IN-CONTEXT ALGEBRA

Anonymous authors
Paper under double-blind review

ABSTRACT

We investigate the mechanisms that arise when transformers are trained to solve arithmetic on sequences where tokens are variables whose meaning is determined only through their interactions. While previous work has found that transformers develop geometric embeddings that mirror algebraic structure, those previous findings emerge from settings where arithmetic value tokens have fixed meanings. We devise a new task in which the assignment of symbols to specific algebraic group elements varies from one sequence to another. Despite this challenging setup, transformers achieve near-perfect accuracy on the task and even generalize to unseen algebraic groups. We develop targeted data distributions to create causal tests of a set of hypothesized mechanisms, and we isolate three mechanisms the models learn: commutative copying where a dedicated head copies answers, identity element recognition that distinguishes identity-containing facts, and closure-based cancellation that tracks group membership to constrain valid answers. Unlike the geometric representations found in fixed-symbol settings, our findings show that models develop symbolic reasoning mechanisms when trained to reason in-context with variables whose meanings are not fixed.

1 INTRODUCTION

Much of the performance of language models (LMs) can be attributed to the power of the token embedding, for example pre-encoding the attribute *female* in the embedding for "`Queen`," (Mikolov et al., 2013) or pre-encoding *divisible-by-two* within the embedding of the token "`108`" (Zhou et al., 2024; Hu et al., 2025; Kantamneni & Tegmark, 2025; Nikankin et al., 2025). Yet the hallmark of abstract reasoning is the ability to work with words and symbols whose meaning is unknown ahead of time. What mechanisms can a transformer language model employ if it is unable to pre-encode solutions in the embeddings of the words?

In this work, we devise a simple in-context learning setting where tokens serve as pure variables, acquiring meaning only through their interactions within each sequence. That will allow us to ask: What computational strategies do transformers develop when deprived of meaningful embeddings?

We adopt a familiar arithmetic problem setting, training small transformers to predict answers to arithmetic problems sampled from finite algebraic groups. What makes our setting special is that each token is a variable that can represent any algebraic element: the meaning of each token is only fixed within a single sequence. Unlike previous studies of the emergence of arithmetic reasoning (Power et al., 2022; Zhang et al., 2022; Nanda et al., 2023; Zhong et al., 2023), solving problems in our setting will force models to infer structure solely from observations of contextual relationships rather than relying on pre-encoded solution information within each token.

Surprisingly, we find that models trained on this task develop fundamentally different reasoning strategies than those that have been previously observed when LMs solve arithmetic. Rather than learning geometric representations of a Fourier basis, we find that the model acquires symbolic reasoning mechanisms based on sparse relational patterns. We identify three primary algorithmic strategies the model employs beyond verbatim copying: commutative copying, identity element recognition, and closure-based cancellation. These findings suggest that the kind of reasoning strategies learned by transformers are highly dependent on the task structure, with symbolic rather than geometric reasoning strategies emerging when tokens carry no consistent meaning across contexts.

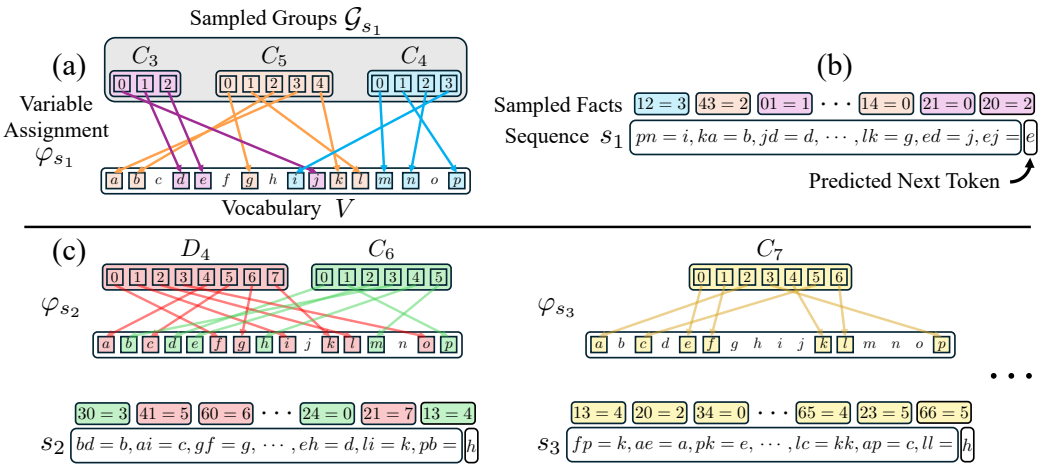


Figure 1: An overview of the data generation process. (a) **Variable Assignment:** We sample a set of algebraic groups and assign the elements of each group a non-overlapping set of vocabulary symbols. (b) **Sequence Generation:** Sampled facts are converted into variable statements via the latent mapping φ_s and concatenated together to form a sequence. (c) **Sample Diversity:** Every sequence is constructed by sampling a new set of groups, defining a new latent mapping, and sampling a new string of facts. The vocabulary symbols are assigned specific meanings within individual sequences, but can take on very different meaning across sequences.

2 Task Description

In this section, we describe our in-context learning task. At a high level, our task involves simulating a mixture of finite algebraic groups.[1] Each task sequence presents several examples of products between elements in a group and the model is trained on the ordinary next-token language modeling objective, with the goal that it will learn to predict the outcome of unseen group products (Figure 1).

More formally, we have a set of m algebraic groups $\mathcal{G} = \{G_1, G_2, \ldots, G_m\}$ that the model is trained to simulate. Recall that for any finite group G, the product of two elements $x, y \in G$ is written as $z = x \cdot y \in G$. We call each such product "$x \cdot y = z$" a *fact*. Training data consists of sequences of k facts written using a vocabulary of variable tokens $v_i \in V$ whose meaning may vary between sequences. In practice, the vocabulary is small, with $N = |V| < \sum_i |G_i|$. A typical sequence s takes the form shown in Equation 1, where individual facts consist of four tokens.

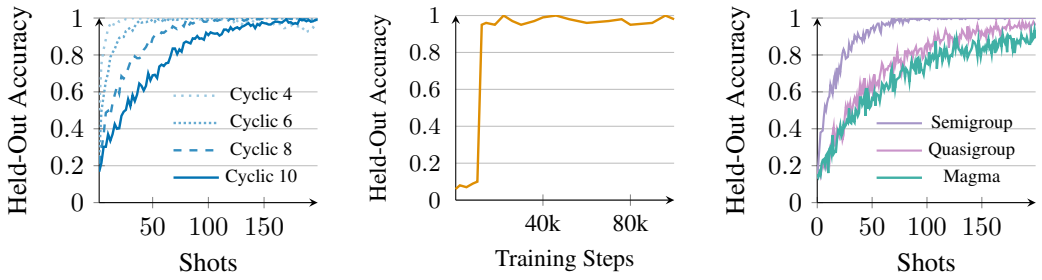
$$s = \text{“}v_{x1}v_{y1} = v_{z1},\ v_{x2}v_{y2} = v_{z2},\ \cdots,\ v_{xk}v_{yk} = v_{zk}\text{”} \tag{1}$$

We describe the positions of a fact with the following terminology: The first element v_{xi}, occupies the "left-slot"; v_{yi} is the "right-slot"; $=$ is the "predictive token"; and v_{zi} is in the "answer-slot".

To generate a training sequence s, we first sample a mixture of groups $\mathcal{G}_s$ from $\mathcal{G}$ with total order less than or equal to the number of variable tokens N. We then construct a latent mapping $\varphi_s : H_s \to V$ that randomly assigns all elements of H_s to tokens in V, where $H_s = \bigcup \mathcal{G}_s$ is the set of all group elements that can appear in s. We ensure that each group in $\mathcal{G}_s$ is assigned a non-overlapping set of variables so that the meaning of each variable within a given sequence is determined by the underlying group structure (Figure 1a).

