Circuit Explained: How Does a Transformer Perform Compositional Generalization

Anonymous authors
Paper under double-blind review

Abstract

Compositional generalization—the systematic combination of known components into novel structures—is fundamental to flexible human cognition, but the mechanisms in neural networks remain poorly understood in both machine learning and cognitive science.Lake & Baroni (2023) showed that a compact encoder-decoder transformer can achieve simple forms of compositional generalization in a sequence arithmetic task. In this work, we identify and mechanistically interpret the circuit responsible for compositional generalization in such a model. Using causal ablations, we isolate the circuit and further show that this understanding enables precise activation edits to steer the model's outputs predictably. We found that the circuit leverages the disentangled representation of position and token so that functional transformations can be applied to positions in a token-independent manner. Our findings advance the understanding of how compositionality can emerge in neural networks and offer testable hypotheses for similar mechanisms in other neural architectures and compositional tasks. Code will be published after double-blind review.

Keywords: Transformer; Mechanistic Interpretability; Compositionality

1 Introduction

Humans excel at compositional generalization—the ability to combine known components to solve novel problems. For example, once we learn the concept of "swap," we can apply it to any two objects, regardless of their identity. This capacity for relational abstraction appears early: even infants generalize rules like same-different or before-after to unfamiliar contexts (Gentner, 1983; Holyoak & Thagard, 1994; Marcus et al., 1999). While cognitive science has long studied this ability, its neural basis remains unknown.

In parallel, compositional generalization has posed a longstanding challenge in machine learning. Building flexible, general-purpose systems requires models to go beyond memorization and exhibit systematic generalization. Classic critiques, such as those by Fodor (1979), argued that connectionist models lack the structure required for compositionality. While this concern shaped decades of skepticism, recent advances with transformer-based models (Vaswani et al., 2017)—and large-scale training paradigms (Kaplan et al., 2020; Bubeck et al., 2023) have provided substantial counterexamples.

In recent work, Lake & Baroni (2023) demonstrated that a compact encoder-decoder transformer can achieve human-like compositional generalization in a symbolic sequence arithmetic task. Here, we leverage the small scale of this model and provide an end-to-end mechanistic interpretation of how it solves this compositional generalization task. We trace the model's behavior to a minimal, interpretable circuit and reverse-engineer the attention dynamics into a human-readable algorithm. We validate the circuit's role by predictably steering the model's output.

This study revealed two core principles underlying the model's success:

1. The model relies on disentangled representations of position (slot) and token (content) embeddings.
2. *Functions* operate on disentangled position embeddings allowing generalized transformations of arbitrary token identities.

While the model and task we have considered are specific, the principles of compositional generalization we have identified may apply to broad compositional tasks in machine learning and cognitive science.

We discuss **Related Work** on *Transformer Circuit Interpretation* and *Compositional Generalization in Transformers* in the Appendix.

2 Experimental Setup

Our experimental setup involves a synthetic function composition task (Figure 1) designed to probe *compositional generalization* in a compact Transformer (Lake & Baroni, 2023). We outline the task structure, the Transformer basics (including attention mechanisms), and the training protocol.

2.1 Task Structure

Each episode consists of a **Support Set** and a **Question** (Figure 1):

Support Set:

Specifies (i) how the *Primitives* are symbol-to-color mappings (e.g., `B` maps to `blue`, written as `[B] = blue`, with `[ ]` as an interpretation function for translating inputs to outputs; or `D` maps to `pink`, written as `[D] = pink`). The Support Set also specifies (ii) *Functions* as symbolic operations that take primitives as arguments, and calls the interpretation function on those arguments in a new order. For instance, the function `S` might take two arguments and evaluate them in the order of (e.g., `[B S D] = [D][B][D] = pink blue pink`. The `2nd` token is always the *function* token; the `1st` and `3rd` tokens are always the *argument* tokens to the *function*. The length of *function* output ranges from 2 to 5.

Question:

Presents a new composition of primitives and functions defined in the Support Set.

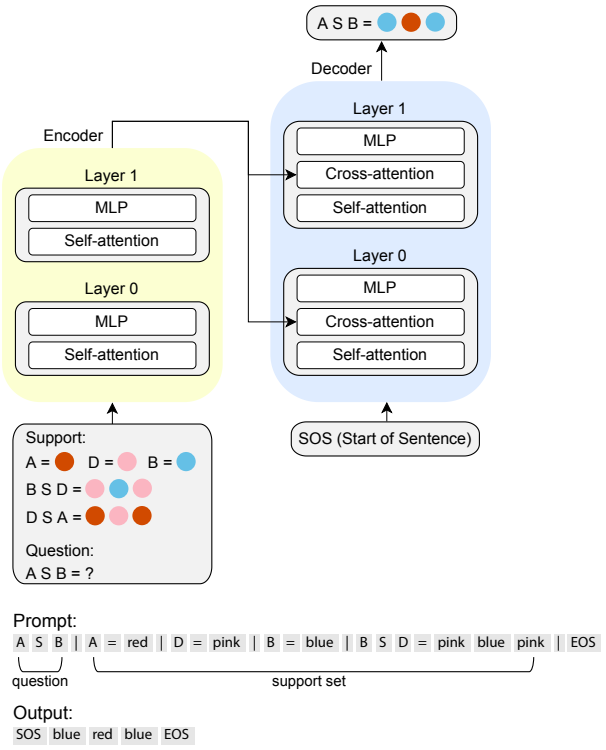

Figure 1: Top, schematic of the transformer model and task. Bottom, the prompt and output format for the compositional generalization task.

The model generates answers to the **Question** as token sequences emitted from the decoder, with a `SOS` (start of sentence) token as the first input to the decoder and an `EOS` (end of sentence) marking the end of the emission. The model operates strictly via in-context learning—weights remain frozen during inference, and test episodes are disjoint from training data. The model must infer latent variable bindings (primitives and functions) from the **Support Set** and dynamically compose these bindings to solve the novel **Question**.

2.2 Model

2.2.1 Transformer Basics

Our transformer uses an encoder-decoder architecture that involves two types of attentions:

- **Self-Attention:** Captures within-sequence interactions. The token embedding matrix $X \in \mathbb{R}^{n_{\text{input}} \times d_{\text{model}}}$ is projected into Queries, Keys, and Values:

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V,$$

where $W_Q, W_K, W_V \in \mathbb{R}^{d_{\text{model}} \times d_{\text{head}}}$ are learnable weight matrices.

- **Cross-Attention:** Enables the decoder to attend to encoder outputs. Here, the Queries (Q) come from the *decoder* tokens, while the Keys (K) and Values (V) come from the *encoder* tokens. We denote the different tokens used to compute $Q\ K\ V$ as $X_Q\ X_K\ X_V$.

The attention mechanism operates through two separate circuits on embedding $X \in \mathbb{R}^{n_{\text{input}} \times d_{\text{model}}}$ for each attention head:

- **QK Circuit** ($W_Q W_K^\top$): Determines from *where* information flows to each token by computing attention scores between token pairs, with higher scores indicating stronger token-to-token relationships:

$$\text{Attention}(Q, K) = \text{softmax}\left(\frac{X_Q W_Q (X_K W_K)^\top}{\sqrt{d_{\text{head}}}}\right) \in \mathbb{R}^{n_{\text{query}} \times n_{\text{key}}},$$

  where *softmax* is applied along the Key dimension and independently for each head.

