Finite State Automata Inside Transformers with Chain-of-Thought: A Mechanistic Study on State Tracking

Yifan Zhang[1,2*], Wenyu Du[3], Dongming Jin[1,2], Jie Fu[4†], Zhi Jin[1,2†]
[1]Key Laboratory of High Confidence Software Technology (PKU), MOE, China
[2]School of Computer Science, Peking University, China
[3]The University of Hong Kong, [4]Shanghai AI Lab
yifanzhang@stu.pku.edu.cn, fujie@pjlab.org.cn, zhijin@pku.edu.cn

Abstract

Chain-of-Thought (CoT) significantly enhances the performance of large language models (LLMs) across a wide range of tasks, and prior research shows that CoT can theoretically increase expressiveness. However, there is limited mechanistic understanding of the algorithms that a Transformer with CoT (denoted as $\texttt{Transformer}_{+\texttt{CoT}}$ in this paper) can learn. Our key contributions are: (1) We evaluate the state tracking capabilities of $\texttt{Transformer}_{+\texttt{CoT}}$ and its variants, confirming the effectiveness of CoT. (2) Next, we identify the circuit (a subset of model components, responsible for tracking the world state), indicating that late-layer MLP neurons play a key role. We propose two metrics, compression and distinction, and show that the neuron sets for each state achieve nearly 100% accuracy, providing evidence of an implicit finite state automaton (FSA) embedded within the model. (3) Additionally, we explore three challenging settings: skipping intermediate steps, introducing data noises, and testing length generalization. Our results demonstrate that $\texttt{Transformer}_{+\texttt{CoT}}$ learns robust algorithms (FSAs), highlighting its resilience in challenging scenarios. Our code is available at https://github.com/IvanChangPKU/FSA.

1 Introduction

Transformer-based large language models (LLMs Touvron et al., 2023; OpenAI, 2023) revolutionize natural language processing (NLP) by demonstrating significant progress across various tasks. However, they still face challenges with basic calculations (Zhou et al., 2023), complex reasoning (Valmeekam et al., 2024; Han et al., 2024), and regular languages (Bhattamishra et al., 2020). Approaches such as Chain-of-Thought (CoT) prompting (Wei et al., 2023) and scratchpads (Nye et al., 2021) address these limitations by generating intermediate reasoning steps. To understand the success of CoT, prior work has analyzed its expressiveness from the perspective of formal language theory and circuit complexity. Theoretical works (Zhang et al., 2024; Qiu et al., 2024; Li et al., 2024) demonstrate that incorporating a linear number of intermediate steps increases the expressive power of transformers, enabling them to represent all **Finite State Automata**, which are a foundational class of automata.

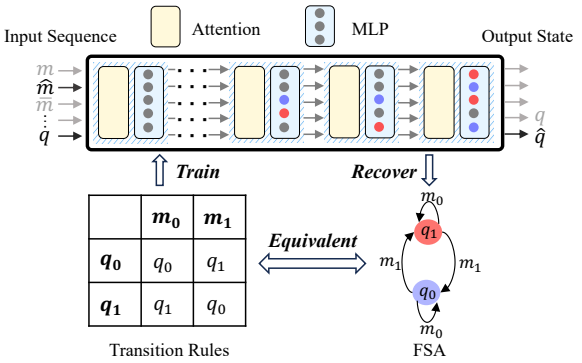


Figure 1: An illustration of one of the simplest state-tracking problems, $\mathbb{Z}_2$. After training on sequences generated from the $\mathbb{Z}_2$ transition rules, $\texttt{Transformer}_{+\texttt{CoT}}$ successfully recovers an implicit finite state automaton (FSA) by differentiating between two internal states—q_0 and q_1—using two distinct and disjoint sets of neurons in the late-layer MLPs. These neuron groups are visually distinguished using colors: neurons corresponding to state q_0 are marked in purple, those for q_1 in red, while neurons not contributing to either state are shown in gray.

However, theoretical expressiveness indicates only upper and lower bounds on what an architecture can express; it does not guarantee successful learning during training in practice. For instance, although recurrent neural networks (RNNs) are theoretically more expressive than transformers in Chomsky's computational hierarchy—being capable of recognizing all regular languages, whereas

*Work done during internship at Shanghai AI Lab.
†Corresponding and co-senior authors.

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 13603–13621
July 27 - August 1, 2025

transformers fail to recognize certain types (*e.g.*, periodic finite-state languages) (Delétang et al., 2023)—they often fail to outperform transformers in practice. This discrepancy is primarily due to issues such as vanishing gradients, which hinder learning long-term dependencies, and difficulties in parallelization, which limit computational efficiency (Hochreiter and Schmidhuber, 1997; Vaswani et al., 2017; Brown et al., 2020). Consequently, some studies investigate the expressiveness of these architectures through the lens of learnability by measuring performance in language modeling. For example, Liu et al. (2023a) demonstrate that training transformers with recency-biased scratchpads improves sequential accuracy. However, even near-perfect next-token prediction accuracy does not imply that generative models reconstruct a true world model (Vafa et al., 2024). This raises a critical question: **Does CoT help transformers recover a world model in the form of FSAs, or do they merely learn shortcuts?**

To address this question, we extend the study of learnability beyond accuracy improvements, performing an internal mechanistic analysis of CoT's success. Specifically, we focus on *state tracking*, a foundational task to evaluate expressiveness (Merrill et al., 2024). In state tracking, a sequence of updates modifies the world state, which is represented as an FSA. The goal is to determine the final state after applying all updates sequentially. State tracking is a core capability of generative models and supports many downstream tasks — such as entity tracking (Kim and Schuster, 2023), chess (Toshniwal et al., 2022), and map navigation (Liu et al., 2023b). Figure 1 illustrates $\mathbb{Z}_2$, one of the simplest state tracking problems[1], along with its corresponding FSA and transition rules.

We begin by comprehensively evaluating the state tracking capabilities of $\texttt{Transformer}_{+\texttt{CoT}}$, comparing it with other models (RNNs), transformer variants (*e.g.*, those with recurrence (Fan et al., 2021; Yang et al., 2022)), and CoT variants (*e.g.*, implicit CoT (Goyal et al., 2024)). Empirically, we show that $\texttt{Transformer}_{+\texttt{CoT}}$ is the only model capable of efficiently learning state tracking for sequences of arbitrary lengths across three groups: $\mathbb{Z}_{60}$, $A_4 \times \mathbb{Z}_5$, and A_5, in both in-distribution and out-of-distribution settings.

[1]The state tracking problem $\mathbb{Z}_2$ is equivalent to parity, a formal language describing binary sequences with specific evenness or oddness properties.

Next, to provide a mechanistic explanation for this success, we apply interpretability techniques to analyze the algorithms learned by $\texttt{Transformer}_{+\texttt{CoT}}$. Using activation patching (Vig et al., 2020), we identify the circuits (specific model components) responsible for state tracking and observe that $\texttt{Transformer}_{+\texttt{CoT}}$ relies heavily on late-layer MLP neurons. These neurons can be effectively grouped into states based on transition rules. To quantify this, we introduce two metrics: compression and distinction. Compression measures the similarity of representations for the same state under different input prompts, while distinction quantifies the separation between different states, even when their inputs are similar. We find nearly 100% accuracy on both metrics at every intermediate step, providing strong evidence that the model reconstructs the world model (*i.e.*, FSAs). For instance, in Figure 1, $\texttt{Transformer}_{+\texttt{CoT}}$ compresses inputs corresponding to two states (*i.e.* q_0 and q_1) by activating two distinct sets of neurons.

In real-world tasks that involve state tracking, state transitions are often implicit, and reasoning structures tend to be imperfect. For example, mathematical reasoning is related to state tracking, which involves keeping track of the problem goal, and chain-of-thought annotations in existing datasets—such as OpenWebMath—contain skipping or noise. To evaluate robustness in challenging scenarios, we assume three experimental settings: skip-step reasoning, noisy scratchpads, and length generalization. Through controlled experiments, we show that $\texttt{Transformer}_{+\texttt{CoT}}$ learns robust algorithms even in noisy environments, suggesting that the underlying FSAs exhibit strong resilience.

In summary, this work is the first to extend the study of learnability and expressiveness through mechanistic interpretation on state tracking, uncovering the underlying algorithms of $\texttt{Transformer}_{+\texttt{CoT}}$. Our contributions are as follows:

1. We conduct a comprehensive evaluation of the state tracking capabilities of $\texttt{Transformer}_{+\texttt{CoT}}$, demonstrating its unique ability to track states of arbitrary lengths across multiple groups ($\mathbb{Z}_{60}$, $A_4 \times \mathbb{Z}_5$, and A_5) in both in-distribution and out-of-distribution settings.
2. Using interpretability techniques, including activation patching, we analyze the learned algorithms in $\texttt{Transformer}_{+\texttt{CoT}}$. We identify

the activation of late-layer MLP neurons and classify them into states based on transition rules, achieving nearly 100% accuracy in metrics of compression and distinction, which confirms the model's reconstruction of the world model (FSAs).

3. We explore $\texttt{Transformer}_{+\texttt{CoT}}$ in three challenging settings and find that it learns resilient algorithms capable of effective state tracking in noisy conditions.

2 Background

2.1 FSA and State Tracking

We adopt the conventional definition of a finite state automaton (FSA) as a tuple $\mathcal{A} = (\Sigma, Q, q_0, \delta)$, where Σ is the alphabet, Q is a set of states, q_0 is the initial state, and δ is the transition function (Hopcroft, 2007). Formally, state tracking can be framed as solving a word problem on a finite monoid $(M, \cdot)$, where the objective is to compute the product $m_1 \cdot m_2 \cdot \ldots \cdot m_n \in M$ (Merrill et al., 2024). When M is finite, the computation can be carried out by a corresponding finite state automaton (M, M, e, δ), where the identity element e acts as the initial state and the transition function is defined as $\delta(m_1, m_2) = m_1 \cdot m_2$ for all $m_1, m_2 \in M$.