Given this latent mapping, we then assemble s by sampling facts from $\mathcal{G}_s$, converting them to variable statements via φ_s, and concatenating them together (Figure 1b). The statement "$\varphi_s(x)\varphi_s(y) = \varphi_s(z)$" only appears in s when there is a corresponding valid fact "$x \cdot y = z$" among the sampled groups $\mathcal{G}_s$. Importantly, while the mapping φ_s is fixed within a sequence, it varies between sequences, ensuring that vocabulary tokens $v_i \in V$ act as variables without fixed global meaning.

[1] While not imperative for understanding our task setup, we provide a brief review of relevant topics from group theory in Appendix A.



(a) **Performance increases with context length.** Accuracy increases monotonically with the number of in-context facts. Groups of higher order require more in-context facts to achieve near-perfect performance.

(b) **Phase transition on non-copyable facts.** Accuracy on queries where copying is impossible remains low early in training but then rises abruptly, indicating that the model eventually learns to generalize beyond simple copying strategies.

(c) **Generalization across algebraic structure.** The model generalizes to unseen groups of order 8 and also achieves non-trivial accuracy on non-group structures (semigroups, magmas).

Figure 2: In-context algebra performance.

3 Can transformers learn in-context algebra?

We train transformer models on the in-context algebra task (§2) and evaluate both their in-distribution performance and their ability to generalize across contexts. We report results for one representative model throughout, but observe qualitatively similar patterns across multiple training runs (see Figure 7 in Appendix B). Our main model is a 4-layer transformer with 8 attention heads per layer and hidden size 1024, trained with next-token prediction on sequences of k=200 algebraic facts ($\sim$ 1000 tokens). The training distribution $\mathcal{G} = \{C_3, \ldots, C_{10}, D_3, D_4, D_5\}$ includes cyclic and dihedral groups of up to order 10, with sequences written using N=16 variable tokens plus the special tokens '=' and ','. Because group-to-variable assignments are randomized per sequence, tokens act as placeholders whose meaning must be inferred from context.

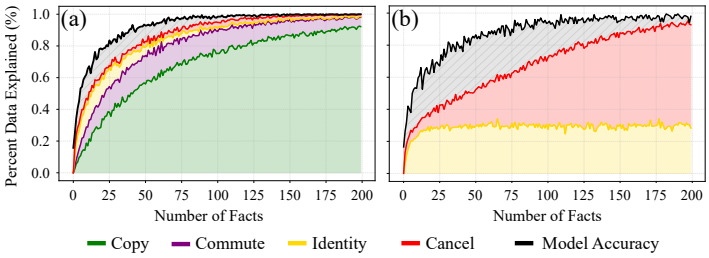
Performance increases with context length. Accuracy increases monotonically with the number of in-context facts k, but the rate of improvement depends on the group order (Fig. 2a). Smaller groups (e.g., $C4$, C_6) reach high accuracy with only a few shots, whereas larger groups (e.g., C_8, C_{10}) require substantially more context to achieve similar performance.

Phase transition on non-copyable queries. At large k, many queries are trivially solvable by copying a previously seen fact (about 90% of queries are copyable at k=200 versus 45% at k=50). To address this, we evaluate the model with held-out data where the final fact "$xy =$" and its commutative pair "$yx =$" never appear elsewhere in the sequence. In this setting, the model still achieves near perfect accuracy, and we observe an abrupt improvement during training (a phase transition) on non-copyable queries (Fig. 2b), suggesting the emergence of strategies beyond verbatim retrieval.

Generalization across algebraic structure. The model also transfers to unseen groups: on the complete set of order-8 groups (including groups excluded from training), the model also achieves near-perfect performance. Interestingly, performance does not collapse on non-group structures such as semigroups or magmas (Fig. 2c). The model still achieves non-trivial accuracy, though generalization remains consistently stronger for true groups.

4 Hypothesizing Model Mechanisms

When analyzing a random in-context algebra sequence, it is possible that multiple algorithms could theoretically produce correct predictions. That can make it challenging to identify which mechanisms the model actually implements. Consider the sequences shown in Equation 2 and Equation 3 that differ only by which fact is bolded. The model could correctly predict "$dp = \boldsymbol{p}$" by either copying the answer from the duplicate fact appearing earlier (i.e., $dp = p$) or by recognizing d is an



Data	$k = 50$	$k = 100$
$\mathcal{D}_{\text{copy}}$	100.0%	100.0%
$\mathcal{D}_{\text{commute}}$	98.0%	99.0%
$\mathcal{D}_{\text{identity}}$	99.0%	100.0%
$\mathcal{D}_{\text{associate}}$	60.2%	56.5%
$\mathcal{D}_{\text{cancel}}$	97.0%	92.0%

(c)

Figure 3: Algorithmic coverage: **(a)** the percentage of training data that can be solved by each mechanism: copying (green), commutative copying (purple), identity recognition (yellow), and closure-based elimination (red) and compare it to the empirical model performance (black). The gray shaded region represents unexplained performance. **(b)** Coverage of sequences where neither form of copying is possible. Identity recognition solves 28.7% of the problems (yellow), and closure-based cancellation can solve an additional 40% (red). **(c)** The model achieves high accuracy on almost all algorithmic distributions (97-100%), except for associative composition (60%).

identity element (i.e., $dc = c$) and applying the identity rule.

$$\text{“}kb = i, dc = c, cl = p, jp = l, \textbf{\textit{dp=p}}, en = e, bb = n, pj = l, dp = \text{”} \tag{2}$$

$$\text{“}kb = i, \textbf{\textit{dc=c}}, cl = p, jp = l, dp = p, en = e, bb = n, pj = l, dp = \text{”} \tag{3}$$

To disambiguate between potential mechanisms, we design five targeted data distributions to test specific algorithms that can solve algebra sequences when a corresponding set of facts is present in the context. We describe each data distribution and the hypothesized algorithm it tests below:

1. **Verbatim Copying ($\mathcal{D}_{\text{copy}}$).** This data tests whether the model copies exact facts from the context. We construct sequences $s \in \mathcal{D}_{\text{copy}}$ to contain at least one duplicate of the final fact.
2. **Commutative Copying ($\mathcal{D}_{\text{commute}}$).** In many groups, knowing that $ab = c$ often implies that $ba = c$. This data tests whether the model copies these commutative facts from the context. We construct $s \in \mathcal{D}_{\text{commute}}$ to contain at least one instance of the commutative fact (e.g., $yx = z$ for final fact $xy =$), and no duplicate facts.
3. **Identity Element Recognition ($\mathcal{D}_{\text{identity}}$).** This data tests whether the model can recognize and apply the identity rule. We construct $s \in \mathcal{D}_{\text{identity}}$ such that the final fact contains the identity element (e.g., $xy = x$) and that at least one prior fact in the context reveals the identity element (e.g., $zy = z$). We remove any duplicate or commutative facts.
4. **Associative Composition ($\mathcal{D}_{\text{associate}}$).** This data tests whether the model can chain fact results together to answer a new fact via associativity. Given a final fact $xy = z$, we construct $s \in \mathcal{D}_{\text{associate}}$ so that it contains a minimum set of facts that would enable a solution via association. For example, the three facts $xg = f$, $gd = y$, $fd = z$ can be composed (i.e., $(xg)d = fd \Rightarrow x(gd) = z \Rightarrow xy = z$) to compute $xy = z$. We make sure to only use triples that do not include duplicate or commutative facts.
5. **Closure-Based Cancellation ($\mathcal{D}_{\text{cancel}}$).** This data tests whether the model can track group membership and appropriately apply the cancellation law to eliminate invalid answers (e.g., $xb = g$ eliminates g as an answer to “$xy =$”). Given a final fact $xy = z$, we construct $s \in \mathcal{D}_{\text{cancel}}$ by including all the facts that share x in the left-slot (e.g., $xb = g$) or y in the right-slot (e.g., $cy = e$), and removing duplicate and commutative facts.