- **OV Circuit** ($W_V W_O$): Controls *what* information gets written to each token position. Combined with the **QK Circuit**, this produces the output of the attention head:

$$Z = \text{Attention}(Q, K) \cdot X_V W_V W_O \in \mathbb{R}^{n_{\text{query}} \times d_{\text{model}}},$$

  where $W_O \in \mathbb{R}^{d_{\text{head}} \times d_{\text{model}}}$ is a learnable weight matrix.

Following attention blocks, each transformer layer includes a position-wise **MLP block**, which applies a non-linear transformation to each token embedding independently. However, prior work by Wang et al. (2022) demonstrated that MLPs are less relevant for induction tasks. Therefore, our circuit analysis focuses exclusively on attention heads.

2.2.2 Model Training

We adopt an encoder-decoder Transformer with **2 layers** in the encoder and **2 layers** in the decoder (Figure 1) with each layer containing **8 attention heads**. Further model details appear in the Appendix.

We train on 10,000 such episodes for 50 epochs and evaluate on 2,000 test (held-out) episodes. The model achieves **94%** accuracy on this test set, indicating strong compositional generalization capabilities. In the test set, primitive assignments and function definitions are conjunctively different from those in the training set (i.e., some primitives or some functions might be in the training set, but not the whole combination of them), preventing a memorization strategy. Please refer to the Appendix for additional details.

3 Results

We first outline the high-level algorithm the model appears to implement. We then describe our circuit discovery process, using causal methods to isolate the attention heads driving compositional behavior. Finally, we validate the mechanism through targeted interventions that predictably steer model outputs. Throughout, we use 1-indexing (count from 1) for tokens.

3.1 The High-Level Algorithm

The overall algorithm the model implements can be described in terms of three logical steps (Figure 2).

1. **Unbind**: The *function* output in the Support is coded in terms of an array of positions associated with the *argument* tokens.
2. **Rebind**: The new *argument* tokens in the Question bind with the original *argument* positions in the Support forming new position-token pairs.

3. **Produce**: Reapply the function coded in terms of positions to the new position-token pairs to form an array of output tokens.

In essence, the model uses token-independent positions as a pointer system that through **unbinding** and **rebinding** can generalize to different token identities.

3.2 Transformer Solution with Attention Operations

Next, we map each step to specific attention heads and walk through the attention operations with a guidance episode in Figure 3.

Nomenclature: given a function definition in the prompt (e.g., '`...| B S D=pink blue pink|...`'), function Left-Hand-Side (LHS) refers to tokens on the left (e.g., '`B S D`'); function Right-Hand-Side (RHS) refers to tokens on the right (e.g., '`pink blue pink`'). Index-on-LHS/RHS refers to the *relative* position of tokens on the LHS/RHS, (e.g., '`B=1st, pink=1st`'). Similarly, index-in-question refers to a token's *relative* position in the *Question*. The attention heads are named by their interpreted roles, which are detailed in the **Circuit Discovery** section.

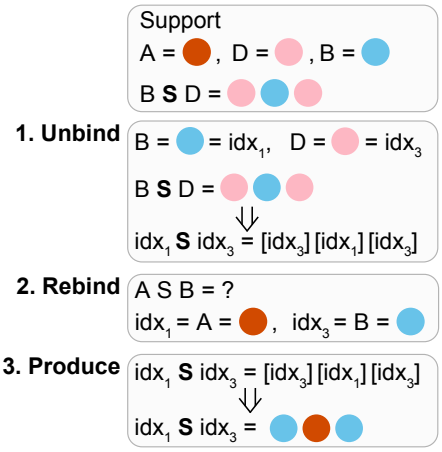

Figure 2: Schematic of the compositional algorithm. For **Unbind**, in '`B S D`', `B` is the `1st` token (idx_1), and '`B=blue`', so '`B=blue`=idx_1'; similar for '`D`'. For **Rebind**, in '`A S B`', `A` is the `1st` token (idx_1), and '`A=red`', so 'idx_1`=A=red`'; similar for '`B`'.

Step 1: Unbind

- **Primitive- and Function-Retrieval Heads** (Figure 3a): Color tokens on the function RHS (`pink`) attend to their associated primitive tokens on the function LHS (`D`), inheriting the latter's index-on-LHS (`3rd`). The heads are detailed in Figure 10.

Step 2: Rebind

- **Question-Broadcast Head** (Fig. 3b): Primitive input tokens in the Support (e.g., `B`) attend to the same primitive tokens in the Question (`B`), inheriting the latter's index-in-question (`3rd`). The head is detailed in Figure 6b.
- **Primitive-Pairing Head** (Fig. 3c): Color tokens (`blue`) attend to their associated primitive tokens (`B`), inheriting the latter's index-in-question (`3rd`). The head is detailed in Figure 6a.

Step 3: Produce

- **RHS-Scanner Head** (Fig. 3d): The `1st` token in the Decoder (`SOS`) attends to the `1st` tokens on the function Right Hand Side (RHS) (`pink`), inheriting the latter's former-inherited index-on-LHS (`3rd`). The head is detailed in Figure 9.
- **Output Head** (Fig. 3e): `SOS` token (with inherited index-on-LHS=`3rd`) attends to color tokens (`blue`) with the same index-in-question (`3rd`), inheriting the latter's token identity (`blue`), and generates the next prediction (`blue`). The head is detailed in Figure 4. Then the `2nd` token in the Decoder (`blue`) repeats the operation with the RHS-Scanner Head until completion of function RHS.

Together, these heads form a modular circuit that performs compositional generalization via pointer-based binding, mirroring symbolic execution through argument extraction, rebinding, and output generation.

