

# Language Models, Graph Searching, and Supervision Adulteration: When More Supervision is Less and How to Make More More

Arvid Frydenlund

University of Toronto, Computer Science

Vector Institute

arvie@cs.toronto.edu

## Abstract

This work concerns the path-star task, a minimal example of searching over a graph. The graph,  $G$ , is star-shaped with  $D$  arms radiating from a start node,  $s$ . A language model (LM) is given  $G$ ,  $s$ , and a target node  $t$ , which ends one of the arms and is tasked with generating the arm containing  $t$ . The minimal nature of this task means only a single choice needs to be made: which of the  $D$  arms contains  $t$ ?

Decoder-only LMs fail to solve this elementary task above  $1/D$  chance due to a learned shortcut that absorbs training supervision. We show how this pathology is caused by excess supervision and we present a series of solutions demonstrating that the task is solvable via decoder-only LMs. **We find that the task’s minimal nature causes its difficulty, as it prevents task decomposition.** Our solutions provide insight into the pathology and its implications for LMs trained via next-token prediction.

## 1 Introduction

The path-star task is a seemingly simple, minimal graph search task intended to exhibit a flaw in the standard next-token prediction paradigm used to train decoder-only autoregressive LMs via teacher-forcing (TF) (Bachmann and Nagarajan, 2024).

Each graph is star-shaped with  $D$  arms rooted at a single start node,  $s$ . The LM is given the complete graph (as a shuffled edge list) and a query,  $(s, t)$ , where  $t$  is a target node that ends an arm. The task is to generate the arm with  $t$  from  $s$  to  $t$  (Fig 1). This requires the LM to choose an arm by initially generating one of the  $D$  leading nodes adjacent to  $s$ , with the rest of the arm being dictated by following edges. Thus, the task’s difficulty lies in choosing the correct leading node,  $l_t$ , necessitating planning and reconstruction of the correct arm from  $t$  to  $l_t$ .

Training via TF conditions the LM on prior ground-truth tokens. This induces learning an undesired shortcut, the *Clever Hans Cheat* (CHC),

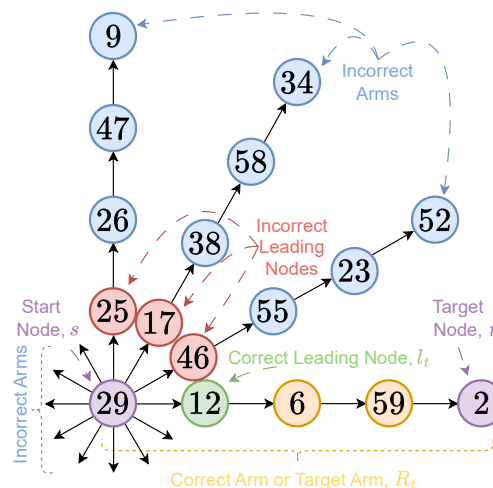


Figure 1: An example path-star graph.  $D = 12$ ,  $M = 5$ ,  $s$  is ‘29’,  $t$  is ‘2’,  $R_t$  is ‘29 12 6 59 2’, and  $l_t$  is ‘12’. We omit eight incorrect arms for space. **The task is to generate  $R_t$  given a query,  $Q = (s, t)$ , and the graph,  $G$ , as a tokenized shuffled edge list** (See Fig. 2).

which allows for trivial prediction of all non-leading nodes via a single edge look-up given the preceding node (given via TF). Thus, all the sequential supervision is absorbed into learning the CHC except for a single target token,  $l_t$ , which becomes the sole support for learning the required arm reconstruction subtask.<sup>1</sup> As a result, LMs fail to generate the correct arm above the random baseline of  $1/D$  chance (Bachmann and Nagarajan, 2024). While it has been shown that decoder-only LMs can express the task (Frydenlund, 2024), **it remains an open question if decoder-only language models trained via teacher-forcing can learn the task.**

### 1.1 Significance of the Failure

LMs are the ubiquitous model for NLP tasks (Brown et al., 2020), as well as for reasoning tasks (Bubeck et al., 2023). These tasks often require planning, which LMs struggle with (Valmeekam et al., 2023b; Kambhampati et al., 2024, Ap. C.1.1).

<sup>1</sup>Why this itself is hard is an open question, see Sec. 2.2.

Potentially, this poor planning performance may be attributable to a fundamental problem with the next-token prediction paradigm. **The path-star task is designed to support such a claim, where the minimal nature of the planning task is meant to isolate and highlight the failure; if standard LMs trained in standard ways fail to solve such a brutally simple task, it calls into question the sufficiency of the standard paradigm.**

This motivated the use of alternative models. Bachmann and Nagarajan (2024) used a ‘teacherless’ model which foregoes TF by conditioning on fully masked input (Monea et al., 2023). Frydenlund (2024) generalized this to non- and iterative-autoregressive models and demonstrated learnability differences between models, where an encoder-only LM could solve the task (on small graphs).

Saparov et al. (2025) showed positive results on path-star graphs with encoder-only models and on more general graph topologies with both encoder- and decoder-only models. *They did not try path-star graphs with decoder-only LMs.* They found that the topology is critical to generalization, but that learning does not scale with graph size (and using scratchpad that performs a depth-first search did not resolve this issue), leading to the claim that ‘transformers struggle to learn to search’.

Yin et al. (2024) and Hu et al. (2025a) both proposed novel model architectures on the perceived deficiency of decoder-only models in solving the path-star task. Yin et al. (2024) trained an auxiliary autoencoder to form *planning latent-states* that encode future tokens and then trained an LM which regressed against these to learn special *planning tokens*. Hu et al. (2025a) introduced a model trained on both forward and backward contexts using two separate forward and backward encoders. Ahn et al. (2025) continued with this motivation but tried to minimize the required architecture changes.

Wu et al. (2024c) put forth a related argument that next-token prediction is potentially problematic for planning tasks due to the cross-entropy loss leading to spurious correlations (see Appx. C.2).

## 1.2 Disproving Prior Claim and Conjecture

The claim that failure to learn the task demonstrates an insufficiency of the standard paradigm is empirically supported. Hence, showing the task to be learnable with standard methods would refute this claim. Bachmann and Nagarajan (2024) also conjectured that this failure will apply to more complex planning tasks. They gave story generation as an

example, but without any empirical testing.

**This work will show that this claim and conjecture are unfounded, or at least extremely limited, by demonstrating that the task is fragile to minor modifications which make the task learnable via standard methods.** We show that the CHC is not the sole cause of the task’s difficulty, and that preventing it is unnecessary for learning the task. We instead show that subtask decomposition is necessary for learning the task, and explain that the failure is caused by excessive or adulterated supervision that prevents this decomposition.

Saparov et al. (2025) showed strong success on the task with encoder-only models and on a similar-looking task with decoder-only models. This success came without discussion with Bachmann and Nagarajan (2024). We will explain this gap as also being attributable to subtle changes in the task that induce subtask decomposition. Like Saparov et al. (2025), we find the task becomes harder to learn with scale. However, it is unnecessary to show scalability to larger graphs to refute prior claims.

## 2 Task, Data, and Tokenization

Each graph,  $G$ , has  $D$  arms of the same length  $M$  (inclusive of  $s$ ) and is constructed by sampling nodes from a set of possible nodes,  $V$ , without replacement, making the graph size  $|G| = D(M - 1) + 1$ . The edges are determined by this sample order. Thus, all nodes are unique and *semanticsless* as they only relate via randomly sampled edges.

The task is tokenized as a sequence consisting of the query,  $Q = (s, t)$ , with start- and end-of-query markers (‘/ s t ?’). Each edge,  $(u, v)$ , is followed by the end-of-edge marker (‘u v l’). See Fig. 2. **The entire graph is provided to the language model as a series of edges, which are randomly shuffled.** This destroys any higher-order structural information about  $G$ , meaning that the task must be solved via planning and edge-following. The source-side input into the model is  $Q$  followed by  $G$  and the end-of-graph marker (‘=’). We place  $Q$  before  $G$  as it is better for decoder-only models (Frydenlund, 2024). Let  $R_t = x_1, \dots, x_M$  be the series of nodes from  $s$  to  $t$  forming the target arm and the sequential target-side supervision.