4.1 Measuring Coverage and Performance on Targeted Distributions

We seek to answer two questions: (1) What fraction of algebra sequences can theoretically be solved by these hypothesized algorithms? and (2) Does the model successfully solve sequences that algorithmic strategies can solve when presented with the appropriate facts in-context?

Algorithmic Coverage. To understand the breadth of data that our hypothesized mechanisms might explain, we implement Python equivalents of all five algorithms (Appendix E) and measure their *coverage*, i.e., the percentage of sequences they can theoretically solve. We apply the algorithms

sequentially in the following order: verbatim copying, commutative copying, identity recognition, associative composition, and closure-based cancellation, where each algorithm is only applied to sequences unsolved by previous mechanisms. We compute algorithmic coverage over both random training sequences (Figure 3a) and held-out sequences where neither form of copying is possible (Figure 3b), using 2000 sequences for each case.

We find that verbatim copying can solve a large percentage of the training data, with its area under the curve (AUC) being 67.9% (Figure 3a, green). Commutative copying accounts for an additional 12.1% of cases (purple), with the identity solving 4.2% (yellow) and closure-based cancellation solving 2.7% (red) for total coverage AUC of 86.9%. In contrast, the model accuracy (black) achieves an AUC of 92.4%, indicating that while our hypothesized algorithms might explain much of the model's empirical training performance, they cannot explain everything the model has learned.

When a sequence cannot be solved via copying or commutative copying, we see a very different trend (Figure 3b). In this more challenging setting, the model achieves a slightly lower AUC of 87.3% (black). Identity recognition is able to solve 28.7% of hold-out cases (yellow), and closure-based cancellation can solve an additional 39.1% (red), bringing the total hold-out coverage AUC to 67.8%. Here, the gap between the model's empirical performance and our algorithmic coverage is larger, particularly for algebra sequences with fewer facts.

Model Performance on Subdistributions. We evaluate the model on sequences sampled from each distribution $\mathcal{D}_i$, and report results at $k = 50$ and $k = 100$ facts (Figure 3c). We find the model gets near perfect performance on the four of the five data distributions that we test: verbatim copying (100.0%), commutative copying (99.0%), identity element recognition (100.0%), and closure-based cancellation (97%). However, the model does not get as good of performance on sequences that test for associative composition (60.2%).

5 Causal Verification of Learned Mechanisms

Based on the results in Section 4.1, we perform causal interventions to understand how the model mechanistically implements the following proposed algorithms: (1) verbatim copying, (2) commutative copying, (3) identity element recognition, and (4) closure-based cancellation.

5.1 Causal Interventions

In order to understand the internal computations underlying the model's capabilities, we use causal interventions (Vig et al., 2020; Meng et al., 2022; Geiger et al., 2025) to verify how the model implements the targeted behavior. This is typically done by implicating model components such as attention heads or directions in its activation space (Wang et al., 2023; Geiger et al., 2024; Mueller et al., 2024). Similar to prior work, we quantify the importance of a component via its indirect effect (IE; Pearl, 2001). We compute IE as the change in probability of the target variable token v_{target} under some intervention across a pair of algebra sequences that differ in a meaningful way (s_{clean}, s_{corrupt}). Equation 4 shows how to compute IE for an attention head $a^{(l,h)}$ at layer l, head h, where activations are patched from s_{clean} into s_{corrupt}:

$$\text{IE}(l,h) = P(v_{\text{target}}|a^{(l,h)}_{s_{\text{clean}}} \rightarrow s_{\text{corrupt}}) - P(v_{\text{target}}|s_{\text{corrupt}}) \tag{4}$$

where $a^{(l,h)}_{s_{\text{clean}}} \rightarrow s_{\text{corrupt}}$ indicates activations $a^{(l,h)}$ are being patched from s_{clean} into s_{corrupt}. The average indirect effect (AIE) can be computed over a dataset $\mathcal{D}$ as:

$$\text{AIE}(\mathcal{D}, l, h) = \frac{1}{|\mathcal{D}|} \sum_{\mathcal{D}} (\text{IE}(l,h)) \tag{5}$$

5.2 Copying and Commutative Copying

In this subsection, we investigate how the model implements verbatim and commutative copying. As shown in Section 4.1, a large percentage of our training data ($\sim 80\%$) can either be solved by verbatim copying or commutative copying, and the model achieves high performance (97-100%) when either form of copying is possible (see Figure 3c).

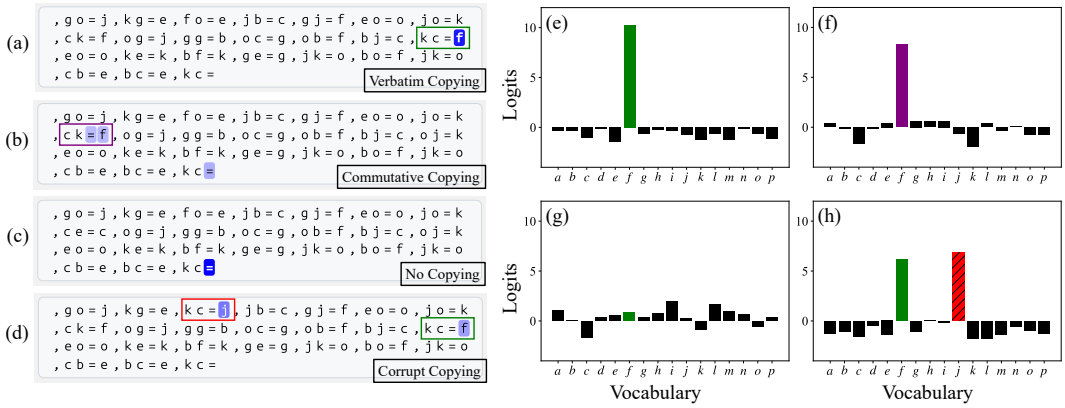


Figure 4: An analysis of copying (§ 5.2). Attention patterns (a-d) and direct logit contributions (e-h) of the copying head (layer 3, head 6) across variations of the same algebra sequence. (a) When verbatim copying is possible, the head attends to the answer-slot of the previous fact "$kc = f$" and (e) directly promotes that token's logit (green). (b) When the exact fact is absent, the head's attention shifts to the answer-slot and predictive token of the commutative fact "$ck = f$" and (f) promotes that token (purple). Note this fact was also in (a) but not attended to, indicating exact facts take precedence over commutative ones. (c) When both exact and commutative facts are absent, the head self-attends and (g) no longer strongly promotes one token. (d) When injecting a matching "corrupted" fact with an incorrect answer ("$kc = j$", red), the head attends to each answer-slot and (h) promotes both variables (green, red).

Verbatim Copying. We search for attention heads responsible for copying the correct answer from the context by computing the average indirect effect (AIE) of each attention head $a^{(l,h)}$ (layer l, head h). We patch the activations of each head $a^{(l,h)}$ from the final predictive token in s_{clean} into the same token position in s_{corrupt}, a randomly sampled sequence where copying is not possible, and measure its IE. We compute $\text{AIE}(\mathcal{D}_{\text{copy}}, l, h)$ over 200 samples from $\mathcal{D}_{\text{copy}}$.

We find a single attention head (layer 3, head 6) with high AIE (0.91) that is primarily responsible for copying, with no other head having an IE higher than 0.08 (Figure 8a). We visualize the attention patterns of this copying head in Figure 4a, and find that it attends to the answer-slot of duplicated facts (shown in green), much like the n-gram heads observed in Akyürek et al. (2024). On $\mathcal{D}_{\text{copy}}$, head 3.6 strongly promotes the logit of the attended-to token (Figure 4e) which can be seen by applying the model's unembedding matrix to the attention head output $U(a^{(l,h)})$. This allows us to understand its output contribution in terms of vocabulary tokens (Nostalgebraist, 2020; Elhage et al., 2021; Dar et al., 2023). The top logit consistently matches the attended-to token (Figure 9).

