What drives the emergence of symbolic mechanisms in large language models?

Melody Zixuan Li
Princeton Neuroscience Institute
Princeton University
ml8939@princeton.edu

Taylor Webb
Princeton Neuroscience Institute
Department of Psychology
Princeton University
taylor.w.webb@gmail.com

Abstract

Recent work has identified a set of emergent symbolic mechanisms that support abstract reasoning in large language models, but it remains unclear what factors drive the emergence of these mechanisms. To address this question, we trained neural networks from scratch on abstract sequence tasks, varying both architectural and data distributional factors, and investigated the effects of these factors on the emergence of symbolic mechanisms. Using representational, attentional, and causal mediation analyses, we first confirmed that transformer-based language models trained from scratch on our task developed symbolic mechanisms. We then found that, across multiple axes of data diversity, greater diversity produced stronger symbolic signatures. Surprisingly, the same signatures emerged in modified transformer architectures and even multilayer perceptrons, indicating that architectural inductive biases are not necessary for the development of symbolic mechanisms. We also observed consistent patterns in an entity-attribute binding task. Together, these findings identify data diversity, not architecture, as the primary driver of emergent symbolic computation in neural networks.

1 Introduction

Can large language models (LLMs) acquire human-like abstract reasoning, and what mechanisms might support this capacity? Many studies have reported that LLMs exhibit strong performance on abstract reasoning tasks [Webb et al., 2023], but debate persists over whether this reflects genuine structured abstraction or surface-level statistical patterns [McCoy et al., 2024, Lewis and Mitchell, 2024]. Symbolic mechanisms have long been hypothesized as necessary for human-like abstraction [Marcus, 2001], yet it has remained unclear whether and how such mechanisms might be implemented and learned by modern neural network models. Yang et al. [2025] provided direct evidence for a set of emergent symbolic mechanisms supporting abstract reasoning in pretrained LLMs, identifying a circuit composed of three types of attention-heads: *symbol abstraction heads* that map tokens to abstract role variables, *symbolic induction heads* that reason over those variables, and *retrieval heads* that map variables back to tokens. A key open question concerns the factors that drive the emergence of these mechanisms. Specifically, does the emergence of symbolic computation in LLMs depend on specific inductive biases of the transformer architecture from which these models are constructed, or does it depend on the distribution of the data used to train these models, or some combination of both?

In this work, we sought to address these questions by training neural networks from scratch on a controlled in-context abstract sequence task, similar to the task used to identify symbolic mechanisms in LLMs by Yang et al. [2025]. First, we confirmed that transformer language models trained from scratch on this task developed symbol abstraction heads, reproducing some of the key mechanistic signatures identified in pretrained LLMs. We then investigated the extent to which these signatures

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

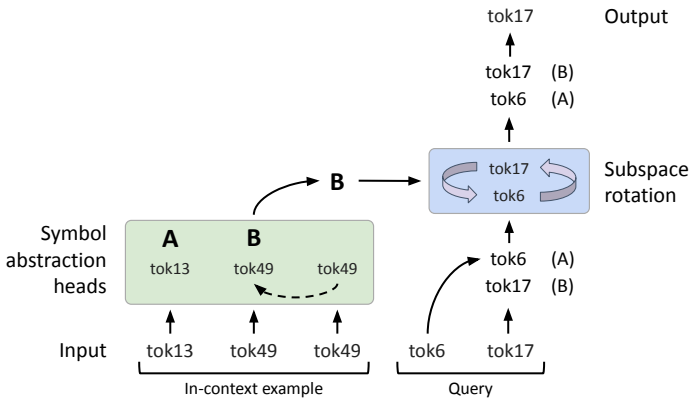


Figure 1: **Illustration of Symbolic Mechanisms Learned by Model.** Problems consist of an in-context example governed by either an ABA or ABB (pictured) rule constructed with arbitrary tokens, followed by an incomplete query example that must be completed according to the same rule. Symbol abstraction heads process the in-context example, identifying the symbol (A or B) associated with the third token. Query tokens are copied into role-specific (A vs. B) subspaces of the residual stream at the final position, and a rule-dependent (ABA vs. ABB) rotation is performed to gate the appropriate token (corresponding to A vs. B) into the output subspace.

are driven by data distributional and architectural factors. We found that both mechanistic and behavioral signatures of symbolic computation scaled with *data diversity*, which we varied along four axes: the number of training examples, the vocabulary size, the number of distinct rules and statistically biased training data. On all four axes, greater diversity produced stronger symbolic signatures. We also found an interaction between data diversity and training dynamics: Models trained with small vocabularies and few rules initially solved the task using a non-symbolic solution based on first-order statistics, but eventually developed symbolic mechanisms with further training, following a pattern similar to grokking [Power et al., 2022, Nanda et al., 2023]. By contrast, we found that the emergence of symbolic mechanisms did not depend on architectural inductive biases. The same mechanistic and behavioral signatures emerged in a set of transformer architectures modified to remove specific inductive biases relevant to symbolic computation and even in a multilayer perceptron (MLP) architecture with much weaker inductive biases. Finally, to test whether these findings extend beyond the abstract sequence task, we introduced a second task that required in-context binding of entities to attributes, where we found that some of the same data-diversity factors governed the emergence of the binding-ID mechanism identified by Feng and Steinhardt [2024], and that a signature of this mechanism emerged in MLPs.

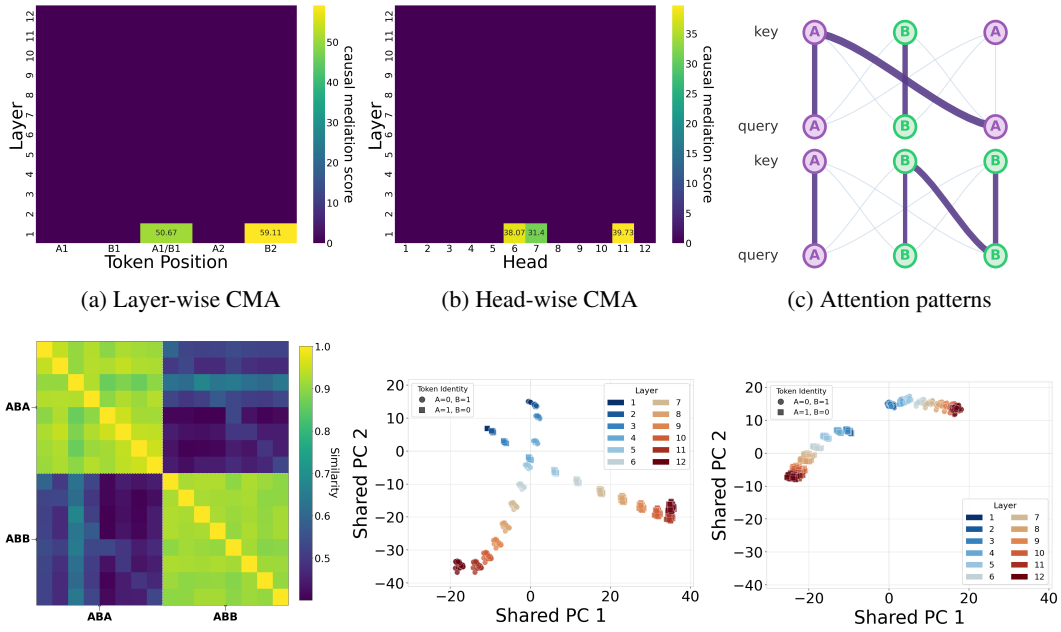
We make the following contributions:

1. We found that transformers trained from scratch on abstract sequence tasks developed a two-stage symbolic mechanism. Symbol abstraction heads inferred the rule, and a rotation mechanism applied it to the query (Section 2).
2. Varying diversity along four axes, we found that it governed whether this mechanism emerged, as shown by causal interventions and out-of-distribution generalization (Section 3).
3. Surprisingly, our experiments revealed that modified transformers, and even plain MLPs, exhibited the same signatures of symbolic computation (Section 4).
4. We further observed that on entity-attribute binding, data diversity likewise determined whether a binding-ID mechanism emerged, while architecture had no impact (Section 5).

2 Analysis of the Solution Learned by the Model

2.1 Architecture of the Learned Solution

We trained transformer language models to perform an abstract sequence task similar to the one used by Yang et al. [2025] to identify emergent symbolic mechanisms in pretrained LLMs. In our



(a) Layer-wise CMA (b) Head-wise CMA (c) Attention patterns

(d) CMA-weighted average RSA (e) PCA: ABA (f) PCA: ABB

Figure 2: **Evidence for Symbol Abstraction Heads and Rotation Mechanism Learned by Model.** **(a)** Layer-wise causal mediation analysis localizes symbolic encoding to layer 1. **(b)** Head-wise causal mediation analysis identifies symbol abstraction heads. **(c)** Query–key attention of the symbol abstraction heads. **(d)** RSA confirms that identified heads encode abstract role identity rather than token surface form. **(e)–(f)** Residual-stream PCA reveals a layer-wise rotation from abstract role representation to output prediction.

task, an in-context example governed by either an ABA or ABB rule, constructed from arbitrary tokens, was followed by a query consisting of two tokens. We trained models to predict the token that would complete the query example according to the rule inferred from the in-context example. We first identified the emergent circuit these models learned to solve the task (Figure 1). The circuit consisted of two major components: 1) *symbol abstraction heads* that computed invariant symbolic representations of the rule governing the in-context example, and 2) a rule-dependent *rotation* operation in the residual stream at the final position that gated the appropriate token (corresponding to A vs. B, depending on the rule) into the output subspace for the model. The following sections describe the evidence for these mechanisms.