As generative models, transformers augmented with chain-of-thought generate state sequences $(q_1 \ldots q_n) \in M^*$ conditioned on input sequences $(m_1 \ldots m_n) \in M^*$. Our work centers on word problems in the context of groups, which are monoids with inverses. We specifically investigate two structures: the cyclic group $\mathbb{Z}_k$, defined by addition modulo k, and the alternating group A_k, a subgroup of the symmetric group S_k comprising all even permutations of k elements.

2.2 Mechanistic Interpretation of MLPs

Geva et al. (2022) demonstrate that multilayer perceptrons (MLPs) contribute additive updates to the residual stream, which can be decomposed into weighted sums of sub-updates. In particular, given input $\mathbf{x^l}$ at layer l, the MLP can be expressed using parameter matrices $\mathbf{K^l}, \mathbf{V^l} \in \mathbb{R}^{d_{\text{mlp}} \times d_{\text{m}}}$, where d_{mlp} is the MLP intermediate dimension and d_{m} is the model dimension. Additionally, a non-linear activation function f is applied as follows:

$$\text{MLP}^l(\mathbf{x^l}) = f(\mathbf{K^l x^l})\mathbf{V^l}$$

Expanding this further, it can be decomposed as:

$$\text{MLP}^l(\mathbf{x^l}) = \sum_{j=1}^{d_{\text{mlp}}} f(\mathbf{x^l} \cdot \mathbf{k_j^l})\mathbf{v_j^l} = \sum_{j=1}^{d_{\text{mlp}}} m_j^l \mathbf{v_j^l}$$

where $\mathbf{k_j^l} \in \mathbb{R}^{d_{\text{m}}}$ and $\mathbf{v_j^l} \in \mathbb{R}^{d_{\text{m}}}$ correspond to the j-th row vectors of $\mathbf{K^l}$ and $\mathbf{V^l}$, respectively. The scalar $m_j^l = f(\mathbf{x^l} \cdot \mathbf{k_j^l})$ represents the activation coefficient for the neuron $\mathbf{v_j^l}$. Notably, when these sub-updates $m_j^l \mathbf{v_j^l}$ are projected into the vocabulary space using the logit lens (Geva et al., 2021), they can be interpreted in a human-understandable way.

3 Evaluating State Tracking Capability Across Architectures

Besides $\texttt{Transformer}_{+\texttt{CoT}}$, there are various theoretical works (Zhang et al., 2024; Fan et al., 2024; Yang et al., 2022; Fan et al., 2021) attempting to inject recurrence into transformers, while another line of work (Goyal et al., 2024; Hao et al., 2024) proposing modifications of chain-of-thought (implicit chain-of-thought in contrast to explicit chain-of-thought). In this section, we will explore the state tracking capability with an empirical lens: *can transformers with/without CoT, and their variants, learn state tracking?*

Datasets: $A_5, A_4 \times \mathbb{Z}_5, \mathbb{Z}_{60}$. Following the formal definition of state tracking in (Merrill et al., 2024; Grazzi et al., 2024), we model state tracking as word problems, and consider three kinds of groups with increasing difficulty: $\mathbb{Z}_{60}$, an abelian group encoding mod-60 addition, $A_4 \times \mathbb{Z}_5$, a non-abelian but solvable group[2], which is a direct product group of one alternating group A_4 (a subgroup of the symmetric group S_4 containing only even permutations) and one cyclic group $\mathbb{Z}_5$, and A_5, the alternating group on five elements, which is the smallest non-solvable subgroup. With the same number of elements 60, the three groups belong to TC^0, TC^0, and NC^1-complete respectively, with varying difficulty mainly deriving from the complexity of learning the group multiplication operation.

Architectures. We denote $\texttt{Transformer}$ as a GPT2 architecture (Radford et al., 2019) with a bounded number of layers. When augmented with chain-of-thought, it is denoted as

[2]Formally, a finite group G is solvable if it has a subnormal series $1 = G_0 < G_1 < \ldots < G_k = G$ such that each factor G_i/G_{i-1} is abelian.

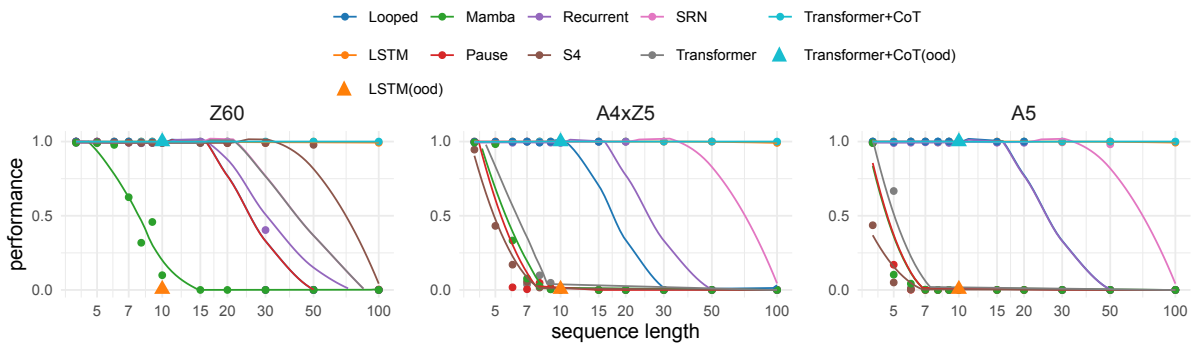


Figure 2: Model accuracy across sequence lengths for $\mathbb{Z}_{60}$, $A_4 \times \mathbb{Z}_5$ and A_5. The x-axis shows sequence lengths (2 to 100), and the y-axis shows sequence accuracy. Each line represents a different model (see legend). Dots indicate in-distribution performance, while triangles indicate out-of-distribution performance (group elements sampled from the full set, including mixed sequences not encountered during training, though within the same length range).

$\texttt{Transformer}_{+\texttt{CoT}}$. To compare with other models, we also consider recurrent neural networks (RNNs Jain and Medsker, 1999 and LSTMs Hochreiter and Schmidhuber, 1997), S4 (Gu et al., 2022), Mamba (Gu and Dao, 2024), implicit chain-of-thought: transformers with pause (denoted as Pause) (Goyal et al., 2024) and other variants of transformers: standard recurrent transformer (denoted as Recurrent) (Yang et al., 2022), looped transformer (denoted as Looped) (Fan et al., 2024).

Task Formulation. In order to disentangle recurrence introduced by CoT from those arising from architectural modifications (*e.g.*, the standard recurrent transformer, where recurrence is inherent to the model architecture), we employ two different task formulations for evaluation: the Token-Tagging (TT) task and the Language Modeling (LM) task. In the TT task, each input token m_i is annotated with its corresponding world state q_i. In contrast, the LM task requires the model to autoregressively generate the sequence of state transitions $(q_1 \dots q_n)$. Specifically, only $\texttt{Transformer}_{+\texttt{CoT}}$ and implicit CoT (Pause) are evaluated using the LM task, whereas other models and variants the TT task. The reason of the difference is that, we use the same set of labels whatever the task is, so as to eliminate the label-related influences on supervised training—similar to the "hint" setting in (Li et al., 2024).

Experimental Setup. We utilize the Python package `Abstract Algebra`[3] to randomly generate three distinct types of group-based datasets. Each dataset consists of input sequences $(m_1 \dots m_n)$ and corresponding state sequences $(q_1 \dots q_n)$, where state transitions follow the multiplication operations of three different algebraic groups: $A_5, A_4 \times \mathbb{Z}_5, \mathbb{Z}_{60}$. For comparison, all models are configured with a single layer and a model dimension of 512. Each model is trained on 1,000,000 sampled sequences of length n, for successively larger values of n, and evaluated the sequence accuracy on a held-out validation set. We set a maximum of 500 training epochs and employ early stopping once the validation accuracy reaches 99%.

In-Distribution Performance. We hold a comprehensive evaluation of models' state tracking capability on word problems, with the same model depth and dimension. Figure 2 gives different models' performance across different input sequence length and different groups. We draw several conclusions:

1. Consistent with theoretical study (Liu et al., 2023a; Merrill et al., 2024), `Transformer` and state-space models, such as S4 and Mamba, are incapable of expressing arbitrary length A_5 word problems, in contrast to RNN and LSTM.
2. CoT definitely extends the expressive power of `Transformer`. In contrast to Pause, which fails with longer sequences, $\texttt{Transformer}_{+\texttt{CoT}}$ can efficiently learn word problems of arbitrary length across multiple groups.
3. $\texttt{Transformer}_{+\texttt{CoT}}$ achieves a dual win in both expressiveness and learnability. Unlike architectural modifications such as Looped or

[3] https://github.com/alreich/abstract_algebra

Recurrent, $\texttt{Transformer}_{+\texttt{CoT}}$ maintains the original Transformer architecture without modification, and thus preserves the inherent advantages of the standard Transformer, including the parallel training and smoother gradient flow. While alternative architectures may postpone accuracy degradation on longer input sequences compared to the standard Transformer, they ultimately fail to converge on sequences of arbitrary length.

