Loop, Think, & Generalize: Implicit Reasoning in Recurrent-Depth Transformers

Harsh Kohli **Srinivasan Parthasarathy** **Huan Sun*** **Yuekun Yao***
The Ohio State University
{kohli.120,parthasarathy.2,sun.397,yao.1267}@osu.edu
https://github.com/OSU-NLP-Group/Loop-Think-Generalize

Abstract

We study implicit reasoning, i.e. the ability to combine knowledge or rules within a single forward pass. While transformer-based large language models store substantial factual knowledge and rules, they often fail to compose this knowledge for implicit multi-hop reasoning, suggesting a lack of compositional generalization over their parametric knowledge. To address this limitation, we study recurrent-depth transformers, which enables iterative computation over the same transformer layers. We investigate two compositional generalization challenges under the implicit reasoning scenario: *systematic generalization*, i.e. combining knowledge that is never used for compositions during training, and *depth extrapolation*, i.e. generalizing from limited reasoning depth (e.g. training on up to 5-hop) to deeper compositions (e.g. 10-hop). Through controlled studies with models trained from scratch, we show that while vanilla transformers struggle with both generalization challenges, recurrent-depth transformers can effectively make such generalization. For systematic generalization, we find that this ability emerges through a three-stage grokking process, transitioning from memorization to in-distribution generalization and finally to systematic generalization, supported by mechanistic analysis. For depth extrapolation, we show that generalization beyond training depth can be unlocked by scaling inference-time recurrence, with more iterations enabling deeper reasoning. We further study how training strategies affect extrapolation, providing guidance on training recurrent-depth transformers, and identify a key limitation, *overthinking*, where excessive recurrence degrades predictions and limits generalization to very deep compositions.

1 Introduction

Large language models (LLMs) (Brown et al., 2020) are known to acquire substantial factual knowledge during pretraining, storing it in their parameters (Geva et al., 2023). However, how effectively this knowledge can be composed for reasoning remains less understood (Dziri et al., 2023; Press et al., 2023). In particular, recent work shows that transformer-based LLMs struggle under **implicit reasoning**, i.e. reasoning within a *single forward pass* without explicit chain-of-thought (CoT) (Wei et al., 2022). Such failures reveal a fundamental limitation of transformers: despite storing rich knowledge, they are often unable to flexibly combine it to solve novel questions. This limitation has important implications for generalization, as many tasks require composing multiple pieces of seen knowledge in novel ways not observed during training (Lake & Baroni, 2018; Berglund et al., 2023).

Why do transformers struggle to combine their parametric knowledge in implicit reasoning? Consider a query such as *"The spouse of the performer of Imagine is"*. Previous work shows that transformers solve this by chaining two facts: first retrieving that *the performer of Imagine is John Lennon* in shallow layers, and then that *the spouse of John Lennon is Yoko Ono* in deeper layers (Biran et al., 2024; Wang et al., 2024a; Yang et al., 2024b). However, since knowledge

*Equal supervision.

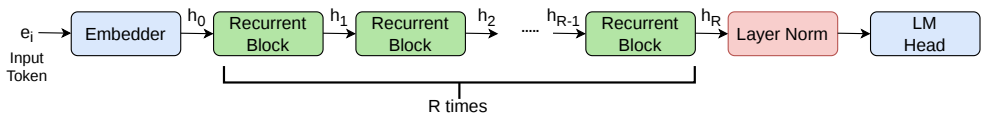

Figure 1: **Recurrent depth model architecture.** The transformer block is repeated R times. The embedding layer and language model head (LM Head) have tied weights. In our experiments, we use a simple looped transformer similar to Saunshi et al. (2025) without design elements such as input injection, gated halting, and middle looping.

is distributed across different layers of the transformer, there is no guarantee that the fact required for a particular query can be accessed correctly. For example, if the fact *the spouse of John Lennon is Yoko Ono* is only stored in shallow layers, deeper layers cannot access it because parameters are not shared across layers. While transformers can be trained to learn to combine such knowledge properly (Wang et al., 2024a; Yao et al., 2025), they fail to compositionally generalize to unfamiliar combinations or deeper recursive combinations.

To address this limitation, we introduce depth-recurrence into transformers, allowing the same set of layers to be applied iteratively. The input sequence is processed multiple times by a shared transformer block, where the output of each iteration serves as input to the next. In contrast to vanilla transformers, where knowledge is tied to specific layers, recurrence enables more flexible access to and composition of parametric knowledge within a single forward process. Such models, known as *recurrent-depth transformers* or *looped transformers*, have recently gained attention as a promising architecture (Dehghani et al., 2019; Geiping et al., 2025; Zhu et al., 2025). While prior work has shown that recurrent-depth transformers improve length generalization (Bansal et al., 2022; Fan et al., 2025), it remains unclear whether they can overcome compositional generalization limitations when reasoning over parametric knowledge.

In this paper, we systematically study whether recurrent-depth transformers can compositionally combine their parametric knowledge *implicitly*. By constructing synthetic datasets, we train models to learn implicit reasoning from scratch. Unlike LLMs trained on vast, opaque web-scale corpora, this setup provides control over the data and mitigates confounding biases introduced during pretraining. Specifically, we characterize two challenges: *systematic generalization* (combining knowledge not used in any composition during training) and *depth extrapolation* (e.g., training on 5-hop reasoning and evaluating on 10-hop).

Our main findings are two-fold. First, **recurrent-depth transformers exhibit strong systematic generalization, while vanilla transformers fail to do so**. We show that this ability emerges through a sharp three-stage *grokking* process, that transitions from memorization to in-distribution generalization, and finally to systematic generalization. We also support this with evidence from the internal activations of models across different training stages.

Second, **recurrent-depth transformers enable depth extrapolation**, generalizing to reasoning depths beyond those observed during training, as inference-time compute (i.e., recurrent iterations) increases. We further find that the training-time recurrence strategy plays a critical role in extrapolation performance, with dynamic recurrence achieving the strongest generalization. Despite these gains, we identify a key limitation: recurrent-depth transformers suffer from overthinking (Bansal et al., 2022), which degrades performance and limits generalization to extremely deep recursions.

2 Related Work

Several small-scale studies pretrain looped or recurrent-depth transformers on synthetic tasks to better understand their behavior in a controlled setting. Our work best aligns with such studies where we are able to cleanly attribute differences in performance and generalization to specific architectural choices and model design decisions. Yang et al. (2024a) demonstrate how "looping" a transformer block helps to better emulate learning algorithms

such as gradient descent for in-context linear regressions, 2-layer neural networks, and decision trees. Fan et al. (2025) show that such looped transformers offer superior length generalization on algorithmic tasks such as parity and binary addition. Saunshi et al. (2025) conduct a larger-scale pretraining with the 250B tokens of the Pile dataset (Gao et al., 2020) and find that looped versions of transformer models of the same effective depth have a greater inductive bias towards reasoning at the cost of memorization and perplexity. Based on these results, they propose a regularization term that encourages certain layers to be closer to each other, thus improving the tradeoff between reasoning and fact recall.

Relative to other works, our targeted setting yields unique insights on training dynamics and model behavior. We demonstrate how weight sharing through recurrence can solve systematic composition where vanilla transformers are known to struggle and extrapolation in multi-hop composition is possible with increased recurrence at inference-time. While Fan et al. (2025) propose looped architectures for length generalization, they assume an oracle number of training iterations based on sample complexity. We believe that our setup is closer to real-world scenarios where task complexity cannot be easily estimated through heuristics (such as input length). Without the assumption of task complexity a priori, we face distinct challenges in training our models. We analyze how best to apply methods like recurrent-depth and common pitfalls to avoid, which can help inform more robust implicit reasoning models in the future. We discuss other related work in Appendix D.

3 Task Formulation

We formally define our implicit reasoning setup using a synthetic multi-hop reasoning task, and categorize three generalization challenges under this formulation: in-distribution generalization, systematic generalization, and depth extrapolation (Figure 2). The latter two can be viewed as out-of-distribution (OOD) generalization. Such tasks have been shown to be difficult for vanilla transformers to learn (Yao et al., 2025), highlighting their limitations in composing parametric knowledge for reasoning (Allen-Zhu & Li, 2023; Yang et al., 2024b).

3.1 Task Definition

Our implicit reasoning task relies on a directed knowledge graph (KG) where nodes represent a set of entities $E = \{e_i\}$ and edges represent a set of relations $R = \{r_j\}$. The KG is composed of atomic (1-hop) facts, each taking the form of a triplet (h, r, t), where $h, t \in E$ and $r \in R$ (here h and t imply the head and tail entities, respectively).

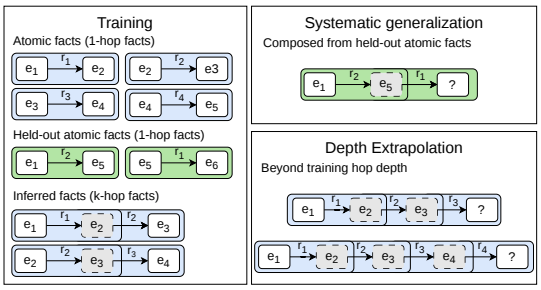

Figure 2: Illustration of systematic and extrapolation generalization tasks with a sample dataset.