2.2 Evidence for Symbol Abstraction Heads

Symbol abstraction heads are attention heads that convert input tokens to invariant symbol-like representations, based on their relations to other tokens [Yang et al., 2025]. They are an emergent instance of the Abstractor architecture proposed by Altabaa et al. [2024]. In the specific case of our task, these heads convert input tokens to invariant representations of abstract symbols (i.e., A or B) based on the relationship to the other tokens within an in-context example (e.g., if the third token is the same as the second token, it will be assigned the symbol B). We identified a set of heads in the first layer of the model that reproduced the key signatures (attentional, representational, and causal) of symbol abstraction heads found in LLMs.

Causal Mediation Analysis. To determine whether our model developed symbol abstraction heads, we performed causal mediation analysis (CMA) [Pearl, 2022, Wang et al., 2022, Todd et al., 2024], patching internal activations between contexts with different rules (ABA vs. ABB) and measuring the effect on the model's outputs. Importantly, context pairs were created in which the rule differed, but the final token in the in-context example was the same (e.g., 'tok13 tok49 tok49', instantiating an ABB rule vs. 'tok49 tok13 tok49', instantiating an ABA rule, but both ending with 'tok49'), ensuring that any effect observed for patching activations at this token position would reflect encoding of an

abstract symbol rather than merely encoding input tokens. When patching activations for the output of all attention heads at a particular layer, we found an effect concentrated in layer 1 for the expected position, which is the third item (A1/B1) in the in-context example (Figure 2a)[1]. When patching activations for the outputs of individual attention heads at this token position, we found a set of three attention heads that produced the expected effect (Figure 2b), thus identifying these heads as candidate symbol abstraction heads.

Attention Analysis. To further confirm the presence of symbol abstraction heads, we analyzed the attention patterns for the identified heads. Consistent with the predicted behavior of symbol abstraction heads, we found that attention was directed from the final token in each in-context example to the previous token that shared the same symbol (i.e., from the third token to the first token for ABA rules, and from the third token to the second token for ABB rules) (Figure 2c).

Representational Similarity Analysis. To test whether the identified heads encode truly invariant symbolic representations rather than token identity, we performed representational similarity analysis (RSA) [Kriegeskorte et al., 2008]. For each identified head, we computed the pairwise similarity between the outputs of that head across pairs of contexts, constructing a similarity matrix reflecting the pattern of similarity across many contexts. Figure 2d shows a weighted average of this matrix, weighted by the CMA scores used to identify these heads. The results confirmed that the outputs of these heads had a high degree of similarity for pairs of contexts that shared the same rule (upper left and lower right quadrants) regardless of the specific tokens used to instantiate those rules, confirming that these heads encoded abstract rules invariantly. The identified attention heads therefore reproduced the key signatures of symbol abstraction heads found in pretrained LLMs.

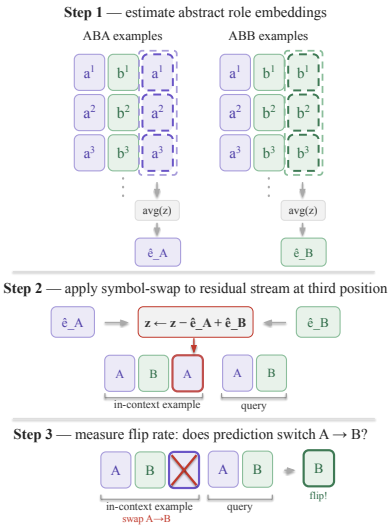
2.3 Evidence for Rule-Dependent Subspace Rotation

After the rule is inferred from the in-context example, it must be applied to the query to generate the final token. Rather than employing symbolic induction heads and retrieval heads as in pretrained LLMs [Yang et al., 2025], we found that this was accomplished in our model via a rule-dependent subspace rotation in the residual stream at the final position. Specifically, the A and B tokens for the query example were stored in role-specific subspaces in early layers, and these subspaces were then rotated in a rule-dependent manner (e.g., the subspaces are rotated only for ABA rules, not for ABB rules) such that the appropriate completion token (A or B, depending on the rule) is available to be read out from a particular subspace in the later layers of the model.

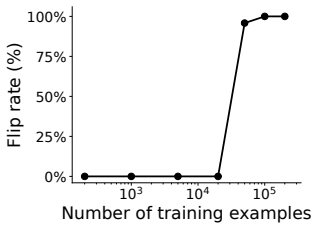
Shared PCA. We confirmed the presence of this rotation-based strategy by performing PCA across all layers jointly on residual-stream activations at the final position, thus constructing a shared subspace in which layer-wise trajectories can be compared directly. Problems were constructed so that the query token was drawn from a fixed pair of tokens, allowing token identity and abstract role (A vs. B) to be dissociated across trials. Figure 2e and Figure 2f show resulting trajectories for ABA and ABB rules, with color encoding layer depth (blue = early, red = late) and marker shape encoding token identity. In early layers, representations clustered by token identity regardless of rule. Across successive layers, the trajectories followed an arc-shaped path in the shared PCA space, converging by the final layer to a configuration that is separable by the correct output token rather than by abstract role. These results are consistent with a near-linear rotation that progressively remaps the abstract role representation into the appropriate output token embedding, depending on the rule.

Decoding role information. We trained linear decoders on the residual stream at the final sequence position and evaluated whether they generalized across rules (ABA to ABB and vice versa). In early layers, decoders transferred based on symbol identity, independent of the rule, whereas in later layers, they transferred based on the rule-dependent correct answer, which differs between the two rules. This transition occurred in the middle layers, consistent with the PCA trajectories. Further details are provided in Appendix B.

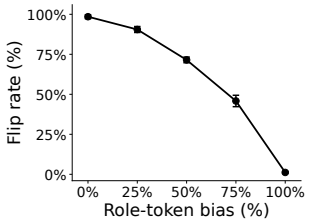
[1] Note that we also found an effect at the final position in the sequence (B2), but this does not unambiguously reflect the abstract rule since the input token also differed at this position.



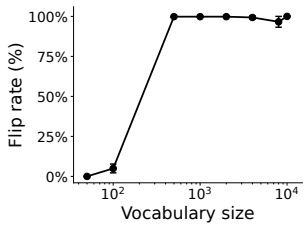
(a) Concept intervention pipeline



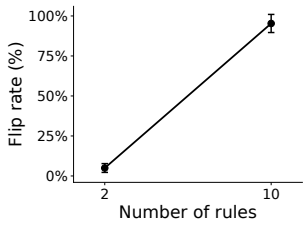
(b) # training examples



(c) role-token bias



(d) vocab. size



(e) # rules

Figure 3: **Assessing the Impact of Data Diversity on Emergent Symbol Processing. (a)** Concept intervention pipeline. Role embeddings $\hat{\mathbf{e}}_A$ and $\hat{\mathbf{e}}_B$ are estimated from residual activations and used to swap the role at a target position. The flip rate is the fraction of prompts whose prediction switches to the answer of the opposite rule. **(b)** Flip rate increases with the number of unique training examples, **(c)** decreases as role-token bias increases, and **(d)** increases with vocabulary size and **(e)** the number of rules. Error bars show the standard deviation across 3 seeds.

3 Data Diversity Drives the Emergence of Symbolic Mechanisms

3.1 Experimental Setup

We varied data diversity along four axes to evaluate its impact on the emergence of symbol abstraction heads. All models used the same hyperparameters as GPT-2 Small [Radford et al., 2019]. The first data diversity axis was the **number of training examples** (from 200 to 200k). The second was **role-token bias** (0% to 100%), defined as the fraction of vocabulary tokens consistently associated with a single role (A or B) during training. These associations were then reversed at test time. The third was **vocabulary size** (V from 50 to 10k). The fourth was the **number of distinct rules** (2 or 10). Across all diversity axes, we hypothesized that increased diversity would reduce the viability of token-level or statistical shortcuts and would therefore encourage models to learn a solution based on genuine relational structure. To compare the emergence of symbolic mechanisms across these settings, we used the concept intervention described below and measured its efficacy. For role-token bias, we additionally evaluated out-of-distribution (OOD) generalization behaviorally. The specific parameter values, data construction procedures, and training details are provided in Appendix A.