Commutative Copying. In many groups, knowing that $ab = c$ will also imply that $ba = c$. To investigate how the model implements such commutative copying, we compute the indirect effects of patching attention head activations from sequences in $s_{\text{clean}} \in \mathcal{D}_{\text{commute}}$, to non-copying sequences s_{corrupt}, where neither verbatim nor commutative copying are possible. We find the same pattern: head 3.6 is again the only head with strong IE (0.48) for commutative copying (Figure 8b). In the absence of duplicate facts, head 3.6 attends to the predictive token and answer-slot of the *commutative* fact (Figure 4b) and similarly promotes the attended-to variable token (Figure 4f).

Non-Copying Sequences. When neither copy-inducing fact is present in the context, head 3.6 often self-attends (Figures 4c), not strongly promoting any token (Figure 4g). However, when the query contains an identity fact, we find head 3.6 has an interesting identity demotion behavior (§ 5.3).

Corrupt Copying. While copying the answer-slot of a commutative fact can solve facts for abelian groups, this doesn't work for non-commutative facts. When analyzing the copying behavior of this head on cyclic and dihedral groups separately, we find that more than 97% of the time it promotes the token it attends to, even if that token is the *wrong answer* (see Figure 9b). We illustrate this behavior in Figure 4d, where we inject a duplicate fact with an incorrect answer and show that head 3.6 attends to both duplicates (red, green) and promotes both of their logits (Figure 4h).

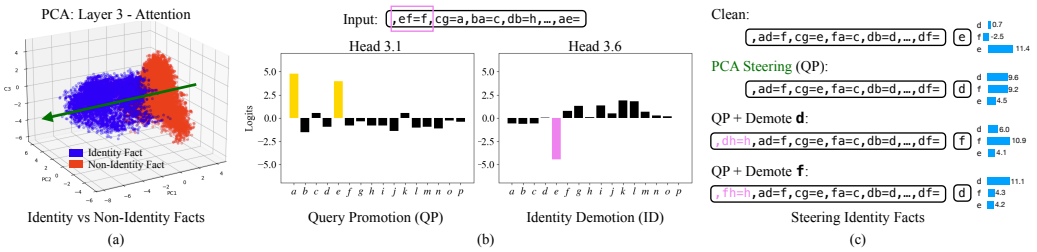


Figure 5: Identity Recognition. (a) PCA decomposition of fact embeddings at the final attention layer reveals a clear separation of identity facts (blue) and non-identity facts (red). (b) Head 3.1 promotes the logits of both variables in the query (a and e), while head 3.6 demotes the logit of the identity variable, e. (c) PCA steering on its own can induce identity behavior, but it promotes both variables in the query to have near equal logits. Inserting a false identity fact for either query variable triggers identity demotion, which along with PCA steering achieves cleaner identity behavior.

5.3 Identity Recognition

Our coverage has analysis revealed that when verbatim and commutative copying are no longer allowed, the identity algorithm can solve close to 30% of all hold-out problems. In this section, we use data from $\mathcal{D}_{\text{identity}}$ to study how the model solves sequences where the query is an identity fact. Recall that an identity element e satisfies $e \cdot x = x \cdot e = x$ for all elements $x \in G$, so that if one variable in the question is known to be the identity, the answer is equal to the other variable.

Our experiments suggest that identity recognition emerges from the interaction of two complementary mechanisms: query promotion, that elevates both variables in the question as potential answers, and identity demotion that suppresses the known identity element. When both mechanisms activate simultaneously, the non-identity token is correctly selected.

Structure from PCA. First, we note that transformer representation reveals a strong signal correlated with the presence of an identity element in the question. To analyze this, we plot attention layer outputs at the predictive token position (the "=" symbol) just before the model predicts an answer. Using PCA on the layer 3 attention outputs (Figure 5a), shows a clear separation between facts containing identity elements (blue) and non-identity facts (red). This separation is invariant to the specific variables in the fact or the underlying group. This suggests the model has learned to recognize and solve identity facts differently from facts that do not contain an identity element.

Query Promotion and Identity Demotion. To analyze the role of the layer 3 attention at predicting identity facts in Figure 5b we use the logit lens (Nostalgebraist, 2020) and find two heads whose logits correlate strongly with identity variables. Head 3.1 promotes both variables in a given fact, serving as a "query promotion" mechanism. This strategy of predicting that the answer is equal to the question is appropriate for problems in which one of the factors is the identity, although on its own it would have the undesirable effect of promoting the identity element itself as the answer.

Head 3.6 acts as an "identity demotion" mechanism, attending to previous identity facts in the context and suppressing the identity token's logit. Combined with the previous strategy, this serves to leave only the non-identity factor as the promoted answer.

Causal Verification. We further have evidence that the dominant PCA direction in representation space causally controls identity recognition. To understand the causal effects, we perform representation steering experiments along this learned direction. When we intervene on the layer 3 attention output of a non-identity fact and steer it toward the identity cluster, the model begins producing equal logits for both query tokens (Figure 5c: i vs. ii).

In addition to query promotion, we can also manipulate the model's identity recognition by introducing false identity facts to influence the identity demotion signal. When we inject a fact incorrectly suggesting one of the query tokens is the identity element, the identity demotion head (3.6) responds by suppressing that token and causes a strong identity prediction (Figure 5c, iii, iv).

On the other hand, when the model is presented with false identity context while the query is a non-identity fact, it typically confuses the prediction. However, if we steer in the negative PCA direction

(away from the identity cluster), the model recovers and correctly predicts the non-identity answer. These steering experiments demonstrate that the learned PCA direction has causal control over the model's identity reasoning, enabling us to both enhance and suppress identity recognition.

5.4 CLOSURE-BASED CANCELLATION

The closure-based cancellation algorithm is a combination of two key submechanisms: (i) tracking which variables belong to the same algebraic group (i.e. the *closure*), and (ii) systematically eliminating invalid answers using the *cancellation law*, which implies that for elements $x, y, z \in G$, if $y \neq z$ then $xy \neq xz$ and $yx \neq zx$.

We hypothesize the algorithm can be understood at a high-level as computing the difference of two sets: S_{closure} - S_{cancel}. Consider the sequence s sampled from $\mathcal{D}_{\text{cancel}}$ shown in Equation 6. For the final query $pe =$, the closure contains all elements that have appeared in facts involving p or e, i.e., $S_{\text{closure}} = \{p, e, f, a, n\}$. The cancellation law then eliminates candidates: p (from $pf = p$), n (from $ee = n$), f (from $ae = f$), and e (from $pp = e$), leaving a as the only valid answer (i.e, $S_{\text{cancel}} = \{p, n, f, e\}$).

$$\text{“}pf = p, ee = n, pf = p, pf = p, ae = f, pp = e, pf = p, pn = f, pp = e, pe = \text{”} \tag{6}$$

We use causal interventions to determine how the model implements these two submechanisms. Our analysis reveals evidence of both a closure subspace, used to promote the logits of variables in the same group, and an elimination subspace that demotes answers based on facts present in the context.

Closure Submechanism. Because we trained our model autoregressively at all sequence positions, the closure submechanism emerges naturally: when predicting the right-slot of a fact like $ab =$, the model must identify which variables could plausibly follow a. These are precisely the elements that belong to the same group (i.e. the closure). In fact, when we analyze the model's predictions at left-slot positions, we find nearly uniform logits across all elements previously associated with that variable, confirming the model has learned how to compute group closure (Figure 10).