A k-hop inferred fact is defined as a chain of k atomic facts connecting a head entity h to a final tail entity t via a sequence of $k-1$ intermediate entities $(i_1, \ldots, i_{k-1})$:

$$(h, r_1, i_1), (i_1, r_2, i_2), \ldots, (i_{k-1}, r_k, t)$$

Given the head entity h and the sequence of relations $r_1, \ldots, r_k$, we use an auto-regressive decoder-only model to predict the final tail entity t. The input prefix is $< e_h >< r_1 >< r_2 > \cdots < r_k >$, and the target is $< e_t >$. Ideally, the model must implicitly perform the k-hop traversal, successively retrieving each intermediate entity $(i_1, \ldots, i_{k-1})$ until it can resolve the final tail entity t.

3.2 Generalization Challenges

Given a generated knowledge graph, we first define the complete atomic fact set as $\mathcal{C} = \{(h, r, t)\}$, and the induced set of k-hop inferred facts from $\mathcal{C}$ as

$$\mathcal{I}_k(\mathcal{C}) = \{(h, r_1, \ldots, r_k, t) \mid \exists\, i_1, \ldots, i_{k-1},\ (h, r_1, i_1), \ldots, (i_{k-1}, r_k, t) \in \mathcal{C}\}.$$

Training set. The training set includes two parts: all possible atomic facts $\mathcal{C}$ together with a set of inferred facts $\mathcal{I}_{train}$ up to a maximum depth k_{train} (e.g. k-hop facts with $k \in [2, k_{train}]$). To characterize different generalization challenges, we partition the atomic fact set into two disjoint subsets $\mathcal{C} = \mathcal{C}_{ID} \cup \mathcal{C}_{OOD}$. The training inferred facts $\mathcal{I}_{train}$ can then be defined as

$$\mathcal{I}_{train} = \{(h, r_1, \ldots, r_k, t) \mid (h, r_1, \ldots, r_k, t) \in \mathcal{I}_k(\mathcal{C}_{ID}),\ k \leq k_{\text{train}}\}.$$

We then define three generalization challenges:

In-distribution generalization. The model is evaluated on inferred facts $(h, r_1, \ldots, r_k, t) \in \mathcal{I}_k(\mathcal{C}_{ID})$ that are not observed during training, equivalent to randomly sample inferred facts from $\mathcal{I}_k(\mathcal{C}_{ID})$ as held-out test set. Despite its simplicity, previous work shows that vanilla transformers can only learn such tasks through extended training (Wang et al., 2024a).

Systematic generalization. The model is evaluated on inferred facts $(h, r_1, \ldots, r_k, t) \in \mathcal{I}_k(\mathcal{C}_{OOD})$, which are induced from atomic facts that are never used in compositions in the training data. This requires the learner to systematically combine its learned knowledge, without having seen combinations of it in training. This setting simulates scenarios where knowledge appears only as plain text in pretraining data (e.g. long-tail knowledge), but never forms answers to reasoning queries during training. Previous work (Wang et al., 2024a) shows that vanilla transformers completely fail on this generalization challenge.

Depth extrapolation. The model is evaluated on inferred facts of greater depth than those included in the training dataset, i.e., $(h, r_1, \ldots, r_k, t) \in \mathcal{I}_k(\mathcal{C}_{\mathcal{ID}})$ with k larger than k_{train}. Solving this requires the learner to infer the underlying rules of the task and iteratively apply them at depths far beyond those observed during training. This setting simulates scenarios where the complexity (i.e. reasoning depth) of training data is limited due to budget constraints, yet we expect the model to generalize beyond training. Such depth generalization poses challenges for vanilla transformers in symbolic (Kim & Linzen, 2020) and knowledge reasoning (Yao et al., 2025). Although related to length generalization, depth extrapolation is conceptually distinct: it measures the depth to which a model can repeatedly apply learned rules over its parametric knowledge beyond training.

4 Recurrent-Depth Transformer

Model architecture. Across all experiments we use a decoder-only transformer with a *recurrent-depth* design illustrated in Figure 1. Concretely, we instantiate a GPT-2 style block with L layers and reuse this for R recurrent iterations, yielding an effective rolled-out depth of $D = L \times R$ layers. At each recurrent iteration the same stack of layers is applied to the current hidden states. This allows the model to allocate more computation (increasing R) at inference time without changing the architecture or re-training the parameters. We exploit this property in our inference-time scaling experiments described in Section 6. Formally, let f_θ denote the L transformer layers with shared parameters θ, and let $h^{(0)}$ be the input sequence after the initial embedding layer. The model computes

$$h^{(r+1)} = f_\theta\big(h^{(r)}; m\big), \qquad r = 0, \ldots, R-1,$$

where m denotes the causal attention and padding masks. The final representation $h^{(R)}$ is passed through a final layer normalization and a tied output projection to produce logits over the vocabulary at each position. We only supervise the next-token distribution at the final position corresponding to the tail entity t. Each entity and relation is represented by a dedicated token ($< e_i >$, $< r_j >$), and the query prefix is mapped to token embeddings.

We adopt a zero-initialization strategy to stabilize training under repeated application of shared weights. Specifically, we initialize the output projection matrices (`c_proj`) of both the multi-head attention and feed-forward blocks to zero, so that each recurrent block is an exact identity mapping at initialization. This ensures that the input-output Jacobian remains stable even when the model is unrolled to a large number of recurrent iterations. This design is motivated by the known instability of deep networks with shared parameters (Agarwala & Schoenholz, 2022), which becomes particularly pronounced in recurrent-depth transformers (Saunshi et al., 2025). Following Zhang et al. (2019), this initialization supports stable optimization under unbounded unrolling of the recurrent iterations.

Stopping strategies of the recurrent iterations. Training the looped transformer requires a stopping strategy to determine the number of recurrent iterations in the forward pass on the input. We consider two stopping strategies: *fixed iteration* and *dynamic iteration*. Fixed iteration determines the recurrent iterations to be the same fixed value for all training instances. The dynamic iteration strategy samples the number of recurrent iterations independently for each training batch. Concretely, for the dynamic model we sample

$$R \sim \text{clip}\big(\text{Poisson}(\lambda), R_{\text{min}}, R_{\text{max}}\big),$$

Unless otherwise stated, we set the hyperparameters to $R_{\text{min}} = 2$ and $R_{\text{max}} = 8$. Such strategies have been shown to be effective in realistic pretraining (Geiping et al., 2025; Zhu et al., 2025), and here we adopt a simple Poisson distribution to control the sampling distribution. Importantly, our strategies contrast with prior studies on the generalization ability of looped transformers, where the iteration number is matched to the complexity of each training instance, assuming oracle access to such complexity. Instead, our setup reflects practical scenarios (Geiping et al., 2025), where the complexity is unknown and computation cannot be allocated precisely in advance.

5 Systematic Generalization

In this section, we study systematic generalization, i.e. whether models can combine parametric knowledge not composed during training for multi-hop tasks. We focus on 2-hop, as Wang et al. (2024a) shows that vanilla transformers already struggle with this simple task.

5.1 Experiment Setup

Dataset. We construct the dataset by instantiating a knowledge graph with $|E| = 2000$ and $|R| = 200$, where each entity has average out-degree 20. We then include all 40k atomic facts, randomly partitioned with 95% C_{ID}, and 5% C_{OOD}, together with 273.6k inferred facts for training. Our in-distribution set includes 3k held-out two-hop inferred facts composed from C_{ID}, and OOD set includes nearly 2k two-hop inferred facts composed from C_{OOD}.

Model. We train our looped transformer with $L = 4$ layers, and use fixed training recurrence with $R \in \{1, 2, 4, 8\}$. The model with $R = 1$ is equivalent to a 4-layer vanilla transformer. We evaluate the accuracy of the predicted token against the gold answer. Absolute position embeddings (APE) (Vaswani et al., 2017) are used as positional embeddings in this setup. We do not use dynamic recurrence in this setting, as systematic generalization in the 2-hop task already emerges from weight sharing under fixed recurrence. In the multi-hop setting, however, it improves extrapolation to more complex samples at inference time and helps alleviate latent overthinking, as discussed in Section 6.

5.2 Results

Recurrent-depth transformers perform systematic generalization, while vanilla transformers do not. On the left of Figure 3, we plot OOD accuracy as a function of training epochs. We find that the vanilla transformer (i.e. $R = 1$) completely fails when the task requires combining unfamiliar atomic facts, while even the simplest $R = 2$ recurrence achieves non-trivial generalization performance. Increasing training iterations further accelerates the

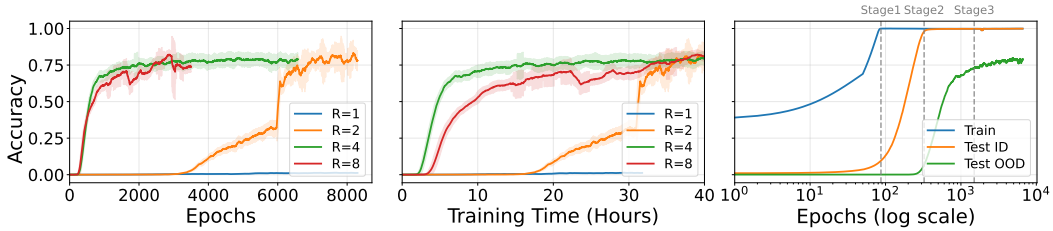

Figure 3: Accuracy curves for recurrent-depth models across training epochs and wall-clock time. **Left:** Test OOD accuracy for models trained with $R \in \{1, 2, 4, 8\}$, plotted against training epochs. Curves are smoothed with a 100-epoch rolling mean, with shading indicating standard deviation. **Middle:** Test OOD accuracy for the same models, plotted against training wall-clock time (hours). **Right:** Accuracy of the $R = 4$ model on the examples from training, ID, and OOD test splits, plotted against training epochs.

convergence, e.g. $R = 4$ converges with 2k epoch, while $R = 2$ takes 7k. This acceleration is not only in terms of training steps, but also in absolute wall-clock time (Figure 3, middle).