3.2 Concept Interventions

Let $\mathbf{z}_{t,\ell}$ denote the residual stream at position t and layer ℓ. Let $t^\star$ be the third token of the in-context example, which is the A symbol in an ABA example and the B symbol in an ABB example. For each layer we define the role embeddings as mean activations at $t^\star$,

$$\hat{\mathbf{e}}_{A,\ell} = \mathbb{E}_{\mathrm{ABA}}[\mathbf{z}_{t^\star,\ell}]\,, \qquad \hat{\mathbf{e}}_{B,\ell} = \mathbb{E}_{\mathrm{ABB}}[\mathbf{z}_{t^\star,\ell}]\,, \tag{1}$$

estimated from n matched prompts of each type. To swap the role on a test prompt, we change the activation at $t^\star$ only. For an ABA prompt,

$$\mathbf{z}_{t^\star,\ell} \;\leftarrow\; \mathbf{z}_{t^\star,\ell} - \hat{\mathbf{e}}_{A,\ell} + \hat{\mathbf{e}}_{B,\ell} \quad \text{for every layer } \ell \text{ simultaneously}, \tag{2}$$

and for an ABB prompt we exchange $\hat{\mathbf{e}}_{A,\ell}$ and $\hat{\mathbf{e}}_{B,\ell}$. When applied to an ABA sequence, the intervention should shift the model's predictions toward those consistent with ABB. We applied the intervention at the third position of the in-context example, the token that determines whether

the example follows ABA or ABB. We measured the *flip rate*, the fraction of trials in which the intervened model outputs the answer that the opposite rule would give. We used the flip rate of this intervention as our primary measure of the extent to which the model possesses invariant, causally relevant symbolic representations. (Figure 3a).

We began by varying the number of unique training examples. Flip rate was near 0% with 200 examples and reached 100% from 100k examples (Figure 3b). In these low-diversity conditions the models did not generalize: training accuracy converged, but test accuracy did not (Appendix Figure 8). To test whether symbolic mechanisms depend on diversity even when models do generalize, we turned to role-token bias. During training, a fraction b of the vocabulary was role-exclusive: half of these tokens appeared only as A and half only as B, while the remaining $(1-b)V$ tokens can fill either role. At b=100% a token's identity fully determines its role, so the training distribution can be solved by learning token–role associations alone. At test time, we reversed the roles of the exclusive tokens. All models generalized in-distribution at every bias level, including 100% bias, with in-distribution accuracy near ceiling (Appendix Figure 9). Flip rate nevertheless fell from 98% at 0% bias to near 0% at 100% bias (Figure 3c). For vocabulary size, flip rate was 0% at V=50, 5% at V=100, and close to 100% from V=500 upward (Figure 3d). At V=100, models generalized but had a low flip rate (consistent with a lack of symbolic mechanisms), so we asked whether training models on more rules would induce a higher flip rate at this vocabulary size. With V=100, flip rate was 5% with 2 rules and 95% with 10 rules (Figure 3e).

Finally, we found an interaction between data diversity and training dynamics. For all experiments described above, we measured flip rate at the point in training when models first reached an accuracy threshold of 95%. For models trained on a small vocabulary size (V=50 and V=100), we found that training models well past the point of convergence eventually led to a sharp increase in the flip rate, consistent with an initial period where models employed a non-symbolic but generalizable solution followed by a later period in which these models eventually developed symbolic mechanisms (Appendix C). Together, these results show that greater data diversity promotes the efficient development of symbolic mechanisms.

3.3 Out-of-Distribution Generalization

For models trained on different levels of role-token bias, we also measured accuracy on test prompts in which the token–role assignments from training were reversed (Figure 4). A model that relies on first-order token statistics should fail on these tests, because the associations it learned now lead to the wrong answer. By contrast, a model that has learned the rule should be unaffected. Consistent with this, we found that models trained on a small vocabulary size (V=100, 2 rules), generalized well when trained with 0% role-token bias (i.e., they generalized well in-distribution), but were unable to generalize when trained with 100% role-token bias (i.e., they could not generalize out-of-distribution). By contrast, models trained with a larger vocabulary size generalized well at all bias levels (i.e., they generalized well both in-distribution and out-of-distribution). These results suggest that, in addition to the mechanistic signature tracked by our concept interventions, data diversity also drove out-of-distribution generalization at the behavioral level, thus conferring a key benefit of symbolic computation.

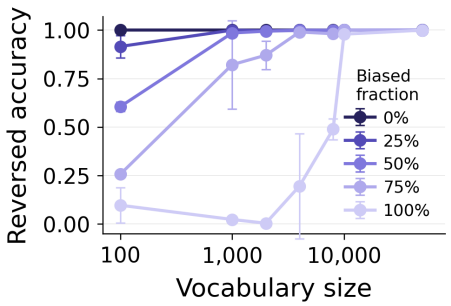
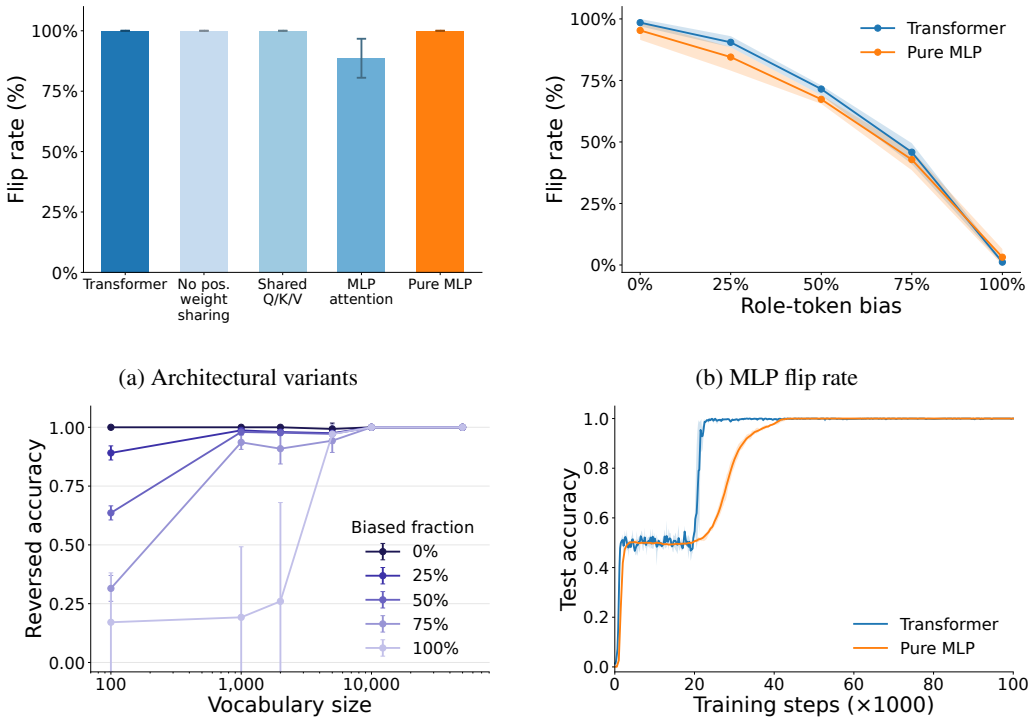


Figure 4: Out-of-distribution generalization

4 Symbolic Mechanisms Do Not Depend on Architectural Inductive Biases

We next asked whether symbolic mechanisms depend on the inductive biases of the transformer architecture. We tested this by evaluating three modified transformers, each ablating a specific architectural feature that seems relevant to symbolic computation, described below.



(a) Architectural variants

(b) MLP flip rate

(c) MLP out-of-distribution generalization

(d) Training dynamics

Figure 5: **Architectural inductive biases do not determine the emergence of symbolic mechanisms. (a)** Flip rate at V=10k for Transformers with specific inductive biases removed, and for MLP models. Error bars show the standard deviation across 3 seeds. **(b)** Flip rate decreases with role-token bias in the same way for the pure MLP and the Transformer. **(c)** MLP models show the same dependence of out-of-distribution (reversed-role) accuracy on vocabulary size and role-token bias as transformers: with high bias and a small vocabulary they fail the reversed test, and reversed accuracy recovers as vocabulary grows. **(d)** Test accuracy over training.

4.1 Modified Transformers

There are several aspects of the transformer architecture that seem relevant to its ability to implement symbolic computation, including 1) the presence of discrete embeddings with shared parameters, potentially enabling a form of compositionality, 2) the use of distinct embeddings for keys, queries, and values, potentially enabling a form of indirection (i.e., the use of one variable to refer to another), and 3) the use of inner products in the attention operation, which are well suited to encoding abstract relations [Webb et al., 2024, Altabaa and Lafferty, 2024]. We evaluated three architectural modifications against a standard transformer control, all trained at $V = 10$k, to test whether specific inductive biases of the transformer architecture are necessary for the emergence of symbolic mechanisms (Figure 5a). The first modified architecture removed weight sharing across sequence positions (No pos. weight sharing), ablating the shared parameters that may enable compositional representations. The second constrained the query, key, and value projection matrices to be identical (Shared Q/K/V), removing the separation that may encourage transformers to implement indirection. The third replaced dot-product attention with a learned scalar attention score computed by a small MLP over concatenated key and query representations (MLP attention), removing an inductive bias that may be helpful for learning abstract relations.

Figure 5a shows the flip rate for our concept intervention at the third token position, measured at the final training checkpoint. Flip rates were high across all of the architectural variants: the control Transformer, the variant without position-specific weights, and the variant with shared Q/K/V all reached 100%, and the MLP attention variant reached about 90% on average, with more

variability across runs. None of these specific Transformer features is therefore necessary for symbolic mechanisms to emerge. This raises the question: do we need attention at all?

