the nodes and their incoming edges are highlighted in the figure. A large fraction of the bilingual trace's incoming signals match signals already present in the monolingual circuit, with AH(18, 6) and AH(19, 1) appearing as bilingual-only cross-lingual additions from `une` to `card`.

Table 5. Logit-edge trace of the bilingual prompt for the top-1 prediction `carte` and the runner-up `lettre`. Weights are signed contributions to the answer token's logit at the final position.

		`carte`		`lettre`	
Source	Tokens	Weight	%	Weight	%
Embedding	une ← une	1.84	8.4	—	—
AH(18, 6) (cross-lingual)	une ← card	0.76	3.5	—	—
AH(19, 1) (cross-lingual)	une ← card	3.33	15.2	—	—
MLP 20	une ← une	2.46	11.2	—	—
MLP 22	une ← une	3.70	16.9	—	—
MLP 23	une ← une	2.36	10.7	—	—
MLP 24	une ← une	6.23	28.4	8.36	48.2
AH(25, 7)	une ← une	1.26	5.8	8.99	51.8
Total		21.93		17.35	

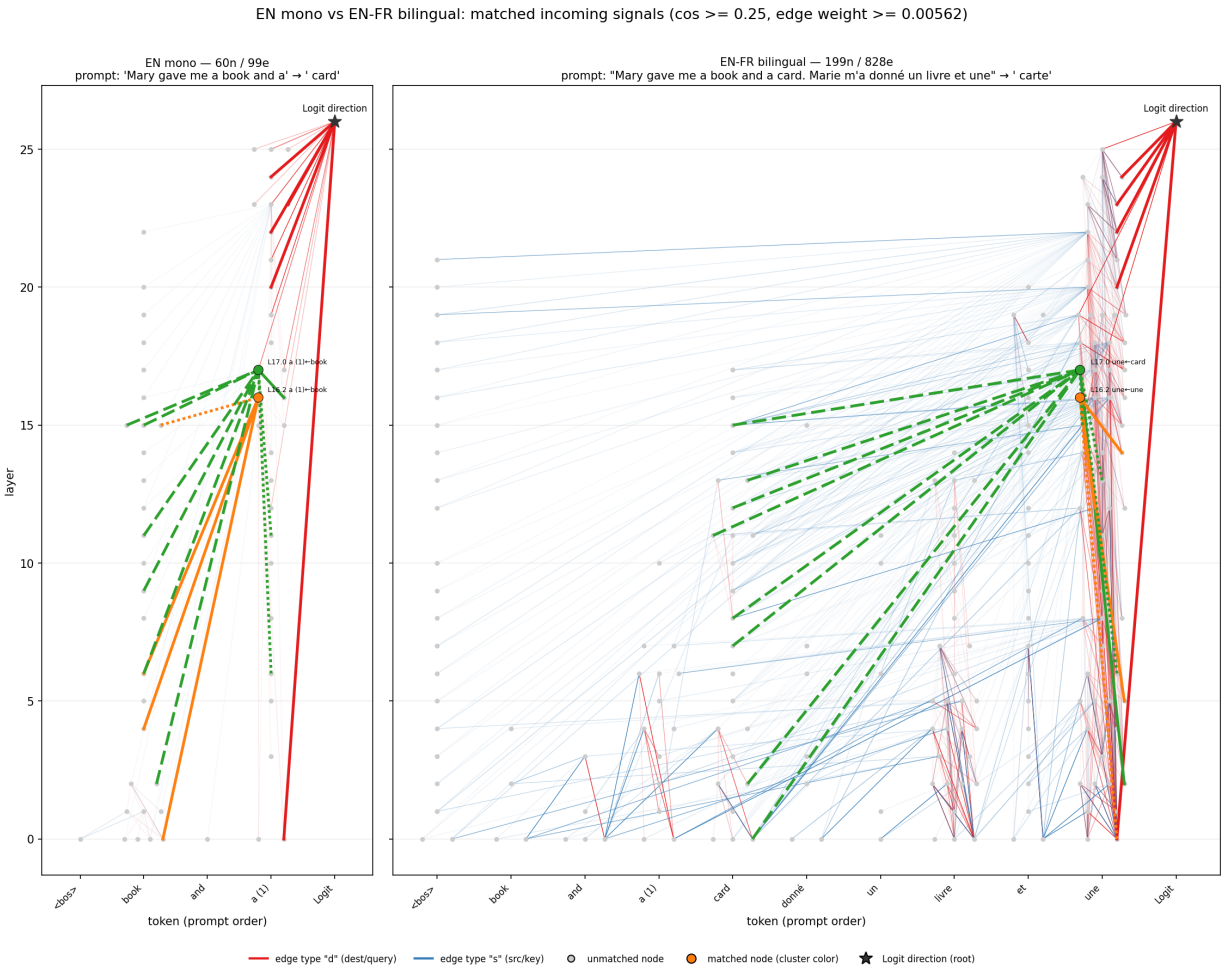


Figure 46. Side-by-side circuits for the English-only prompt "`Mary gave me a book and a card`" (left, predicting `card`) and the bilingual prompt "`Mary gave me a book and a card. Marie m'a donné un livre et une`" (right, predicting `carte`). A node in one circuit is matched to its counterpart in the other when they share the same attention head (or MLP) and the cosine similarity between their aggregated incoming-signal vectors exceeds 0.25; matched nodes and their incoming edges are highlighted.

K. Compute Resources

All ACC++ experiments fall into three buckets, run on different hardware.

ACC vs. ACC++ comparisons (Appendix E). These were run locally on a Mac with an M4 chip and 128 GB of unified memory; no discrete GPU was used.

Balanced IOI tracing (Section 3, Appendix H) and multilingual IOI tracing (Section 3, Appendix I). For GPT-2 Small and Pythia-160M, tracing was performed on CPU-only cluster nodes with 28 cores and 128 GB of RAM; per-prompt tracing time on these nodes is approximately 5–10 seconds. For Gemma-2 2B, tracing was performed on cluster nodes equipped with NVIDIA A100 40 GB GPUs, with per-prompt tracing time of approximately one minute. The full balanced IOI run (3,000 prompts $\times$ 3 models) and the multilingual runs were carried out on these same nodes.

Autointerpretation pipeline (Appendix F). The full autointerpretation pipeline (Pile activation collection, top-K extraction, explainer with DeepSeek-R1, judge with Gemma-3-27B) was run on cluster nodes with 4$\times$NVIDIA L40S GPUs. End-to-end execution across all three models (13,406 GPT-2 / 15,316 Pythia / 22,223 Gemma signals) takes approximately five days of wall-clock time on these nodes.

Preliminary and exploratory experiments not reported in the paper required additional compute on the same hardware, but their cost is small compared to the runs above.