Systematic generalization emerges through a three-stage grokking dynamic. We further analyze the training dynamics of the $R = 4$ model (right panel of Figure 3) to understand how systematic generalization emerges. We observe a three-stage dynamic: In the first stage, the model overfits the training set, with only training accuracy improving. In the second stage, in-distribution generalization emerges after prolonged training beyond memorization, a phenomenon referred to as *grokking*. In the final stage, systematic generalization arises only after the model achieves near-perfect in-distribution accuracy, occurring at a much later point than training overfitting (e.g., 10^4 vs. 10^2 epochs).

Analyzing model internals with logit lens. We use the logit lens technique (nostalgebraist, 2020) to examine how models represent the bridge entity and the final target during different stages of training. After each layer and recurrent iteration, we project the intermediate hidden states through the final layer norm and language modeling head to obtain logits over the output vocabulary. For 2-hop inputs of the form (h, r_1, r_2), where h is the head entity and r_1, r_2 are the two relations, we measure at each effective depth the accuracy of predicting the bridge entity at the r_1 position and the target entity at the r_2 position. Figure 4 shows logits lens for an $R = 2$ recurrent-depth model on Training, Test ID, and Test OOD splits, across checkpoints corresponding to the three training stages. We compare against an iso-FLOP 8-layer vanilla transformer with matched effective depth. The vanilla model exhibits only two training stages and fails to achieve non-zero systematic generalization regardless of training time, consistent with Wang et al. (2024a). We also include a causal analysis for systematic generalization in Appendix E.

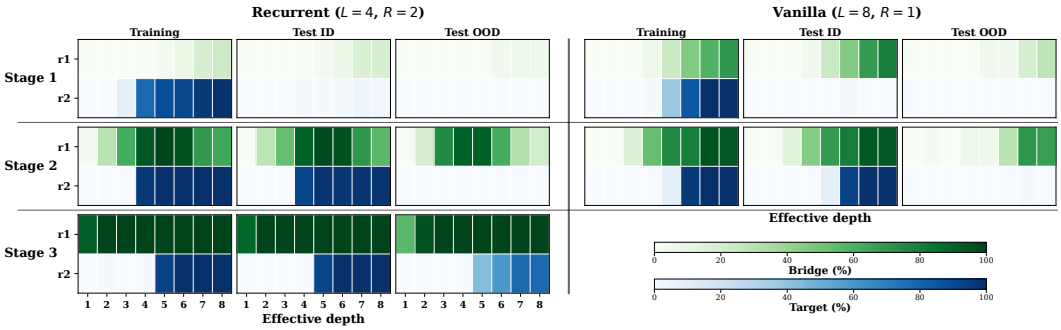

Figure 4: Accuracy of predicting bridge and target entities using logit lens at corresponding token positions for the recurrent-depth ($R = 2$) and the 8-layer vanilla transformer.

Grokking marks a transition from memorization to systematic generalization. We first focus on the recurrent-depth model (Figure 4 left panels), which exhibits distinct mechanisms across the three stages. In Stage 1, the model predicts targets without reliably decoding the bridge, indicating memorization. In Stage 2, the bridge becomes decodable, followed by correct target prediction for in-distribution data at deeper effective depths. Only in Stage 3 does the model succeed on OOD inputs, marking a transition from rote learning to systematic composition. In contrast, although vanilla transformers (right panels) can recover the bridge entity on Test OOD inputs, they fail to perform the second-hop reasoning, as they lack incentives to encode OOD facts in deeper layers.

6 Depth Extrapolation

In this section, we study depth extrapolation, i.e., whether the model can perform deeper recursions when combining its parametric knowledge than those observed during training.

6.1 Experiment Setup

Curriculum training. Different from 2-hop scenarios, learning k-hop tasks generally requires training the model with an easy-to-hard curriculum over hop depth (k) as suggested in Yao et al. (2025). Specifically, we start with training on atomic and 2-hop facts until an accuracy threshold (95%) is reached on a held-out 2-hop test split. Next, 3-hop data is included in the training for the next stage of our curriculum until 95% is achieved on the held-out 3-hop test split. This process is repeated for each hop level $k \in \{2, \ldots, N_{\text{max}}\}$. To prevent forgetting, at each stage we jointly train on all previously introduced facts (e.g., at stage $k = 5$ we train on atomic and 2-hop through 5-hop facts).

Due to this threshold-based curriculum, training facts beyond the model's capability are never exposed. That is, if the model fails to achieve above-threshold accuracy on k-hop queries, training terminates and $(k+1)$-hop data is never introduced. We define the largest such k as the *learnable recursion depth* of the model.

Dataset. We construct the dataset by instantiating a knowledge graph with $|E| = 200$ entities and $|R| = 10$ relations, where each entity has an average out-degree of 10. We additionally impose a permutation constraint on the knowledge graph to avoid learning shortcut solutions, for which we provide details in Appendix B. We pre-generate 2k atomic facts and 15k k-hop inferred facts for each $k \in [2, K_{max}]$, with $K_{max} = 40$. During training, these facts are progressively introduced following the curriculum described above.

For each model, we evaluate on 750 held-out k-hop facts. Facts with k up to the model's learnable recursion depth form the in-distribution test set, while those beyond it constitute

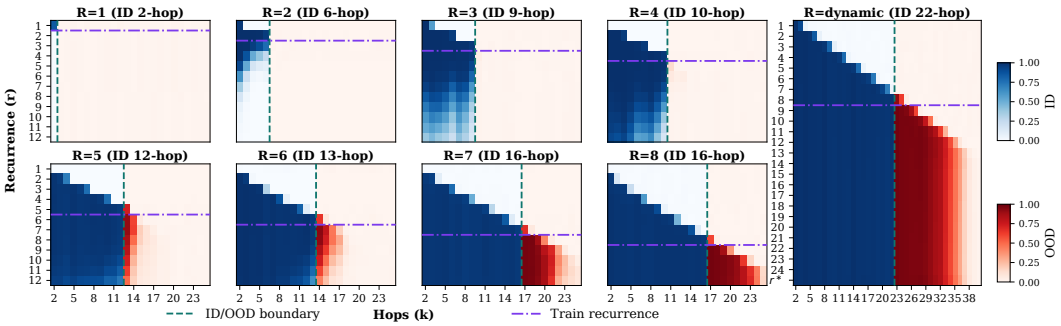

Figure 5: Accuracy of recurrent-depth models on multi-hop composition, trained under various fixed and dynamic recurrence setups. The violet dash-dotted horizontal line indicates the training recurrence (or maximum in the dynamic setting), while the teal dashed vertical line marks the maximum ID generalization achieved by each model. The x-axis shows hop complexity and the y-axis shows inference-time recurrence (r^* denotes adaptive halting).

the extrapolation test set. This split is model-dependent, reflecting each model's maximum achievable reasoning depth.

Model. We follow the model setups in Section 5.1, except that we use $R \in \{1,2,3,4,5,6,7,8\}$ for fixed iteration. Here we use no positional embeddings (NoPE) (Kazemnejad et al., 2023; Wang et al., 2024b), which shows better generalization in pilot studies. In addition to results with $L = 4$, we present results with varying model size and dynamic train recurrence in Appendix J, and across random seed initializations in Appendix I. We also experiment with larger vanilla transformer models, iso-FLOP with recurrent-depth models using $R \in \{2, 4, 6\}$ and report results in Appendix G.

6.2 In-Distribution Generalization

Scaling up training-time iteration increases the learnable recursion depth of looped transformers. Looking at the blue area of Figure 5, we find that increasing training recurrent iterations accordingly improves the ID generalization. This is consistent with previous findings (Wang et al., 2024a; Yao et al., 2025) that the learnable recursion depth of a transformer is bounded by the depth of its layers, and we demonstrate that for looped transformers, scaling up training recurrent iterations can also increase its "effective depth", without relying on additional parameters. Training with dynamic iteration further increases the learnable recursion depth over the fixed iteration, suggesting that the fixed iteration is not an optimal design choice. Interestingly, more recurrent iterations do not always translate into larger learnable depth. (e.g., both R=7 and R=8 learns up to 16-hop task).