4.2 MLP Models

To test whether symbolic mechanisms require attention at all, we trained a plain MLP with no attention. The MLP received as input the concatenation of all token embeddings in the sequence. Residual connections were included between layers, as in the Transformer. The network had 4 hidden layers, and the output layer projected from d_{model} to the vocabulary using only the slice of the hidden state corresponding to the final sequence position (Appendix A.2). This MLP reached a flip rate of 100% in all five runs (Figure 5a), indicating that symbolic mechanisms can emerge in a model without attention.

Concept intervention results. We then asked whether symbolic mechanisms in the MLP depend on data diversity in the same way as they do in the Transformer. Figure 5b shows flip rate as a function of role-token bias for both models, using the manipulation described in Section 3. Flip rate decreased monotonically with bias for both the Transformer and the MLP, indicating that the emergence of symbolic mechanisms is driven by data diversity for both Transformers and MLPs.

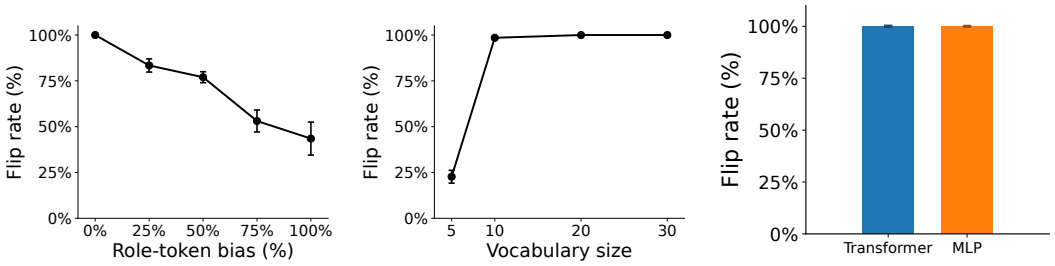
Out-of-distribution generalization. MLP models also showed a qualitatively similar pattern of out-of-distribution generalization (Figure 5c). At $V = 100$, accuracy on reversed-role prompts dropped as role-token bias increased, whereas at $V = 10$k, generalization was perfect across all bias levels. These results indicate that out-of-distribution generalization was driven by data diversity in MLPs just as it was in Transformers.

Training efficiency. Finally, we investigated whether transformers and MLPs show differences in the efficiency with which they developed symbolic mechanisms over the course of training. Figure 5d shows accuracy over training for transformer and MLP models at $V = 10$k. The transformer reached ceiling after roughly 20k steps and the MLP after roughly 40k. This suggests that, while data diversity is the main driver of symbolic computation, the transformer offers a modest training efficiency advantage, accelerating convergence without changing the nature of the final solution.

5 Data Diversity Also Drives the Emergence of Entity-Attribute Binding

To test whether the effect of data diversity extends beyond the ABA/ABB task, we introduced a second task that required *entity–attribute binding*. Each input contains two entity–attribute pairs followed by a query entity, $e_1\, a_1\, e_2\, a_2\, e_q$, and the model must output the attribute bound to the queried entity: a_1 if $e_q = e_1$, and a_2 if $e_q = e_2$. For example, the sequence `Alice friendly Bob tall Bob` has the answer `tall`. In the experiments, entities and attributes are arbitrary tokens, and each sequence uses a new pairing, so the answer cannot be predicted from token identity alone. That is, the model must link each entity to the attribute that follows it in-context. The task is built around the *binding ID mechanism* identified by Feng and Steinhardt [2024], in which a language model binds an entity to its attribute by giving both the same abstract identifier, independent of token identity. Here, this means that e_1 and a_1 would be bound (via vector addition) to one identifier and e_2 and a_2 would be bound to another. The model answers the query by first looking up the identifier of e_q and then retrieving the attribute that's bound to this identifier. As in Section 2, we performed causal mediation analysis and representational similarity analysis to confirm that the binding task also used invariant, symbol-like binding-ID representations rather than token-specific ones; full results are provided in Appendix E.

Mirroring the role-token bias results in Section 3, increasing role-token bias degraded the binding-ID flip rate, from 100% at 0% bias to 44% at 100% bias (Figure 6a). Consistent with the vocabulary-size results in Section 3, the flip rate also depended strongly on vocabulary size, with a flip rate of 23% at $V{=}5$, and flip rates at or near ceiling for all larger vocabularies (Figure 6b). As in Section 4, this effect did not depend on the architectural inductive biases of the Transformer. Both a standard Transformer and an MLP displayed a binding-ID flip rate of 100% (Figure 6c). Together, these results indicate that the effect of data diversity on the emergence of symbolic mechanisms extends beyond the ABA/ABB rule task to a qualitatively different abstract reasoning task.



(a) Flip rate vs. role-token bias (b) Flip rate vs. vocabulary size (c) Transformer vs. MLP

Figure 6: **Data diversity drives the emergence of symbolic binding mechanisms. (a)** Increasing role-token bias results in decreased binding-ID flip rates. **(b)** The binding-ID flip rate increases with vocabulary size. **(c)** Both MLPs and Transformers display high flip rates, indicating that both have developed symbolic binding mechanisms.

6 Related Work

Abstract reasoning in neural networks. A central debate in cognitive science and AI concerns whether neural networks perform genuine structured abstraction or simply exploit surface-level statistical patterns [Marcus, 2001, McCoy et al., 2024, Lewis and Mitchell, 2024, Griffiths et al., 2026]. Webb et al. [2023] demonstrated emergent analogical reasoning in LLMs, though its mechanisms remained unclear. Our work builds on Yang et al. [2025], by asking what factors drive the emergence of a symbolic circuit for abstract reasoning in LLMs.

Mechanistic interpretability and in-context learning mechanisms. Our methods draw on a toolkit for reverse-engineering neural network computations. Causal mediation analysis [Pearl, 2022, Meng et al., 2022, Wang et al., 2022, Todd et al., 2024] localizes computations to components; we use it to identify symbol abstraction heads and characterize their causal role. Representational similarity analysis [Kriegeskorte et al., 2008] complements this by distinguishing abstract from token-specific encodings. Our concept intervention method extends the activation patching paradigm of Meng et al. [2022] and the function vector framework of Todd et al. [2024].

Relational inductive biases and inner products. A complementary line of work asks how architectural design shapes learned representations. The relational bottleneck hypothesis [Webb et al., 2024] proposes that constraining networks to process relational rather than object-level information encourages symbol-like representations. Abstractors [Altabaa et al., 2024] implement this via relational cross-attention. Inner product spaces are well-suited to encoding abstract relations [Altabaa and Lafferty, 2024], suggesting that the attention mechanism may be a useful inductive bias for symbolic computation. Our finding that MLPs developed the same signatures qualifies this: inner products may help (especially in terms of training efficiency) but are not necessary given sufficient data diversity.

Data and curriculum effects. A growing body of evidence shows that training data distributions shape learned representations and strategies. Chan et al. [2022] showed that in-context learning emerges from distributional properties like burstiness and rare-class frequency. Uselis et al. [2025] similarly found that compositional generalization in vision models is driven by data diversity, with greater combinatorial coverage inducing a linearly factored representation. Data diversity has also been linked to compositional generalization [Hupkes et al., 2020]. We extend this to the mechanistic level: data diversity shapes not only behavioral generalization but the qualitative nature of the underlying mechanisms, independently of architecture.

Training dynamics. Language models often acquire different abilities at different points in training, sometimes abruptly. In-context learning emerges together with induction heads in a sudden phase change [Olsson et al., 2022, Nallamala, 2024], and in-context learning develops in discrete stages [Hoogland et al., 2025]. Generalization can also follow memorization after long training [Li et al., 2025], as in grokking [Power et al., 2022, Nanda et al., 2023]. Our low-diversity models showed a similar ordering, first generalizing in-distribution via a statistical solution, and then developing

symbolic mechanisms only after a long delay. How such phases relate to the emergence of specific symbolic mechanisms remains to be tested.

7 Conclusion

What determines whether a neural network develops symbolic mechanisms? We found that data diversity consistently promoted symbolic mechanisms across four axes, while architectural inductive biases were not necessary for their emergence. These findings extend to a second abstract reasoning task, suggesting that data diversity plays a central role in determining when and how symbolic mechanisms emerge.

Limitations and Future Work. The solution learned by our models partially reproduced the symbolic mechanisms observed in pretrained LLMs. Our models developed symbol abstraction heads, which LLMs use to *infer* abstract rules from in-context examples, but not the symbolic induction heads and retrieval heads that LLMs use to *apply* these rules to new examples. We hypothesize that greater rule diversity (in addition to token diversity) and rule uncertainty may be needed to induce these additional mechanisms, but this remains to be tested. Rather than attempting to reproduce the full suite of mechanisms employed by large-scale models, we chose symbol abstraction heads as a testbed for the factors that drive emergent symbolic computation. Two further limitations concern training dynamics and architecture. We trained small models from scratch on controlled tasks and found that low-diversity models first learned a statistical solution and developed symbolic mechanisms only after a long delay. How such dynamics unfold in large models trained under much more complex regimes, with mixed data, long training, and post-training, is unclear. Our concept interventions also showed that MLPs reached the same flip rates as Transformers, but how symbolic mechanisms are actually implemented in an MLP remains an open question.