Out-of-Distribution Performance. Liu et al. (2023a) analyze transformers' failure in out-of-distribution of parity, and argue that transformers learn a shortcut solution, which compute the parity by counting the number of 1s in the input sequences and compute mod-2, thus failing to generalize to sequences with unseen sum of 1s. Zhang et al. (2024) discuss the role of chain-of-thought in computability, and point that CoT simulates the recurrent connection by iteratively encode and decode back and forth between states and tokens. The expressiveness of $\texttt{Transformer}_{+\texttt{CoT}}$ in state tracking stems from its ability to retrieve prior computations by appending the previous step's state to the end of the scratchpad.

To investigate whether $\texttt{Transformer}_{+\texttt{CoT}}$ learns an algorithm based solely on the input sequences $(m_1 \dots m_n)$ or combines input and scratchpad as theoretical work expects, we divide the elements of the three groups into proper subsets. And we train the model on sequences with m belonging to one proper subset, but evaluate on the full set. We stress that, through restricting the input sequences into separate subsets, the possible state sequences remain the same, for the reason that any subset can express the whole group through group operation. If the model learns a shortcut solution, that attends to only input, it can not generalize to sequences with group elements sampled from the full set, because the model has not seen mixed input sequences in the training set. As previously outlined, both LSTM and $\texttt{Transformer}_{+\texttt{CoT}}$ successfully learn all word problems in-distribution, and we test their performance out-of-distribution. Results in Figure 2 show that $\texttt{Transformer}_{+\texttt{CoT}}$ achieves perfect generalization on three groups $\mathbb{Z}_{60}, A_4 \times \mathbb{Z}_5, A_5$ in contrast to LSTM, implying that the model attends to not only input but also scratchpad. We provide more experimental settings in Appendix B. The out-of-distribution performance eliminates the possibility that the model learns specific shortcuts similar to those Liu et al. (2023a) find on $\mathbb{Z}_2$ group, but *this does not rule out the possibility that other shortcuts exist, which necessitates an interpretation on the mechanism.*

4 Mechanism Inside Transformers with Chain-of-Thought: FSAs

Both transformers with chain-of-thought and recurrent neural networks achieve perfect performance in-distribution, and the former even generalizes well out-of-distribution. Due to the transformer's black-box nature, the mechanism behind its state tracking implementation remains unknown. Consequently, we cannot answer questions such as what algorithm the model has learned to keep track of the world state, and how well the model can generalize. Considering that the word problems involve a series of state transitions, the model's state computation is dynamic and sequential while generating intermediate steps. In this section, we first try to analyze the circuit used by transformers with chain-of-thought to keep track of the world state, and then conduct a deeper component analysis to interpret the mechanism.

4.1 Circuit Localization

To understand the mechanism of state tracking, we will first localize the circuit (a subset of components) using activation patching (Vig et al., 2020). We formalize each word problem as a prompt $\mathbf{p_i} = (m_1 \dots m_n | q_1 \dots q_{i-1})$, where $(m_1 \dots m_n)$ is the problem description and q_i is the resulting state token at i-th step.[4] At each intermediate step, we sample a prompt $\mathbf{p_i}$ with its corresponding state token q_i, and then sample a counterfactual prompt $\mathbf{p_i'}$ that yields a different state token q_i'. We perform intervention experiments by replacing the activation of a specific MLP layer or attention head with the pre-computed activation from $\mathbf{p_i'}$, and then assess how this affects the probabilities of answer tokens. To assess the importance of a component at i-th step, we compute the following intervention effect (IE) metric (Nikankin et al., 2024), averaged over all prompts. The IE measures the mean relative change in the model's output probabilities for both q_i and q_i':

$$\text{IE}(q_i, q_i') = \frac{1}{2}\left[\frac{\text{P}^*(q_i') - \text{P}(q_i')}{\text{P}(q_i')} + \frac{\text{P}(q_i) - \text{P}^*(q_i)}{\text{P}^*(q_i)}\right]$$

[4] In this paper, the scratchpad step index starts at 0.

where $\mathrm{P}(q)$ and $\mathrm{P}^*(q)$ denote the model's predicted probability for token q before and after the intervention, respectively.

We localize the circuit in $\texttt{Transformer}_{+\texttt{CoT}}$ responsible for tracking the world state at each intermediate step in A_5 word problems. As shown in Figure 3, the circuit responsible for state tracking primarily consists of MLPs rather than attention heads. Moreover, the circuit remains largely consistent across steps: MLP0 (McDougall et al., 2023) at the position of m_i and late-layer MLPs at the last position consistently play a key role in state tracking at the i-th step. Overall, given input sequences $(m_1 \dots m_n) \in M^*$ concatenated with scratchpad tokens $(q_1 \dots q_{i-1}) \in M^*$, it is primarily the MLPs—particularly those in the later layers—that implement the state transitions.

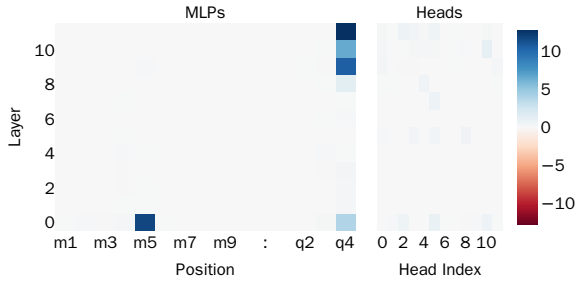


Figure 3: Activation patching results for A_5 word problems of length 10 at 4-th step, with the prompt $\mathbf{p_5} = (m_1 \dots m_{10} | q_1 \dots q_4)$ and the label q_5. The color intensity reflects the magnitude of the IE value. The left and right subfigures represent MLPs and attention heads, respectively, emphasizing that the circuit is primarily composed of MLP0 and late-layer MLPs.

4.2 MLP Neuron Analysis

Having identified the circuit, we then hold deeper component analysis of how late-layer MLPs implement state tracking.

We begin by grouping all possible prompts into distinct subsets based on their resulting states. Specifically, any prompt $(m_1 \dots m_n | q_1 \dots q_{i-1})$ at i-th step belongs to the same subset if it reduces to the same state q_i under the group operation. For simplicity, we refer to such a subset as the $\mathcal{Q}$ prompt subset, where $\mathcal{Q}$ denotes the resulting state. To quantify the contributions of the MLPs, we compute the contribution of each late-layer MLP to promoting $\mathcal{Q}$ in the residual stream using the logit lens (Geva et al., 2021), evaluated across all $\mathcal{Q}$ prompt subsets. As shown in Figure 4, the last three layers of MLP play a significant role, with MLP11 primarily driving the state transition, contributing an average of 72% to the logit increase.

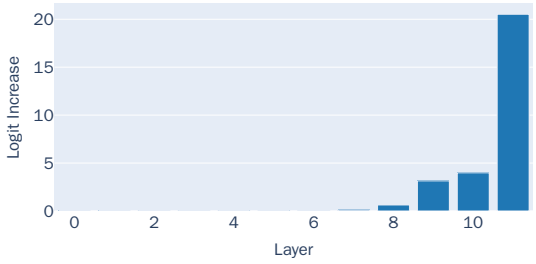


Figure 4: Average logit increase in the layer representation for different MLPs. The final three layers of the MLP primarily contribute to promoting the resulting state $\mathcal{Q}$, with MLP11 contributing the most.

We distinguish prompts $(m_1 \dots m_n | q_1 \dots q_{i-1})$ according to the tuple (m_i, q_{i-1})[5], referred to as (m, q) pairs for brevity. We then compute the activation coefficient m_j^l of the j-th MLP neuron in layer l across all such pairs. Late-layer MLP neurons are selectively activated by a specific subset of (m, q) pairs. Specifically, the top 60 most-activated (m, q) pairs for a given neuron are distributed along individual rows or columns in the two-dimensional (m, q) grid. Notably, the transition rules dictate that exactly 60 (m, q) pairs reduce to the state $\mathcal{Q}$, and these pairs are similarly distributed across rows or columns in the grid. This observation prompts the following research questions:

- **Q1:** Do the observed activation patterns indicate a group of MLP neurons that activated by one $\mathcal{Q}$ prompt subset, where the (m, q) pairs reduce to the same state $\mathcal{Q}$?
- **Q2:** Is it possible to classify MLP neurons based on the resulting states to which their activations correspond?

To address these research questions, we conduct neuron classification experiments (see Appendix D) to categorize all MLP neurons at layer l. Our findings reveal that neurons in layers MLP11, MLP10, and MLP9 can be mapped to specific states with success rates of 90.0%, 79.6%, and 38.7%, respectively. Notably, approximately 15.5% of MLP11 neurons achieve a perfect F1 score relative to the

[5]According to the transition rules, the resulting state q_i is determined by the input m_i and the preceding state q_{i-1}. Hence, each prompt can be represented as a tuple (m_i, q_{i-1}).

ground truth subset of (m, q) pairs, suggesting an N-to-1 mapping between neurons and states.

For Q1, most neurons exhibit high F1 scores with respect to the ground truth subset, indicating that these neurons are selectively activated by the $\mathcal{Q}$ prompt subset. For Q2, we systematically classify activated late-layer MLP neurons across intermediate steps, achieving high success rates. Notably, we observe that 15.5% of MLP11 neurons exhibit an N-to-1 mapping to specific states. These results suggest that transformers with chain-of-thought primarily keep track of the world state via the late-layer MLP neurons. These neurons exhibit selective activation in response to specific subsets of (m, q) pairs and contribute to promoting the correct state token $\mathcal{Q}$ into the residual stream. Furthermore, accurate state transitions are achieved through the collective sub-updates of multiple neurons associated with the target state $\mathcal{Q}$. Beyond the late-layer MLPs, we also analyze another critical component of the circuit: MLP0, which primarily achieves effective embedding of the input m_i, facilitating subsequent state transitions. Further details are provided in Appendix G.