Each experiment uses a model trained from scratch on graphs with static  $D$  and  $M$ , so different-sized graphs are not mixed during training (except in Sec. 3.6). We avoid uncontrollable biases from natural data by not using pretrained models.

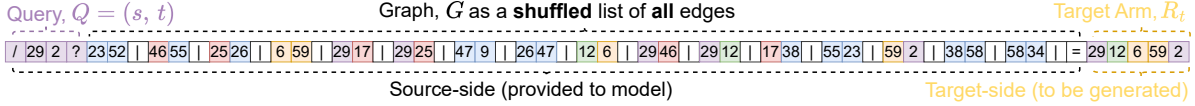


Figure 2: A *tokenization* corresponding to Fig. 1. We omit any edges belonging to the omitted incorrect arms.

Frydenlund (2024) identified that the original experimental design leads to spurious correlations and overfitting due to the task’s large sample space,

$$Z = \frac{|V|!}{(|V| - D(M - 1) - 1)!} \times D. \quad (1)$$

To this end, they proposed using ‘structured samples’. While this helped, it did not resolve overfitting and increased training time. Instead, we use an online dataset that generates new samples during training. Saparov et al. (2025) also used an online dataset. Sanford et al. (2024b) found better learnability with online training for the  $k$ -hop task. We also minimize the space by using  $|V| = |G|$ .

Information can only be routed into the future due to the decoder’s causal constraint. This increases the task’s difficulty as the LM must learn two separate routing rules subject to edge  $(u, v)$  proceeding or succeeding  $(v, w)$ . To avoid this, Frydenlund (2024) introduced an ‘arm-wise shuffle’ which only shuffles arms relative to each other. However, this allows for a trivial solution by looking back  $M - 1$  positions from  $t$  to predict  $l_t$ . Instead, we present a ‘causal-wise shuffle’ where each arm is in sequential order but not contiguous. This alternative setup acts as a control to indicate if the causal constraint is causing difficulties.

## 2.1 Supervision Adulteration

We discussed how the models will overfit due to spurious correlations in the data. The CHC is also a shortcut learnt due to overfitting; however, this is a different kind of overfitting, as it is not caused by the data, but rather by the way the task is constructed. In particular, the CHC is a shortcut caused by providing excess supervision or *adulteration*.

Consider the various ways the task is supervised. In a supervised learning framework, we generally regard the target labels as ‘the supervision’ as they can be human-annotated. With the next-token prediction paradigm, we forego human annotation by using a self-supervised rule for generating the targets. In both cases, the targets are a function of the input and thus the choice of input is just as much a form of supervision as the targets themselves. Under this view, the model is provided with three

types of supervision during training: the target-side labels, target-side inputs, and source-side inputs.

The path-star task (PST) is designed to induce a bad interaction between these three types of supervision under standard training. For a given step  $i$ , the LM is trained to predict the target-side label  $x_i \in R_t$ . However, it will be given  $x_{<i}$ , including  $x_{i-1}$  as target-side input due to TF. This induces learning a trivial single-edge lookup as the edge  $(x_{i-1}, x_i)$  is provided in the source-side input.<sup>2</sup> Fig. 5a illustrates the CHC as a single-edge lookup.

Thus excessive supervision partitions the sequential target-label supervision into supporting two tasks: the desired PST, supported by a single target label, and the undesired single-edge lookup task, supported by the remaining labels. This indicates that the task is not constructed properly to induce learning the PST. **We will demonstrate that 1) this bad interaction, and thus the CHC, can be avoided in various ways and, 2) avoiding the CHC is not actually critical for learning the task if we consider other ways the task is supervised.**

Task construction is a form of supervision encompassing multiple design decisions. We considered the target-side above, however, the source-side representation is also supervised. For example, how the  $G$  is shuffled matters (edge-wise vs. arm-wise (Frydenlund, 2024) or causal-wise as in Tbl. 3), the decision to place  $Q$  after or before  $G$  or which tokens to include in the query (Sec. 3.5).

There are also non-representational forms of supervision in creating the training data and training procedure. Bachmann and Nagarajan (2024) consider training and evaluating each model of graphs of the exact same size. This is done to explicitly dismiss out-of-domain effects.<sup>3</sup> Alternatively, we can train the models on various sizes (Sec. 3.6). To elucidate why this is supervision, we could supervise the order of data to guide training from easy to hard via curriculum learning (Bengio et al., 2009).

<sup>2</sup>To foreshadow Sec. 3.4, this would not be possible if that edge was not provided in the source-side inputs.

<sup>3</sup>“For each experiment, we generate the training and test graphs from the same distribution ... with fixed  $[D]$ ,  $[M]$  and  $[|V|]$ . Thus, any failure we demonstrate is an in-distribution failure, and does not arise from the inability to generalize to different problem lengths” (Bachmann and Nagarajan, 2024).

Our choices of supervision to include  $s$  and  $t$  in  $Q$  and only considering same-sized graphs leads to learning shortcuts for trivially predicting  $s$  and  $t$ . These are not bigram-based like the CHC, but positional as they are given on the source-side and always appear in the same place on the target-side.

## 2.2 Sensitivity Conjecture

Why learning  $l_t$  from a single target token is difficult is an open question. Frydenlund (2024) conjectured it relates to the task being sensitive to a single token,  $t$ . Hu et al. (2025a) provided a construction of parity as a path-star task that only generates  $l_t$ , implying it is at least as hard to solve as parity. Parity is maximally sensitive and known to be extremely difficult to learn with transformers (Bhatamishra et al., 2023; Hahn and Rofin, 2024). This conjecture motivates some methodology, however, we find little empirical support for it (Sec. 3.5).

## 3 Methods and Experiments

We use decoder-only models with 2 heads, 64 dim. embeddings, 256 dim. feed-forward layers, and learned positional embeddings. We use  $L = 8$  layers for most experiments. Having  $M < L$  allows for the linear graph reconstruction alg. to be learnt. Frydenlund (2024) proved  $\mathcal{O}(\log(M))$  layers are sufficient for this in theory. This was empirically demonstrated by Yin et al. (2024); Saparov et al. (2025) who use  $L < M$ . Sanford et al. (2024b) demonstrated that this  $\mathcal{O}(\log(M))$  alg. can be learnt for a related task. We use a learning-rate of  $5 \times 10^{-4}$ , a batch-size of 1024, and do not use dropout or a scheduler. We train for 100M samples.

We use  $D \in \{2, 3, 4, 5\}$ ,  $M \in \{5, 7, 9, 12, 15\}$  but only up until we observe unsuccessful trials.

We modify the task setting from Bachmann and Nagarajan (2024) by a) placing  $Q$  before  $G$ , b) using an online dataset to avoid overfitting, and c) setting  $|V| = |G|$  instead of 100. **These changes are immaterial to any conjecture regarding an inability to plan and do not prevent learning the CHC or its apparent effect on the task.** We confirm the PST is still unlearnable (even on minimal graphs with only  $|G| = 9$  nodes) in this setting in Fig. 3 (full results are in Tbl. 1 in Appx. B.1).

Fig. 3 shows the PST is learnable with causal-wise shuffling, **indicating that the causal constraint accounts for some of the task’s difficulty.** All further experiments use ‘edge-wise’ shuffling.

**In the following sections, we will introduce a**

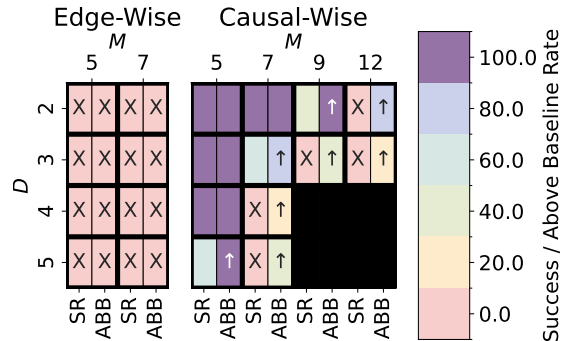


Figure 3: Baseline results. We report the *Success Rate* (SR) where the model predicts  $> 95\%$  sequential accuracy over  $n = 5$  seeded trials and *Above-Baseline* (ABB) where the model predicts  $> (100/D + 10)\%$  sequential accuracy. This happens when the model can predict  $l_t$  above  $1/D$  chance. As such, when  $ABB > SR$  ( $\uparrow$ ), it implies that the model has overcome the main challenge of the PST and would have learnt the task had it been provided with more training time in these cases. An ‘x’ further indicates no trials learnt the task.