Acknowledgments and Disclosure of Funding

This work was supported by the Courtois Chair in Fundamental Research - V (Neuroscience) at the Université de Montréal awarded to T.W. We are grateful to Mila (mila.quebec) and Princeton University for the compute resources, software, and technical help that made our experiments possible.

References

Awni Altabaa and John Lafferty. Approximation of relation functions and attention mechanisms. *arXiv preprint arXiv:2402.08856*, 2024.

Awni Altabaa, Taylor Webb, Jonathan Cohen, and John Lafferty. Abstractors and relational cross-attention: An inductive bias for explicit relational reasoning in transformers. In *International Conference on Learning Representations*, volume 2024, pages 24597–24613, 2024.

Stephanie Chan, Adam Santoro, Andrew Lampinen, Jane Wang, Aaditya Singh, Pierre Richemond, James McClelland, and Felix Hill. Data distributional properties drive emergent in-context learning in transformers. *Advances in neural information processing systems*, 35:18878–18891, 2022.

Jiahai Feng and Jacob Steinhardt. How do language models bind entities in context? In *International Conference on Learning Representations (ICLR)*, 2024. URL `https://arxiv.org/abs/2310.17191`.

Thomas L. Griffiths, Brenden M. Lake, R. Thomas McCoy, Ellie Pavlick, and Taylor W. Webb. Whither symbols in the era of advanced neural networks? *Trends in Cognitive Sciences*, 2026. doi: 10.1016/j.tics.2026.02.003.

Jesse Hoogland, George Wang, Matthew Farrugia-Roberts, Liam Carroll, Susan Wei, and Daniel Murfet. Loss landscape degeneracy and stagewise development in transformers. *Transactions on Machine Learning Research*, 2025. First posted as "The Developmental Landscape of In-Context Learning".

Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. Compositionality decomposed: How do neural networks generalise? *Journal of Artificial Intelligence Research*, 67:757–795, 2020.

Nikolaus Kriegeskorte, Marieke Mur, and Peter A Bandettini. Representational similarity analysis-connecting the branches of systems neuroscience. *Frontiers in systems neuroscience*, 2:249, 2008.

Martha Lewis and Melanie Mitchell. Evaluating the robustness of analogical reasoning in large language models. *arXiv preprint arXiv:2411.14215*, 2024.

Melody Zixuan Li, Kumar Krishna Agrawal, Arna Ghosh, Komal Kumar Teru, Adam Santoro, Guillaume Lajoie, and Blake A. Richards. Tracing the representation geometry of language models from pretraining to post-training. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. arXiv:2509.23024.

Gary F Marcus. *The algebraic mind: Integrating connectionism and cognitive science*. The MIT Press, 2001.

R Thomas McCoy, Shunyu Yao, Dan Friedman, Mathew D Hardy, and Thomas L Griffiths. Embers of autoregression show how large language models are shaped by the problem they are trained to solve. *Proceedings of the National Academy of Sciences*, 121(41):e2322420121, 2024.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35:17359–17372, 2022.

Gautam Reddy Nallamala. The mechanistic basis of data dependence and abrupt learning in an in-context classification task. In *International conference on learning representations*, volume 2024, pages 19226–19239, 2024.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. *arXiv preprint arXiv:2301.05217*, 2023.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, et al. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022.

Judea Pearl. Direct and indirect effects. In *Probabilistic and causal inference: the works of Judea Pearl*, pages 373–392. 2022.

Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets. *arXiv preprint arXiv:2201.02177*, 2022.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Eric Todd, Millicent Li, Arnab Sen Sharma, Aaron Mueller, Byron Wallace, and David Bau. Function vectors in large language models. In *International conference on learning representations*, volume 2024, pages 17282–17333, 2024.

Arnas Uselis, Andrea Dittadi, and Seong Joon Oh. Does data scaling lead to visual compositional generalization? *arXiv preprint arXiv:2507.07102*, 2025.

Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small. *arXiv preprint arXiv:2211.00593*, 2022.

Taylor Webb, Keith J Holyoak, and Hongjing Lu. Emergent analogical reasoning in large language models. *Nature Human Behaviour*, 7(9):1526–1541, 2023.

Taylor W Webb, Steven M Frankland, Awni Altabaa, Simon Segert, Kamesh Krishnamurthy, Declan Campbell, Jacob Russin, Tyler Giallanza, Randall O'Reilly, John Lafferty, et al. The relational bottleneck as an inductive bias for efficient abstraction. *Trends in Cognitive Sciences*, 28(9): 829–843, 2024.

Yukang Yang, Declan Campbell, Kaixuan Huang, Mengdi Wang, Jonathan Cohen, and Taylor Webb. Emergent symbolic mechanisms support abstract reasoning in large language models. In *International Conference on Machine Learning (ICML)*, 2025.

A Experimental Details

A.1 Task and data

ABA/ABB. Each prompt contains one in-context example that follows a rule, followed by a query that the model must complete under the same rule. The input is five tokens, A1 B1 A1 A2 B2 (ABA) or A1 B1 B1 A2 B2 (ABB), and the model predicts the sixth token (A2 for ABA, B2 for ABB). There is no separator or start token. The four tokens A1, B1, A2, B2 are distinct and are drawn without replacement from the vocabulary `tok_1`, ..., `tok_`V, freshly for every prompt, so surface identity carries no information about the rule. The two rules are equally likely.

Binding. A prompt is $[e_1, a_1, e_2, a_2, q]$, where $e_1 \neq e_2$ are entities, $a_1 \neq a_2$ are attributes, q is one of the two entities (chosen with probability $1/2$), and the target is the attribute paired with q. The vocabulary has about 20,000 tokens.

Dataset sizes. Table 1 lists the training and evaluation sets used in each experiment.

Table 1: Datasets. "Prompts" counts distinct training prompts.

Experiment	Vocabulary	Train prompts	Validation prompts
Vocabulary size (ABA/ABB)	50–10,000	2,000,000	2,000
Number of training samples	10,000	200–200,000	500
Number of rules	100	2,000,000	5,000
Role-token bias	100–20k	5,000,000	5,000
Architecture variants	10,000	5,000,000	10,000
Binding (role-token bias)	20,000	5,000,000	20,000

The role-token bias datasets additionally contain a standard test set (no role-exclusive tokens) and a reversed-role test set of 5,000 prompts each.

A.2 Models

All models are trained from scratch. Tokens are mapped to ids by a fixed lookup table with a few special tokens (pad, unk, bos, eos, sep).

Transformer. GPT-2 as implemented in HuggingFace `transformers` (4.56.2) with the GPT-2-small configuration: 12 layers, 12 attention heads, hidden size 768, MLP width 3072, GELU, pre-LayerNorm, learned absolute position embeddings, tied input and output embeddings, and dropout 0.1 on embeddings, attention and residual connections. Weights use the default GPT-2 initialization. The model has about 86M parameters at $V = 100$ and 94M at $V = 10{,}000$.

Architecture variants (Section 4). All variants keep the Transformer above except for the one change listed.

- *No positional weight sharing.* Every input position has its own MLP weights (`c_fc`, `c_proj`) in every block; attention weights stay shared across positions.
- *Shared Q/K/V.* A single projection $W_s \in \mathbb{R}^{768\times768}$ per layer produces queries, keys and values ($q = k = v = W_s h$), which makes the attention scores symmetric.
- *MLP attention.* The dot-product score of each head is replaced by a two-layer network applied to every (query, key) pair, $s(q, k) = W_2\,\mathrm{GELU}(W_1[q; k]+b_1)+b_2$, with hidden size equal to the head dimension (64). The softmax, causal mask, value and output projections are unchanged.
- *Pure MLP (no attention).* The input is padded to 16 positions. Token and learned position embeddings (dimension 64) are summed and the sequence is flattened into a 1024-dimensional vector. Four residual blocks follow, each LayerNorm, Linear $1024\to2048$, GELU, dropout 0.1, Linear $2048\to1024$; then a final LayerNorm, a reshape to 16×64, and a tied unembedding. There is no causal mask, so every position can see every other; the target token is never given as input, and the prediction is read out at the last context position. The model has about 17M parameters.

Binding models. The Transformer is the same GPT-2-small configuration with a context length of 10. The MLP is the pure MLP above with 8 blocks instead of 4 (about 35M parameters).

A.3 Training

All models are trained with the HuggingFace `Trainer`: AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$) with no weight decay, gradient-norm clipping at 1.0, fp16 mixed precision, a linear warmup over the first 10% of steps followed by cosine decay to zero, and per-device batch size 24 on one GPU. The peak learning rate is 10^{-4} in all experiments except the MLP-attention variant (3×10^{-4}, two GPUs with per-device batch 24, i.e. an effective batch of 48). Dropout is 0.1.