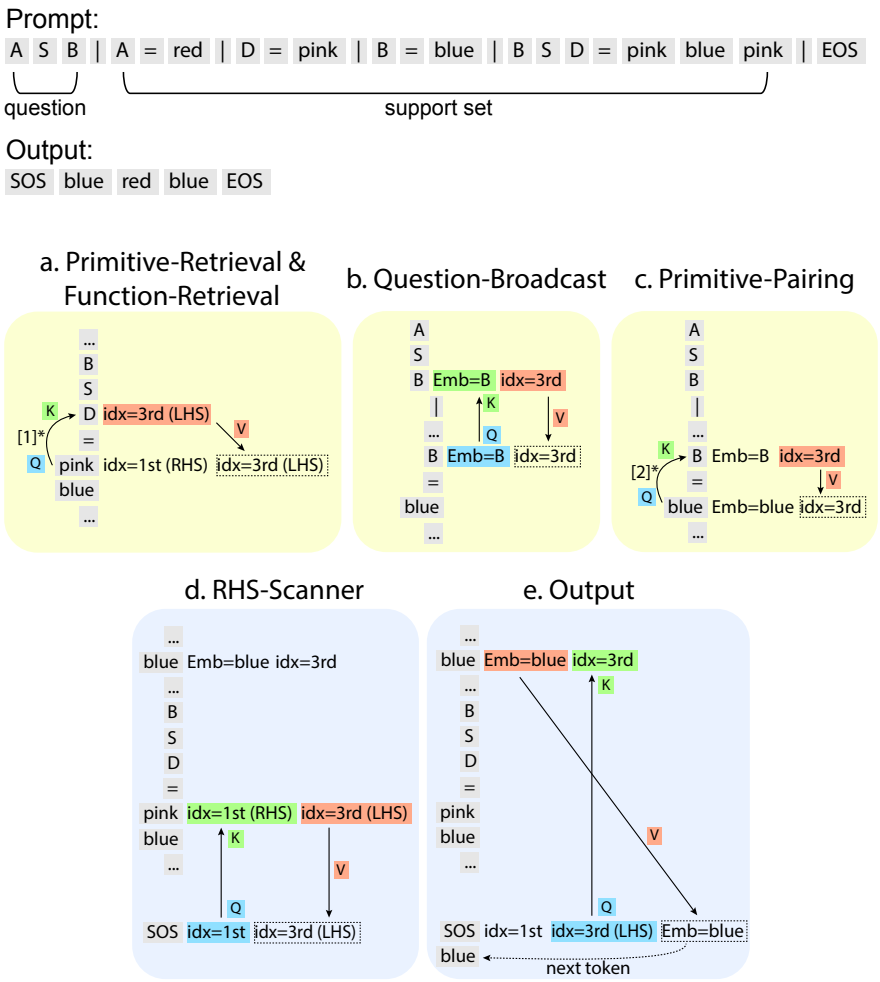

Figure 3: Summary of circuit for compositional generalization. Top, the example episode's input and output. For a-e, the yellow boxes indicate self-attention heads and the blue boxes indicate cross-attention heads. Titles refer to the functional attention heads that execute the steps (details in Circuit Discovery section). We unfold all relevant information superimposed in tokens' embeddings and highlight their roles in attention operations. [1]*, the QK alignment discussed in Primitive-Retrieval Head section. [2]*, the QK alignment discussed in Primitive-Pairing Head section.

3.3 Circuit Discovery

Nomenclature: for attention heads, *Enc-self-0.5* stands for *Encoder, self-attention, layer 0, head 5*; similarly, *Dec-cross-1.5* stands for *Decoder, cross-attention, layer 1, head 5.*

3.3.1 Output Head (Dec-cross-1.5; Figure 4b)

We discovered the model's circuit backwards from the unembedding layer using *logit attribution* (nostalgebraist, 2020), which measures each decoder attention head's linear contribution to the final token logits (adjusted by the decoder's output *LayerNorm*). We identified **Dec-cross-1.5** (decoder cross attention layer 1 head 5) as the primary contributor (Figure 4a).

Dec-cross-1.5's Q tokens always attend to the K tokens from the Encoder that are the *next* predicted ones. For example, in Figure 4b, the `SOS` token attends to instances of `red` in the Support Set, which is indeed the correct next output prediction. This attention accuracy (i.e., max-attended token being the next-emitted

token) of Dec-cross-1.5 remains above 90% for the first three tokens in the responses across all test episodes (Figure 4c), with Dec-cross-1.1 and -1.3 partially compensating beyond that point.

These observations suggest that Dec-cross-1.5's OV circuit feeds token identities directly to the decoder unembedding layer (output layer). Specifically, we observe that the output of the OV circuit, XW_vW_o, align closely (strong inner product) with the unembedding vectors of the corresponding tokens (Figure 4d). Hence, we designate Dec-cross-1.5 as the *Output Head* (while Dec-cross-1.1 and -1.3 perform similar but less dominant roles).

Next, we show how the Output Head identifies the correct token through QK interactions.

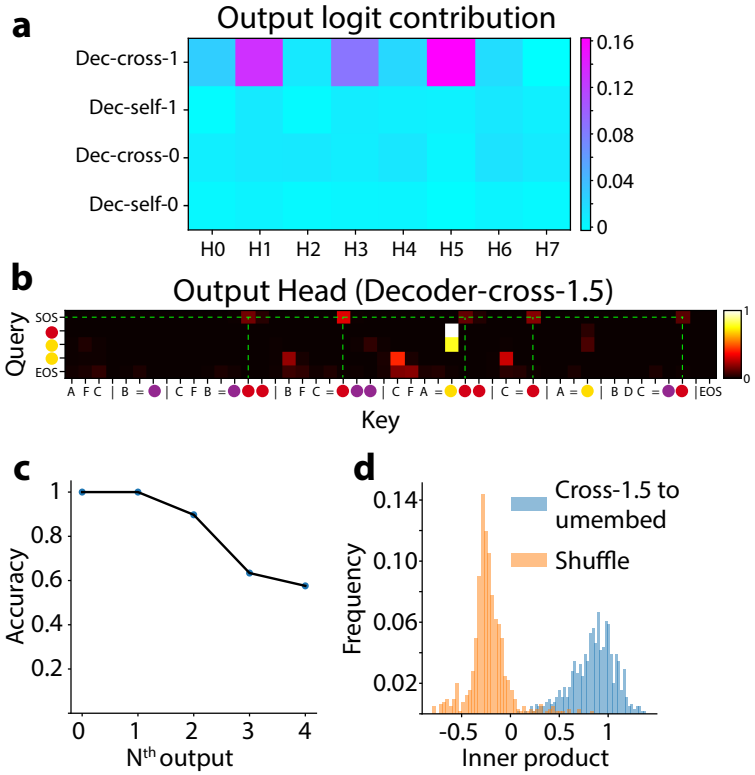

Figure 4: (a) Logit contributions of each decoder head to the logits of correct tokens (fraction to total logits). (b) Attention pattern of Dec-cross-1.5. (c) For Dec-cross-1.5, the percentage of attention focused on the next predicted token. (d) For Dec-cross-1.5, alignment (inner product) between its OV output (e.g., $x_{red}W_vW_o$) and the corresponding unembedding vector (e.g., Unemb_{red}). We estimated the null distribution by randomly sampling unembedding vectors.

3.3.2 The K-Circuit to the Output Head

We first determine which encoder heads critically feed into the Output Head's K. To do this, we performed *path-patching* (Wang et al., 2022) by ablating all but one single encoder head and then measuring how much of Output Head's QK behavior (*i.e.*, attention accuracy) remained. During these experiments, Output Head's Q were *frozen* using clean-run activations. Here we report patching results with mean-ablation (qualitative similar to random-sample ablation) (details in Appendix).

Through this process, we identified **Enc-self-1.1** and **Enc-self-0.5** as the primary contributors to Output Head's K, acting in a sequential chain (Figure 5). Next, we show how they sequentially encode symbols' index-in-question critical for the QK alignment.

Primitive-Pairing Head (Enc-self-1.1; Figure 6a) This head exhibits a distinct attention pattern that pairs each color token with its associated primitive symbol token (e.g., in the Support Set, all instances of `red` attend to `C`). In other words, Enc-self-1.1 relays information (described below, as computed by e.g.,

Enc-self-0.5) from the primitive symbols to their corresponding color tokens via its QK circuit. Hence, we call Enc-self-1.1 the *Primitive-Pairing Head*.