**method or a slight modification to the task which allows the task to be learnt. In general, these will use standard teacher-forcing and next-token prediction. They will also be orthogonal to each other. Importantly, the success of each method can be explained as avoiding supervision adulteration and so inducing subtask decomposition** (one example of this is illustrated in Fig. 5).

### 3.1 Token Masking

We first consider token masking to address the adulteration. This will discourage learning the CHC by preventing conditioning on fully observed prior ground-truths during training, thus breaking the bad supervision interaction by modifying the target-side inputs. This is motivated by the limited successes of ‘teacher-less’, iterative-, and non-autoregressive models (Frydenlund, 2024).

A main innovation from these models is that we do not need to employ full masking, unlike the ‘teacher-less’ and non-autoregressive models and we can keep the causal parameterization of the model, unlike the iterative- and non-autoregressive models. Importantly, this can be achieved via ubiquitous data-noising methods used with standard TFed training. In particular, we can employ either token dropout/masking (Gal and Ghahramani, 2016; Bowman et al., 2016) or token replacements via scheduled sampling<sup>4</sup> (Bengio et al., 2015) (or

<sup>4</sup>As we know how the model generates, we skip implementing scheduled sampling and just sample from  $V$  instead.

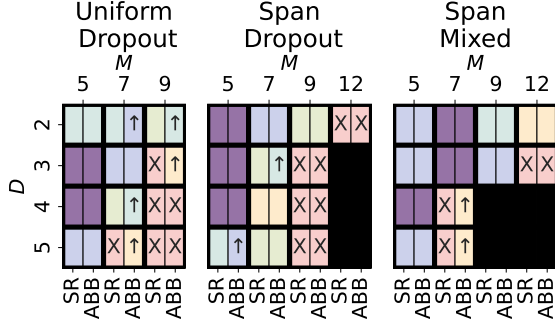


Figure 4: Masking results (full Tbl. 3 in Appx. B.2).

a mix). Replacement has the benefit of providing an anti-CHC learning signal as the model can not trust edge look-ups, but it introduces more complex noise. We try both uniform sampling of the sequence length and contiguous span sampling to shun consecutive ground-truths (Joshi et al., 2020).

### 3.1.1 Results and Discussion

Fig. 4 shows that masking makes the task learnable but struggles as  $D$  and  $M$  scale. We find minor differences in the two masking types and try mixing them, as they may provide different benefits (token replacement tells the model not to trust single-edge lookups while a masked token prevents these).

Finding that a given method makes the PST learnable but does not scale will be a consistent pattern across methods. **Our focus for these methods is, a), showing that the task becomes learnable and, b), explaining why.** We conjecture about limitations in scalability in Sec. 5 which were also observed by Saparov et al. (2025, see Appx. C.2).

### 3.1.2 Unadulterated Task Decomposition

We show how masking prevents the CHC in Fig. 5. First, consider the CHC in Fig. 5a and the needed algorithm for predicting  $l_t$  in Fig. 5b. The CHC learns a forward alg. from the prior token, while the required alg. must work backward from the target query. Figs. 5c and 5d show how masking induces multi-edge lookups. In Fig. 5c, when all prior tokens are masked, it induces learning a subset of steps for the required alg. **This provides a deeper explanation for why masking works beyond preventing the CHC; it induces task decomposition.** This also explains why unadulterated sequential supervision is important. Decomposition can occur because arm reconstruction is inherently recursive.

Fig. 5d shows that having unmasked prior tokens may lead to learning a forward alg. While these

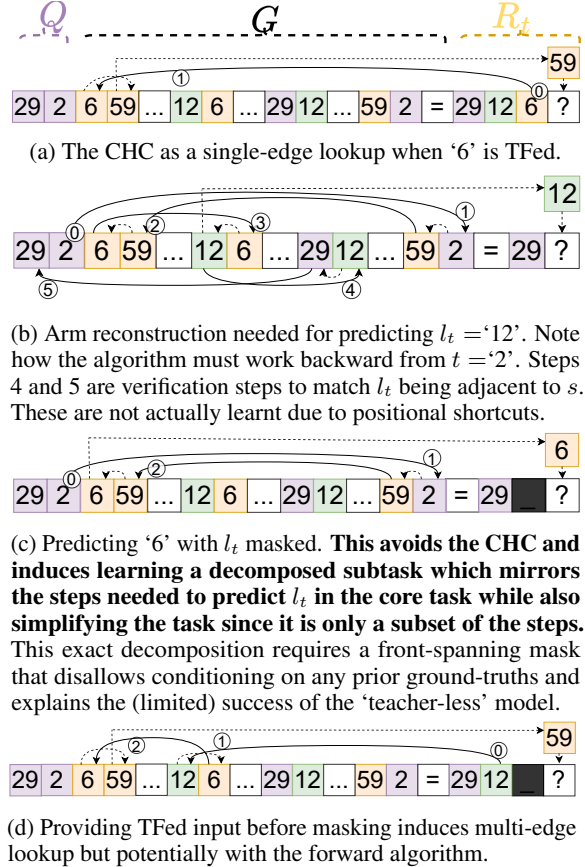


Figure 5: Algorithmic steps performed in the CHC and arm reconstruction, also with masking (blacked-out).

seem similar to a human, they are different algs. and it's unclear if they mutually support each other in terms of learning to predict  $l_t$ , i.e., does a subtask need to exactly mirror the core task for subtask decomposition to help learning the core task.

### 3.2 Alternative Sequential Distributions

We consider learning a distribution over the next tokens instead of the next token, i.e., multi-token prediction. This is done via learning a belief-state,  $B$ , which is a hidden-state that supports making future predictions via some linear function of  $B$  i.e.,  $P_B(x_{i:M} | f(B = x_{<i}))$  (Hu et al., 2025a).

We present three simple methods for learning this future distribution,  $P_B$ : bag-of-words (BoW), label-smoothing (LS), and ranking. Yin et al. (2024) used a BoW baseline with  $R_t$  as the bag. This performed nearly as well as their proposed model and solved the task in the majority of cases.<sup>5</sup> We exclude prior tokens ( $< i$ ) from the bag so as to only contain future tokens at each step. Note BoW is equivalent to LS with uniform smoothing.

BoW is based on the inductive bias that nodes

<sup>5</sup>They used pretrained GPT2 models in their experiments.

in  $R_t$  are more important than nodes in the other arms. We can extend this with another inductive bias which assumes that near-present tokens are more important than far-future tokens, i.e., that the order matters. This can be achieved using monotonically decreasing label weights. Thus, LS requires hand-crafted weights to form a hand-crafted distribution that the model tries to match. We can avoid this ad-hoc approach via an equivalence between LS and ranking (Frydenlund et al., 2022).

### 3.2.1 Ranking-into-the-Future (RITF)

We generalize from LS to ranking-into-the-future by constructing rank targets and training with a rank-based loss, providing a structured loss over multiple tokens at each time-step. As we are using future tokens, the structure is over the sequence, and the sequential order is used for the rank-order.

Let the future distribution at a single step  $i$  be  $P_{B,i}(x_i \succ x_{i+1} \succ \dots \succ x_M)$  such that the scores of sequential tokens decreases monotonically from time-step  $i$ . Let  $\sigma_i = f(B = x_{<i})$  be a vector of these scores or logits. Then we use a pair-wise hinge loss over the entire sequence in  $R_t$  s.t.

$$L_B = \sum_{i=1}^M \sum_{j=i}^M \sum_{k=j+1}^M \max(0, 1 - (\sigma_i[j] - \sigma_i[k])) \quad (2)$$

We incorporate another bias that ranks tokens in  $R_t$  above all others, encoding the concept that the correct arm is more important than the others.<sup>6</sup> This creates very dense supervision with  $M(M-1)/2$  intra- and  $M^2(|V|-M)$  inner-arm pairs. Initial experimentation found this was crucial. This makes sense as, when the model fails to learn to predict  $l_t$ , it learns a uniform distribution over the set of leading nodes, meaning they have equal scores. This secondary bias creates supervision that  $l_t$  is more important than the other leading nodes. Note that this is already implicitly done in any cross-entropy loss, including BoW and LS, as cross-entropy works by promoting the singular ground-truth while demoting all other nodes.