A.4 Intervention and flip rate

Role vectors. For every layer ℓ we take the residual stream after block ℓ at the third token of the in-context example (where the ABA and ABB examples differ). The ABA vector z^{ℓ}_{ABA} is the mean of this activation over $2n$ ABA prompts, and z^{ℓ}_{ABB} is the mean over $2n$ ABB prompts. The ABA and ABB prompts are built from the same tokens (each of n token pairs is used in both orders, and the in-context and query tokens are shared), so the two sets differ only in the rule. We use $n = 500$ in all reported results, i.e. 1,000 prompts per rule.

Intervention. On a test ABA prompt we add $z^{\ell}_{\text{ABB}} - z^{\ell}_{\text{ABA}}$ to the residual stream at that position, at every layer simultaneously (and the reverse for ABB prompts), this is equivalent to replacing the rule component z_{ABA} by z_{ABB} while keeping the token-specific component. All tokens are unchanged. For the pure MLP, which has no per-position streams, the vector is added to the flattened hidden vector at every block. For the 5-token rules, the vector is estimated and applied at the last token of the in-context example (position 4), where ABABA and ABABB first differ.

Flip rate. A test prompt *flips* if the model's prediction after the intervention exactly equals the correct answer under the opposite rule for that prompt. Test prompts use fresh tokens, 200 per rule for the reported results (400 prompts in total) with duplicates removed. For the out of distribution bias experiments the flip rate is measured on reversed-role prompts, using role vectors estimated from role-respecting prompts of the same bias level.

Uncertainty. Points are means over 3-5 seeds; error bars are standard deviations (rules, vocabulary size) or the min–max range (bias, architectures), as stated in each caption.

A.5 Data-diversity manipulations

Vocabulary size. The number of distinct tokens in the training dataset, $V \in \{50, 100, 500, 1000, 2000, 4000, 8000, 10000\}$.

Number of training samples. The number of distinct training prompts, $N \in \{200, 1000, 5000, 20000, 50000, 100000, 200000\}$ at $V = 10{,}000$.

Number of rules. For $R = 10$ the rules are length-5 patterns: ABABA, ABABB, ABCBA, ABCBB, ABCCC, AAABA, AAABB, AABAA, AABAB, AABBA, at $V = 100$. The $R = 2$ point is the plain ABA/ABB task at $V = 100$.

Role-token bias. A fraction $b \in \{0, 0.25, 0.5, 0.75, 1\}$ of the vocabulary is role-exclusive during training, split evenly into a set that only fills role A and a set that only fills role B; the remaining $(1 - b)V$ tokens can fill either role. At test time the two role-exclusive sets are swapped (the reversed-role test) while the rule is unchanged. For the binding task the exclusive sets are entity-only and attribute-only tokens.

A.6 Convergence definitions

Accuracy is evaluated on the validation set every 250 steps and flip rate at every saved checkpoint. A quantity *converges* at the first checkpoint at which its rolling mean over three checkpoints reaches the

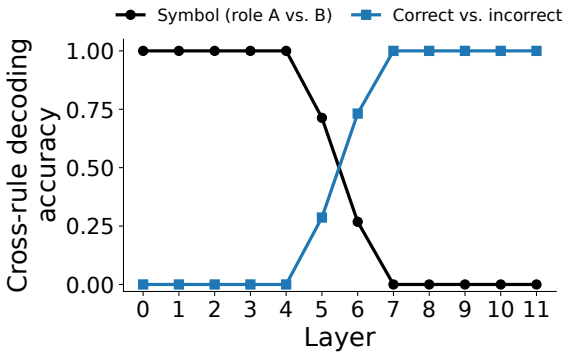


Figure 7: **Cross-rule decoding by layer.** Probes trained on one rule and tested on the other. Layers 0–4 decode symbol identity perfectly; their correct/incorrect score is 0% because the same direction points to the wrong token under the new rule. Layers 7–11 show the reverse.

threshold and at least 90% of the remaining checkpoints stay above 95% of the threshold. Accuracy convergence uses a threshold of 95% in the vocabulary-size and number-of-rules plots. Flip-rate convergence uses 90%, and runs that never reach it are excluded from the ratio plots. The flip rate at accuracy convergence is read from the checkpoint nearest to the accuracy-convergence step.

B Decoder Experiments

To test the account in Section 2.3 that early layers carry abstract symbol identity while later layers carry the correct answer, we trained linear probes on the residual stream at the final query position of a model (vocabulary size 2000). The query always used the same two tokens, so token identity was constant across trials and only (i) which of the two tokens plays role A and (ii) which rule is active vary. Each trial has one in-context example with fresh tokens. We used 400 trials per rule. For each layer, we trained a logistic-regression probe (L2 penalty, $C = 1$) on raw activations from one rule (ABA or ABB) to predict whether the first query token is the correct answer, then evaluated the same probe on the *other* rule against two different labels:

- **symbol generalization**: whether the first query token has the role the probe was trained to detect (role A for an ABA-trained probe, role B for an ABB-trained probe)
- **correct/incorrect generalization**: whether the first query token is the correct answer, which flips meaning between the two rules.

Within a single rule the two labels coincide, so only the cross-rule test separates them. Chance is 50%. A probe that tracks role identity scores 100% on the first label and 0% on the second, because the correct token changes role between the rules; a probe that tracks the answer scores the reverse. We average over both training directions (ABA→ABB and ABB→ABA).

Symbol generalization was 100% in layers 0–4 and fell to 71% and 27% in layers 5 and 6 and 0% afterwards, while correct/incorrect generalization rose from 0% (layers 0–4) through 29% and 73% (layers 5 and 6) to 100% from layer 7 onward (Figure 7). Because the cross-rule probe is evaluated on a rule it was never trained on, this is evidence that early layers represent role identity abstractly and later layers represent the answer itself.

C Training Dynamics Across Data Diversity Levels

This section gives the training dynamics behind the data-diversity results in the main text. For each level of each diversity axis, we track task accuracy and the third-token flip rate at every saved checkpoint. We include all four data-diversity axes: number of training samples (Figure 8), role-token bias (Figure 9), vocabulary size (Figure 10), and number of rules (Figure 11). Unless stated otherwise, accuracy is shown in black and flip rate in blue; lines are the mean over seeds and the shaded band spans the minimum and maximum over seeds. Accuracy is measured on freshly generated test

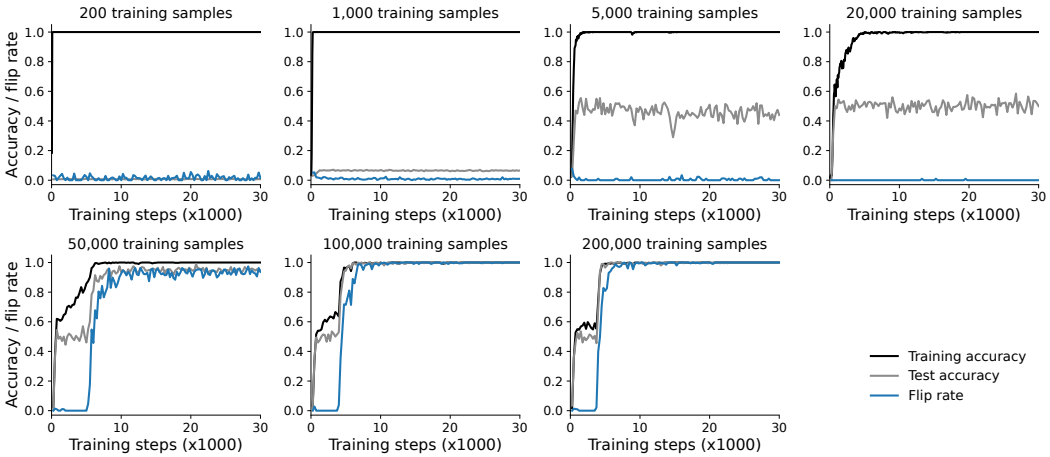


Figure 8: **Timecourses by number of training samples** (vocabulary size 10,000). Flip rate is measured on in-distribution prompts.