To investigate which upstream heads feed into the OV circuit of the Primitive-Pairing Head, we applied a sequential variant of path-patching, isolating the chain:

$$\begin{gathered}\text{Upstream heads (e.g. Enc-self-0.5)} \longrightarrow \\ \text{Primitive-Pairing Head } (V) \longrightarrow \\ \text{Output Head } (K),\end{gathered}$$

while mean-ablating all other direct paths to Output Head's K. We identified **Enc-self-0.5** as an important node (Figure 6b).

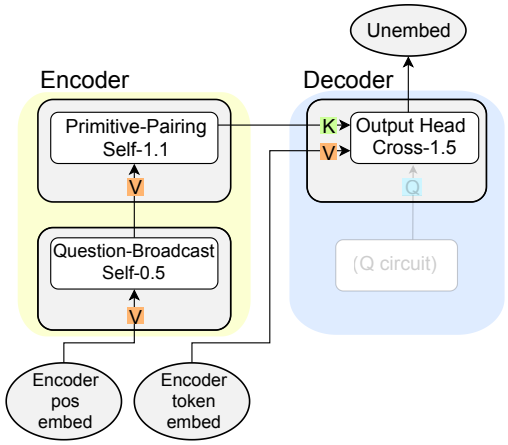

Figure 5: Enc-self-1.1 and Enc-self-0.5 serve as the main contributors of the K-circuit for the Output Head.The K-circuit encodes primitive symbols' index-in-question.

Question-Broadcast Head (Enc-self-0.5; Figure 6b) All input symbol in the Support Set attend to their copies in the input Question. In other words, Enc-self-0.5 broadcasts question-related information (including token identity and position) across symbols in the Support Set (henceforth the Question-Broadcast Head). We hypothesize that the primitive symbols' index-in-question is the critical information passed from the Question-Broadcast Head's Z through the Primitive-Pairing Head's Z and lastly into the Output Head's K.

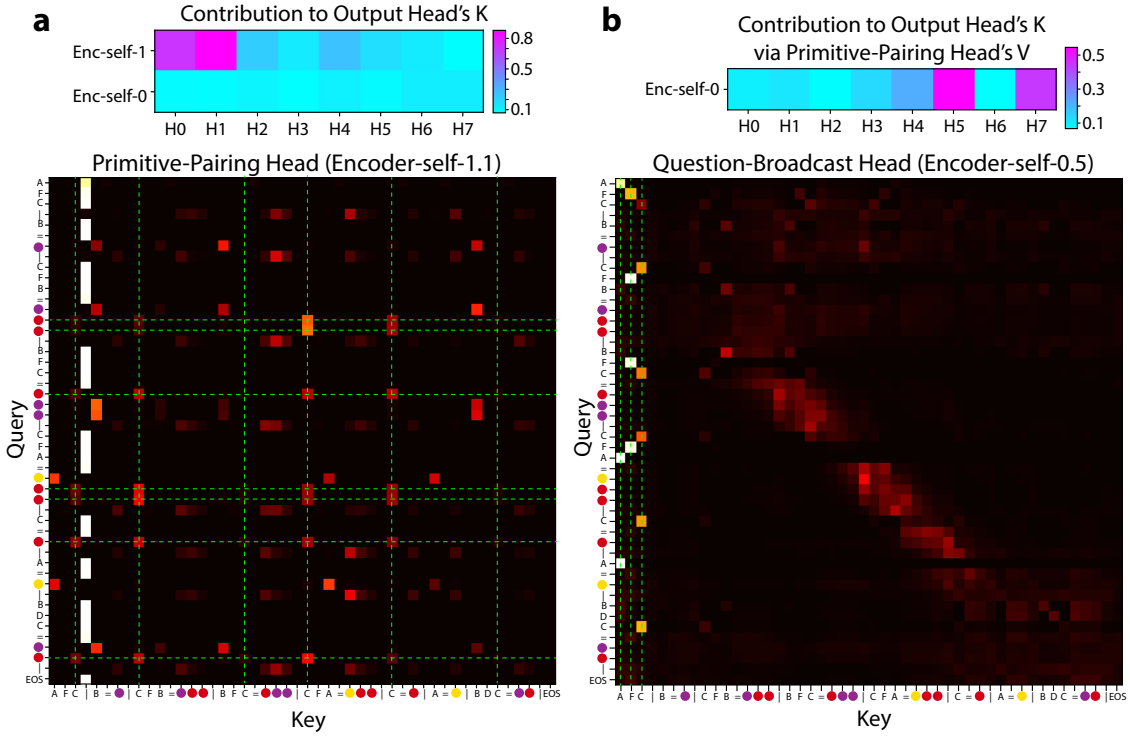

Figure 6: (a) Top, contributions to Output Head's performance (percentage of attention on the correct next token) via K. Bottom, attention pattern of Enc-self-1.1. (b) Top, contributions to the Output Head's performance through the Primitive-Pairing Head's V. Bottom, attention pattern of Enc-self-0.5.

Index-In-Question Tracing To validate this hypothesis, we examined the Question-Broadcast Head's Z for each *primitive-symbol* token. We reduced these outputs to two principal components and colored each

point by its index-in-question. As illustrated in Figure 7a, the Question-Broadcast Head's Z exhibit clear clustering, indicating that the index-in-question is robustly encoded at this stage (quantified by the R^2 score, i.e., the amount of variance explained by index identity, details in Appendix). We further confirmed that the Primitive-Pairing Head's Z preserves index-in-question (Figure 7b) and that the resulting Output Head's K also reflect the same clustering (Figure 7c).

Causal Ablation Finally, we verified that this circuit indeed causally propagates index-in-question. Ablating the Question-Broadcast Head's Z (together with the similarly functioning Enc-self-0.7) obliterates the clustering in the Primitive-Pairing Head's Z; ablating the Primitive-Pairing Head's Z (together with similarly functioning Enc-self-1.0) disrupts the clustering in the Output Head's K (Figure 7). We therefore conclude that the Question-Broadcast Head, the Primitive-Pairing Head and heads with similar functions form a crucial K-circuit pathway, passing index-in-question information from primitive tokens to their associated color tokens in the Output Head's K.

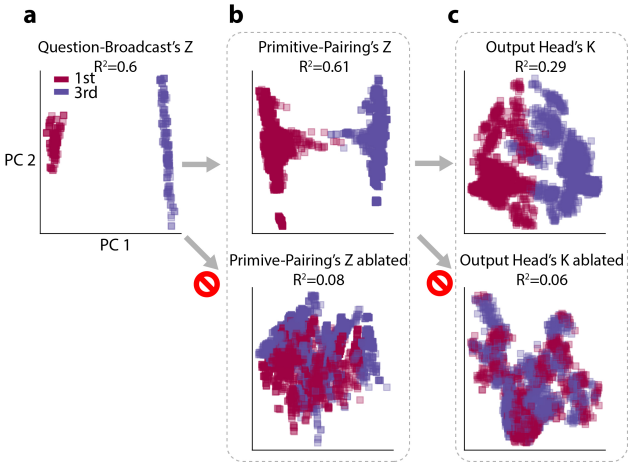

Figure 7: Principal Components Analysis (PCA) of token embeddings, colored by their associated index-in-question. Concretely, for a prompt like '`A S B | A=red | B=blue | ...`', in (a), points are the Z of '`A`' and '`B`' in the Support (`A` labeled `1st`, `B` labeled `3rd`); in (b), points are the Z of '`red`' and '`blue`' in the Support (`red` labeled `1st`, `blue` labeled `3rd`); in (c), points are the K of '`red`' and '`blue`' in the Support (`red` labeled `1st`, `blue` labeled `3rd`). The distinct clusters suggest strong index information. R^2 score quantifies the percentage of total variance explained by the index identity.