Phase transition from prolonged training to rapid learning. We observe that models require a prolonged training phase to acquire low-hop tasks, after which they rapidly generalize to much more complex samples (Figure 6). We illustrate this for the model trained with dynamic recurrence by plotting training steps against the compositional complexity achieved on in-distribution data. This suggests that the main difficulty lies in discovering the underlying compositional rule. Once such a rule is internalized, the model can quickly extend it to samples of much higher complexity. In Figure 6 we observe how the model required over 1.3 million steps for grokking up to 4-hop train samples but was quickly able to learn up to 19-hop samples very few additional steps. Beyond that, for even more complex samples, while each new hop requires additional training steps to cross the 95% threshold, the model still achieves strong generalization (>90%) on hops 20, 21, and 22 within fewer than 8k extra steps per hop which is commensurate with the steps required for each additional hop from 4 through 19. By loading a trained checkpoint (20, say) and continuing training exclusively on 21-hop samples instead of the data mix consisting of samples from all previous stages in our curriculum, generalization over 90% on the new split can be achieved in as little as 50 additional steps of training.

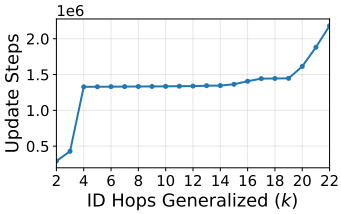

Figure 6: Cumulative gradient updates required to first generalize to each hop complexity.

6.3 Depth Extrapolation

Scaling inference-time iterations unlocks depth extrapolation. In Figure 5, we observe that when using the same number of recurrent iterations as in training, all models struggle to generalize to tasks of higher complexity than those seen during training. However, this limitation is immediately alleviated when we increase the number of inference-time iterations, with more iterations enabling generalization to progressively harder tasks. Notably, this scaling effect only emerges for $R > 4$, suggesting that sufficient training-time iterations are a prerequisite for benefiting from increased inference-time computation.

The effect of training iteration strategy on extrapolation. The above results characterize the maximum reasoning depth each model can achieve under the curriculum setting, but

they do not disentangle whether the differences (e.g. $R = 6$ generalizing to 17-hop vs. $R = 8$ to 24-hop) are due to more training iterations or exposure to more complex training data (e.g. $R = 6$ is trained up to 13-hop, while $R = 8$ up to 16-hop). To isolate the effect of the training iteration strategy, we train all models on the same data (up to 12-hop) and evaluate extrapolation on 12–24 hop tasks.

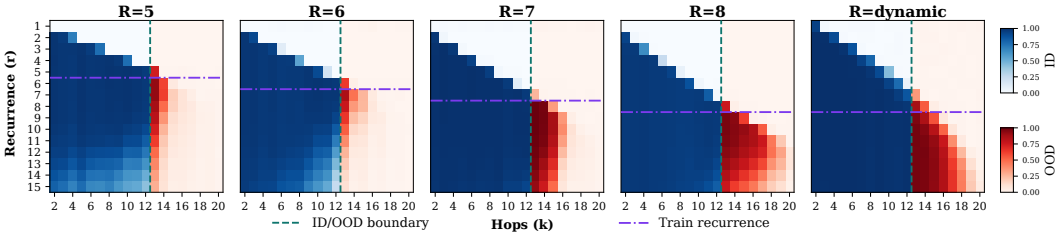

Figure 7: Accuracy of models trained with fixed recurrence $R \in \{5, 6, 7, 8\}$ and dynamic recurrence on up to 12-hop samples.

From Figure 7, we first find that among fixed-iteration models, increasing the number of training-time iterations substantially improves extrapolation when scaling inference-time iterations (e.g. $R = 6$ extrapolates up to 14-hop, while $R = 8$ reaches 19-hop). Second, under the same training data, the dynamic iteration strategy achieves comparable extrapolation performance to the $R = 8$ model (both reaching 19-hop). This contrasts with Figure 5, where the dynamic strategy generalizes to significantly higher complexity than $R = 8$.

These results suggest that the maximum number of iterations used during training determines the extrapolation range (i.e. how far beyond the training complexity the model can generalize), while dynamic iteration effectively exploits this range, since it enables a larger learnable recursion depth. Hence, when training data is sufficiently complex, dynamic iteration should be preferred over fixed strategies with the same maximum iteration budget. We provide further evidence of this through matched-data experiments in Appendix H.

Scaling inference-time iterations is limited by overthinking. Despite strong generalization performance, we observe performance degradation when using excessively large inference-time iterations. For example, increasing iterations beyond 15 for the R=dynamic does not improve OOD performance (Figure 5), a phenomenon known as *overthinking* (Bansal et al., 2022). We study this along two key axes: (1) inference iterations, i.e. how increasing the inference iterations affects the model's prediction confidence, and (2) task complexity, i.e. how increasing task complexity impacts this confidence.

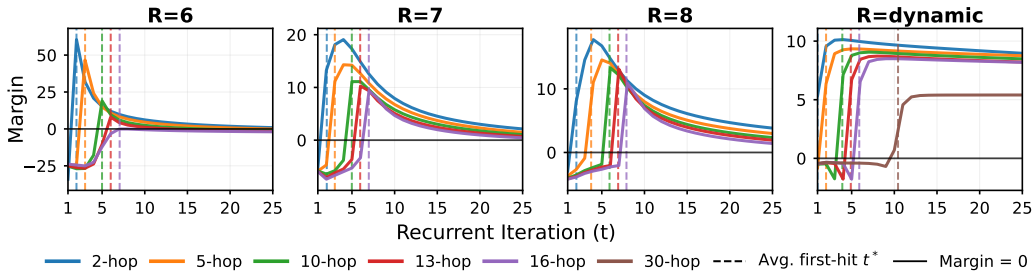

Figure 8: Average logit margin across recurrent iterations for fixed-recurrence ($R = 6, 7, 8$) and dynamic-recurrence models. Curves show different hop complexities, and dashed vertical lines indicate the average first-hit iteration t^* (when the correct entity is predicted).

In Figure 8, we analyze the logit margin, defined as the difference between the logit of the correct entity and that of the strongest competing token, and make two observations. First, across all models and tasks, the margin increases with inference iterations until reaching a peak, and then consistently declines as iterations continue. This suggests that overthinking arises as the iteration number grows, regardless of the task complexity. Notably, the dynamic

model exhibits a much slower margin decay compared to fixed-iteration models, indicating that dynamic iteration is more robust to overthinking. Second, the peak margin decreases as task complexity increases across all models. This implies that for more complex tasks, the model's predictions are inherently less confident and therefore more susceptible to degradation under additional iterations. As a result, the benefit of increasing inference iterations diminishes for highly complex tasks. Note that although previous work mitigates overthinking by injecting inputs information in every iteration (Bansal et al., 2022; Geiping et al., 2025), we find that such methods do not resolve the issue in implicit reasoning.

Adaptive halting improves inference efficiency. We flexibly halt recurrence in order to allocate compute proportionate to input complexity. Prior work (Geiping et al., 2025) proposes adaptive halting based on the output distribution $p_t(\cdot \mid x)$, terminating when the change between successive iterations becomes small, measured by $\mathrm{KL}(p_t, |, p_{t-1}) < \varepsilon_{\mathrm{KL}}$. In our setting, we find that this criterion often halts the recurrence prematurely, leading to suboptimal performance. To address this, we additionally incorporate the entropy of the output distribution, $H(p_t)$, and only stop when both the divergence is small, and the prediction is confident, i.e.,

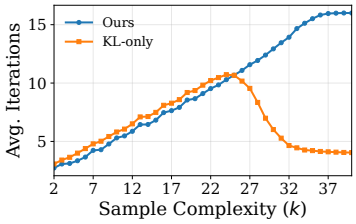

Figure 9: Average recurrent iterations using adaptive halting vs. sample complexity.

$$\mathrm{KL}(p_t \parallel p_{t-1}) < \varepsilon_{\mathrm{KL}} \qquad \text{and} \qquad H(p_t) < H_{\text{thresh}}.$$

In Figure 9, we plot the number of iterations using both methods against the hop count (k), demonstrating that ours results in better allocation of inference-time compute in accordance with task complexity. While the output distribution may change very little across iterations, the entropy is still high, indicating that the model remains uncertain despite this apparent convergence. We compare results using the two methods in Figure 10. In our setting, combining KL divergence with entropy therefore provides a more reliable halting signal and yields better overall results.

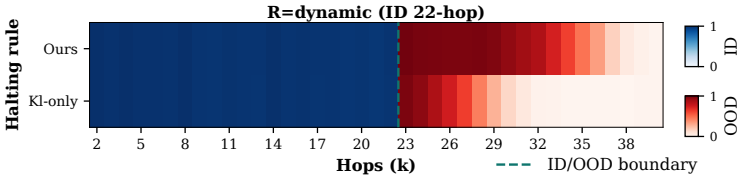

Figure 10: Comparison of adaptive halting based on KL-divergence and entropy (ours) versus KL-divergence alone (Geiping et al., 2025).

7 Conclusion

We study whether recurrent-depth transformers can compositionally use parametric knowledge for implicit multi-hop reasoning, a task that is challenging for vanilla transformers. Through controlled experiments on models trained from scratch, we show that recurrent-depth transformers successfully address both compositional generalization challenges. Systematic generalization emerges through a three-stage grokking dynamic, as the model transitions from rote memorization to generalizable solutions. Depth extrapolation is enabled by scaling inference-time compute, where additional iterations allow for greater reasoning depth, although latent overthinking limits performance on highly complex tasks. Overall, our results highlight recurrent-depth transformers as a promising architecture for compositional reasoning over parametric knowledge.