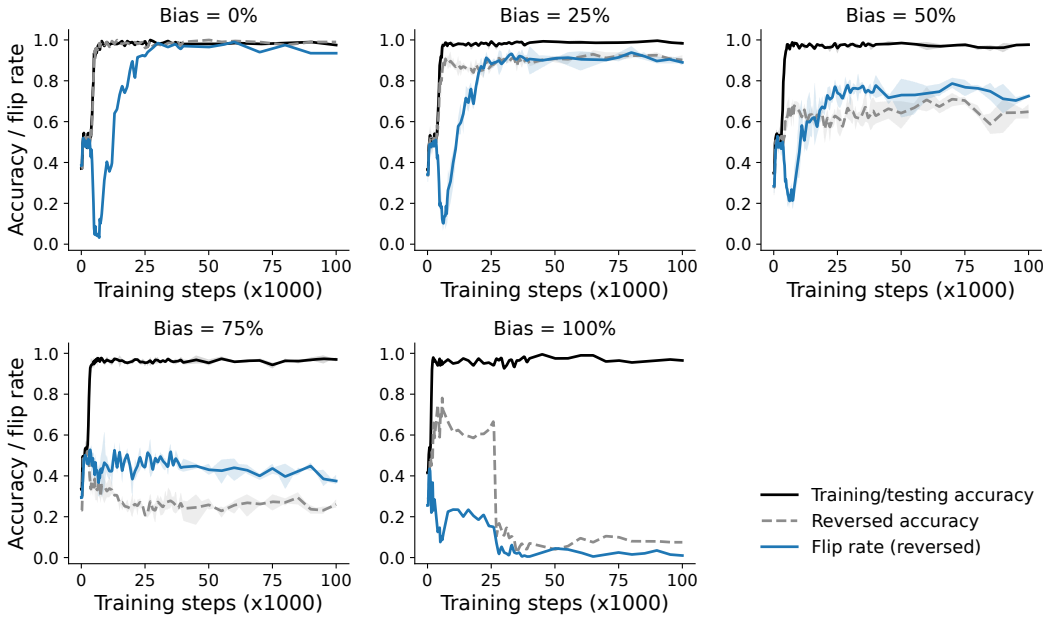


Figure 9: **Timecourses by role-token bias** (vocabulary size 100). Solid black: in-distribution accuracy. Dashed gray: accuracy on reversed-role (out-of-distribution) prompts. Blue: flip rate on reversed-role prompts. In-distribution accuracy is near ceiling at every bias level, while reversed accuracy and reversed flip rate fall as bias increases.

prompts. Figure 12 summarizes the same data as the ratio of the flip-rate convergence step to the accuracy convergence step.

D Effect of Model Size

To test whether the delayed emergence of the symbolic mechanism depends on model capacity, we compared the default GPT-2-small model used throughout the paper (12 layers, 12 heads, hidden size 768; 85.9M parameters) with a much smaller model (2 layers, 2 heads, hidden size 128; 0.53M parameters) on the ABA/ABB task at vocabulary size 50.

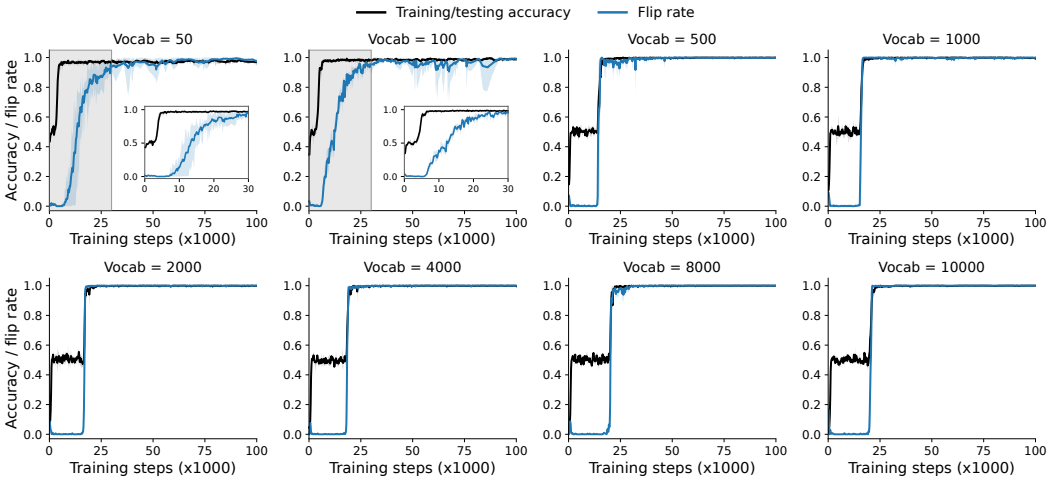


Figure 10: **Timecourses by vocabulary size**. Shaded boxes and insets zoom on the first 30,000 steps for the two smallest vocabularies. At small vocabularies ($V \leq 100$) flip rate rises well after accuracy has converged; from $V = 500$ upward the two curves rise together.

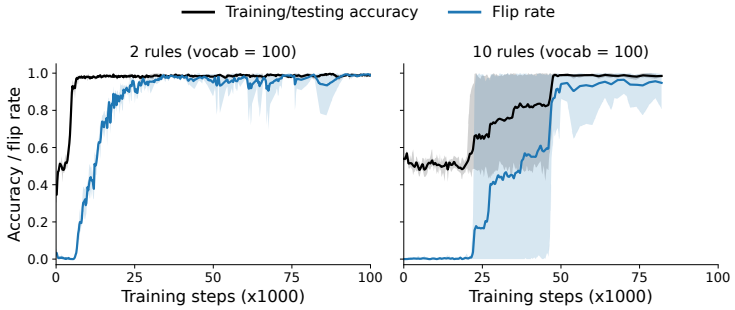


Figure 11: **Timecourses by number of rules** (vocabulary size 100). The 2-rule condition is the plain ABA/ABB task. The 10-rule condition uses ten length-5 rules.

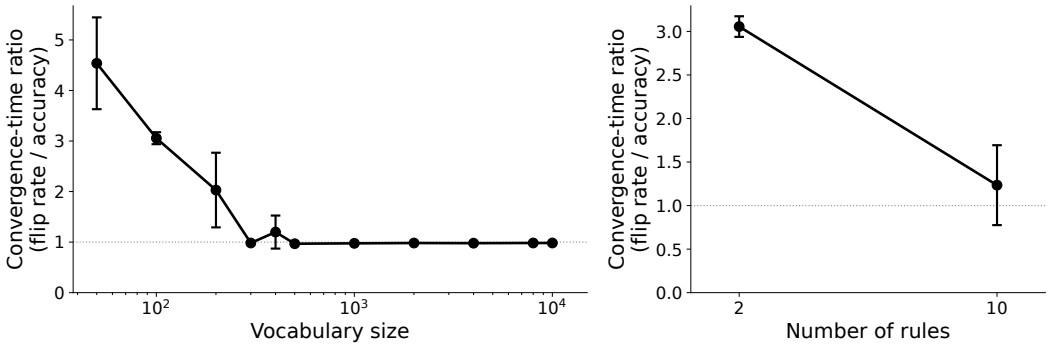


Figure 12: **Ratio of flip-rate to accuracy convergence step across diversity levels.** A ratio of 1 means the mechanism converges together with accuracy. Points are means over runs. The ratio is about 1 for vocabulary sizes $V \geq 500$ and for the number-of-rules axis, and larger for the smallest vocabularies.

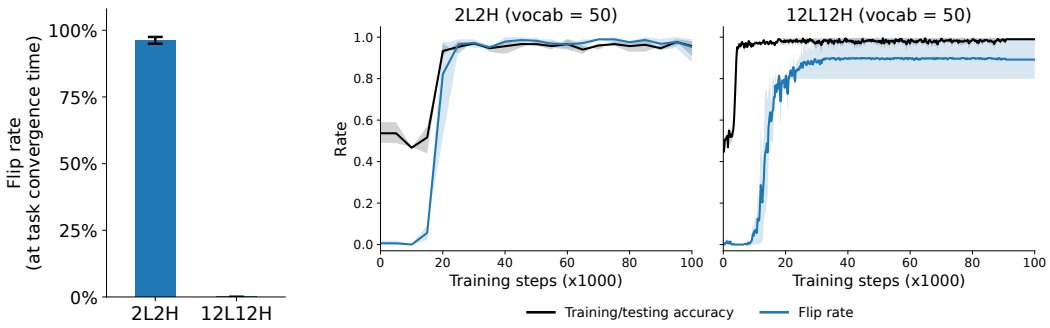


Figure 13: **Effect of model size on the emergence of the symbolic mechanism** (vocabulary size 50). **Left:** flip rate at the step where accuracy first converges, mean ± standard deviation over seeds. **Right:** training dynamics of the two models: accuracy and flip rate over training.

Smaller models skip the statistical phase. At the step where accuracy converges, the 2L2H model already had a flip rate of $96.2\% \pm 1.3\%$, whereas the 12L12H model had a flip rate of 0% (Figure 13). The 12L12H model reached 95% accuracy much earlier, but its flip rate only rose about $6.5\times$ later; in the 2L2H model the two converge together. Thus the smaller model does not show the initial statistical phase at all: it appears to go directly to symbolic signatures, although it takes longer, in training steps, to get there than the base model takes to solve the task.

Capacity versus diversity. We think this fits naturally alongside the vocabulary-size result. Reducing capacity, by using a smaller model, removes the option of a first-order statistical solution outright, so there is no memorization phase to pass through. Reducing diversity, by using a smaller vocabulary at fixed model size, merely makes the statistical solution viable for longer before the pressure toward the compressed, symbolic solution eventually wins out.

E Mechanistic Analysis of Entity-Attribute Binding

We tested whether the entity-attribute binding task (Section 5) is solved with binding IDs, mirroring the analyses used for the ABA/ABB task in Section 2: an intervention, a causal mediation analysis, and a representational similarity analysis.