3.3.3 The Q-Circuit to the Output Head

Having established the role of the K-circuit, we next investigate where its Q originates. We again relied on *sequential path-patching* to pinpoint which decoder heads ultimately provide the Output Head's Q. We identified **Dec-cross-0.6** as the main conduit for the Q values of the Output Head. Enc-self-1.0 and -1.2 supply positional embeddings that enable the decoder to track primitive symbol's index-on-LHS, thereby completing the QK alignment for correct predictions (Figure 8).

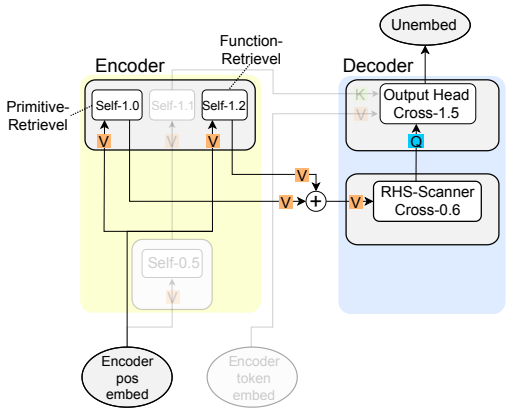

Figure 8: Schematic of the Q-circuit. The Output Head inherits its Q from Dec-cross-0.6, which aggregates positional information passed from Enc-self-1.0 and Enc-self-1.2. The Q-circuit encodes primitive symbols' index-on-LHS.

RHS-Scanner Head (Dec-cross-0.6; Figure 9b) We identify **Dec-cross-0.6** as the dominant contributor to the the Output Head's Q (Figure 9a). Analyzing Dec-cross-0.6's attention patterns reveals that each Q token (from Decoder in the cross-attention) sequentially attends to the color tokens (in the Support Set) on the function's RHS (Figure 9b). For example, the first Decoder token (`SOS`) attends to the first RHS tokens (`purple`, `red`, `yellow`), and the second query token (`red`) attends to the second RHS tokens (`red`, `purple`, `red`), and so on. This iterative scanning mechanism enables the decoder to reconstruct the transformation defined by the function. Hence we call Dec-cross-0.6 the RHS-Scanner Head.

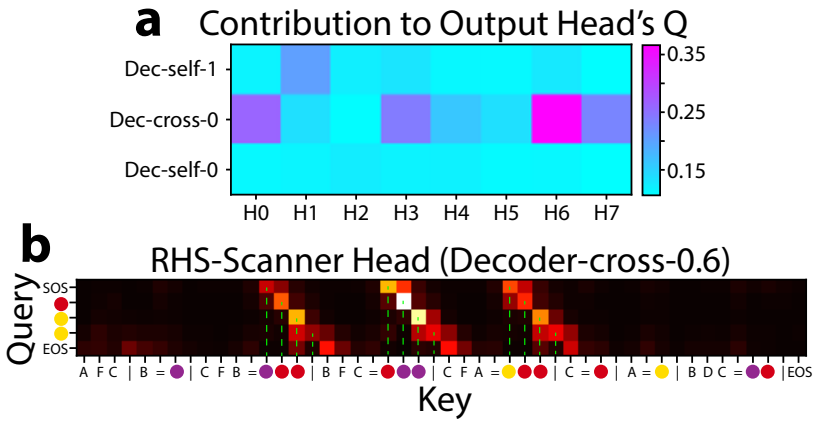

Figure 9: (a) Contribution to Output Head's performance via Q.(b) Attention pattern of Dec-cross-0.6.

Primitive-Retrieval Head (Enc-self-1.0; Figure 10b) and Function-Retrieval Head (Enc-self-1.2; Figure 10c) Next, we looked for critical encoder heads that feeds to the RHS-Scanner Head and finally contributes to the Output Head's Q. Unlike the K-circuit discovery, where "keep-only-one-head" ablations is sufficient, multiple heads appear to contribute partial but complementary information. To isolate their roles, we measured drops in the output head's accuracy when ablating each encoder head individually while keeping the others intact (the "ablate-only-one-head" approach, more discussion in Appendix).

This analysis highlighted **Enc-self-1.0** and **Enc-self-1.2** as critical (Figure 10a). In Enc-self-1.0, within the Support Set, each color token on the RHS attends back to its corresponding symbol on the LHS, inheriting that symbol's token and positional embedding (henceforth the Primitive-Retrieval Head) (Fig. 10b). Meanwhile, Enc-self-1.2 is similar, such that each color token on the RHS attends back to its function symbol on the LHS, passing that token and positional embedding on to the color token (henceforth the Function-Retrieval Head) (Fig. 10c).

Why do the color tokens on the RHS attend back to both kinds of information on the LHS? We reason that if a color token on the RHS were to encode it's primitive symbol's index-on-LHS: for example, in '`...| D=pink | B S D=pink blue pink |...`', `pink` were to encode `3rd` inherited from `D` (`D` is `3rd` in '`B S D`'), the *absolute* position of `D` must be compared with the *absolute* position of the `S` to yield a *relative* position. Now that with the Primitive- and Function-Retrieval Heads, each RHS color token carries two positional references: (1) the associated LHS primitive, and (2) the function symbol, we hypothesize that by comparing these references, the model can infer the primitive symbols' index-on-LHS for each of the associated color tokens on the RHS.

Index-On-LHS Tracing To confirm that our discovered circuit genuinely encodes the index-on-LHS in the Output Head's Q, we conducted three complementary ablation experiments summarized in Figure 11:

- **Retaining only the Primitive- and Function-Retrieval Heads** When all other encoder heads are ablated, the RHS-Scanner Head's Z still carries index-on-LHS that propagate to the Output Head's Q, indicating that these two heads alone provide sufficient index information.
- **Ablating the Primitive- or Function-Retrieval Head individually** Ablating either head disrupts the clustering by index-on-LHS in the RHS-Scanner Head's Z, demonstrating that both heads are necessary to preserve the full index information.
- **Ablating the RHS-Scanner Head (together with Dec-cross-0.0 and -0.3)** These decoder heads share similar attention patterns that track color tokens on the function's RHS. When all three are ablated, clusterings by index-on-LHS are eliminated from the Output Head's Q.