To investigate this mechanism causally, we sample counterfactual pairs (s, s') from $\mathcal{D}_{\text{cancel}}$ that have different closure and elimination sets (S_i, S_i'). The pairs are constructed so that when intervening between sequences, a counterfactual answer will arise from applying the set difference operation $(S_{\text{closure}} - S'_{\text{cancel}}) = v_{\text{CF}}$. Inspired by previous work (Geiger et al., 2024; Prakash et al., 2025) showing subspaces can encoding high-level causal variables, we train a subspace W to capture the model's representation of this closure set. To do this, we perform subspace-level patching from s_{clean} into s_{corrupt} (s'), and train W to maximize the likelihood of producing the expected output v_{CF} that would arise if the subspace represented the model's closure set (Equation 7), based on our set difference model.

$$P(v_{\text{CF}} | (W a^l_{\text{clean}} + (I - W) a^l_{\text{corrupt}}) \rightarrow a^l_{\text{corrupt}}) \tag{7}$$

We can measure its accuracy on how often the model's predicted answer under intervention matches the expected counterfactual target. We train a 32-dimensional W on the model's layer 3 attention output a^l, and find that it can achieve good intervention accuracy (99.8%) after only ten epochs of 1000 data pairs (Figure 11). Details about how we construct W are in Appendix D.

For the closure subspace, we train probes (Alain & Bengio, 2017) to understand what the subspace has learned, and how it represents variables. We train a probe to detect the presence of a variable within the subspace, when it is in the group closure or not. We find probes are able to identify with high accuracy when a variable is in the closure subspace (97-99%), and that these variable-level probes partially align with the model's unembedding matrix (Figure 12), furthering evidence that the closure subspace promotes variables it has seen before in the context.

Cancellation Submechanism. To understand the cancellation submechanism, we train a subspace using a similar construction to the above, but vary the patching setup. If this new subspace W' captures the elimination set, then it should generate the counterfactual answer arising from the opposite set difference, where the closure comes from the corrupt sequence s', and the elimination set comes from the clean sequence s: $(S'_{\text{closure}} - S_{\text{cancel}}) = v'_{\text{CF}}$.

We similarly train this subspace and find it also achieves high intervention accuracy, indicating it successfully represents the elimination set. Intervening in this subspace, we can eliminate arbitrary variables, so that they do not become answers from the model.

6 PHASE TRANSITIONS IN THE ACQUISITION OF ALGEBRAIC REASONING

We find that the model undergoes distinct phase transitions (Appendix: Fig. 6). Across seeds and configurations (2-6 layers, 4-16 heads), the same sequence of stages recurs, each marked by a sharp drop in loss. The earliest ability to emerge is **group closure**: the model learns that combining two elements always yields another valid group element. This appears in left-slot predictions of equations like $ab{=}c$, where the model distributes probability nearly uniformly across all valid candidates. This is followed by **contextual copying**, first reproducing facts verbatim and then extending to commutativity-aware copying (ba after ab in abelian groups).

Later stages emerge more gradually. The model develops **identity recognition**, steadily improving on identity-related facts and forming distinct representation clusters for identity elements. In parallel, it acquires **elimination reasoning**, applying cancellation laws and closure constraints to rule out inconsistent candidates (e.g., in "$ax =?$", excluding elements already paired with a). Unlike closure and copying, these abilities do not show sharp transitions but appear jointly, suggesting they build on copying: once the model can retrieve and recombine facts, it can also infer identities and apply elimination strategies.

7 RELATED WORK

Arithmetic as a testbed for interpretability. Arithmetic tasks have long served as controlled settings for mechanistic interpretability. In the *grokking* setup, small transformers trained on modular arithmetic first memorize training data before converging to interpretable, generalizing solutions with periodic embeddings (Power et al., 2022; Nanda et al., 2023; Zhong et al., 2023; Stander et al., 2024). Pretrained LLMs exhibit similar periodic structure in their number embeddings (Zhou et al., 2024; Hu et al., 2025; Kantamneni & Tegmark, 2025; Nikankin et al., 2025), enabling modular arithmetic without explicit training. However, these works assume fixed symbol meanings. More closely related to our setting, He et al. (2024) show that transformers trained on multiple permutations of a single group develop hierarchical "circle-of-circles" embeddings, while Zhong & Andreas (2024) demonstrate that models with trained embeddings, but otherwise random weights can still implement algorithmic solutions. We extend this line by removing fixed meanings altogether, requiring models to solve problems where token referents vary arbitrarily between sequences.

Mechanisms of in-context learning. The ability of transformers to learn from demonstrations has been attributed to several mechanisms. Early work identified *induction heads* that implement copying via prefix matching (Elhage et al., 2021; Olsson et al., 2022), while theory has framed ICL as Bayesian inference (Xie et al., 2022) or gradient-descent-like adaptation (von Oswald et al., 2023). More recent studies found *function vectors* that capture task-level structure (Todd et al., 2024; Hendel et al., 2023), and showed that token representations can flexibly adapt to context (Park et al., 2025; Gopalani et al., 2024; Wurgaft et al., 2025; Minegishi et al., 2025).

Symbolic reasoning and causal interpretability. Neural systems have long been studied as potential mechanisms for symbol manipulation, from Tensor Product Representations (Smolensky, 1990) and Holographic Reduced Representations (Plate, 1995) to recent cognitive-science studies of emergent symbolic reasoning in modern networks (Yang et al., 2025; Swaminathan et al., 2023). More recently, mechanistic interpretability also started mapping internal reasoning circuits in transformers (Li et al., 2023; Brinkmann et al., 2024; Prakash et al., 2024; Saparov et al., 2025; Wu et al., 2025), often using causal interventions (Mueller et al., 2024; Geiger et al., 2024; 2025).

8 CONCLUSION

Our findings challenge the hypothesis that geometric embeddings are the primary mechanism by which transformers solve algebraic problems. In our setting, when tokens carry no fixed meaning, we have analyzed the mechanisms learned by transfornmer LMs in detail and found that the models abandon geometric strategies entirely and develop symbolic mechanisms instead. Our observations suggests that the geometric representations observed in previous arithmetic studies may be artifacts of fixed symbol meanings rather than universal computational principles. Understanding when and why models choose different computational strategies remains an important open question for future interpretability work.

Ethics Statement

This paper aims to advance the foundational understanding of in-context learning and transformers. While such research may influence future model development and deployment, we cannot meaningfully anticipate these downstream impacts within the scope of this work.

References

Ekin Akyürek, Bailin Wang, Yoon Kim, and Jacob Andreas. In-context language learning: Architectures and algorithms. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=3Z9CRr5srL.

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes, 2017. URL https://openreview.net/forum?id=ryF7rTqgl.

Jannik Brinkmann, Abhay Sheshadri, Victor Levoso, Paul Swoboda, and Christian Bartelt. A mechanistic analysis of a transformer trained on a symbolic multi-step reasoning task. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 4082–4102, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.242. URL https://aclanthology.org/2024.findings-acl.242/.

Guy Dar, Mor Geva, Ankit Gupta, and Jonathan Berant. Analyzing transformers in embedding space. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16124–16170, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.893. URL https://aclanthology.org/2023.acl-long.893.

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. *Transformer Circuits Thread*, 2021. URL https://transformer-circuits.pub/2021/framework/index.html.

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

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

Pulkit Gopalani, Ekdeep Singh Lubana, and Wei Hu. Abrupt learning in transformers: A case study on matrix completion. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=O9RZAEp34l.

Tianyu He, Darshil Doshi, Aritra Das, and Andrey Gromov. Learning to grok: Emergence of in-context learning and skill composition in modular arithmetic tasks. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=aVh9KRZdRk.

Roee Hendel, Mor Geva, and Amir Globerson. In-context learning creates task vectors. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 9318–9333, 2023. URL https://aclanthology.org/2023.findings-emnlp.624.

Alston S Householder. Unitary triangularization of a nonsymmetric matrix. *Journal of the ACM (JACM)*, 5(4):339–342, 1958.

Xinyan Hu, Kayo Yin, Michael I Jordan, Jacob Steinhardt, and Lijie Chen. Understanding in-context learning of addition via activation subspaces. *arXiv preprint arXiv:2505.05145*, 2025. URL https://arxiv.org/abs/2505.05145.