### 3.2.2 Results and Discussion

Fig. 6 shows that RITF is superior to both BoW and LS. For LS we use a stepped monotonically decreasing weight (Frydenlund et al., 2022). We believe this does not work as it couples the inductive bias with loss scaling (so future tokens have tiny weights). We know the inductive bias is not at

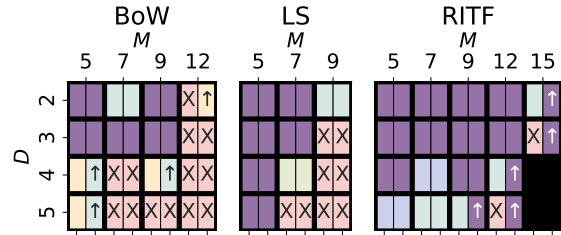


Figure 6: Alt. distro. results (full Tbl. 4 in Appx. B.3).

fault, as it is the same as in RITF. **This shows that it is easier to specify rules than specific weights.**

More importantly, future predictions make the task learnable for multiple related reasons. First, the multi-token loss requires multi-edge lookups and so induces decomposition. Second, it avoids adulteration by skipping adjacent inputs for all targets at  $> i + 1$ . **This is the same as masking, except instead of using noised input to cause the skip, it is implicitly defined as part of the loss.** The first step is also fully masked, thus inducing the desired backward alg. for all tokens. Third, by applying this at each time-step, **the loss induces novel dense decomposition across the sequence** where learning  $P_{B,i+1}$  is a sub-problem of learning  $P_{B,i}$  (see Appx. A.2 for an expanded discussion).

### 3.3 Scratchpads (SP) to Increase Supervision

SPs predict an intermediate sequence before the target sequence, providing auxiliary input and target supervision (Nye et al., 2022). Both the reverse arm order and the arm-wise graph shuffle make the task trivial, and so would be obvious SPs. These are problem-specific and do not prevent adulteration and, so, are not insightful. We present alternatives.

Instead, for arm reconstruction (AR-SP), we generalize the reverse order to generate the arm nodes as a BoW in any order. As there are  $M!$  orderings, we use LS over the choices. This unifies the auxiliary BoW and single next-token distributions in Sec. 3.2 since, *the next  $M$  tokens are the BoW*. We can avoid LS by determining a canonical ordering via sorting by node values. **This introduces node semantics.** This may provide strong supervision as nodes in  $R_t$  need to be identified and then ordered, which requires making comparisons that will not match the source-side edges, thus avoiding the bad interaction that causes the CHC.

We also use SPs that reconstruct the entire graph by ordering the arms (GR-SP). Full reconstruction would cause adulteration, so we just match leading- and target-node pairs. We order the arms by leading- or target-node value, again, introducing

<sup>6</sup>This is not included in Eq. 2. See Appx. B.3.

semantics (see Fig. 21 in Appx. B.5).

### 3.3.1 Results and Discussion

AR-SP results can be seen in Tbl. 5 in Appx. B.4. We report the sequential accuracy for  $R_t$  and the SP separately to isolate where any errors occur. The reverse SP is trivially learnt as expected. Some BoW trials are successful. Importantly, this can be explained by inducing subtask decomposition (Appx. A.3). While the BoW SP make the PST learnable (on small scales), its performance is disappointing, since an obvious solution (to a human) exists, which would be to generate the BoW in the reverse order, however, this solution is not found. As Tbl. 5 shows, the models fail to correctly predict the SP and then fails to predict  $R_t$  when conditioning on the incorrect SP. This failure is informative as it shows **the reverse solution is only trivial when it is provided with direct supervision** (the same sequence supervision that causes the CHC).

The performance of the sorted SP is harder to explain. Sorting naturally decomposes, but, by design, is agnostic to graph edges (except for the identification step). It may be that this subtask does not mutually support learning the PST task. However, the issue is that the model fails to learn to sort at all at scale, so we suspect that this the same scaling issue affecting the other methods.

GR-SP results can be seen in Tbl. 6 in Appx. B.5. These only learn to solve the task in 4/80 trials and this is exceedingly informative. We have four variants of the GR-SP, as we can go from leading-to-target-nodes or vice versa, and then either sort by leading- or target-node values. Fig. 22 in Appx. B.5 plots the accuracy of each SP token across training. The models learn to correctly identify the needed sets of leading and target nodes. This is done by single-edge shortcuts; leading nodes by adjacency to  $s$  and targets nodes by counting degree. The models also correctly learn to sort either the leading- or target-nodes. This means that for the SP where we go from leading-to-targets, sorted by leading nodes, the model can correctly identify the first leading-node but fails to connect it with its paired target. The same thing happens using targets-to-leading, sorted by target nodes.

**In both cases, the model knows which arm to reconstruct, and can condition on either the correct leading or target node, but still does not learn the actual reconstruction! Here, all the model needs to do is deterministic path-following with no planning to choose the correct**

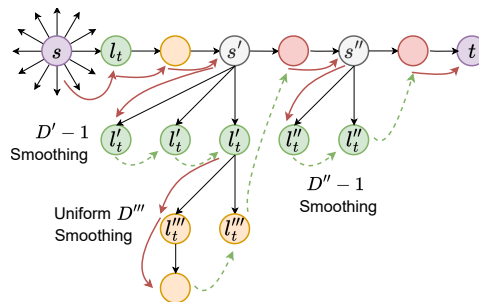


Figure 7: Red and green-dashed arrows form the desired pre-order traversal; red edges are included in  $G$ , while green edges are not, and hence are immune to the CHC.

**arm. This begs the question: if the solution is deterministic and does not require choices, is this actually a planning problem?** These and the BoW SP results indicate that arm reconstruction is what makes the task hard – not planning. Thus, these negative results are consistent with and indirectly support the theory that supervised task decomposition is necessary, since the scratchpads the fail are those that do not induce decomposition.

### 3.4 From Path-Star to Tree-Star

Here, we partially prevent the CHC and induce task decomposition via the source-side modification of changing the graph topology from paths to trees.

One way of preventing the CHC is to remove or modify edges in  $G$  to prevent single-edge lookups. We achieve this by generalizing the task to consider arms as trees instead of paths. In particular, we train on trees but evaluate on path-star graphs. Training on trees slightly changes the training objective as we are not generating the arm – which is more generally the shortest path from  $s$  to  $t$  – but rather an equivalent pre-order traversal of the tree. This introduces a problem in that such a traversal requires a planner tree to determine the order of multiple child nodes in the traversal. Encoding it as a planner tree will induce new undesired shortcuts.

We employ a trick to avoid this. By task definition,  $t$  must be the last generated token. This precludes any subtree containing  $t$  from being generated before the others. Then, given a subtree containing  $D'$  child nodes (including  $t$ ), the distribution over these children being valid continuations of a traversal is asymmetric in that  $t$  is excluded but uniform over the remaining  $D' - 1$  choices. However, this only works for subtrees containing  $t$ . This is achieved via LS over valid child nodes during training. Call these  $D'$ -ary trees. See Fig. 7.

Following this logic, we can design the tree to

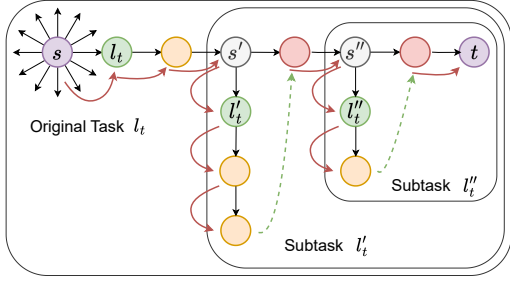


Figure 8: A split tree. Each split induces a decomposition and there is only a single valid pre-order traversal.

D	D-ary Trees			Split Trees					
	5	M	7	5	7	M	9	12	15
2	↑	X	↑	↑	↑	↑	↑	X	↑
3	↑	↑	↑	↑	↑	X	↑	X	↑
4	X	↑	X	↑	↑	X	X	X	X
5	X	↑	X	↑	X	X	X	X	X

Figure 9: Split tree results (Tbl. 8 in Appx. B.7)

resolve any ordering ambiguity and avoid needing LS with a deterministic traversal by only allowing subtrees containing  $t$  to have a max of 2 children and all others one child, i.e., structurally-lopsided binary trees. Call these *split trees*. See Fig. 8.