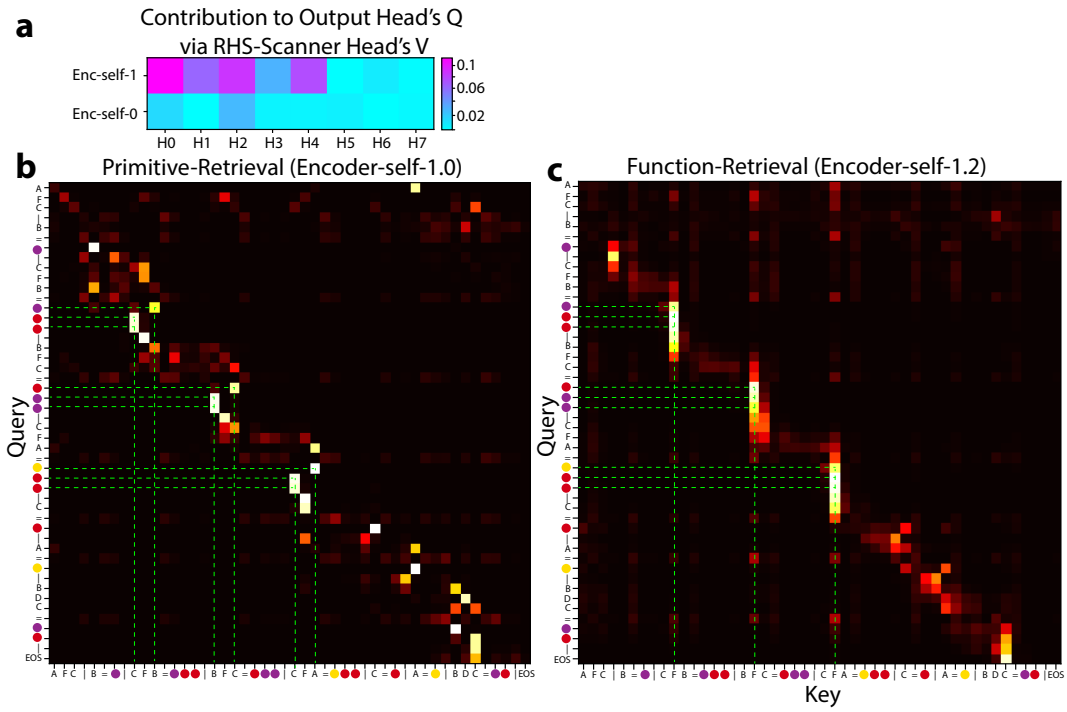

Figure 10: (a) Contribution to Output Head's performance via Q. (b) Contribution to Output Head's performance via the RHS-Scanner's V. (c) Attention pattern of Dec-cross-0.6. (d) and (e) Attention patterns of Enc-self-1.0 and Enc-self-1.2.

Thus, we conclude that the Q-circuit depends on the RHS-Scanner Head to capture the index-on-LHS information supplied by the Primitive- and Function-Retrieval Heads. By aligning these Q signals with the K, the model consistently determines which token to generate next.

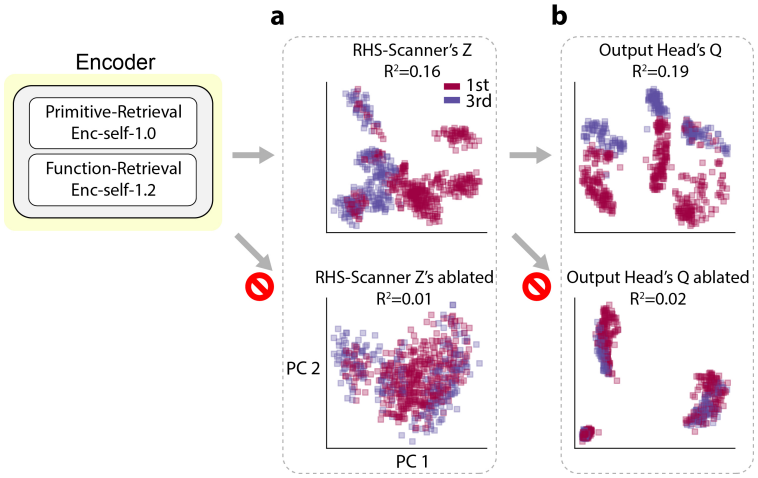

Figure 11: PCA for token embeddings labeled by index-on-LHS. Concretely, for an episode with prompt '`A S B | A=red | B=blue | B S A=red blue red`' and prediction '`SOS blue red blue EOS`', in (a), points are the Z of '`SOS`' and '`blue`' in the decoder input tokens (`SOS` is labeled `3rd`, because `SOS` attends to the `red` on function RHS, and `A` is the `3rd` on the LHS; similarly, `blue` is labeled `1st`); in (b), points are the Q of decoder input tokens (`SOS` is labeled `3rd`, `red` is labeled `1st`). R^2 score quantifies the percentage of total variance explained by the index identity.

3.4 Targeted Perturbation Steers Output

Our circuit analysis indicates that the Output Head's K-circuit encodes the primitive symbols' *index-in-question*, while its Q-circuit encodes their *index-on-LHS*. If the model indeed relies on this positional indexing for token prediction, then perturbing (i.e., swapping) this index information should systematically alter the attention patterns and consequently the model's behavior.

Perturbation Alters Attention Patterns Concretely, consider an input like '`A S B | A=red | B=blue | ...`', where '`red`' inherits `1st` (from `A`) and '`blue`' inherits `3rd` (from `B`) in the K-circuit. If the Q-circuit expects a token with index value of `1st` ('`red`' in this case), swapping these positional embeddings between `A` and `B` at the earliest node (Question-Broadcast Head's V) of the K-circuit—while freezing the Q-circuit—should revert the Output Head's attention from '`red`' to '`blue`' (Figure 12a).

Indeed, performing this targeted swap of positional embeddings caused the Output Head to shift its attention predictably from '`red`' to '`blue`', corresponding precisely to their swapped positions (Figure 12b). A control experiment using random positional embedding shuffles did not produce this systematic attention shift, confirming the causal role of positional indexing in the Output Head's QK alignment.

Perturbation Partially Alters Model Outputs We further assessed whether this targeted perturbation affects the final output logits. Post-perturbation, model accuracy dropped from **94%** to **76%**, remaining above chance but far below the original performance. Correspondingly, the logits for originally correct tokens decreased while those for swapped tokens increased (Figure 12c). The partial logit swap aligns with earlier observations (Figure 4), where Dec-cross-1.1 and -1.3 also provide significant contributions to final token logits. Again, random positional embedding shuffles did not replicate this systematic logit alteration.

Together, these targeted perturbations confirm the causal role of the identified compositional circuit, demonstrating that precise manipulation of internal activations can systematically steer model outputs.

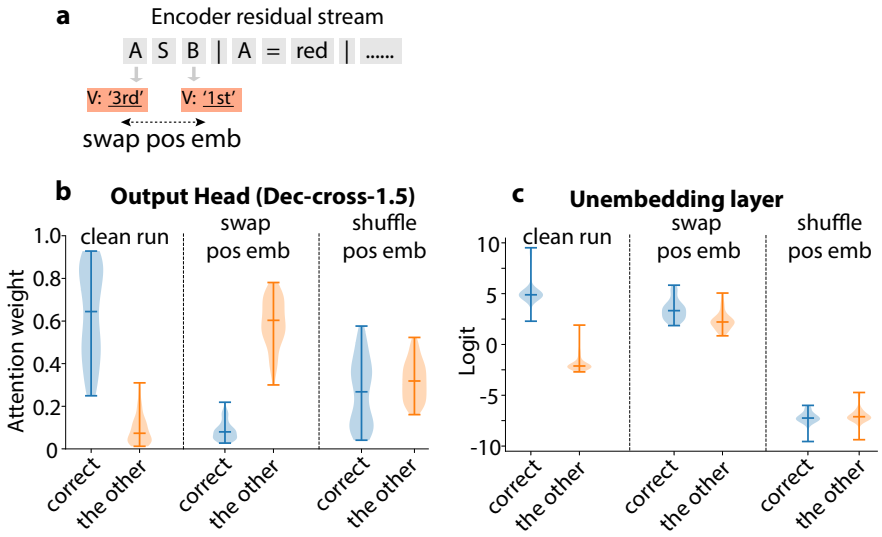

Figure 12: (a) Schematic illustrating the targeted swap of position embeddings. (b) Attention weights from the Output Head comparing the original correct token and swapped token across three conditions: unperturbed (left), targeted position swap (middle), and random shuffle control (right). (c) Similar comparison as (b), but for final output logits rather than attention weights.