Acknowledgments

The authors of this paper express their gratitude to Boshi Wang for valuable discussions on this work, and for motivating the permutation-based task formulation for depth exploration described in Appendix B. We also thank Yupei Du for valuable feedback on this paper, and the OSU NLP and DMRL groups for their thoughtful comments.

This research was supported in part by NSF CAREER #1942980, Schmidt Sciences, Coefficient Giving (formerly Open Philanthropy) and Ohio Supercomputer Center (Ohio Supercomputer Center, 1987). The views and conclusions contained herein are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notice herein.

References

Atish Agarwala and Samuel S Schoenholz. Deep equilibrium networks are sensitive to initialization statistics. In *International Conference on Machine Learning*, pp. 136–160. PMLR, 2022.

Zeyuan Allen-Zhu and Yuanzhi Li. Physics of language models: Part 3.1, knowledge storage and extraction. *arXiv preprint arXiv:2309.14316*, 2023.

Mikita Balesni, Tomek Korbak, and Owain Evans. The two-hop curse: Llms trained on $a \to b, b \to c$ fail to learn $a \to c$. *arXiv preprint arXiv:2411.16353*, 2024.

Arpit Bansal, Avi Schwarzschild, Eitan Borgnia, Zeyad Emam, Furong Huang, Micah Goldblum, and Tom Goldstein. End-to-end algorithm synthesis with recurrent networks: Extrapolation without overthinking. *Advances in Neural Information Processing Systems*, 35: 20232–20242, 2022.

Lukas Berglund, Meg Tong, Max Kaufmann, Mikita Balesni, Asa Cooper Stickland, Tomasz Korbak, and Owain Evans. The reversal curse: Llms trained on" a is b" fail to learn" b is a". *arXiv preprint arXiv:2309.12288*, 2023.

Eden Biran, Daniela Gottesman, Sohee Yang, Mor Geva, and Amir Globerson. Hopping too late: Exploring the limitations of large language models on multi-hop queries. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 14113–14130, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.781. URL https://aclanthology.org/2024.emnlp-main.781/.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Róbert Csordás, Kazuki Irie, and Jürgen Schmidhuber. The neural data router: Adaptive control flow in transformers improves systematic generalization. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=KBQP4A_J1K.

Róbert Csordás, Kazuki Irie, Jürgen Schmidhuber, Christopher Potts, and Christopher D Manning. MoEUT: Mixture-of-experts universal transformers. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=ZxVrkm7Bjl.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HyzdRiR9Y7.

Hanwen Du, Yuxin Dong, and Xia Ning. Latent thinking optimization: Your latent reasoning language model secretly encodes reward signals in its latent thoughts. *arXiv preprint arXiv:2509.26314*, 2025.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D. Hwang, Soumya Sanyal, Sean Welleck, Xiang Ren, Allyson Ettinger, Zaid Harchaoui, and Yejin Choi. Faith and fate: limits of transformers on compositionality. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

Ying Fan, Yilun Du, Kannan Ramchandran, and Kangwook Lee. Looped transformers for length generalization. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=2edigk8yoU.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Jonas Geiping, Sean Michael McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. In *ES-FoMo III: 3rd Workshop on Efficient Systems for Foundation Models*, 2025. URL https://openreview.net/forum?id=D6o6Bwtq7h.

Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. Dissecting recall of factual associations in auto-regressive language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 12216–12235, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.751. URL https://aclanthology.org/2023.emnlp-main.751/.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason E Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=Itxz7S4Ip3.

Jie Huang and Kevin Chen-Chuan Chang. Towards reasoning in large language models: A survey. In *Findings of the Association for Computational Linguistics, ACL 2023*, pp. 1049–1065. Association for Computational Linguistics (ACL), 2023.

Da Ju, Stephen Roller, Sainbayar Sukhbaatar, and Jason E Weston. Staircase attention for recurrent processing of sequences. *Advances in neural information processing systems*, 35: 13203–13213, 2022.

Amirhossein Kazemnejad, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Payel Das, and Siva Reddy. The impact of positional encoding on length generalization in transformers. *Advances in Neural Information Processing Systems*, 36:24892–24928, 2023.

Najoung Kim and Tal Linzen. COGS: A compositional generalization challenge based on semantic interpretation. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 9087–9105, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.731. URL https://aclanthology.org/2020.emnlp-main.731/.

Harsh Kohli, Sachin Kumar, and Huan Sun. GroundCocoa: A benchmark for evaluating compositional & conditional reasoning in language models. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 8280–8295, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.420. URL https://aclanthology.org/2025.naacl-long.420/.

Brenden Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International conference on machine learning*, pp. 2873–2882. PMLR, 2018.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1eA7AEtvS.

Ziyue Li, Yang Li, and Tianyi Zhou. Skip a Layer or Loop It? Learning Program-of-Layers in LLMs. In *Forty-third International Conference on Machine Learning (ICML)*, 2026. URL https://arxiv.org/pdf/2606.06574.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Wenquan Lu, Yuechuan Yang, Kyle Lee, Yanshu Li, and Enqi Liu. Latent chain-of-thought? decoding the depth-recurrent transformer. *arXiv preprint arXiv:2507.02199*, 2025.

nostalgebraist. interpreting gpt: the logit lens, 2020. URL https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens.

Ohio Supercomputer Center. Ohio supercomputer center, 1987. URL https://ror.org/01apna436.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 5687–5711, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.378. URL https://aclanthology.org/2023.findings-emnlp.378/.

Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi. Reasoning with latent thoughts: On the power of looped transformers. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=din0lGfZFd.

Herbert A. Simon and Allen Newell. Human problem solving: The state of the theory in 1970. *American Psychologist*, 26:145–159, 1971. URL https://api.semanticscholar.org/CorpusID:14405801.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Boshi Wang, Xiang Yue, Yu Su, and Huan Sun. Grokking of implicit reasoning in transformers: a mechanistic journey to the edge of generalization. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2024a. Curran Associates Inc. ISBN 9798331314385.

Jie Wang, Tao Ji, Yuanbin Wu, Hang Yan, Tao Gui, Qi Zhang, Xuan-Jing Huang, and Xiaoling Wang. Length generalization of causal transformers without position encoding. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 14024–14040, 2024b.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Liu Yang, Kangwook Lee, Robert D Nowak, and Dimitris Papailiopoulos. Looped transformers are better at learning learning algorithms. In *The Twelfth International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=HHbRxoDTxE.

Sohee Yang, Elena Gribovskaya, Nora Kassner, Mor Geva, and Sebastian Riedel. Do large language models latently perform multi-hop reasoning? In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10210–10229, Bangkok, Thailand, August 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.550. URL https://aclanthology.org/2024.acl-long.550/.

Yuekun Yao, Yupei Du, Dawei Zhu, Michael Hahn, and Alexander Koller. Language models can learn implicit multi-hop reasoning, but only if they have lots of training data. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 9695–9713, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. URL https://aclanthology.org/2025.emnlp-main.490/.

Hongyi Zhang, Yann N. Dauphin, and Tengyu Ma. Fixup initialization: Residual learning without normalization, 2019. URL https://arxiv.org/abs/1901.09321.

Rui-Jie Zhu, Zixuan Wang, Kai Hua, Tianyu Zhang, Ziniu Li, Haoran Que, Boyi Wei, Zixin Wen, Fan Yin, He Xing, et al. Scaling latent reasoning via looped language models. *arXiv preprint arXiv:2510.25741*, 2025.

A Limitations

Through our paper, we evaluate vanilla and recurrent-depth transformers on a single family of implicit reasoning problems. While we cleanly isolate systematic generalization and depth exploration in compositional reasoning, we do not cover the entire range of reasoning problems relevant to modern language models (Huang & Chang, 2023). In order to maintain a controlled setting and properly study the behavior of recurrent-depth models, our experiments are intentionally small-scale and focused. This potentially limits direct extrapolation of results to modern LLMs whose performance and behavior is shaped by many additional factors like tokenizer choices, heterogeneous internet-scale data, pretraining data, post-training procedures and optimization at a much larger scale. Althought we study somewhat larger models in Appendix J, these are still significantly limited in scale relative to frontier LLMs today.

Additionally, our task formulation abstracts away many aspects of real-world language use. Inputs are presented in a highly structured form with dedicated entity and relation tokens with a very limited vocabulary. This removes challenges that might arise from surface-form variation, underspecification, distractor information, and distribution shift in natural language. Therefore, we do not claim complete and immediate transfer of positive results to LLMs trained with recurrent-depth at scale. However, we hope to isolate architectural effects in a controlled setting and use them to inform future designs for more robust implicit reasoning in future LLMs.

B Dataset

To investigate how our implicit reasoning models scale to deeper k-hop composition tasks, we design a dataset from a permutation-based knowledge graph. In this setup, we found that models can sometimes achieve high accuracy via *shortcuts* rather than performing the intended multi-step traversal. For large k, the final tail entity t can become nearly determined by a short suffix of the relation sequence. Thus, a model may learn a shallow mapping from the trailing relations to the answer instead of retrieving the intermediate entities. To overcome this, We first construct a set of atomic facts over $|E| = 200$ entities and $|R| = 10$ relations. For each relation $r \in R$, we sample a random permutation π_r over the entity indices and define the atomic facts as

$$\forall e_i \in E : \quad (e_i, r, e_{\pi_r(i)}).$$

Thus, each relation acts as a bijection over the entity set. Since the out-degree is set to $d = |R| = 10$, every entity has exactly one outgoing edge for each relation. This ensures

that each relation has full coverage over entities. To generate a k-hop example, we sample a starting atomic fact and then iteratively extend it by following outgoing edges from the current tail entity. At each step, one outgoing relation-edge pair is selected uniformly at random. Intermediate entities are used only during construction and are not included in the final training example.