### 3.4.1 Results and Discussion

Fig. 9 shows that training on split trees makes the task learnable. This can again be explained as inducing task decomposition. In Fig. 8, each subtask (marked as primes) is similar to a path-star graph with  $D' = 2$  where a new start node,  $s'$ , is any node with two children and the correct leading node,  $l'_t$ , is the first node of the subtree not containing  $t$ . It is interesting that experiments where  $D > 2$  work since the induced subtask is restricted to  $D' = 2$  and thus does not match the original task.

We find that the  $D$ -ary trees do not learn the task outside of a few cases, despite introducing a similar decomposition. We conjecture that smoothing over subtrees with  $D$  choices creates a deficient subtask that mirrors the adulterated PST task since we are explicitly forcing the model to learn a uniform distribution over leading nodes and this matches the undesired learnt behaviour of models that fail the task, i.e., we are directly teaching the model to learn the exact behaviour we are trying to avoid.

Training on trees and evaluating path-star graphs may look like an exotic solution, but we stress this is actually a generalization of graph topology and one that does not go far enough. We suspect that

D	Query Subset			GST (New)		GST (Orgl.)	
	5	M	9	5	7	M	5
2	↑	↑	↑	↑	↑	↑	↑
3	X	X	X	↑	↑	X	X
4	X	X	X	↑	↑	X	X
5	X	X	X	↑	↑	X	X

Figure 10: Results for general query methods. Full Tbl. 7 in Appx. B.6, along with original-setting experiments.

the best graph topology would be one that allows for perfect decomposition, where each subtask exactly mirrors the core task except for a change in the number of recurrent steps needed to solve the task. That is, the choice of graph topology will affect subtask homology. This conjecture would explain the necessity of ‘balanced’ graphs beyond preventing shortcuts (Saparov et al., 2025).<sup>7</sup>

Training on trees and then evaluating path-star graphs is also interesting since it is counterintuitive from the perspective of in-domain learning; we require training on trees to generalize to paths when direct training on paths fails. Ironically, the PST is defined as it is to rule out out-of-domain effects (Bachmann and Nagarajan, 2024). From the perspective of adulteration, paths are an excessively informative graph structure. Also, training on paths to evaluate paths is more direct task supervision than training on trees to evaluate paths.

### 3.5 Generalized Queries

Given the sensitivity conjecture, we consider if providing more than one node from  $R_t$  in  $Q$  will decrease sensitivity. We do this by sampling a subset of  $R_t$  (in any order to avoid adulteration). This is similar to token masking in that we are supporting prediction via providing multiple tokens from  $R_t$  to condition on, except this is being applied on the source-side. During inference, only  $t$  is given.

#### 3.5.1 Results and Discussion

Fig. 10 shows that using a subset of  $R_t$  makes the task learnable for small graphs. To verify that that is due to decreasing sensitivity, we tried sampling a single node as a general single target (GST) from  $R_t$ . We find that not only does this make the task learnable, it performs better than using a subset. We expect this is because a subset introduces too much noise. However, if this does not work because of

<sup>7</sup>Shortcuts are probably the symptom, not the illness.



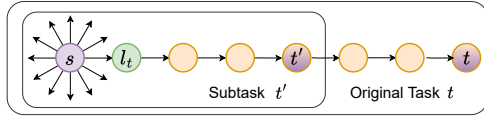


Figure 11: Sampling  $t$  as task decomposition.  $t$  is the original target at position  $M$  and  $t'$  is a sampled target.

reduced sensitivity, why does it work? Because it induces task decomposition (illustrated in Fig. 11).

Hu et al. (2025a) performed this same experiment and found it did not learn the task.<sup>8</sup> To explain this contradiction, we experiment using the original task settings (using an offline dataset,  $|V| = 100$ , and  $Q$  after  $G$ ). Here we find that the task is much harder to learn, with only 3/20 trials succeeding (Tbl. 7 in Appx. B.6). This implies that it would be easy to find only negative results, especially if seeded trials were not used. We argue further about issues with hyperparameters in Appx. B.6. **This highlights the importance of using an online dataset, and how other issues – not related to planning – contribute to the PST being unlearnable in the original experimental setting.**

### 3.6 Generalized Length Decomposition

Given the above, it should be obvious now that a direct way to induce subtask decomposition would be to supervise the training process by sampling different-sized graphs. Tbl. 9 in Appx. B.8 shows this makes the task learnable as expected. Training on various values of  $D$  does not seem to help. Combining general length and target sampling improves performance over just doing the former. We expect these results would be better if given more training time, as this introduces a lot of noise.

**These results show that training on various lengths is not just for out-of-domain generalization, but also promotes in-domain learning. This also prevents positional shortcuts.**

## 4 Conclusion

We have demonstrated that the path-star task, which was seemingly unlearnable, is actually learnable with decoder-only models. We have shown how the original task is designed with adulterated supervision, explained why this makes it unlearnable due to obscuring or absorbing decomposition supervision, and shown that preventing the CHC is not critical for learning the task given some decomposition supervision. We developed multiple

methods to overcome this lack of supervision, with all retaining the next-token prediction paradigm with standard training via teacher-forcing.<sup>9</sup>

**We have empirically demonstrated that decomposition is critical for learning to search over graphs.** This is strongly supported by the fact that the methods we have developed are all orthogonal to each other but can all be explained as inducing subtask decomposition in some form.<sup>10</sup>

A few informative negative results suggest that the core difficulty of the task does not concern planning at all but rather graph reconstruction and that seemingly trivial solutions will not be found unless directly supervised. We also show that graph reconstruction is made more difficult for decoder-only models due to causal constraints via the causal-wise shuffling experiments.

Our work serves as a bridge between Bachmann and Nagarajan (2024) and Saparov et al. (2025) by providing explanation for why the graph search task presented by the former is seemingly unlearnable, while the very similar graph search task presented by the latter is learnable; the latter provided decomposition supervision (specifically they incorporated general graph topologies and lengths), while the former did not due to adulteration. Note, these still permit the CHC. While this is not critical for learning the task, preventing all shortcuts, say by masking, may be important for generalization.

As searching is recursively defined, decomposition is inherent in the original task. Thus, we can induce it using the original supervision – provided we are careful not to adulterate it. This contrasts with other tasks that require introducing scratchpads or extra supervisory information – done by modifying the task itself or learning a secondary subtask that is decompositional (Wies et al., 2023).

**If one were concerned about the implications that the empirical results of the path-star task had on the sufficiency of the next-token prediction paradigm for planning tasks, this work alleviates those concerns. If one is skeptical about such conjectures, this validates their beliefs with an explanation of why the PST in its original form is unlearnable. Our findings show that the task is fragile, where minor changes induce decomposition and make it learnable, which indicates these issues will not apply to – or be nearly as potent to – complex tasks as conjectured.**

<sup>8</sup>And they performed this experiment explicitly to determine if subtask decomposition makes the task learnable.

<sup>9</sup>Except for the future distributions.

<sup>10</sup>See Appx. A for a summary and comparison of methods.

## 5 Limitations

**Scaling issues:** The major limitation of our work is that each method fails to scale with either  $D$  or  $M$ . We believe that using graph topologies that allow for stronger and more consistent decomposition, where each subtask mirrors the main task, will be key for scalability. Thus, we think that path-star experimental setting is not a suitable environment to consider how the methods we have developed will scale to larger graphs. Instead, we believe using more general graphs would be a better environment and leave this to future work. However, [Saparov et al. \(2025\)](#) observed similar scaling issues using more general graphs, which suggests that the issue does not stem from topology by itself (see Appx. C.2 for a comparison of our works). Still, this would let us avoid or isolate issues related to using topology when considering scale.

We conjecture that different issues may affect the scaling of  $D$  and  $M$ .  $M$  scales with the number of model layers ([Frydenlund, 2024](#)). It is unclear if learning the log alg. is harder than the linear alg.

**Model parameterization:** [Frydenlund \(2024\)](#) empirically demonstrated differences in performance on the PST between various model parameterizations. The difference between decoder-only models and the others is due to the causal constraint. We show that overcoming the constraint makes the task learnable, but not exactly what induces learning to overcome it. We do, however, make a connection to the target-side encoder models by showing that masking, the same method used to train these models, makes the task learnable due to decomposition. Now that we have successfully shown that decoder-only models can learn the task, we can better explore the learnability conditions between the models in future work. Differences in parameterizations have been shown to be important in other symbolic tasks ([Ye et al., 2025a,b](#)).