Subhash Kantamneni and Max Tegmark. Language models use trigonometry to do addition. *arXiv preprint arXiv:2502.00873*, 2025.

Kenneth Li, Aspen K Hopkins, David Bau, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Emergent world representations: Exploring a sequence model trained on a synthetic task. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=DeG07_TcZvT.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26, 2013.

Gouki Minegishi, Hiroki Furuta, Shohei Taniguchi, Yusuke Iwasawa, and Yutaka Matsuo. In-context meta learning induces multi-phase circuit emergence. In *ICLR 2025 Workshop on Building Trust in Language Models and Applications*, 2025. URL https://openreview.net/forum?id=LNMfzv8TNb.

Aaron Mueller, Jannik Brinkmann, Millicent Li, Samuel Marks, Koyena Pal, Nikhil Prakash, Can Rager, Aruna Sankaranarayanan, Arnab Sen Sharma, Jiuding Sun, Eric Todd, David Bau, and Yonatan Belinkov. The quest for the right mediator: A history, survey, and theoretical grounding of causal interpretability, 2024. URL https://arxiv.org/abs/2408.01416.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=9XFSbDPmdW.

Yaniv Nikankin, Anja Reusch, Aaron Mueller, and Yonatan Belinkov. Arithmetic without algorithms: Language models solve math with a bag of heuristics. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=O9YTt26r2P.

Nostalgebraist. Interpreting GPT: The logit lens. URL https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens, 2020.

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. *Transformer Circuits Thread*, 2022. URL https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html.

Core Francisco Park, Andrew Lee, Ekdeep Singh Lubana, Yongyi Yang, Maya Okawa, Kento Nishi, Martin Wattenberg, and Hidenori Tanaka. ICLR: In-context learning of representations. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=pXlmOmlHJZ.

Judea Pearl. Direct and indirect effects. In *Proceedings of the Seventeenth Conference on Uncertainty and Artificial Intelligence, 2001*, pp. 411–420. Morgan Kaufman, 2001.

T.A. Plate. Holographic reduced representations. *IEEE Transactions on Neural Networks*, 6(3): 623–641, 1995. doi: 10.1109/72.377968.

Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets, 2022. URL https://arxiv.org/abs/2201.02177.

Nikhil Prakash, Tamar Rott Shaham, Tal Haklay, Yonatan Belinkov, and David Bau. Fine-tuning enhances existing mechanisms: A case study on entity tracking. In *Proceedings of the 2024 International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=8sKcAWOf2D. arXiv:2402.14811.

Nikhil Prakash, Natalie Shapira, Arnab Sen Sharma, Christoph Riedl, Yonatan Belinkov, Tamar Rott Shaham, David Bau, and Atticus Geiger. Language models use lookbacks to track beliefs, 2025. URL https://arxiv.org/abs/2505.14685.

Abulhair Saparov, Srushti Ajay Pawar, Shreyas Pimpalgaonkar, Nitish Joshi, Richard Yuanzhe Pang, Vishakh Padmakumar, Mehran Kazemi, Najoung Kim, and He He. Transformers struggle to learn to search. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=9cQB1Hwrtw.

Paul Smolensky. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence*, 46(1):159–216, 1990. ISSN 0004-3702. doi: https://doi.org/10.1016/0004-3702(90)90007-M.

Dashiell Stander, Qinan Yu, Honglu Fan, and Stella Biderman. Grokking group multiplication with cosets. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=hcQfTsVnBo.

Sivaramakrishnan Swaminathan, Antoine Dedieu, Rajkumar Vasudeva Raju, Murray Shanahan, Miguel Lazaro-Gredilla, and Dileep George. Schema-learning and rebinding as mechanisms of in-context learning and emergence. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 28785–28804. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/5bc3356e0fa1753fff7e8d6628e71b22-Paper-Conference.pdf.

Eric Todd, Millicent Li, Arnab Sen Sharma, Aaron Mueller, Byron C Wallace, and David Bau. Function vectors in large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=AwyxtyMwaG. arXiv:2310.15213.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. Investigating gender bias in language models using causal mediation analysis. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 12388–12401. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/92650b2e92217715fe312e6fa7b90d82-Paper.pdf.

Johannes von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent, 2023. URL https://arxiv.org/abs/2212.07677.

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*, 2023. URL https://openreview.net/forum?id=NpsVSN6o4ul.

Yiwei Wu, Atticus Geiger, and Raphaël Millière. How do transformers learn variable binding in symbolic programs?, 2025. URL https://arxiv.org/abs/2505.20896.

Daniel Wurgaft, Ekdeep Singh Lubana, Core Francisco Park, Hidenori Tanaka, Gautam Reddy, and Noah D. Goodman. In-context learning strategies emerge rationally, 2025. URL https://arxiv.org/abs/2506.17859.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An explanation of in-context learning as implicit bayesian inference. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=RdJVFCHjUMI.

Yukang Yang, Declan Iain Campbell, Kaixuan Huang, Mengdi Wang, Jonathan D. Cohen, and Taylor Whittington Webb. Emergent symbolic mechanisms support abstract reasoning in large language models. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=y1SnRPDWx4.

Yi Zhang, Arturs Backurs, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, and Tal Wagner. Unveiling transformers with lego: a synthetic reasoning task. *arXiv preprint arXiv:2206.04301*, 2022. URL https://arxiv.org/abs/2206.04301.

Ziqian Zhong and Jacob Andreas. Algorithmic capabilities of random transformers. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=plH8gW7tPQ.

Ziqian Zhong, Ziming Liu, Max Tegmark, and Jacob Andreas. The clock and the pizza: Two stories in mechanistic explanation of neural networks. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=S5wmbQc1We.

Tianyi Zhou, Deqing Fu, Vatsal Sharan, and Robin Jia. Pre-trained large language models use fourier features to compute addition. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=i4MutM2TZb.

A GROUP THEORY

In this section, we review relevant terms from group theory that are used in our analysis.

A **group** $(G, \cdot)$ is a non-empty set G equipped with a binary operation $\cdot : G \times G \to G$ that satisfies the following properties:

- **Associativity.** For all elements $x, y, z \in G$: $(x \cdot y) \cdot z = x \cdot (y \cdot z)$
- **Identity.** There exists an identity element $e \in G$ such that $e \cdot g = g \cdot e = g$ for all $g \in G$
- **Invertibility.** For each element $g \in G$, there exists an inverse element $g^{-1} \in G$ such that $g \cdot g^{-1} = g^{-1} \cdot g = e$

For notational convenience, we refer to a group $(G, \cdot)$ simply as G.

A group G is called **abelian** (or **commutative**) if for all $x, y \in G$, the following holds: $x \cdot y = y \cdot x$.

Our training data consists of two main families of groups: cyclic groups and dihedral groups.

A **cyclic group** of order n, denoted C_n, consists of all powers of a single generator element: $C_n = \{e, g, g^2, \ldots, g^{n-1}\}$ where $g^n = e$ is the identity. Every cyclic group C_n has the same structure as doing arithmetic modulo n. For example, in C_5, multiplying group elements (e.g., Equation 8) works exactly like adding numbers mod 5 (e.g., Equation 9):

$$g^3 \cdot g^4 = g^2 \tag{8}$$

$$\equiv 3 + 4 = 2 \pmod 5 \tag{9}$$

A **dihedral group** D_n is the group of symmetries of a regular n-gon (square, hexagon, etc.), with order $|D_n| = 2n$. Its elements consist of n rotations and n reflections. We note that all cyclic groups are abelian, and dihedral groups are non-abelian for $n \geq 3$.

An important consequence of group structure is the **cancellation law**, which states that we can "cancel" common terms in equations. Specifically, for any group G with elements a, b, c:

- *Left cancellation:* If $ab = ac$, then $b = c$
- *Right cancellation:* If $ba = ca$, then $b = c$

Equivalently (by contrapositive): if $y \neq z$, then $xy \neq xz$ and $yx \neq zx$. This rule guarantees that distinct group elements produce distinct products which helps ground our understanding of the closure-based cancellation mechanism described in Section 5.4.