C Training details and additional parameters

For most experiments, we use an embedding dimension of 768, 12 attention heads and a recurrent block of 4 transformer layers. However, in Appendix J we experiment with recurrent blocks of greater depth. Similarly, in Section 5.2 we compare with a vanilla transformer of larger depth (8) in our systematicity analysis. AdamW optimizer (Loshchilov & Hutter) with a learning rate of 10^{-4}, weight decay of 0.01 and a linear warmup schedule of 2000 steps is used in all of our experiments. We use a batch size of 512 and 128 for the systematicity and extrapolation experiments respectively. In our adaptive halting method, we use fixed thresholds $\varepsilon_{\text{KL}} = 0.01$ and $H_{\text{thresh}} = 3.00$, and halt recurrence when both $\text{KL}(p_t \parallel p_{t-1}) < \varepsilon_{\text{KL}}$ and $H(p_t) < H_{\text{thresh}}$.

D Other related work

Implicit Reasoning LLMs. Weight-sharing in transformers has been used as a strategy for parameter-efficiency starting with Universal Transformers (Dehghani et al., 2019) and ALBERT (Lan et al., 2020). An earlier closely related architecture is the ladder model by Ju et al. (2022), which repeatedly applies a shared Transformer core to the full input sequence, yielding recurrence in depth. Recent work revisits this idea as a mechanism for implicit reasoning and scaling computation at inference-time. Csordás et al. (2024) introduce Mixture-of-Experts Universal Transformers (MoEUT), combining recurrent layer sharing with sparse experts to improve the parameter-compute trade-off. Recurrent-depth models trained at LLM scale such as Huginn (Geiping et al., 2025) and Ouro (Zhu et al., 2025) have shown promise and often exceed the performance of larger, non-recurrent models on popular benchmarks. Another approach to latent reasoning, Coconut (Hao et al., 2025) replaces discrete reasoning tokens with continuous hidden states, enabling reasoning in latent space rather than through explicit text similar to recurrent-depth models. Complementarily, Li et al. (2026) introduce Program-of-Layers (PoLar), a lightweight predictor that dynamically skips or repeats segments of frozen pretrained LLM layers for each input, improving reasoning accuracy often with fewer executed layers.

Studies on Implicit Reasoning in Recurrent Depth LLMs. There is a growing body of work analyzing the improved performance of models through mechanisms like recurrent-depth architectures that enable reasoning in latent space. At LLM scale, Lu et al. (2025) study the internals of the Huginn-3.5B using logit lens by projecting the intermediate representations through an unembedding matrix or through the models specialized coda layers (coda-lens). They report little evidence of actual latent reasoning and inconsistencies in interpretability across recurrent blocks. Conversely, Du et al. (2025) find that trajectories of hidden states or "latent thoughts" that lead to correct outcomes are distinguishable from those that lead to incorrect outcomes through certain metrics. The representations of correct trajectories have a higher entropy, anisotropy, and intrinsic dimension but a lower effective rank indicating that correct thinking processes carry richer information with less noise, and generate more expressive latent representations. Both works take a pretrained, generalist LLM (Huginn-3.5B) and focus on probing its latent computations.

Compositional Generalization. Many recent studies have examined the performance of transformer-based language models in compositional generalization, motivated by the view that such step-by-step reasoning is central to human intelligence (Simon & Newell, 1971) and a proxy for analyzing how models can learn internal mechanisms for combining facts instead of emitting long rationales in the form of CoT. Using 3 representative tasks, Dziri et al. (2023) demonstrate that transformers reduce compositional tasks to linearized

subgraph matching that fails with increasing complexity. Wang et al. (2024a) show that transformers learn implicit 2-hop compositional generalization only through ”grokking” and that even then systematicity (OOD generalization) is never observed. Kohli et al. (2025) construct a logically grounded dataset to benchmark on compositional generalization and conditional reasoning, and show that even frontier LLMs struggle on tasks of sufficient compositional complexity. Experiments by Yao et al. (2025) indicate that transformers are capable of multi-hop ID generalization but each hop requires exponentially larger amounts of data and that this issue is partially mitigated through curriculum learning. They also show that intermediate entities are retrieved *layerwise* and that, in order to achieve k-hop generalization, the numbers of layers needs to grow linearly in k. Balesni et al. (2024) finetune pretrained LLMs on synthetic or semi-synthetic composition tasks and find that the models are only able to achieve composition when the two synthetic facts co-occur in the same finetuning document or test-time prompt or when one of the hops is a natural fact in the pretraining corpus. On algorithmic compositional tasks, Csordás et al. (2022) introduce the Neural Data Router, augmenting a weight-shared Transformer with copy gates and geometric attention to learn adaptive control flow and improve generalization to longer and deeper compositions.

E Causal analysis of systematic generalization

We additionally perform a causal analysis using activation patching together with logit-lens decodability, and present the results in Figure 11. In the figure, the color shading of each node indicates decodability: for the top row, how strongly the bridge entity b is decoded at token position r_1, and for the bottom row, how strongly the target entity t is decoded at position r_2, across effective depth. The thickness of the curved arrows shows causal strength measured by activation patching: we corrupt the first-hop prefix (h, r_1), restore the clean activation at position r_1 at a given effective depth, and measure how much the final target prediction is recovered. We use the term “causal site” in the figure to denote the layer at position r_1 that exhibits the strongest causal effect under activation patching.

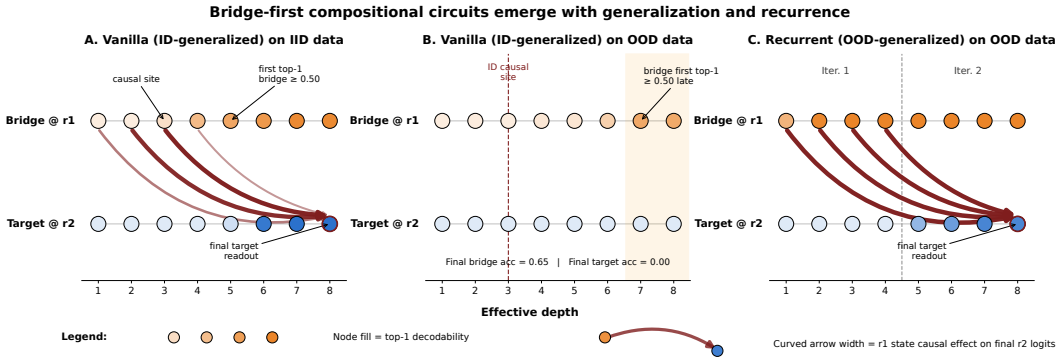

Figure 11: Bridge-mediated composition across effective depth in our systematicity setup.

Panel A shows that, on ID data, the bridge becomes decodable at r_1 in shallow layers (layers 5), and these layers exhibit causal effects on the final answer prediction. Panel B shows that, on OOD data, the vanilla model only recovers the bridge at much deeper layers (layer 7+), and the target never becomes decodable. This suggests that the model can access the OOD first-hop rule only too late to use it for the second-hop composition. Panel C shows that the recurrent-depth model alleviates this issue: all layers in the first iteration at position r_1 show causal effects on the final answer prediction, and subsequent recurrent iterations can reuse the same parametric rule to complete the second-hop composition.

F Experiments with default initialization

Here we report results obtained using the default Gaussian initialization for the `c_proj` matrices, rather than the zero-initialization described in Section 4. As observed from Figure 12, the results are mixed across training runs, and increasing recurrence does not yield a clear or consistent pattern of improved ID or OOD generalization as well as robustness to latent overthinking. Only the model trained with $R = 7$, hows strong signs of inference-time scaling and robustness to latent overthinking. In the case of the model trained with $R = 5$, there is no OOD generalization with increased recurrence at inference-time. For most of the other runs, we see performance degradation due to latent overthinking.

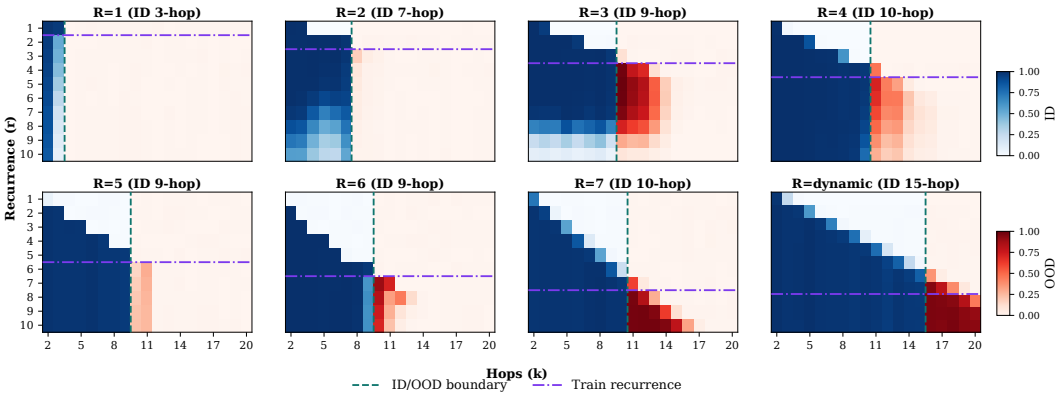

Figure 12: Results with default initialization for training recurrences $R \in \{1, \ldots, 7\}$ and dynamic recurrence.