4.3 Transformers with Chain-of-Thought Recover FSAs

Near-perfect next-token prediction does not guarantee that a generative model has fully reconstructed a world model. For instance, in the cumulative Connect-4 task, a generative model with uniform probability can still achieve near-perfect next-token prediction (Vafa et al., 2024). To address this, we focus on state tracking, where world model FSAs can effectively compress distinct sequences converging to the same state while distinguishing sequences leading to different states, regardless of their superficial similarity. This subsection explores how the generative model $\texttt{Transformer}_{+\texttt{CoT}}$, which achieves near-perfect sequence accuracy on word problems, recovers FSAs by analyzing its internal structure at the granularity of MLP neurons.

Our analysis builds on findings from Section 4.2, which suggest that $\texttt{Transformer}_{+\texttt{CoT}}$ performs state tracking by activating specific groups of neurons. To evaluate the model's ability to recover FSAs, we examine its sequence compression and distinction capabilities. For compression, we assess whether the model activates the same set of neurons for prompts converging to the same state. For distinction, we verify whether the model activates distinct neuron sets for prompts leading to different states. To quantify these properties, we propose two metrics. For two prompts, $\mathbf{p_i}$ and $\mathbf{p_i'}$, that converge to the same state, we define the compression metric as:

$$\text{Compression} = \frac{|\mathrm{N}_{\mathbf{p_i}} \cap \mathrm{N}_{\mathbf{p_i'}}|}{|\mathrm{N}_{\mathbf{p_i}} \cup \mathrm{N}_{\mathbf{p_i'}}|}$$

where $\mathrm{N}_{\mathbf{p_i}}$ and $\mathrm{N}_{\mathbf{p_i'}}$ denote the sets of neurons activated by prompts $\mathbf{p_i}$ and $\mathbf{p_i'}$, respectively. Conversely, for prompts $\mathbf{p_i}$ and $\mathbf{p_i'}$ leading to distinct states, we define the distinction metric as:

$$\text{Distinction} = 1 - \frac{|\mathrm{N}_{\mathbf{p_i}} \cap \mathrm{N}_{\mathbf{p_i'}}|}{|\mathrm{N}_{\mathbf{p_i}} \cup \mathrm{N}_{\mathbf{p_i'}}|}$$

We calculate compression and distinction metrics pairwise for all prompts, averaging them at each intermediate step. Our results demonstrate that $\texttt{Transformer}_{+\texttt{CoT}}$ effectively compresses prompts sampled from the $\mathcal{Q}$ prompt subset while distinguishing prompts from different subsets, achieving near-perfect world model recovery with an average metric score of 0.991. This indicates efficient capture of the underlying FSA structure. Extending our analysis beyond the group A_5, we identify similar FSAs for $\mathbb{Z}_{60}$ and $A_4 \times \mathbb{Z}_5$, as well as in larger-scale models, with detailed results in Appendix I. Collectively, we conclude that **Transformers with Chain-of-Thought recover FSAs on state tracking, even at the granularity of MLP neurons**.

5 Robustness and Generalizability of FSAs

Sections 3 and 4 have demonstrated positive results in the recovery of FSAs. However, the word problems considered are too idealistic, with sequence length controlled and noise excluded. In this section, we will investigate the robustness and generalizability of the FSAs within $\texttt{Transformer}_{+\texttt{CoT}}$, considering intermediate step skipping, scratchpad noise, and length generalization. We find that refining the training data distribution can steer the model toward better adaptation to the aforementioned scenarios, except for length generalization. Addressing the latter may require exploring alternative approaches, such as modifying the model architecture, improving training strategies, etc.

5.1 Skipping Steps

In practical training datasets, such as OpenWebMath[6], reasoning trajectories include implicit or skipped intermediate steps. For example, the length of $(q_1 \dots q_n)$ may be less than n, with some intermediate steps skipped. In our experiment settup, we introduce a skipping probability to omit each intermediate state q_i (except q_n) and train $\texttt{Transformer}_{+\texttt{CoT}}$ on this dataset. To investigate the model's ability to perform intermediate step jumps, we employ a linear classifier (Gurnee et al., 2023) to probe the states embedded in the residual stream at the last position. Specifically, we probe the token q_i at the last position, given the prompt $(m_1 \dots m_n | q_1 \dots q_{i-1})$. From the results in Figure 5, we can find that q_i, q_{i+1}, and q_{i+2} are all embedded in the residual stream, suggesting that the model has learned single-step reasoning, two-step reasoning (skipping one step), as well as three-step reasoning (skipping two steps). Additionally, we propose possible mechanisms for skipping and design experiments for further analysis, as detailed in appendix K. Finally, we conclude that incorporating skipping examples into the training set facilitates the adaptation of the FSAs within $\texttt{Transformer}_{+\texttt{CoT}}$ to the scenario.

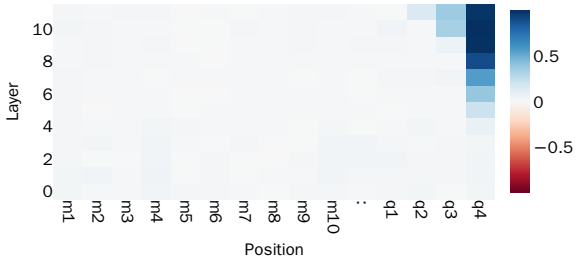


Figure 5: Probing results for state q_i at 4-th step. We can find the positive results appearing two positions earlier, demonstrating that the model has learned skip-step reasoning, including the ability to skip one or two intermediate steps.

5.2 Noise in Scratchpad

We analyze the impact of injecting noise into the scratchpad states $(q_1 \dots q_{i-1})$, where at least one state is false. We categorize noise into two types: (1) any false state except q_{i-1}, and (2) a false q_{i-1}. We evaluate the accuracy of model's next-token prediction under these conditions. For the first type of noise, where noise affects any state other than q_{i-1}, the model maintains near-perfect next-token prediction accuracy. The reason is obvious: the model depends primarily on the input m_i and the preceding state q_{i-1} for the state transition at the i-th step, while disregarding intermediate states prior to q_{i-1}. For the second type of noise, where q_{i-1} is false, the model's performance declines significantly, with next-token prediction accuracy falling to approximately 0.017.

Here, we aim to address the scenario by adjusting the dataset distribution. Specifically, we train the model using data containing the second type of noise in the scratchpad states and achieve a substantial improvement in robustness: the accuracy increases from 0.017 to 0.896. To investigate whether the model accurately recovers the correct state tracking from noise, we employ a linear classifier (Gurnee et al., 2023) at the final position. Specifically, given a corrupted prompt $(m_1 \dots m_n | q_1 \dots \hat{q}_{i-1})$ (where $\hat{q}_{i-1}$ denotes an incorrect state), we probe the correct next state q_i, the false previous state $\hat{q}_{i-1}$ and the true previous state q_{i-1}. We find that the activation sustains positive probing results for $\hat{q}_{i-1}$ across layers until the final one, where it demonstrates a high average accuracy of 0.943 for q_i and 0.379 for q_{i-1}. In autoregressive models, the residual stream at the preceding position of q_{i-2} in the sequence $(m_1 \dots m_n | q_1 \dots \hat{q}_{i-1})$ encodes the correct input state q_{i-1}. Based on this, we propose a possible mechanism that the model's attention heads attend to the activation at the preceding position, thus retrieving the correct input state q_{i-1} and consequently updating the state to q_i. We have also designed experiments to analyze the hypothesis, as detailed in Appendix L. We conclude that refining dataset distribution can help in maintaining the accuracy and reliability of the FSAs inside $\texttt{Transformer}_{+\texttt{CoT}}$.

5.3 Length Generalization

Lastly, we explore length generalization, a key challenge in developing transformer-based LLMs. To enhance length generalization, we replace GPT2's original absolute positional embedding with NoPE (No Positional Embedding), which has been shown in prior work (Kazemnejad et al., 2024) to outperform other explicit positional embedding methods, including both absolute and relative approaches. Notably, in our experiments, NoPE exhibits a certain degree of length generalization, whereas ab-

[6] https://github.com/keirp/OpenWebMath

solute and relative positional embeddings show negligible generalization. We choose LSTM for comparison for the reason that it efficiently learns all investigated word problems on sequences of arbitrary length.

Table 1 illustrates the length generalization capabilities of various models on unseen sequence lengths during training. LSTM achieves perfect generalization, while transformers with chain-of-thought exhibit weak performance. Analyzing the failure from the perspective of MLP neurons, we identify an intriguing "U-turn" phenomenon: the precision of activated neurons, which, after initially decreasing, exhibits an increase during the final few steps of inference. Further details are provided in Appendix M. We conclude that improving the length generalization ability of the model requires other approches, such as modifications to model architecture, loss functions, or optimization algorithms.

	length						
Models	20	21	22	23	24	25	30
LSTM	1.00	1.00	1.00	1.00	1.00	1.00	1.00
$\texttt{Transformer}_{+\texttt{CoT}}$	0.99	0.98	0.87	0.53	0.24	0.09	0.00

Table 1: Comparison of length generalization performance between LSTM and $\texttt{Transformer}_{+\texttt{CoT}}$. Models are trained on sequences of mixed lengths up to 20 and tested on seen (20) and unseen (>20) lengths.

6 Related Work

Theoretical studies (Merrill et al., 2024; Liu et al., 2023a) integrate circuit complexity and algebraic formal language theory to categorize different models and word problems into distinct computational complexity classes, such as TC^0 and NC^1. In particular, Merrill et al. (2024) demonstrate that NC^1-complete word problems cannot be expressed by models within the TC^0 class. In contrast, our work investigates $\texttt{Transformer}_{+\texttt{CoT}}$ through the lens of mechanistic interpretability, revealing a dual advantage in expressiveness and learnability, while also uncovering FSA structures within the model.