For completeness, we briefly describe other algebraic structures we test on that lack some subset of group properties:

- A **semigroup** is a set with an *associative* binary operation, but does not require an identity element or inverses.
- A **quasigroup** is a set where equations $ax = b$ and $ya = b$ always have unique solutions for any a, b, but the operation need not be associative. Latin squares are examples of finite quasigroups.
- A **magma** is simply a set equipped with a binary operation, with no other required structural properties.

B Training Details

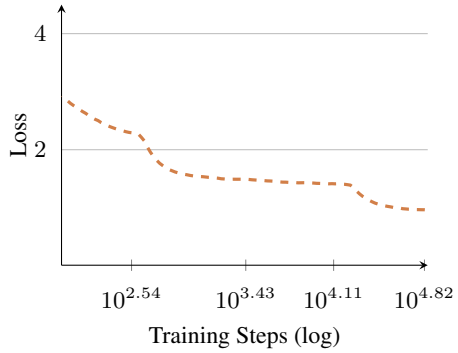


Figure 6: **Phase transitions during training.** Loss over training steps on a log-scale shows three drops that align with discrete behavioral changes.

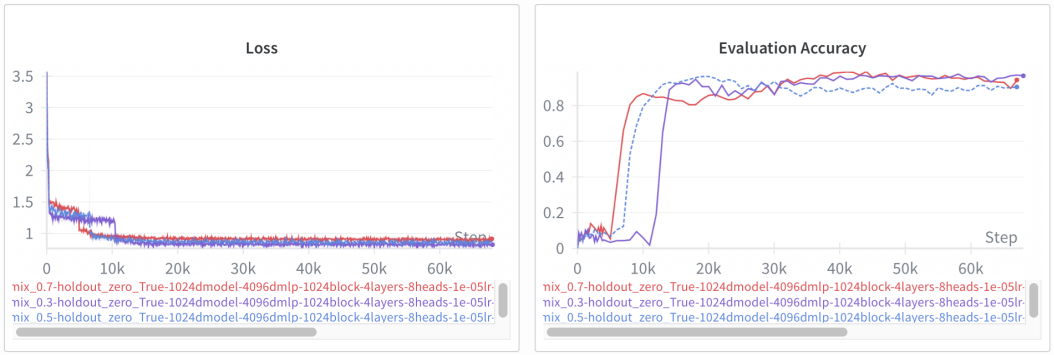


Figure 7: **Consistency across runs.** We observe qualitatively similar patterns across multiple training runs, both in terms of phase transitions and the order in which different capabilities are acquired.

C Additional Results on Copying

In this section, we provide additional results related to the copying and commutative copying mechanisms. Figure 8a shows heatmap of the causal effect of patching from verbatim copying sequences for each attention head in the model. Figure 8b shows a similar heatmap of causal effects for each attention head when patching from commutative copying sequences into no-copy sequences.

Figure 9 shows how often each attention head promotes the correct answer token for copying sequences, when applying the unembedding matrix to its output (i.e., $U(a^{(l,h)})$ (Nostalgebraist, 2020; Elhage et al., 2021). We find that the highest logit promoted by head 3.6 almost always matches the target answer for both verbatim and commutative copying sequences.

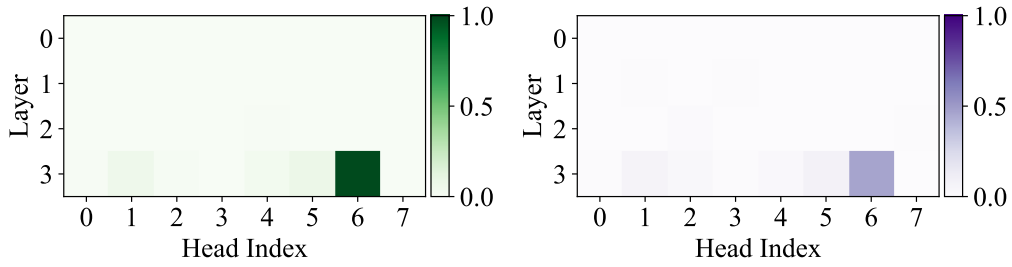


Figure 8: (a) Average causal indirect effect (Equation 5) for each attention head when patching from copying sequences into non-copying sequences, where darker green indicates a stronger change in probability. A single head (layer 3, head 6) is strongly implicated in verbatim copying behavior (AIE=0.91). (b) The same head is implicated when performing patching from commutative copying sequences into non-copying sequences, though the causal effect is slightly weaker (AIE=0.479).

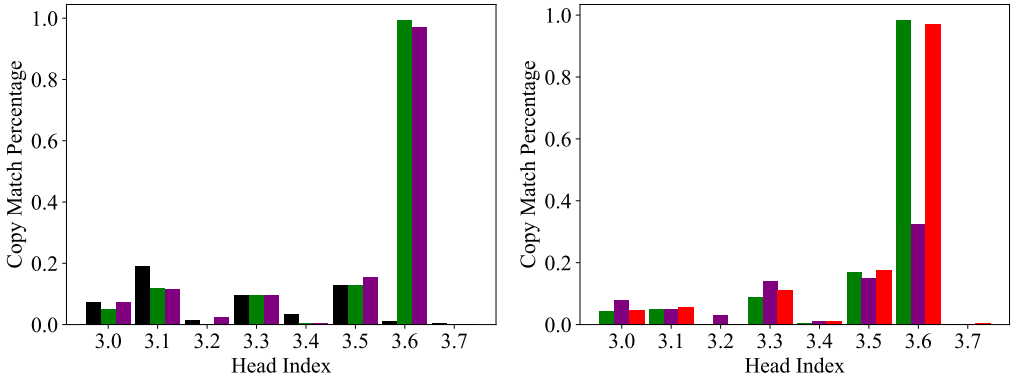


Figure 9: Decoding the output of each attention head at the final layer via the model's unembedding matrix reveals how often an attention head's highest logit matches the correct correct answer on copying sequences. (a) For cyclic groups, we see one head stand out: head 3.6 matches the correct answer more than 99.5% of the time for verbatim copying sequences (green), and 97% of the time for commutative copying sequences (purple), while almost never promoting the correct answer on non-copying sequences (black). (b) For dihedral groups, where not all facts are commutative, we see a similar same trend for exact copying sequences (green), while for commutative copying the head only matches the correct answer 32.5% of the time (purple). However, if we measure whether the highest decoded logit matches the most attended-to token, this happens 97% of the time (red). This suggests head 3.6 is blindly copying whatever it attends to even if that variable is not the correct answer. While this strategy would work for any commutative pair of facts, it cannot solve non-commutative pairs found in dihedral groups.

D Closure and Elimination Subspaces

D.1 Subspace Patching

In this subsection, we describe our how we construct the subspaces we train to characterize the closure and elimination mechanism. We use using a set of learned Householder unit-vectors $v_i \in \mathbb{R}^d$, to construct an orthogonal matrix $Q = H_k H_{k-1} \cdots H_1 \in \mathbb{R}^{d \times d}$ from a series of Householder matrices, $H_i = I - 2 v_i v_i^T$, (Householder, 1958). The first k columns of Q, denoted $Q_k \in \mathbb{R}^{d \times k}$, represent the directions of our intervention subspace. We construct our subspace projection as $W = Q_k Q_k^T$ and perform interventions by mixing information from h_{clean} into h_{corrupt} as: $W h_{\text{clean}} + (I - W) h_{\text{corrupt}} \to h_{\text{corrupt}}$.