To further examine this instability, we train the $r = 5$ model with five different random seeds. As shown in Figure 13, the resulting behaviors vary substantially across seeds. Seed 1 is not affected by latent overthinking, but it also shows no evidence of inference-time scaling by solving harder compositional examples with increased recurrence. Seed 2 is likewise stable, but does exhibit scaling with additional recurrence. Seeds 3, 4, and 5 show varying degrees of inference-time scaling, but also experience performance degradation as recurrence increases, consistent with latent overthinking.

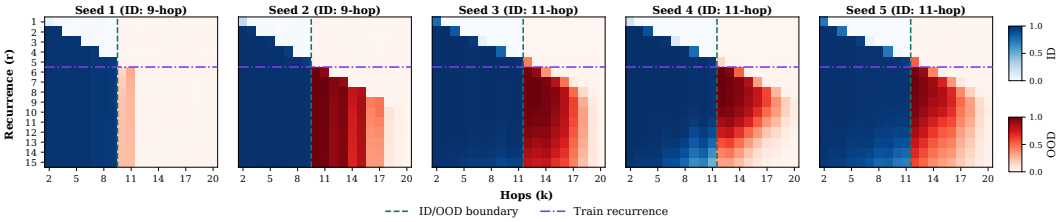

Figure 13: Five random-seed runs for the $R = 5$ model with default initialization.

G Experiments with larger vanilla transformers

We present additional results for larger vanilla transformers in Figure 14. Specifically, we train 8-layer, 16-layer, and 24-layer vanilla transformers using the same setup as in Section 6.1. These results show that deeper vanilla models often achieve better ID generalization and can go beyond the 2-hop or 3-hop generalization observed in the 4-layer vanilla baseline. This is consistent with prior work showing that in-distribution generalization can improve with greater depth in vanilla transformers [1]. However, their ID generalization remains weaker than recurrent-depth models with the same effective depth. More importantly, the vanilla models still do not exhibit OOD generalization to more complex samples, and

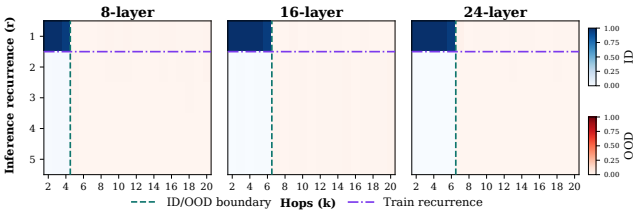

Figure 14: Depth extrapolation with larger vanilla transformers.

they lack the flexibility to scale computation according to sample complexity in the way recurrent-depth models can.

H Additional experiments with matched-hop data

We provide additional extrapolation results for models with different loop iterations trained on matched-hop data in Figure 15. From top to bottom, the rows show results for models trained on data up to 8-hop, 10-hop, and 12-hop, while each column corresponds to a different training recurrence setting. The general trend is consistent with Figure 7: models trained with higher recurrence achieve stronger OOD generalization through inference-time scaling.

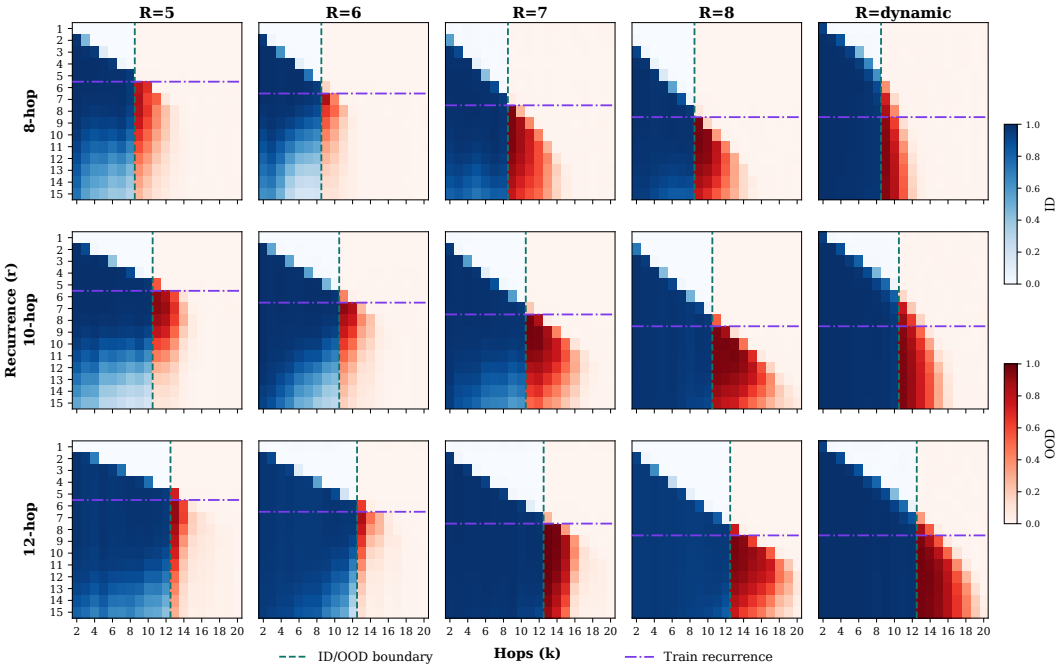

Figure 15: Matched-data experiments with varying train-time recurrence.

I Results with different random seeds

We run our fixed-recurrence models on 2 separate random seed initializations (including the run shown in Figure 5) and with to 40 recurrent iterations at inference time. On both runs, we notice the same general trend of increasing ID and OOD generalization with higher train-time recurrence as well as issues due to latent overthinking (with the sole exception being the model trained with $r = 8$ in the second run). The results are presented in Figures 16 & 17.

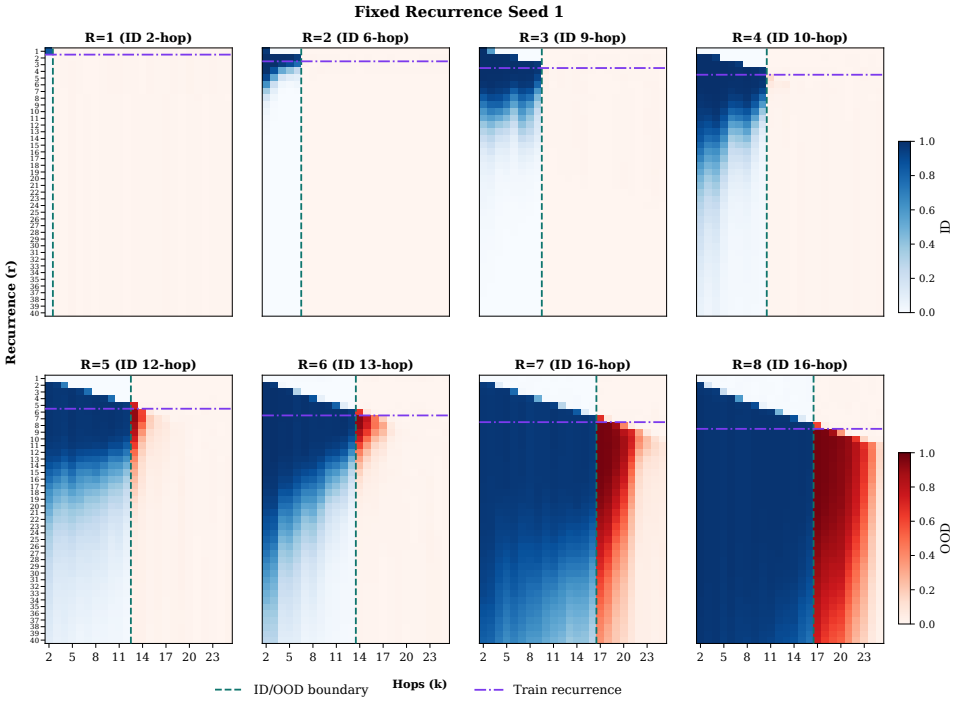

Figure 16: Results for Seed 1.