Vafa et al. (2024) argue that next-token prediction is a fragile metric for evaluating generative models and propose novel evaluation metrics for world model recovery, drawing inspiration from Myhill-Nerode boundary theory (Myhill, 1957; Nerode, 1958). Their metrics—such as distinction recall and precision—evaluate how effectively a model distinguishes between prefixes that lead to different states. In contrast, our proposed metric operates at the level of individual MLP neurons, measuring how the model internally differentiates between input sequences. This offers a more fine-grained, mechanistic perspective on the model's internal dynamics.

7 Conclusions

In this work, we investigate the learnability of $\texttt{Transformer}_{+\texttt{CoT}}$ through the lens of mechanistic interpretability in state tracking. We begin by reaffirming the success of $\texttt{Transformer}_{+\texttt{CoT}}$ in both in-distribution and out-of-distribution settings. Next, we identify the key components of the circuit, specifically the neurons in the last layers of the MLP, which play a critical role. We find evidence of an implicit FSA within the model, where each state is compressed by a distinct group of neurons. Finally, we evaluate the model in three challenging settings and show that the learned FSAs are robust to noise and skipping but struggle with generalization to significantly longer sequences. This suggests the need for architectural or optimization improvements. Our findings reveal that $\texttt{Transformer}_{+\texttt{CoT}}$ achieves near-perfect performance in next-token prediction while internalizing an FSA-like structured state representation, bridging the gap between expressiveness and learnability. This work provides insights into structured reasoning for sequential tasks.

8 Acknowledgments

This research is supported by the National Natural Science Foundation of China under Grant Nos. 62436006, 62192731, 62192730. Jie Fu is supported by Shanghai Artificial Intelligence Laboratory.

Limitations

In this paper, we define state tracking as word problems involving cyclic and symmetric groups, which form the basis of our conclusions. Our study focuses solely on GPT2 models, leaving the exploration of additional models, such as Llama, for future research. Furthermore, while we have made efforts to address more realistic word problems, state tracking in real-world tasks is intricate, making it challenging to separate state tracing abilities from other skills.

References

Satwik Bhattamishra, Kabir Ahuja, and Navin Goyal. 2020. On the Ability and Limitations of Transformers to Recognize Formal Languages. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7096–7116, Online. Association for Computational Linguistics.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Grégoire Delétang, Anian Ruoss, Jordi Grau-Moya, Tim Genewein, Li Kevin Wenliang, Elliot Catt, Chris Cundy, Marcus Hutter, Shane Legg, Joel Veness, and Pedro A. Ortega. 2023. Neural networks and the chomsky hierarchy. *Preprint*, arXiv:2207.02098.

Angela Fan, Thibaut Lavril, Edouard Grave, Armand Joulin, and Sainbayar Sukhbaatar. 2021. Addressing some limitations of transformers with feedback memory. *Preprint*, arXiv:2002.09402.

Ying Fan, Yilun Du, Kannan Ramchandran, and Kangwook Lee. 2024. Looped transformers for length generalization. *Preprint*, arXiv:2409.15647.

Mor Geva, Avi Caciularu, Kevin Wang, and Yoav Goldberg. 2022. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 30–45, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. *Preprint*, arXiv:2012.14913.

Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. 2024. Think before you speak: Training language models with pause tokens. *Preprint*, arXiv:2310.02226.

Riccardo Grazzi, Julien Siems, Jörg K. H. Franke, Arber Zela, Frank Hutter, and Massimiliano Pontil. 2024. Unlocking state-tracking in linear rnns through negative eigenvalues. *Preprint*, arXiv:2411.12537.

Albert Gu and Tri Dao. 2024. Mamba: Linear-time sequence modeling with selective state spaces. *Preprint*, arXiv:2312.00752.

Albert Gu, Karan Goel, and Christopher Ré. 2022. Efficiently modeling long sequences with structured state spaces. *Preprint*, arXiv:2111.00396.

Wes Gurnee, Neel Nanda, Matthew Pauly, Katherine Harvey, Dmitrii Troitskii, and Dimitris Bertsimas. 2023. Finding neurons in a haystack: Case studies with sparse probing. *Preprint*, arXiv:2305.01610.

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Wenfei Zhou, James Coady, David Peng, Yujie Qiao, Luke Benson, Lucy Sun, Alex Wardle-Solano, Hannah Szabo, Ekaterina Zubova, Matthew Burtell, Jonathan Fan, Yixin Liu, Brian Wong, Malcolm Sailor, Ansong Ni, Linyong Nan, Jungo Kasai, Tao Yu, Rui Zhang, Alexander R. Fabbri, Wojciech Kryscinski, Semih Yavuz, Ye Liu, Xi Victoria Lin, Shafiq Joty, Yingbo Zhou, Caiming Xiong, Rex Ying, Arman Cohan, and Dragomir Radev. 2024. Folio: Natural language reasoning with first-order logic. *Preprint*, arXiv:2209.00840.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. 2024. Training large language models to reason in a continuous latent space. *Preprint*, arXiv:2412.06769.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

John E. Hopcroft. 2007. *Introduction to Automata Theory, Languages, and Computation*, 3rd edition. Pearson Addison Wesley.

L. C. Jain and L. R. Medsker. 1999. *Recurrent Neural Networks: Design and Applications*, 1st edition. CRC Press, Inc., USA.

Amirhossein Kazemnejad, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Payel Das, and Siva Reddy. 2024. The impact of positional encoding on length generalization in transformers. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Najoung Kim and Sebastian Schuster. 2023. Entity tracking in language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3835–3855, Toronto, Canada. Association for Computational Linguistics.

Zhiyuan Li, Hong Liu, Denny Zhou, and Tengyu Ma. 2024. Chain of thought empowers transformers to solve inherently serial problems. *Preprint*, arXiv:2402.12875.

Bingbin Liu, Jordan T. Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. 2023a. Transformers learn shortcuts to automata. *Preprint*, arXiv:2210.10749.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. 2023b. Agentbench: Evaluating llms as agents. *Preprint*, arXiv:2308.03688.

Callum McDougall, Arthur Conmy, Cody Rushing, Thomas McGrath, and Neel Nanda. 2023. Copy suppression: Comprehensively understanding an attention head. *Preprint*, arXiv:2310.04625.

William Merrill, Jackson Petty, and Ashish Sabharwal. 2024. The illusion of state in state-space models. *Preprint*, arXiv:2404.08819.

John Myhill. 1957. Finite automata and the representation of events. *WADD Technical Report*, 57:112–137.

Anil Nerode. 1958. Linear automaton transformations. *Proceedings of the American Mathematical Society*, 9(4):541–544.

Yaniv Nikankin, Anja Reusch, Aaron Mueller, and Yonatan Belinkov. 2024. Arithmetic without algorithms: Language models solve math with a bag of heuristics. *Preprint*, arXiv:2410.21272.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. 2021. Show your work: Scratchpads for intermediate computation with language models. *Preprint*, arXiv:2112.00114.

OpenAI. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Ruizhong Qiu, Zhe Xu, Wenxuan Bao, and Hanghang Tong. 2024. Ask, and it shall be given: Turing completeness of prompting. *Preprint*, arXiv:2411.01992.

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

Shubham Toshniwal, Sam Wiseman, Karen Livescu, and Kevin Gimpel. 2022. Chess as a testbed for language model state tracking. *Preprint*, arXiv:2102.13249.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

Keyon Vafa, Justin Y. Chen, Ashesh Rambachan, Jon Kleinberg, and Sendhil Mullainathan. 2024. Evaluating the world model implicit in a generative model. *Preprint*, arXiv:2406.03689.

Karthik Valmeekam, Kaya Stechly, and Subbarao Kambhampati. 2024. Llms still can't plan; can lrms? a preliminary evaluation of openai's o1 on planbench. *Preprint*, arXiv:2409.13373.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA. Curran Associates Inc.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. 2020. Investigating gender bias in language models using causal mediation analysis. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

Jiewen Yang, Xingbo Dong, Liujun Liu, Chao Zhang, Jiajun Shen, and Dahai Yu. 2022. Recurring the transformer for video action recognition. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14043–14053.

Xiang Zhang, Muhammad Abdul-Mageed, and Laks V. S. Lakshmanan. 2024. Autoregressive + chain of thought = recurrent: Recurrence's role in language models' computability and a revisit of recurrent transformer. *Preprint*, arXiv:2409.09239.

Hattie Zhou, Arwen Bradley, Etai Littwin, Noam Razin, Omid Saremi, Josh Susskind, Samy Bengio, and Preetum Nakkiran. 2023. What algorithms can transformers learn? a study in length generalization. *Preprint*, arXiv:2310.16028.

A Symbol Description

In this section, we briefly introduce the symbols used in this article as in Table 2.

B Separating Elements in Groups

In Section 3, we separate all m into proper subsets to explore the models' performance. As Table 3 shows, we divide m in $\mathbb{Z}_{60}$, $A4 \times \mathbb{Z}_5$ and A_5 according to values, orders and permutation types, respectively. We then prove that the division is reasonable, for the reason that every proper subset can generate the whole group.