E.1 Binding-ID intervention

Each input lists two entity-attribute pairs followed by a query entity, and the model must output the attribute bound to it. Under the binding-ID hypothesis, the model tags each pair with an ID (ID0 or ID1) and answers by matching the query's ID. We therefore intervene at the query position: for a query on the first entity we subtract the mean residual vector associated with ID0 and add the one associated with ID1 (and vice versa for the second entity). The two ID vectors are the mean residual-stream vector at the query position over many prompts in which the query is the first-pair entity (ID0) or the second-pair entity (ID1), estimated on one half of the entities and tested on the held-out half, so the test queries are novel tokens. A *flip* means the model now outputs the attribute bound to the other entity.

E.2 Causal mediation analysis

We use two patching experiments at the query position, each producing a score for every layer and token position and for every layer and attention head. In Stage 1 (binding-ID retrieval), the base prompt is $[e_1, a_1, e_2, a_2, e_1]$ and the patched prompt swaps which slot each pair occupies, $[e_2, a_2, e_1, a_1, e_1]$. Both have the same correct answer, a_1, because the entity-attribute binding is preserved; a head that reads the binding ID transfers the base prompt's slot and makes the patched prompt output a_2. In Stage 2 (token retrieval), the patched prompt swaps the attributes in place, $[e_1, a_2, e_2, a_1, e_1]$, so the correct answer changes; a head that reads the attribute token at the queried slot transfers the base prompt's attribute. A head's score is the change in the logit difference between

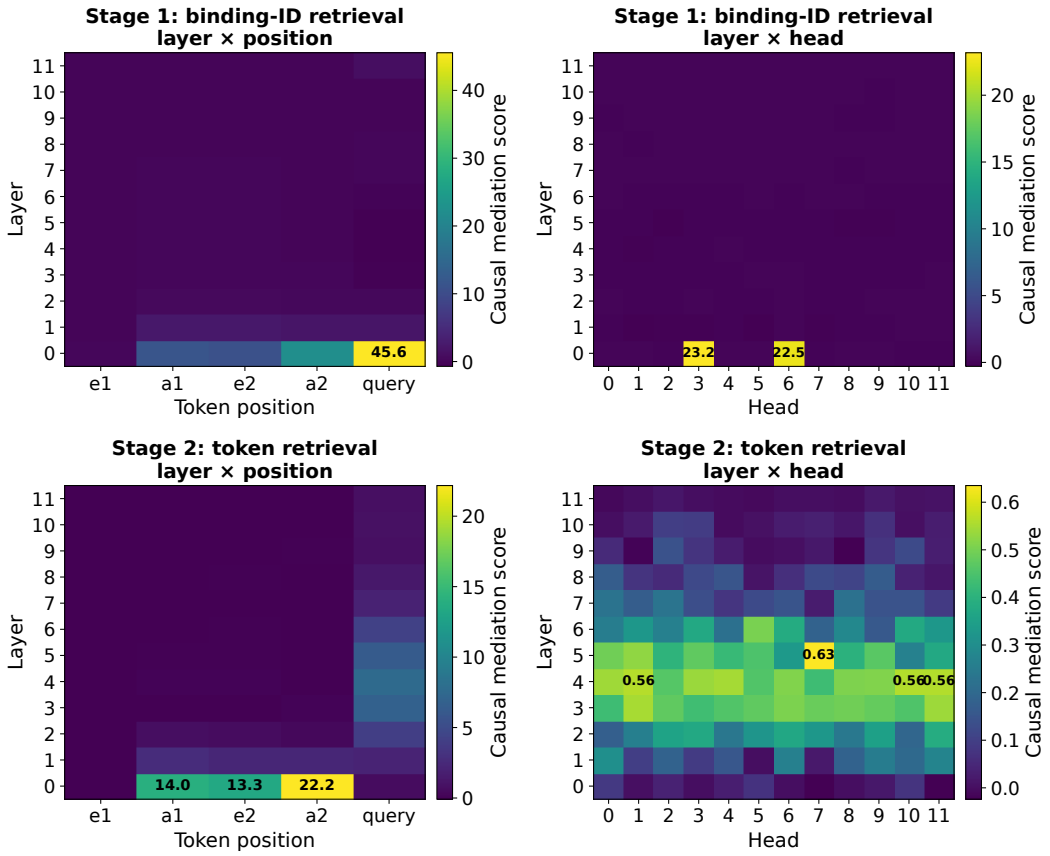


Figure 14: **Causal mediation analysis for binding.** *Top:* Stage 1 (binding-ID retrieval). *Bottom:* Stage 2 (token retrieval). *Left:* layer × token position (patching attention and MLP output at one position). *Right:* layer × head (patching one head's output at the query position). Layers are numbered from 0.

the causal answer and the original answer when its output at the query position is patched from the base into the patched prompt.

For Stage 1 the effect is concentrated in layer 0 at the query position, and at the head level in two heads (layer 0, heads 3 and 6). For Stage 2 the effect is much more distributed: the head-level scores are spread over layers 2–6, strongest in layers 3–5 (peak at layer 5, head 7), and the layer-wise map additionally shows a large effect in layer 0 at the attribute positions.

E.3 Representational similarity analysis

To test what the identified heads represent, we computed per-head representational similarity at the query position. Each problem is rendered twice with the same tokens and the query fixed to the first entity: $[e_1, a_1, e_2, a_2, e_1]$ (the query's pair is in slot 0, ID0) and $[e_2, a_2, e_1, a_1, e_1]$ (slot 1, ID1). At the query position the token is identical across the pair while the binding ID differs, so a head that carries the ID gives a two-block similarity matrix (ID0 against ID0 and ID1 against ID1 similar, across-ID dissimilar), whereas a head that carries token identity gives a diagonal across the two blocks. We collected the head outputs for 20 problems (40 prompts), computed pairwise cosine similarities, and averaged the matrices over heads weighted by their CMA scores, keeping only heads whose score exceeds the mean by more than 0.5 standard deviations.

The Stage 1 matrix had a strong two-block structure, which indicates that the Stage 1 heads encode the binding ID rather than token identity. The Stage 2 matrix keeps a block structure but also shows a diagonal linking the same token across the two IDs.

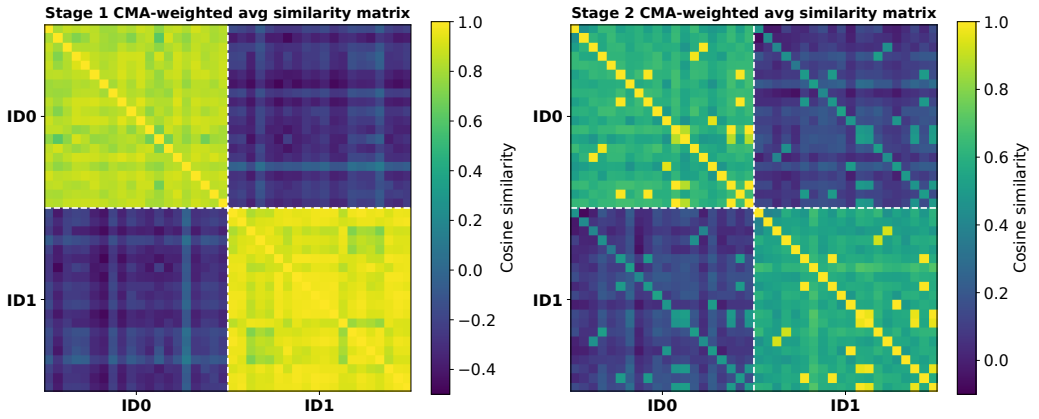


Figure 15: **CMA-weighted representational similarity for binding.** Average cosine-similarity matrices over heads with CMA $z > 0.5$, ordered by binding ID (ID0 prompts first, ID1 second). *Left:* Stage 1 (binding-ID retrieval). *Right:* Stage 2 (token retrieval).

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We have emphasized the main contribution of this paper in both the abstract and introduction sections.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We have discussed the limitations of our work in Conclusion.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.

- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper has no theory.

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide a detailed description of the methods used in our work in Sections 2–3 and Appendix A.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.

- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We release code here: https://github.com/melodylizx/symbolic-mechanisms-emergence

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide a detailed description of our experimental setups in Sections 2–3 and Appendix A.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.

- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: To ensure the reliability of our findings, we repeated our experiments on multiple seeds and reported the standard error as error bars in the figures.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We present rough details about our compute infrastructure in the Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our study (1) does not involve any interaction with human subjects, and (2) utilizes datasets that do not contain personally identifiable information, have not been withdrawn by their authors, and are not included on the NeurIPS list of deprecated datasets.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Impact statements has been included in the conclusions.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper uses synthetic data and small models trained on abstract token sequences, which pose no risk for misuse.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite the codebase used in this work.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: No assets are released.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This work neither involves crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: Not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: We used an AI coding and writing assistant (Claude, Anthropic) during the preparation of this submission. Specifically, Claude assisted with: (1) implementing and debugging training and analysis scripts for some experiments in Sections 3–5; (2) helping with routine code modifications and identifying implementation issues during the development of these experiments; and (3) revising and improving the clarity and presentation of the manuscript text. The AI assistant was used as a supporting tool and did not determine the experimental questions, hypotheses, experimental designs, or analytical methods. All code used to produce the reported results was reviewed and verified by the human authors, and all experimental design choices, analyses, interpretations, and scientific claims were specified, reviewed, and verified by the human authors.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.