4 Discussion

In this work, we investigated how a compact transformer model performs compositional generalization on a synthetic sequence arithmetic task. Using *path-patching* and *causal ablations*, we uncovered a detailed QK

circuit that encodes index information from both the Question and the function's LHS. We further showed that precisely swapping these positional embeddings predictably alters the model's behavior, confirming the causal role of this circuit in supporting compositional generalization. These results demonstrate that even for complex functions, transformers can implement structured and interpretable mechanisms.

4.1 Limitations and Future Work

Model Scale. Our circuit analysis focused on a relatively small transformer. Establishing whether similar interpretable circuits exist in larger models remains an important open question for future work.
Partial Perturbations. Although our targeted activation edits successfully influenced the Output Head's behavior, they did not fully control the predicted tokens. This is likely due to the distributed nature of the mechanism, where multiple heads perform overlapping roles. Coordinated interventions across these redundant heads will require more systematic and automated methods, which we aim to develop in future work.
Specialized Compositional Generalization The identified circuit is brittle and highly specialized to the compositional tasks we defined. Indeed, simply repositioning the *function* symbol (e.g., '`A S B`' to '`S A B`') completely disrupts performance to chance level, which indicates that the model always assumes the *function* symbol to be on the `2nd` position, lacking flexibility in symbol recognition. More flexible compositionality involving multiple or recursive functions clearly requires more complex circuit mechanisms, which remains an important direction for future exploration.

Despite these limitations, our findings offer new insights for both machine learning and cognitive science. First, we provide a rigorous, end-to-end circuit analysis methodology of a transformer solving a complex task. We demonstrate that such interpretability can guide targeted steering of model output through activation-level edits. This circuit-tracing workflow can serve as a template for future interpretability efforts across different transformer architectures and tasks.

Second, we identify two high-level principles that underlie successful compositional generalization in this model: (1) token identity (content) and token position (slot) are represented in a disentangled fashion, and (2) *functions* operate by transforming positional slots, abstracting away from token identities. These principles generate testable hypotheses for other neural architectures—for example, whether similar representational signatures exist in recurrent architectures and diffusion models when solving this type of composition. They also offer a guide for detecting mechanisms for other types of compositional problems.

We hope this research motivates targeted experiments and analyses on compositional generalization in cognitive science, and inspires further work on the mechanistic foundations of compositionality in large-scale models across broader task domains.

References

Adithya Bhaskar, Alexander Wettig, Dan Friedman, and Danqi Chen. Finding transformer circuits with edge pruning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, November 2024.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. Sparks of artificial general intelligence: Early experiments with GPT-4. *arXiv [cs.CL]*, March 2023.

Arthur Conmy, Augustine N Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability. *arXiv [cs.LG]*, April 2023.

DeepSeek-AI, Aixin Liu, Feng, and Others. DeepSeek-V3 technical report. *arXiv [cs.CL]*, December 2024.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds,

Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. A mathematical framework for transformer circuits. `https://transformer-circuits.pub/2021/framework/index.html`, 2021. Accessed: 2025-2-4.

Jerry A Fodor. *The language of thought*. The Language and Thought Series. Harvard University Press, London, England, July 1979.

D Gentner. Structure-mapping: A theoretical framework for analogy. *Cogn. Sci.*, 7(2):155–170, June 1983.

Nicholas Goldowsky-Dill, Chris MacLeod, Lucas Sato, and Aryaman Arora. Localizing model behavior with path patching. *arXiv [cs.LG]*, April 2023.

Michael Hanna, Ollie Liu, and Alexandre Variengien. How does GPT-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. In *Thirty-seventh Conference on Neural Information Processing Systems*, November 2023.

Stefan Heimersheim and Jett Janiak. A circuit for python docstrings in a 4-layer attention-only transformer. 2023.

Felix Hofstätter. Explaining the transformer circuits framework by example. 2023.

Keith J Holyoak and Paul Thagard. *Mental Leaps: Analogy in creative thought*. The MIT Press, December 1994.

Aliyah R Hsu, Georgia Zhou, Yeshwanth Cherapanamjeri, Yaxuan Huang, Anobel Y Odisho, Peter R Carroll, and Bin Yu. Efficient automated circuit discovery in transformers using contextual decomposition. *arXiv [cs.AI]*, June 2024.

Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. Compositionality decomposed: how do neural networks generalise? *arXiv [cs.CL]*, August 2019.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv [cs.LG]*, January 2020.

Brenden M Lake and Marco Baroni. Human-like systematic generalization through a meta-learning neural network. *Nature*, October 2023.

LawrenceC, Adrià Garriga-alonso, Nicholas Goldowsky-Dill, ryan_greenblatt, Ansh Radhakrishnan, Buck, and Nate Thomas. Causal scrubbing: a method for rigorously testing interpretability hypotheses [redwood research]. `https://www.lesswrong.com/posts/JvZhhzycHu2Yd57RN/causal-scrubbing-a-method-for-rigorously-testing`, 2022. Accessed: 2025-2-5.

G F Marcus, S Vijayan, S Bandi Rao, and P M Vishton. Rule learning by seven-month-old infants. *Science*, 283(5398):77–80, January 1999.

nostalgebraist. interpreting GPT: the logit lens. `https://www.alignmentforum.org/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens`, 2020. Accessed: 2025-2-5.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. In-context learning and induction heads. *arXiv [cs.LG]*, September 2022.

Daking Rai, Yilun Zhou, Shi Feng, Abulhair Saparov, and Ziyu Yao. A practical review of mechanistic interpretability for transformer-based language models. *arXiv [cs.AI]*, July 2024.

Clayton Sanford, Daniel Hsu, and Matus Telgarsky. Transformers, parallel computation, and logarithmic depth. *arXiv [cs.LG]*, February 2024.

Ashish Vaswani, Noam M Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Neural Inf Process Syst*, 30:5998–6008, June 2017.

Kevin Ro Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. In *The Eleventh International Conference on Learning Representations*, September 2022.

Mingze Wang and Weinan E. Understanding the expressive power and mechanisms of transformer for sequence modeling. *arXiv [cs.LG]*, February 2024.

Dylan Zhang, Curt Tigges, Zory Zhang, Stella Biderman, Maxim Raginsky, and Talia Ringer. Transformer-based models are not yet perfect at learning to emulate structural recursion. *Trans. Mach. Learn. Res.*, 2024, January 2024.

Zhongwang Zhang, Pengxiao Lin, Zhiwei Wang, Yaoyu Zhang, and Zhi-Qin John Xu. Complexity control facilitates reasoning-based compositional generalization in transformers. *arXiv [cs.CL]*, January 2025.