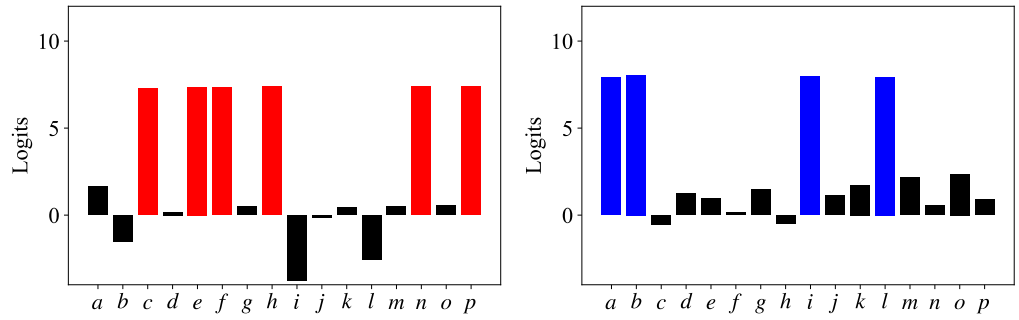


Figure 10: Closure submechanism (§5.4). When predicting the right-slot of a fact, the model produces nearly uniform logits over all variables previously associated with the left-slot in the context. Given the same sequence, but with different left-slot variables (h vs. b), the model produces logits over (a) the six elements connected to h: $\{c, e, f, h, n, p\}$, or (b) the four variables associated with b: $\{a, b, i, l\}$.

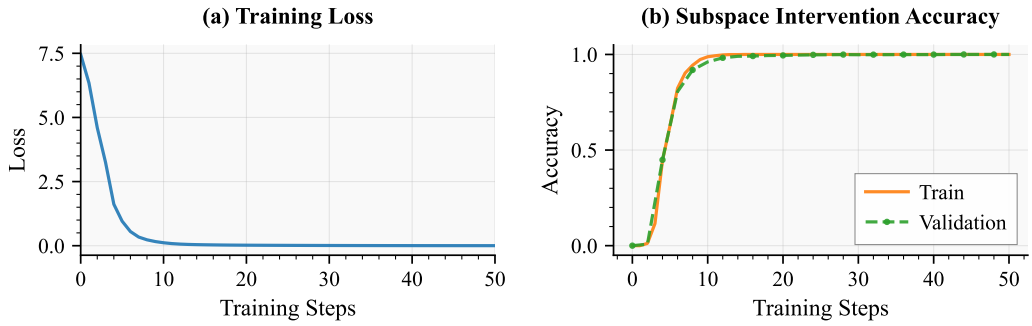


Figure 11: (a) Training loss and (b) intervention accuracy when training a 32 dimensional closure subspace. The subspace quickly achieves 100% intervention accuracy on both the train data and a validation set. We find the learned closure subspace is able to promote the variables of any group.

E Computing Data Coverage

In this section, we provide code implementing how we check for coverage of each algorithm tested in Section 4.

```python
def check_copyable(sequence):
""" sequence (str): A sequence of consecutive algebra facts.
ex: ",fk=i,kn=g,cd=d,kh=c,in=c,nf=h,cg=g,if=n,gf=c,id=h,cg=g,df=g"
"""
    facts = sequence.split(',')
    query = facts[-1]
    return any([fact.split('=')[0] == query.split('=')[0]
                for fact in facts[:-1]])
```

Code Block 1: Python implementation to check if verbatim copying could solve the given algebra sequence.

```python
def check_reverse_copyable(sequence):
""" sequence (str): A sequence of consecutive algebra facts.
ex: ",fk=i,kn=g,cd=d,kh=c,in=c,nf=h,cg=g,if=n,gf=c,id=h,cg=g,df=g"
"""
    facts = sequence.split(',')
    query = facts[-1]
    return any([fact.split('=')[0] == query.split('=')[0][::-1]
                for fact in facts[:-1]])
```

Code Block 2: Python implementation to check if commutative copying could solve the given algebra sequence.

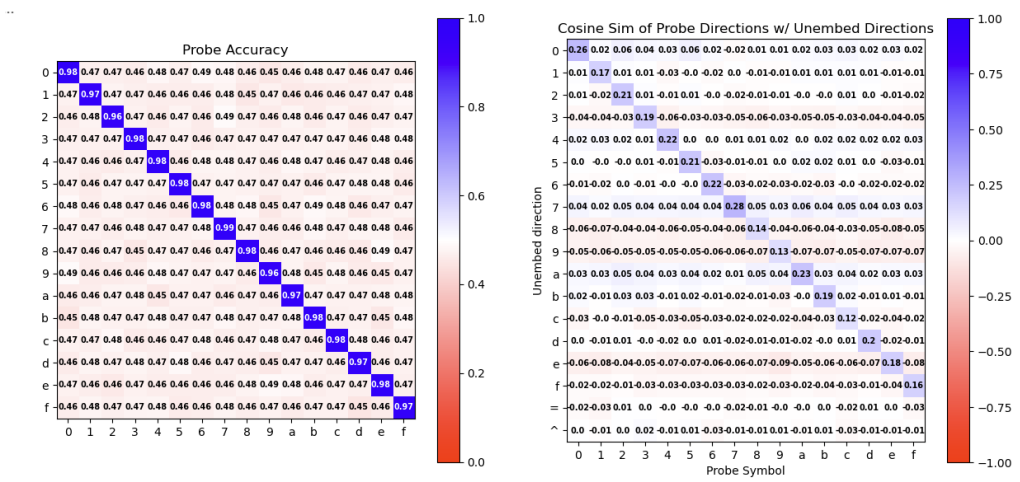


Figure 12: We train probes on the closure subspace that test for the presence of each variable. We find that (a) probes are able to accurately predict when a variable will be in the closure, and (b) the probe directions weakly align with the model's unembedding direction for their respective token.

```python
def check_identity_solvable(sequence):
""" sequence (str): A sequence of consecutive algebra facts.
ex: ",fk=i,kn=g,cd=d,kh=c,in=c,nf=h,cg=g,if=n,gf=c,id=h,cg=g,df=g"
"""
    facts = sequence.split(',')
    query = facts[-1]

    left_identity = [fact[0] == fact[-1] and fact[1] in query.split('=')[0] for fact in
     facts[1:-1]]
    right_identity = [fact[1] == fact[-1] and fact[0] in query.split('=')[0] for fact in
     facts[1:-1]]

    return any(left_identity or right_identity)
```

Code Block 3: Python implementation to check if identity recognition could solve the given algebra sequence.

```python
def check_closure_elimination_solvable(sequence):
""" sequence (str): A sequence of consecutive algebra facts.
ex: ",fk=i,kn=g,cd=d,kh=c,in=c,nf=h,cg=g,if=n,gf=c,id=h,cg=g,df=g"
"""
    facts = sequence.split(',')
    query = facts[-1]

    share_symbol = [fact for fact in facts[-1:1] if query[0] in fact or query[1] in fact]

    share_a_on_left = [fact for fact in facts if fact[0] == a]
    share_b_on_right = [fact for fact in facts if fact[1] == b]

    share_symbol_slots = share_a_on_left + share_b_on_right

    def get_closure_set(facts):
        return set(''.join([x for x in facts]).replace('=', ''))

    set_closure = get_closure_set(share_symbol) # includes answers
    answer_closure = get_closure_set([x[-1] for x in share_symbol_slots])

    return len(set_closure - answer_closure) == 1 and (set_closure - answer_closure) ==
     sequence[-1]
```

Code Block 4: Python implementation to check if a closure-based elimination rule could solve the given algebra sequence.

F USE OF LARGE LANGUAGE MODELS

As per the ICLR 2026 author guidelines, we provide details about our use of large language models (LLMs) in the preparation of this manuscript.

LLMs were primarily used as a general-purpose tool to aid and polish writing, both at the sentence level (e.g., grammar or re-wording sentences), and at the paragraph level (e.g., re-organizing sentences in a paragraph). When considering LLM suggestions, the resulting text went through many subsequent editing rounds. We also used LLMs to answer code-related questions for plotting data used in figures. LLM use did not contribute in any way that we would consider equal to the level of a contributor.