**The value of the path-star task:** Given our findings, we argue that the PST is not suitable for evaluating the performance of novel methods or architectures for planning ([Yin et al., 2024](#); [Hu et al., 2025a](#), among others).<sup>11</sup> We also argue that graph search is not a stand-in for general search. Given the finesse required to learn the PST and issues with scalability, we suggest that graph search may be hard due to issues that only apply to these experimental settings and caution against making claims based on these tasks being used as surrogates for

other search tasks on natural data (see below for positive uses of the task). While our goal of this work is to show that next-token prediction is sufficient for learning the task, we do not consider what method is best. Given the above, we have introduced the novel ranking-into-the-future method, but have not evaluated its potential for planning on a suitable task. We frame RITF as a change in distribution from next-token prediction. This is important to the discussion of the paradigm in this work, but may not be important for other uses.

**Semantics as high-order graph structure:** The path-star task is semantless. This often applies to other minimal graph search tasks. While we do little experimentation in this direction, we conjecture that this plays an important role in learnability, where the right semantics would probably make graph reconstruction significantly easier. This relates to not using pretrained models, which would inherit natural language semantics.

**Why is task decomposition necessary:** We have empirically shown that task decomposition is necessary for learning the task. While it is intuitive why supervising decomposition will help learning, we do not explain why it is necessary. Thus, characterizing the core underlying difficulty is still an open question. We expect solving this will be insightful for learnability theory. **Thus we believe the original adulterated form of the path-star task is of scientific value and hope that research into it will continue.** We did not find positive results trying to decrease sensitivity with multiple query nodes, but our results are inconclusive and may be due to an increase in noise. Thus, the sensitivity conjecture is still a possible explanation.

**Shortcuts:** The PST is a great framework for exploring shortcut learning. Different variations result in different shortcuts, especially when using the SPs. One question we have is if shortcuts, once learnt, actively harm learning the desired task or if they are just benign symptoms of other issues?

**Alternative examples of adulteration:** We describe adulteration as a generic issue but only consider the context of graph searching. We believe it is a useful term for identifying similar issues which can be solved via similar methods. For example, [Chang and Bisk \(2025\)](#) considered trying to learn to count by providing a model with a contiguous sequence of numbers and training using next-token prediction. As they point out, this will be ineffective due to trivial bigram shortcuts. This is because they have presented the task in an adulterated form.

<sup>11</sup>They also both perform additional non-PST experiments.

## Acknowledgments

We thank Frank Rudzicz and Gagandeep Singh for helping to proofread, and Rich Zemel for suggesting some clarifying experiments.

Resources used in preparing this research were provided, in part, by the Province of Ontario, the Government of Canada through CIFAR, and companies sponsoring the Vector Institute.

## References

- Emmanuel Abbe, Samy Bengio, Aryo Lotfi, and Kevin Rizk. 2024a. Generalization on the unseen, logic reasoning and degree curriculum. *Journal of Machine Learning Research*, 25(331):1–58.
- Emmanuel Abbe, Samy Bengio, Aryo Lotfi, Colin Sandon, and Omid Saremi. 2024b. [How far can transformers reason? the globality barrier and inductive scratchpad](#). In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- Kwangjun Ahn, Alex Lamb, and John Langford. 2025. Efficient joint prediction of multiple future tokens. *arXiv preprint arXiv:2503.21801*.
- Réka Albert and Albert-László Barabási. 2002. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47.
- Zeyuan Allen-Zhu and Yuanzhi Li. 2023. Physics of language models: Part 1, context-free grammar. *arXiv preprint arXiv:2305.13673*.
- Zeyuan Allen-Zhu and Yuanzhi Li. 2024. [Physics of language models: Part 3.1, knowledge storage and extraction](#). In *Forty-first International Conference on Machine Learning*.
- Chenyang An, Shima Imani, Feng Yao, Chengyu Dong, Ali Abbasi, Harsh Shrivastava, Samuel Buss, Jingbo Shang, Gayathri Mahalingam, Pramod Sharma, et al. 2024. Next-token prediction task assumes optimal data ordering for llm training in proof generation. *arXiv preprint arXiv:2411.00863*.
- Cem Anil, Yuhuai Wu, Anders Johan Andreassen, Aitor Lewkowycz, Vedant Misra, Vinay Venkatesh Ramasesh, Ambrose Slone, Guy Gur-Ari, Ethan Dyer, and Behnam Neyshabur. 2022. [Exploring length generalization in large language models](#). In *Advances in Neural Information Processing Systems*.
- Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. 2021. [Structured denoising diffusion models in discrete state-spaces](#). In *Advances in Neural Information Processing Systems*.
- Gregor Bachmann and Vaishnavh Nagarajan. 2024. [The pitfalls of next-token prediction](#). In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 2296–2318. PMLR.
- Tanja Baeumel, Josef van Genabith, and Simon Ostermann. 2025. The lookahead limitation: Why multi-operand addition is hard for llms. *arXiv preprint arXiv:2502.19981*.
- Nishant Balepur, Shramay Palta, and Rachel Rudinger. 2024. [It’s not easy being wrong: Large language models struggle with process of elimination reasoning](#). In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 10143–10166, Bangkok, Thailand. Association for Computational Linguistics.
- Albert-László Barabási and Réka Albert. 1999. [Emergence of scaling in random networks](#). *Science*, 286(5439):509–512.
- Nora Belrose, Zach Furman, Logan Smith, Danny Halawi, Igor Ostrovsky, Lev McKinney, Stella Biderman, and Jacob Steinhardt. 2023. Eliciting latent predictions from transformers with the tuned lens. *arXiv preprint arXiv:2303.08112*.
- Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1, NIPS’15*, page 1171–1179, Cambridge, MA, USA. MIT Press.
- Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48.
- Lukas Berglund, Meg Tong, Maximilian Kaufmann, Mikita Balesni, Asa Cooper Stickland, Tomasz Korbak, and Owain Evans. 2024. [The reversal curse: LLMs trained on “a is b” fail to learn “b is a”](#). In *The Twelfth International Conference on Learning Representations*.
- 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.
- Satwik Bhattamishra, Michael Hahn, Phil Blunsom, and Varun Kanade. 2024. [Separations in the representational capabilities of transformers and recurrent architectures](#). In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- Satwik Bhattamishra, Arkil Patel, Varun Kanade, and Phil Blunsom. 2023. [Simplicity bias in transformers and their ability to learn sparse Boolean functions](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume*