Theorem 1 *Every separated proper subset can generate the whole group under group operation.*

Symbol	Description
Σ	Finite set of characters
Q	Finite set of states in the automaton
q_0	Start state, an element of the state set Q
δ	State-transition function
$(M, \cdot)$	A monoid
M^*	Set of all possible sequences from M
e	The starting state
$\mathbb{Z}_k$	Cyclic group of order k
S_k	Symmetric group of order k
TC^0	A complexity class in computational complexity theory, consisting of problems solvable by uniform constant-depth threshold circuits with a polynomial number of gates.
NC^1	A complexity class containing problems solvable by uniform logarithmic-depth Boolean circuits with a polynomial number of gates and bounded fan-in.
$\mathbb{Z}_{60}$	An abelian group encoding mod-60 addition
A_5	The alternating group on five elements
$A_4 \times \mathbb{Z}_5$	A non-abelian but solvable group
$\mathbf{p_i}$	Prompt at the ith step
q_i	Resulting state at the ith step
$\mathcal{Q}$	Resulting state q_i
m_i	The ith input of word problems
q_{i-1}	The preceding state of word problems
(m, q)	(m_i, q_{i-1})
$\hat{q}_i$	Error state
$\mathbf{p_i'}$	Counterfactual prompt at the ith step
q_i'	Counterfactual result state at the ith step
P	Pre-intervention probability distribution
P*	Post-intervention probability distribution
$\mathbf{x^l}$	Input at layer l
$\mathbf{K^l}$	The weight matrix at layer l
$\mathbf{V^l}$	The weight matrix at layer l
d_{mlp}	the MLP intermediate dimension
d_{m}	the model dimension
f	Non-linear activation function
$\mathbf{k_j^l}$	the j-th row vectors of $\mathbf{K^l}$
$\mathbf{v_j^l}$	the j-th row vectors of $\mathbf{V^l}$
m_j^l	Activation coefficient at the jth neuron
$\mathrm{N}_{\mathbf{p_i}}$	Set of activated neurons on prompt $\mathbf{p_i}$
$\mathrm{N}_{\mathbf{p_i'}}$	Set of activated neurons on prompt $\mathbf{p_i'}$

Table 2: Symbols used in this paper

Proof B.1 *For Cyclic group, every element is a generator, so that proper subsets can generate* $h \in \mathbb{Z}_{60}$.

For alternating group, the 3-cycle and 5-cycle in A_5 *can both generate the group.*

For direct product group, we continue the proof as follows:

The elements of order 15 in the direct product group $A_4 \times \mathbb{Z}_5$ *are those elements of the form* (g, h)*, where* $g \in A_4$ *is a 3-cycle (order 3) and* $h \in \mathbb{Z}_5$ *is a nonzero element (order 5). Although these elements of order 15 form a proper subset of the entire group, their projections onto the factors* A_4 *and* $\mathbb{Z}_5$ *separately generate the full groups:*

- *The 3-cycle in* A_4 *can generate* A_4 *(since* A_4 *is generated by its 3-cycles).*
- *Any nonzero element in* $\mathbb{Z}_5$ *is a generator, thus can generate the entire* $\mathbb{Z}_5$.

By the generation theorem of direct product groups, if a subset has surjective projections onto each factor, then the group it generates must be the entire direct product. Thus, all elements of order 15 can generate the entire group $A_4 \times \mathbb{Z}_5$.

We train the model with sequences sampled from one subset, and evaluate on sequences with m sampled from the group. We emphasize that although the model did not encounter mixed sequences during training, the proper subset is able to generate the entire group. Therefore, all possible state transitions are included in the data. $\texttt{Transformer}_{+\texttt{CoT}}$ success in out-of-distribution shows that the learned algorithm does not rely on finding a certain pattern in the input sequence.

Surprisingly, LSTM fails to generalize in the task. We hypothesize that the disparity in out-of-distribution performance stems from the difference in task formulation: the LM task can integrate all elements of the group into the sequence by concatenating each possible state after the input sequence.

C More Experimental Details

In Section 4, we train a GPT2-small model with 12 layers, 12 heads, and a hidden dimension of 768 on a synthetic group dataset. Specifically, we use 2 NVIDIA A100 GPUs, employing mixed-precision bfloat16 for improved computational efficiency. For one group experiment, each run takes approximately 50 minutes. We configure the AdamW optimizer with a weight decay of 0.01 and momentum parameters $\beta = (0.9, 0.999)$, alongside a constant learning rate of 0.0001 and a batch size of 512. The GPT2-small model is trained on a mixture of word problems with n ranging from 2 to 20, and optimized via gradient-based methods with a cross-entropy loss function. Early stopping is implemented once the test accuracy reaches 99%, after which the model is used for MI experiments.

D Classification Algorithms

We provide the late-layer MLP neurons classification procedures as follows:

1. Utilize the priori transition rules to precompute all ground truth sets of (m, q) pairs.
2. Measure all activation coefficient m_j^l across all (m, q) pairs.
3. Utilize the logit lens to calculate the logits of tokens embedded in $\mathbf{v}_\mathbf{j}^\mathbf{l}$, and convert to a 2D pattern, where the cell in index (m, q) is the logit of the result state $\mathcal{Q} = m \cdot q$.
4. Multiply the intermediate results of the previous two steps element-wise, resulting in effective logit contribution of the neuron to the corresponding state for each (m, q) pair.
5. Extract the top 60 (m, q) pairs from the activation pattern as the prediction.
6. Compute the F1 scores between prediction and all labels. The neuron can be classified to state with F1 score no less than threshold $\theta = 0.2$.[7]

E Expected F1 Score of Randomly Activated Neurons

In Section 4.2, we empirically set the value of θ as 0.2 to classify MLP neurons, which is non-trivial compared with random F1 score 0.017. And we provide detailed calculation process of randomly activated neurons' F1 score as follows.

Let S be a set containing 3600 elements: $S = \{0, 1, 2, \ldots, 3599\}$, and let L be the label subset consisting of the first 60 elements: $L = \{0, 1, 2, \ldots, 59\}$. We randomly sample 60 elements from S to form a subset x: $x \subseteq S$ and $|x| = 60$. Our goal is to compute the expected values of precision, recall, and F1 score for the subset x with respect to the label subset L.

Define $T = |x \cap L|$ as the number of correctly selected elements. Since x is chosen uniformly at random from S, and given that L contains 60

[7] Here the value of θ is an empirical setting, significantly higher than the random value of 0.017. See Appendix E for details on computing the random value.

Group	Proper Subsets
A_5	**Identity and Double Transpositions:** 0, 3, 8, 11, 12, 13, 14, 27, 30, 33, 41, 43, 47, 53, 55, 59 **3-Cycles:** 1, 2, 4, 5, 6, 7, 9, 10, 15, 19, 22, 24, 28, 29, 37, 39, 40, 49, 51, 52 **5-Cycles:** 16, 17, 18, 20, 21, 23, 25, 26, 31, 32, 34, 35, 36, 38, 42, 44, 45, 46, 48, 50, 54, 56, 57, 58
$A4 \times \mathbb{Z}_5$	**Order 15:** 6, 7, 8, 9, 11, 12, 13, 14, 21, 22, 23, 24, 26, 27, 28, 29, 31, 32, 33, 34, 36, 37, 38, 39, 46, 47, 48, 49, 51, 52, 53, 54 **Other Orders:** 0, 15, 40, 55, 5, 10, 20, 25, 30, 35, 45, 50, 1, 2, 3, 4, 16, 17, 18, 19, 41, 42, 43, 44, 56, 57, 58, 59
$\mathbb{Z}_{60}$	< 30: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29 ≥ 30: 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59

Table 3: Division of elements for three groups in out-of-distribution.

elements, T follows a hypergeometric distribution. Its expected value is:

$$\mathbb{E}[T] = \frac{|x| \cdot |L|}{|S|} = \frac{60 \times 60}{3600} = 1.$$

The precision and recall are defined by

$$\text{Precision} = \frac{|x \cap L|}{|x|} = \frac{T}{60}$$

$$\text{Recall} = \frac{|x \cap L|}{|L|} = \frac{T}{60}$$

Thus, their expected values are

$$\mathbb{E}[\text{Precision}] = \mathbb{E}\left[\frac{T}{60}\right] = \frac{\mathbb{E}[T]}{60} = \frac{1}{60},$$

$$\mathbb{E}[\text{Recall}] = \frac{1}{60}.$$

The F1 score is the harmonic mean of precision and recall:

$$\text{F1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}.$$

Since $\text{Precision} = \text{Recall} = \frac{T}{60}$, we substitute to obtain

$$\text{F1} = \frac{2 \cdot \frac{T}{60} \cdot \frac{T}{60}}{\frac{T}{60} + \frac{T}{60}} = \frac{2 \cdot \frac{T^2}{3600}}{\frac{2T}{60}} = \frac{T}{60}.$$

Hence, the expected F1 score is

$$\mathbb{E}[\text{F1}] = \mathbb{E}\left[\frac{T}{60}\right] = \frac{1}{60} \approx 0.01667.$$

F Activated Neurons Analysis

Having classified neurons into states as described in Section 4.2, We can compute the precision and recall of the activated neurons, averaged over all $\mathcal{Q}$ prompt subsets, during the model's single-step state transitions. Specifically, we select the top-K activated neurons based on the activation coefficient m_j^l and calculate precision and recall using the classified neurons as ground truth labels. The results presented in Figure 6 indicate that across intermediate steps, the model activates MLP11 neurons with high average precision (0.797) but relatively low average recall (0.253). We emphasize that the high precision accounts for the model's accuracy in solving word problems, whereas the low recall suggests that the neurons activated by a given $\mathcal{Q}$ prompt subset can vary across different steps.