A Appendix

A.1 Related Work

Transformer circuit interpretation. Mechanistic interpretability of transformers began with analysis of simplified models, identifying attention heads as modular components that implement specific functions. In their seminal work, Elhage et al. (2021) and Olsson et al. (2022) introduced "induction heads" as critical components for in-context learning in small attention-only models. These heads perform pattern completion by attending to prior token sequences, forming the basis for later work on compositional generalization. Case studies have dissected transformer circuits for specific functions, such as the 'greater than' circuit (Hanna et al., 2023), the 'docstring' circuit (Heimersheim & Janiak, 2023), the 'indirect object' circuit (Wang et al., 2022), and the 'max of list' circuit (Hofstätter, 2023). These case studies successfully reverse-engineered the transformer into the minimal-algorithm responsible for the target behavior.

To facilitate identification of relevant circuits, researchers have proposed circuit discovery methods such as logit lens (nostalgebraist, 2020), path patching (Goldowsky-Dill et al., 2023), causal scrubbing LawrenceC et al. (2022). For large-scale transformers, automated circuit discovery methods are also proposed (Conmy et al., 2023; Hsu et al., 2024; Bhaskar et al., 2024). So far, transformer interpretability work still requires extensive human efforts in the loop for hypothesis generation and testing. We point to a review paper for a more comprehensive review (Rai et al., 2024).

Compositional generalization in transformers. In their study, Hupkes et al. (2019) evaluated compositional generalization ability on different families of models, and found that transformers outperformed RNN and ConvNet in systematic generalization, i.e., recombination of known elements, but still incomparable to human performance. Zhang et al. (2024) pointed out that transformers struggle with composing recursive structures. Recently, Lake & Baroni (2023) showed that after being pre-trained with data generated by a 'meta-grammar', small transformers (less than 1 million parameters) can exhibit human-like compositional ability in novel in-context learning cases. This is in line with the success of commercial large language models (LLM) in solving complex out-of-distribution reasoning tasks (Bubeck et al., 2023; DeepSeek-AI et al., 2024), where compositional generalization is necessary.

Several studies highlighted factors that facilitate transformer's compositional ability. Wang & E (2024) identified initialization scales as a critical factor in determining whether models rely on memorization or rule-based reasoning for compositional tasks. Zhang et al. (2025) revealed that low-complexity circuits enable out-of-distribution generalization by condensing primitive-level rules. (Sanford et al., 2024) identified logarithmic depth as a key constraint for transformers to emulate computations within a sequence. Here, we offer a complementary mechanistic understanding of how transformers perform compositional computations.

A.2 Transformer Model

We adopt an encoder-decoder architecture, which naturally fits the task by allowing the encoder to process the prompt (Question + Support) with bidirectional self-attention and the decoder to generate an output sequence with causal and cross-attention. Specific hyperparameters include:

- Token embedding dimension: $d_{\text{model}} = 128$
- Attention embedding dimension: $d_{\text{head}} = 16$
- Eight attention heads per layer (both encoder and decoder)
- Pre-LayerNorm (applied to attention/MLP modules) plus an additional LayerNorm at the encoder and decoder outputs
- Standard sinusoidal positional embeddings

The encoder comprises two layers of bidirectional self-attention + MLP, while the decoder comprises two layers of causal self-attention + cross-attention + MLP. We train the model by minimizing the cross-entropy loss (averaged over tokens) using the Adam optimizer. The learning rate is initialized at 0.001 with a warm-up phase over the first epoch, then linearly decays to 0.00005 over training. We apply dropout of 0.1 to both input embeddings and internal Transformer layers, and train with a batch size of 25 episodes. All experiments are performed on an NVIDIA A100 GPU.

A.3 Task Structure

In each episode, the *Support Set* and *Question* are concatenated into a single prompt for the encoder, with question tokens placed at the start. Question, primitive assignments, and function assignments are separated by '`|`' tokens, while primitive and function assignments are identified by '`=`'. Overall, there are 6 possible colors and 9 symbols that may serve as either color primitives or function symbols. Each episode contains 2–4 function assignments and 3–4 color assignments.

A function may be a single-argument (`arg func`) or double-argument (`arg1 func arg2`) function. The function's right-hand side (RHS) describes how arguments are transformed, generated by randomly sampling up to length-5 sequences of arguments and mapping them to color tokens. Each prompt ends with an '`EOS`' token. During decoding, the model begins with an '`SOS`' token and iteratively appends each newly generated token until it emits '`EOS`'.

We randomly generate 10,000 episodes for training and 2,000 for testing, ensuring that the primitive and function assignments in testing episodes do not overlap with those in the training set.

A.4 Path Patching

Path patching is a method for isolating how a specific *source node* in the network influences a particular *target node*. It proceeds in three runs:

1. **Clean Run:** Feed the input through the model normally and *cache* all intermediate activations (including those of the source and target nodes).
2. **Perturbed Run:** Freeze all direct paths into the target node using their cached activations from the clean run. For the *source node* alone, replace its cached activation with *mean-ablated* values. Record the new, perturbed activation at the target node.
3. **Evaluation Run:** Supply the target node with the perturbed activation from Step 2, then measure any resulting changes in the model's output. This quantifies how the source node's contribution (altered via mean-ablation) affects the target node's behavior.

Chained Path Patching. When analyzing circuits that span multiple nodes in sequence, we extend path patching in a *chain-like* manner. For instance, to evaluate a chain $A \to B \to C$:

- We first perform path patching on the sub-path $B \to C$ as usual.
- Next, to capture how A specifically influences B, we isolate and record A's effect on B via mean-ablation on all other inputs to B.
- Finally, we patch that recorded activation into B and evaluate its effect on C.

For a chain of length N, we run $N + 1$ forward passes, ensuring the measured impact on the target node reflects only the chained pathway. This approach precisely attributes the model's behavior to the intended sequence of dependencies.

Two Modes of Ablation. To assess how individual heads or nodes contribute to the target node, we use two complementary modes:

1. **Keep-only-one-head:** Mean-ablate all direct paths to the target node except for *one* node, which retains its clean-run activation. If the target node's performance remains stable, this single node is *sufficient* for driving the relevant behavior. However, this method may fail when multiple heads each provide partial information that is only collectively sufficient.
2. **Ablate-only-one-head:** Keep all source nodes from the clean run except one, which is mean-ablated. If performance degrades, that ablated node is *necessary*. However, if the node's information is *redundant* or duplicated across other paths, the target node's performance will not significantly change.

By combining both modes, we identified the putative QK-circuit of the output head. We then validate the circuits by inspecting the information they propagates and causally erasing the information by ablating specific upstream nodes.

A.5 R^2 Score

To quantify how much an activation dataset $\mathbf{Y}$ encodes a particular latent variable $\mathbf{Z}$, we compute a linear regression of $\mathbf{Z}$ (one-hot encoded) onto $\mathbf{Y}$ and measure the explained variance:

$$R^2 = 1 - \frac{SS_{res}}{SS_{total}}.$$

An R^2 value of 1.0 indicates that $\mathbf{Z}$ fully explains the variance in $\mathbf{Y}$, whereas an R^2 near 0.0 implies $\mathbf{Z}$ provides no information about $\mathbf{Y}$.