- 1: Long Papers*), pages 5767–5791, Toronto, Canada. Association for Computational Linguistics.
- Jing Bi, Yuting Wu, Weiwei Xing, and Zhenjie Wei. 2024. Enhancing the reasoning capabilities of small language models via solution guidance fine-tuning. *arXiv preprint arXiv:2412.09906*.
- Ning Bian, Xianpei Han, Le Sun, Hongyu Lin, Yaojie Lu, Ben He, Shanshan Jiang, and Bin Dong. 2024. [ChatGPT is a knowledgeable but inexperienced solver: An investigation of commonsense problem in large language models](#). In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 3098–3110, Torino, Italia. ELRA and ICCL.
- Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. 2016. [Generating sentences from a continuous space](#). In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 10–21, Berlin, Germany. Association for Computational Linguistics.
- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D 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 Ziegler, Jeffrey Wu, Clemens Winter, Chris 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 *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.
- Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*.
- Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. 2024. [Medusa: Simple LLM inference acceleration framework with multiple decoding heads](#). In *Forty-first International Conference on Machine Learning*.
- Ziwei Chai, Tianjie Zhang, Liang Wu, Kaiqiao Han, Xiaohai Hu, Xuanwen Huang, and Yang Yang. 2023. [Graphllm: Boosting graph reasoning ability of large language model](#). *arXiv preprint arXiv:2310.05845*.
- Mohna Chakraborty, Adithya Kulkarni, and Qi Li. 2023. [Zero-shot approach to overcome perturbation sensitivity of prompts](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5698–5711, Toronto, Canada. Association for Computational Linguistics.
- Yingshan Chang and Yonatan Bisk. 2025. [Language models need inductive biases to count inductively](#). In *The Thirteenth International Conference on Learning Representations*.
- Changyu Chen, Xiting Wang, Ting-En Lin, Ang Lv, Yuchuan Wu, Xin Gao, Ji-Rong Wen, Rui Yan, and Yongbin Li. 2024a. [Masked thought: Simply masking partial reasoning steps can improve mathematical reasoning learning of language models](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5872–5900, Bangkok, Thailand. Association for Computational Linguistics.
- Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. 2023a. [Accelerating large language model decoding with speculative sampling](#). *arXiv preprint arXiv:2302.01318*.
- Nuo Chen, Yuhan Li, Jianheng Tang, and Jia Li. 2024b. [Graphviz: An instruction-following language model for graph computational problems](#). In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD '24*, page 353–364, New York, NY, USA. Association for Computing Machinery.
- Xinyun Chen, Ryan Andrew Chi, Xuezhi Wang, and Denny Zhou. 2024c. [Premise order matters in reasoning with large language models](#). In *Forty-first International Conference on Machine Learning*.
- Yanda Chen, Chen Zhao, Zhou Yu, Kathleen McKeown, and He He. 2023b. [On the relation between sensitivity and accuracy in in-context learning](#). In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 155–167, Singapore. Association for Computational Linguistics.
- Yongqiang Chen, Binghui Xie, Kaiwen Zhou, Bo Han, Yatao Bian, and James Cheng. 2023c. [Positional information matters for invariant in-context learning: A case study of simple function classes](#). *arXiv preprint arXiv:2311.18194*.
- Ta-Chung Chi, Ting-Han Fan, Li-Wei Chen, Alexander Rudnicky, and Peter Ramadge. 2023. [Latent positional information is in the self-attention variance of transformer language models without positional embeddings](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1183–1193, Toronto, Canada. Association for Computational Linguistics.
- David Chiang and Peter Cholak. 2022. [Overcoming a theoretical limitation of self-attention](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7654–7664, Dublin, Ireland. Association for Computational Linguistics.
- Hanseul Cho, Jaeyoung Cha, Pranjal Awasthi, Srinadh Bhojanapalli, Anupam Gupta, and Chulhee Yun.

- 2024a. [Position coupling: Improving length generalization of arithmetic transformers using task structure](#). In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- Hanseul Cho, Jaeyoung Cha, Srinadh Bhojanapalli, and Chulhee Yun. 2024b. Arithmetic transformers can length-generalize in both operand length and count. *arXiv preprint arXiv:2410.15787*.
- Francois Chollet, Mike Knoop, Gregory Kamradt, and Bryan Landers. 2024. Arc prize 2024: Technical report. *arXiv preprint arXiv:2412.04604*.
- François Chollet. 2024. Openai o3 breakthrough high score on arc-agi-pub. <https://arcprize.org/blog/oai-o3-pub-breakthrough>. Accessed: 2024-12-26.
- Zheng Chu, Jingchang Chen, Qianglong Chen, Weijiang Yu, Tao He, Haotian Wang, Weihua Peng, Ming Liu, Bing Qin, and Ting Liu. 2024. [Navigate through enigmatic labyrinth a survey of chain of thought reasoning: Advances, frontiers and future](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1173–1203, Bangkok, Thailand. Association for Computational Linguistics.
- Andrew Cohen, Andrey Gromov, Kaiyu Yang, and Yuandong Tian. 2025. Spectral journey: How transformers predict the shortest path. *arXiv preprint arXiv:2502.08794*.
- Antonia Creswell, Murray Shanahan, and Irina Higgins. 2023. [Selection-inference: Exploiting large language models for interpretable logical reasoning](#). In *The Eleventh International Conference on Learning Representations*.
- Xinnan Dai, Haohao Qu, Yifen Shen, Bohang Zhang, Qihao Wen, Wenqi Fan, Dongsheng Li, Jiliang Tang, and Caihua Shan. 2024a. How do large language models understand graph patterns? a benchmark for graph pattern comprehension. *arXiv preprint arXiv:2410.05298*.
- Xinnan Dai, Qihao Wen, Yifei Shen, Hongzhi Wen, Dongsheng Li, Jiliang Tang, and Caihua Shan. 2024b. Revisiting the graph reasoning ability of large language models: Case studies in translation, connectivity and shortest path. *arXiv preprint arXiv:2408.09529*.
- Mark Z Danielewski. 2000. *House of Leaves: The Remastered, Full-Color Edition*. Pantheon.
- Artur Back de Luca and Kimon Fountoulakis. 2024. [Simulation of graph algorithms with looped transformers](#). In *Forty-first International Conference on Machine Learning*.
- Gregoire Deletang, 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](#). In *The Eleventh International Conference on Learning Representations*.
- Xiang Deng, Yu Su, Alyssa Lees, You Wu, Cong Yu, and Huan Sun. 2021. [ReasonBERT: Pre-trained to reason with distant supervision](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6112–6127, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Wenxuan Ding, Shangbin Feng, Yuhan Liu, Zhaoxuan Tan, Vidhisha Balachandran, Tianxing He, and Yulia Tsvetkov. 2024. [Knowledge crosswords: Geometric knowledge reasoning with large language models](#). In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 2609–2636, Bangkok, Thailand. Association for Computational Linguistics.
- Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, Xu Sun, Lei Li, and Zhifang Sui. 2024. [A survey on in-context learning](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1107–1128, Miami, Florida, USA. Association for Computational Linguistics.
- Mengnan Du, Fengxiang He, Na Zou, Dacheng Tao, and Xia Hu. 2023. Shortcut learning of large language models in natural language understanding. *Communications of the ACM*, 67(1):110–120.
- Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Sean Welleck, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D. Hwang, Soumya Sanyal, Xiang Ren, Allyson Ettinger, Zaid Harchaoui, and Yejin Choi. 2023. [Faith and fate: Limits of transformers on compositionality](#). In *Thirty-seventh Conference on Neural Information Processing Systems*.
- Guy Emerson. 2020. [What are the goals of distributional semantics?](#) In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7436–7453, Online. Association for Computational Linguistics.
- P Erdős and A Réwi. 1959. On random graphs i. *Publ. math. debrecen*, 6(290-297):18.
- Lizhou Fan, Wenyue Hua, Lingyao Li, Haoyang Ling, and Yongfeng Zhang. 2024. [NPHardEval: Dynamic benchmark on reasoning ability of large language models via complexity classes](#). In *Proceedings of the*

- 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 4092–4114, Bangkok, Thailand. Association for Computational Linguistics.
- Lizhe Fang, Yifei Wang, Khashayar Gatmiry, Lei Fang, and Yisen Wang. 2025. [Rethinking invariance in in-context learning](#). In *The Thirteenth International Conference on Learning Representations*.
- Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. 2024. [Talk like a graph: Encoding graphs for large language models](#). In *The Twelfth International Conference on Learning Representations*.
- Guhao Feng, Bohang Zhang, Yuntian Gu, Haotian Ye, Di He, and Liwei Wang. 2023. [Towards revealing the mystery behind chain of thought: A theoretical perspective](#). In *Thirty-seventh Conference on Neural Information Processing Systems*.
- Dan Friedman, Alexander Wettig, and Danqi Chen. 2023. [Learning transformer programs](#). In *Advances in Neural Information Processing Systems*, volume 36, pages 49044–49067. Curran Associates, Inc.
- Arvid Frydenlund. 2024. [The mystery of the pathological path-star task for language models](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 12493–12516, Miami, Florida, USA. Association for Computational Linguistics.
- Arvid Frydenlund, Gagandeep Singh, and Frank Rudzicz. 2022. [Language modelling via learning to rank](#). *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(10):10636–10644.
- Yarin Gal and Zoubin Ghahramani. 2016. [A theoretically grounded application of dropout in recurrent neural networks](#). In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.
- Yuyao Ge, Shenghua Liu, Wenjie Feng, Lingrui Mei, Lizhe Chen, and Xueqi Cheng. 2024. [Graph descriptive order improves reasoning with large language model](#). *CoRR*, abs/2402.07140.
- Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. 2020. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673.
- Anastasios Gerontopoulos, Spyros Gidaris, and Nikos Komodakis. 2025. Multi-token prediction needs registers. *arXiv preprint arXiv:2505.10518*.
- Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. [Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies](#). *Transactions of the Association for Computational Linguistics*, 9:346–361.
- Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. [Mask-predict: Parallel decoding of conditional masked language models](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6112–6121, Hong Kong, China. Association for Computational Linguistics.
- Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Roziere, David Lopez-Paz, and Gabriel Synnaeve. 2024. [Better & faster large language models via multi-token prediction](#). In *Forty-first International Conference on Machine Learning*.
- Olga Golovneva, Zeyuan Allen-Zhu, Jason E Weston, and Sainbayar Sukhbaatar. 2024. [Reverse training to nurse the reversal curse](#). In *First Conference on Language Modeling*.
- Sebastian Goodman, Nan Ding, and Radu Soricut. 2020. [TeaForN: Teacher-forcing with n-grams](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8704–8717, Online. Association for Computational Linguistics.
- 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](#). In *The Twelfth International Conference on Learning Representations*.
- Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. In *International Conference on Learning Representations*.
- Jiatao Gu and Xiang Kong. 2021. [Fully non-autoregressive neural machine translation: Tricks of the trade](#). In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 120–133, Online. Association for Computational Linguistics.
- Jiayan Guo, Lun Du, Hengyu Liu, Mengyu Zhou, Xinyi He, and Shi Han. 2023. [Gpt4graph: Can large language models understand graph structured data? an empirical evaluation and benchmarking](#). *arXiv preprint arXiv:2305.15066*.
- Pei Guo, WangJie You, Juntao Li, Yan Bowen, and Min Zhang. 2024a. [Exploring reversal mathematical reasoning ability for large language models](#). In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 13671–13685, Bangkok, Thailand. Association for Computational Linguistics.
- Qingyan Guo, Rui Wang, Junliang Guo, Xu Tan, Jiang Bian, and Yujiu Yang. 2024b. [Mitigating reversal curse in large language models via semantic-aware permutation training](#). In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 11453–11464, Bangkok, Thailand. Association for Computational Linguistics.