G MLP0 in Circuit

Activation patching results in Section 4.1 show that the circuit mainly consists of MLP0 and late-layer MLPs. Given a prompt $(m_1 \dots m_n | q_1 \dots q_{i-1})$, the late-layer MLPs implement state transition in the postion q_{i-1}, while the MLP0 mainly achieves effective word embedding at the position of m_i. As Figure7 shows, the MLP0 promotes input m_i into the residual stream, which will then be transferred to the last position of q_{i-1} by attention heads for

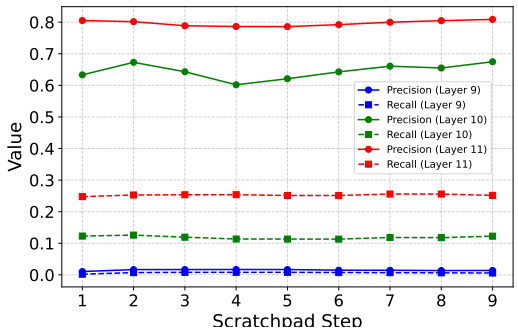


Figure 6: Averaged precision and recall of top-K activated neurons across different intermediate steps.

subsequent state transitions.

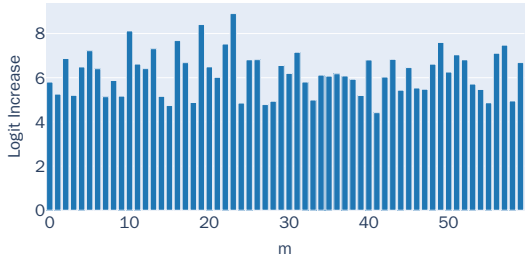


Figure 7: Average input m logit increase in the layer representation for MLP0. The MLP0 mainly implements effective word embedding in the circuit.

H Attention Heads

The activation patching results in Section 4.1 reveal that the circuit is primarily composed of MLPs, which predominantly encode state transitions. Additionally, attention heads within the circuit facilitate the flow of information across positions. We identify the principal attention heads using the activation patching results shown in Figure 3 and analyze their activation patterns. The findings, presented in Figure 9, reveal two distinct attention patterns:

1. Across all steps, specific attention heads directly transfer information from the position of m_i to the position of q_{i-1}.
2. The model develops a secondary pattern: information from positions $m_1 \dots m_n$ is relayed backward to the position of ":", which separates the input and scratchpad and marks the start of state tracking, thus enabling the model to attend to this relay position during inference.

I Automta still exists on other groups with large model scale.

We have found a FSA within GPT2-small on A_5 as in Section 4.3, in this section, we will extrapolate this conclusion to other groups and larger model size. We first conduct analogous experiments on $A_4 \times \mathbb{Z}_5$ and $\mathbb{Z}_{60}$. We find that MLP11 neurons can still be classified according to transition rules, and the model retains the ability to compress and distinguish different input sequences, achieving average compression and distinction metrics of 0.992 and 0.997, respectively. Moreover, we explore whether FSAs still exist with model size larger. Specifically, we repeat the experiments, increasing the model size from GPT2-small (124M) to GPT2-large (744M), and find FSAs still exist. Results are shown in Figure 10.

J Noisy State Tracking

We make an assumption that skipping, noise, and length generalization are present in real-world scenarios. For example, consider the following example sampled from the OpenWebMath corpus. The CoT reasoning misses a step in computing the derivative of $2x$ in the second to last line. Nevertheless, given the vast scale of the pretraining corpus, it is hard to quantify the proportion of noisy reasoning. Due to the existence of noise and the difficulty from vast corpora, we investigate the robustness of FSAs through controlled experiments.

What is the derivative of $f(x) = (e^{2x})(\ln(x))$?
Mar 3, 2017
$f'(x) = e^{2x}\left(2\ln x + \frac{1}{x}\right)$
Explanation:
The derivative of $\ln x$ *is* $\frac{1}{x}$
The derivative of $e^{g(x)}$ *is* $e^{g(x)} \cdot g'(x)$
The derivative of $h(x) \cdot l(x)$ *is* $h'(x) \cdot l(x) + h(x) \cdot l'(x)$
Then
$f'(x)$
$= (e^{2x})' \cdot \ln x + e^{2x} \cdot \frac{1}{x}$
$= e^{2x} \cdot 2 \cdot \ln x + e^{2x} \cdot \frac{1}{x}$
$= e^{2x}\left(2\ln x + \frac{1}{x}\right)$

K Skipping Steps

According to the probing results in Figure 5, we propose two possible mechanisms for skipping in $\texttt{Transformer}_{+\texttt{CoT}}$: one is to use continuous layers to achieve skip-step state transitions, and the

other is to use a single layer to achieve skip-step transitions. The key difference is that the former depends on the skipped states, while the latter does not. To explore what algorithms the model uses in skipping, we design intervention experiments by suppressing skipped tokens. Specifically, given the prompt $(m_1 \dots m_n | q_1 \dots q_{i-1})$, we evaluate the probing changes for q_{i+1} at the last position, while suppressing token q_i. We observe that suppressing q_i significantly reduces positive probing results to nearly zero at the final position with skipping probabilities varying from 0 to 1, suggesting that $\texttt{Transformer}_{+\texttt{CoT}}$ implements skipping through continuous layers.

L Noise in Scratchpad

Given a corrupted prompt $(m_1 \dots m_n | q_1 \dots \hat{q}_{i-1})$ (where $\hat{q}_{i-1}$ represents an incorrect state), we probe the incorrect previous state $\hat{q}_{i-1}$, the correct previous state q_{i-1}, and the correct next state q_i at the last position. The results in Figure 11 show that the activation maintains positive probing results for $\hat{q}_{i-1}$ across layers, up to the final one, where it exhibits a high average accuracy of 0.943 for q_i and 0.379 for q_{i-1}.

In addition, we propose one possible mechanism that at the i-th step, the model mitigates the influence of the incorrect input state $\hat{q}_{i-1}$ in the final layer through attending to the preceding position. This allows it to recover the correct input state q_{i-1} and accurately update to q_i. To verify this, we mask the position of q_{i-2} in the final layer and analyze the resulting changes in probing accuracy for state q_i at the last position. The results in Figure 11 show that after masking, the average score drops from above 0.9 to nearly zero, indicating that attending to q_{i-2} in the last layer is crucial for accurate state transitions. In contrast, we hold ablation experiments with a model trained without noise, and we observe minimal change in the probing accuracy for state q_i while masking. This suggests that the FSA may develop adaptive attention mechanisms to deal with noise in the scratchpad.

M Length Generalization

Numerous factors contribute to poor length generalization in models. Here, we analyze the model's failure to generalize to longer sequences from the perspective of MLP neurons. We evaluate the precision and recall of activated neurons (see Section F) on word problems with sequence lengths exceeding those in the training set. As shown in Figure 8, both precision and recall drop significantly when the scratchpad steps surpass the maximum steps encountered during training. Notably, when the sequence length slightly exceeds the training range, a "U-turn" phenomenon emerges in precision during the final inference steps, which vanishes as the length increases further. To investigate this "U-turn" phenomenon, we conducted experiments with models trained on various length ranges. Our findings reveal that this effect occurs within a margin of approximately 150% of the training length, beyond which it disappears. The phenomenon suggests that activating the correct neurons may not be the primary factor limiting length generalization. In conclusion, enhancing the model's length generalization capability likely requires alternative approaches, such as modifications to the model architecture, loss function, or optimization algorithm.

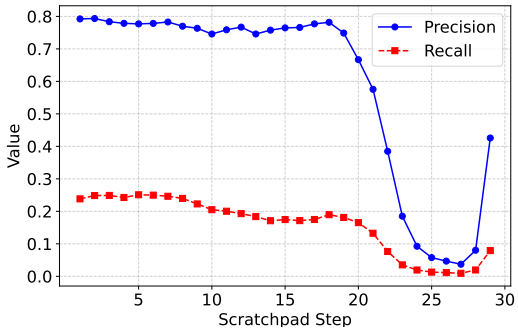
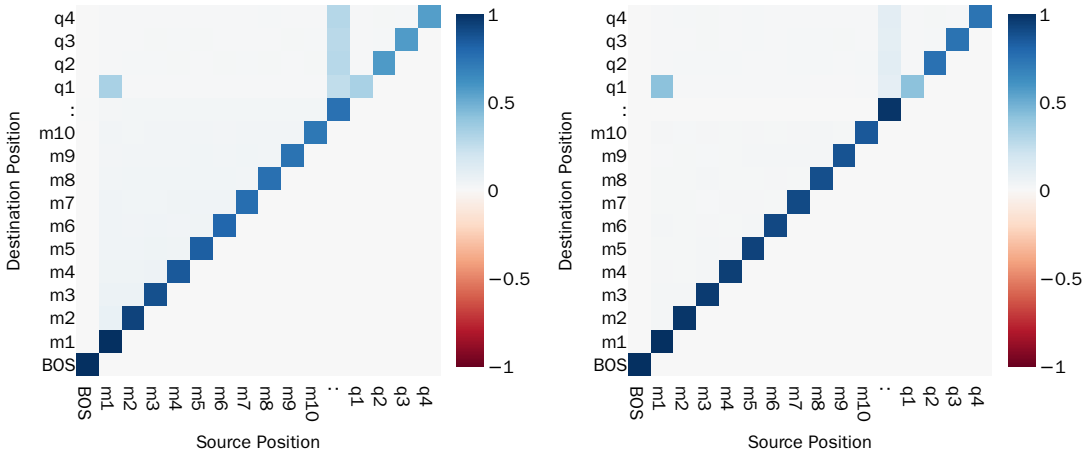
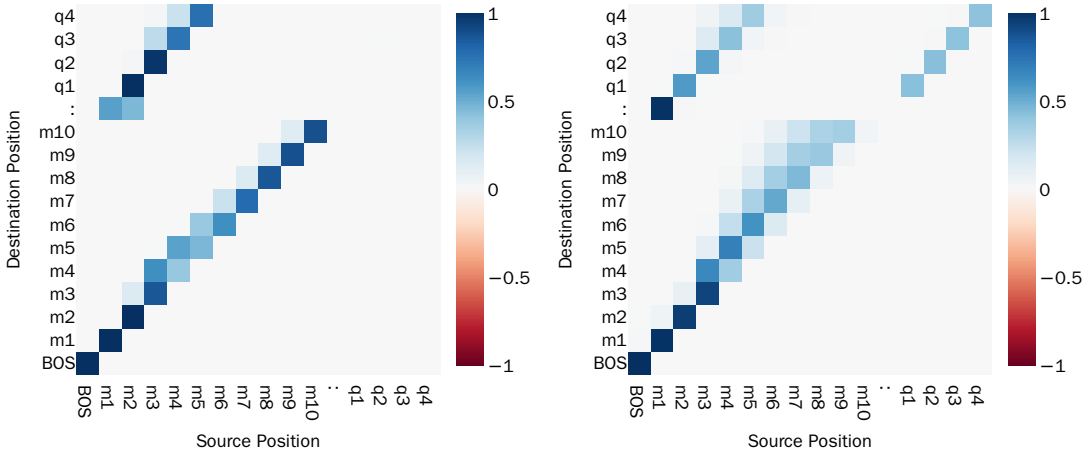