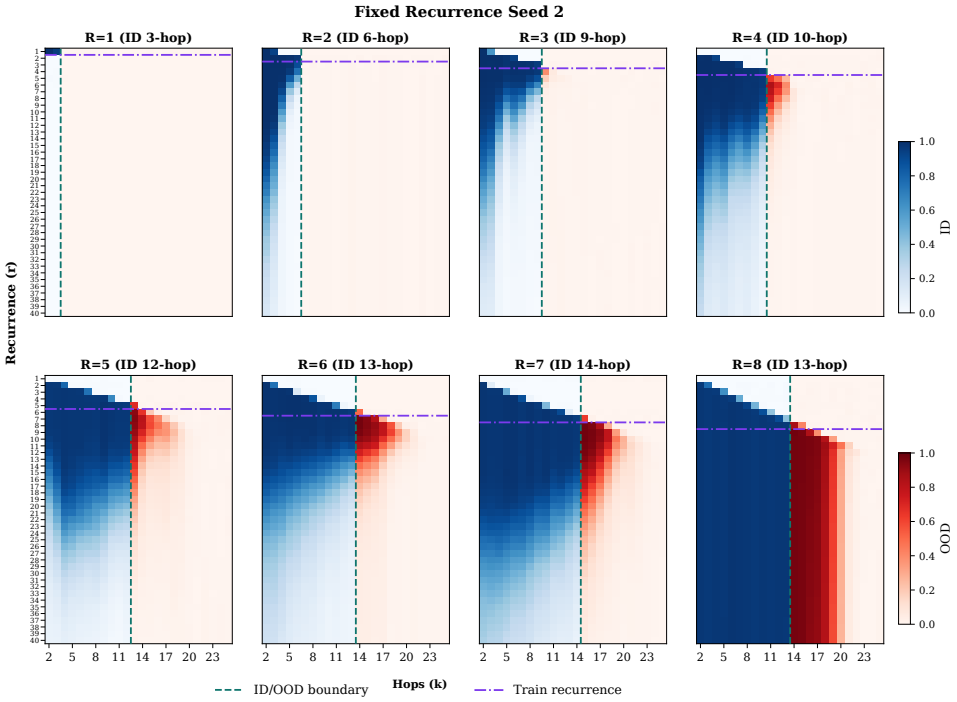

Figure 17: Results for Seed 2.

We similarly plot the performance of the dynamic recurrence model across three different seeds and observe a generally high ID generalization and OOD extrapolation. Across all seeds, the dynamic recurrence model is robust to the latent overthinking phenomenon and adaptive halting (r^*) works reliably to stop recurrence when not necessary.

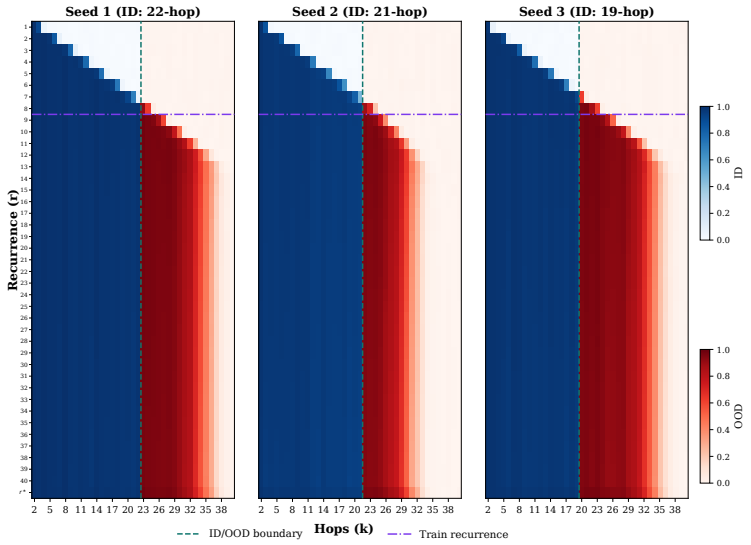

Figure 18: Dynamic recurrence with 3 difference random seed initializations.

J Extrapolation with varying model size and dynamic recurrence

We carry out additional experiments to study how extrapolation behavior changes with model size and with training recurrence in our dynamic recurrence setting. In addition to our default 4-layer model, we train larger models with 6 and 8 layers. We also vary the maximum recurrence used during dynamic-recurrence training, considering settings with maximum train-time recurrence of 8, 12, and 16. For these, the recurrence at each training iteration is sampled from a Poisson distribution with means 4, 6, and 8 respectively, with a minimum recurrence of 2 in all cases.

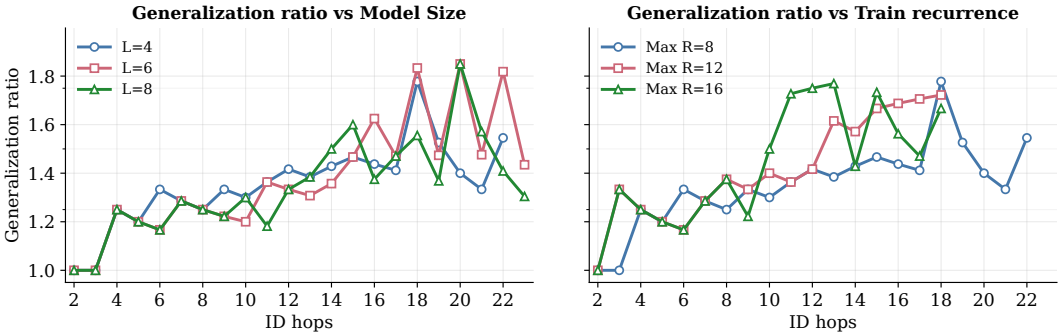

Figure 19: Generalization ratio with changing model size and train recurrence.

To summarize extrapolation under these settings, we plot the *generalization ratio*, defined as the maximum hop complexity to which a model can generalize with at least 60% accuracy with inference-time scaling, divided by the maximum hop complexity that the model has generalized to during training. This can be viewed as the ratio of maximum OOD generalization to maximum ID generalization. A ratio of 1 indicates no extrapolation beyond the level reached during training, while larger values indicate successful extrapolation to more complex compositions at test time.

Figure 19 shows that models initially begin with a ratio close to 1. In particular, when models have only generalized to 2-hop composition during training, they do not yet extrapolate to more complex samples. As training progresses and the models have seen more complex compositions through our curriculum learning setup, the ratio increases. However, across both model-size variations and different maximum train-time recurrence settings, we do not

observe a clear or consistent trend indicating that either larger models or larger train-time recurrence systematically improves this ratio.

K Illusion of very deep composition through shortcuts

Prior to adopting the permutation-based dataset construction described in Appendix B, we observed what appeared to be very deep compositional generalization. As shown in Figure 20, models trained on ID examples up to 40 hops achieved strong OOD performance even at 80 hops for models with $R = 4$, $R = 8$, and dynamic recurrence. This indicates that the model is able to resolve multiple hops or compositions in a single layer. To inspect this closely, we perform a causal activation-patching analysis. We first select a clean 60-hop test example whose relation sequence ends in a particular suffix, and a second random example of the same hop type. We then run both examples through the model and cache their hidden states. Next, for each token position and each recurrent iteration-layer location (as well as the embedding layer), we replace the hidden state in the clean run with the corresponding hidden state from the random run, and measure the resulting change in the logit margin of the correct final answer. Figure 21 visualizes this change relative to the clean run.

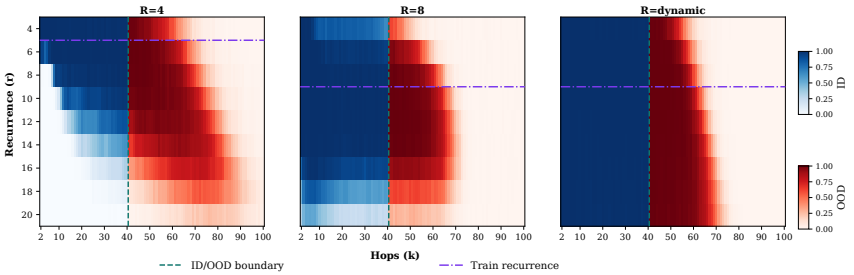

Figure 20: Apparent deep compositional generalization in the pre-permutation dataset.

This analysis reveals that the model is not solving the full compositional chain. Instead, the strongest causal effects are concentrated on a short suffix of the relation sequence. The final tail entity can often be inferred from trailing relations alone, without explicitly retrieving the intermediate entities. The model learns a shallow mapping from a suffix of the relation sequence to the answer, rather than performing full multi-hop composition. These observations motivate the permutation-based knowledge graph introduced in Appendix B. Under that construction, each relation acts as a permutation over entities, so the correct final entity cannot be recovered from a short suffix alone, and the model must compose the full sequence of relations to solve the task.

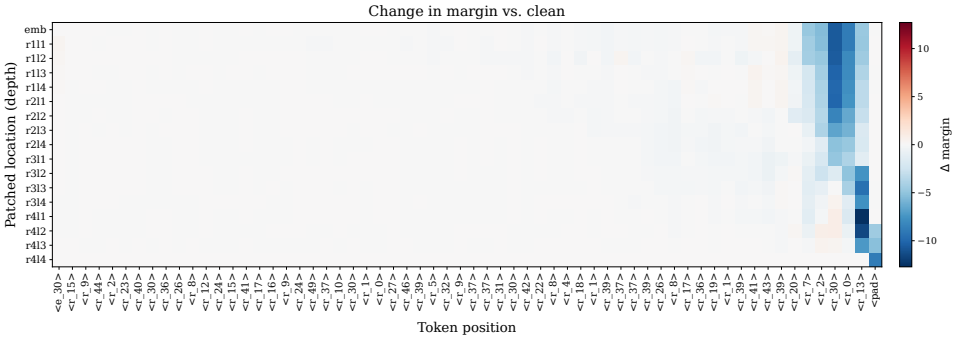

Figure 21: Activation-patching analysis on a 60-hop example. Each cell shows the change in logit margin for the correct answer, relative to the clean run, after replacing a single hidden state in the clean example with the corresponding hidden state from a random 60-hop example. Large changes are concentrated towards the end of the relation sequence, indicating shortcut-based prediction rather than genuine multi-hop retrieval.