Figure 8: MLP11 neurons precision and recall across intermediate steps on word problems with length 30. The model is trained on sequences with length ranging from 2 to 20.

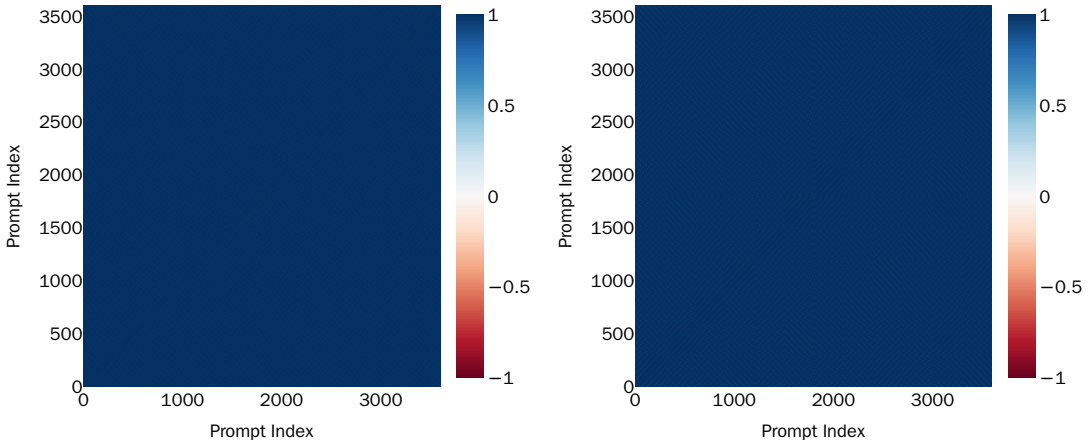


(a) (b)

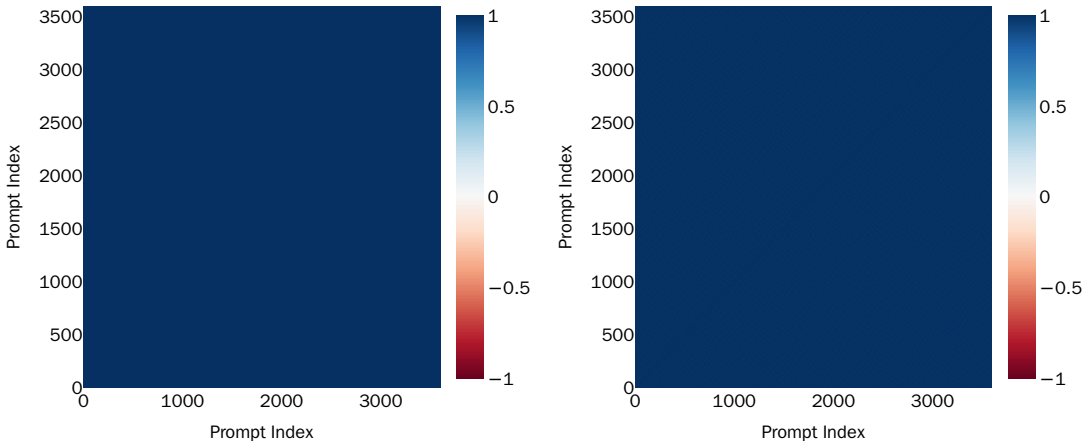


(c) (d)

Figure 9: Attention patterns of the principal attention heads. Specific attention heads send information directly from m_i to q_{i-1}. Besides, the model learns another pattern: it passes information backward from m_1 to m_n backwards until the relay position of ":", so that the model can attend to the relay position during inference.



(a) (b)



(c) (d)

Figure 10: (a): Compression and distinction metrics for MLP11 on A_5. (b): Compression and distinction metrics for MLP11 on $A_4 \times \mathbb{Z}_5$. (c): Compression and distinction metrics for MLP11 on $\mathbb{Z}_{60}$. (d): Compression and distinction metrics for MLP35 on A_5 with larger model size (GPT2-large). The entire square's dark color can be interpreted as the metric being nearly 1 for any pairwise combination of prompts.

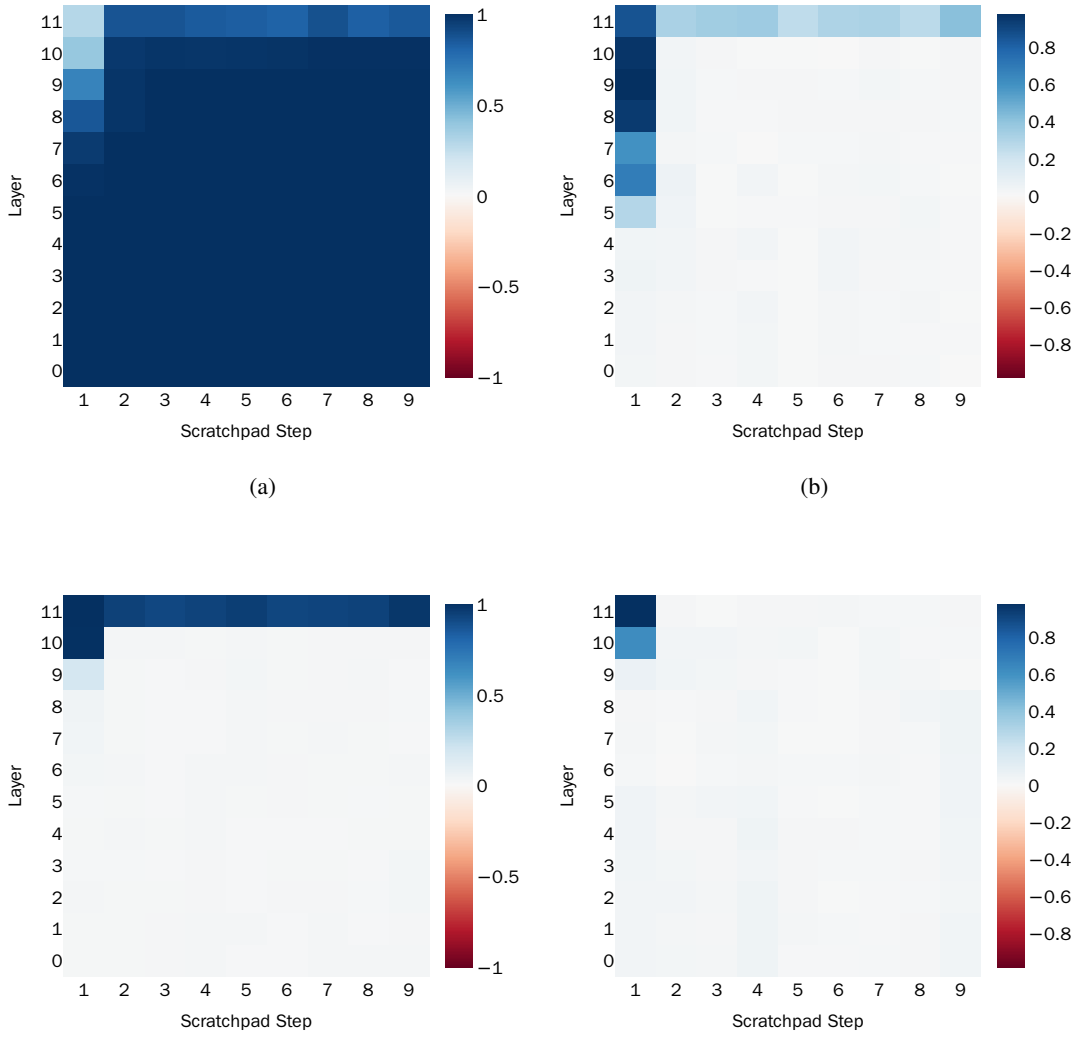


Figure 11: (a): Probing results for $\hat{q}_{i-1}$. (b): Probing results for q_{i-1}. (c): Probing results for q_i. (d): Probing results for q_i while masking the preceding position. The model activation maintains high probing performance for $\hat{q}_{i-1}$ across layers, reaching a final layer average accuracy of 0.943 for q_i and 0.379 for q_{i-1}. However, masking the position of q_{i-2} leads to a sharp drop in the probing score—from above 0.9 to nearly zero.