- Michael Hahn, Dan Jurafsky, and Richard Futrell. 2021. Sensitivity as a complexity measure for sequence classification tasks. *Transactions of the Association for Computational Linguistics*, 9:891–908.
- Michael Hahn and Mark Rofin. 2024. [Why are sensitive functions hard for transformers?](#) In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14973–15008, Bangkok, Thailand. Association for Computational Linguistics.
- Haoyu Han, Yaochen Xie, Hui Liu, Xianfeng Tang, Sreyashi Nag, William Headden, Yang Li, Chen Luo, Shuiwang Ji, Qi He, et al. 2025. Reasoning with graphs: Structuring implicit knowledge to enhance llms reasoning. *arXiv preprint arXiv:2501.07845*.
- Simon Jerome Han, Keith James Ransom, and Andrew Perfors. 2022. Human-like property induction is a challenge for large language models. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, 44 (44).
- Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. 2024. Parameter-efficient fine-tuning for large models: A comprehensive survey. *arXiv preprint arXiv:2403.14608*.
- Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Wang, and Zhiting Hu. 2023. [Reasoning with language model is planning with world model](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8154–8173, Singapore. Association for Computational Linguistics.
- Yiding Hao, Dana Angluin, and Robert Frank. 2022. [Formal language recognition by hard attention transformers: Perspectives from circuit complexity](#). *Transactions of the Association for Computational Linguistics*, 10:800–810.
- Adi Haviv, Ori Ram, Ofir Press, Peter Izsak, and Omer Levy. 2022. [Transformer language models without positional encodings still learn positional information](#). In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1382–1390, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- DongNyeong Heo, Daniela Noemi Rim, and Heeyoul Choi. 2024. N-gram prediction and word difference representations for language modeling. *arXiv preprint arXiv:2409.03295*.
- David Herel and Tomas Mikolov. 2023. Thinking tokens for language modeling. *8th Conference on Artificial Intelligence and Theorem Proving*.
- John Hewitt and Christopher D. Manning. 2019. [A structural probe for finding syntax in word representations](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138, Minneapolis, Minnesota. Association for Computational Linguistics.
- Paul W Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. 1983. Stochastic blockmodels: First steps. *Social networks*, 5(2):109–137.
- Edward S. Hu, Kwangjun Ahn, Qinghua Liu, Haoran Xu, Manan Tomar, Ada Langford, Dinesh Jayaraman, Alex Lamb, and John Langford. 2025a. [Learning to achieve goals with belief state transformers](#). In *The Thirteenth International Conference on Learning Representations*.
- Michael Y Hu, Jackson Petty, Chuan Shi, William Merrill, and Tal Linzen. 2025b. Between circuits and chomsky: Pre-pretraining on formal languages imparts linguistic biases. *arXiv preprint arXiv:2502.19249*.
- Jie Huang and Kevin Chen-Chuan Chang. 2023. [Towards reasoning in large language models: A survey](#). In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1049–1065, Toronto, Canada. Association for Computational Linguistics.
- Jin Huang, Xingjian Zhang, Qiaozhu Mei, and Jiaqi Ma. 2024a. [Can LLMs effectively leverage graph structural information through prompts, and why?](#) *Transactions on Machine Learning Research*.
- Sukai Huang, Trevor Cohn, and Nir Lipovetzky. 2024b. Chasing progress, not perfection: Revisiting strategies for end-to-end llm plan generation. *arXiv preprint arXiv:2412.10675*.
- Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. 2022. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International conference on machine learning*, pages 9118–9147. PMLR.
- Xinting Huang, Andy Yang, Satwik Bhattamishra, Yash Sarrof, Andreas Krebs, Hattie Zhou, Preetum Nakkiran, and Michael Hahn. 2025. [A formal framework for understanding length generalization in transformers](#). In *The Thirteenth International Conference on Learning Representations*.
- Md Shamim Hussain, Mohammed J Zaki, and Dharmashankar Subramanian. 2024. [Triplet interaction improves graph transformers: Accurate molecular graph learning with triplet graph transformers](#). In *Forty-first International Conference on Machine Learning*.
- Kazuki Irie. 2024. Why are positional encodings nonessential for deep autoregressive transformers? revisiting a petroglyph. *arXiv preprint arXiv:2501.00659*.
- Kazuki Irie, Albert Zeyer, Ralf Schlüter, and Hermann Ney. 2019. [Language modeling with deep transformers](#). In *Interspeech 2019*, pages 3905–3909.

- Samy Jelassi, David Brandfonbrener, Sham M. Kakade, and Eran Malach. 2024. [Repeat after me: Transformers are better than state space models at copying](#). In *Forty-first International Conference on Machine Learning*.
- Bowen Jiang, Yangxinyu Xie, Zhuoqun Hao, Xiaomeng Wang, Tanwi Mallick, Weijie J Su, Camillo Jose Taylor, and Dan Roth. 2024a. [A peek into token bias: Large language models are not yet genuine reasoners](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 4722–4756, Miami, Florida, USA. Association for Computational Linguistics.
- Juyong Jiang, Fan Wang, Jiasi Shen, Sungju Kim, and Sunghun Kim. 2024b. [A survey on large language models for code generation](#). *arXiv preprint arXiv:2406.00515*.
- Bowen Jin, Gang Liu, Chi Han, Meng Jiang, Heng Ji, and Jiawei Han. 2024. [Large language models on graphs: A comprehensive survey](#). *IEEE Transactions on Knowledge and Data Engineering*.
- Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. [SpanBERT: Improving pre-training by representing and predicting spans](#). *Transactions of the Association for Computational Linguistics*, 8:64–77.
- Ehsan Kamaloo, Nouha Dziri, Charles Clarke, and Davood Rafiei. 2023. [Evaluating open-domain question answering in the era of large language models](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5591–5606, Toronto, Canada. Association for Computational Linguistics.
- Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Mudit Verma, Kaya Stechly, Siddhant Bhambri, Lucas Paul Saldyt, and Anil B Murthy. 2024. [Position: LLMs can’t plan, but can help planning in LLM-modulo frameworks](#). In *Forty-first International Conference on Machine Learning*.
- Liwei Kang, Zirui Zhao, David Hsu, and Wee Sun Lee. 2024. [On the empirical complexity of reasoning and planning in LLMs](#). In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 2897–2936, Miami, Florida, USA. Association for Computational Linguistics.
- Jungo Kasai, Keisuke Sakaguchi, Ronan Le Bras, Dragomir Radev, Yejin Choi, and Noah A. Smith. 2024. [A call for clarity in beam search: How it works and when it stops](#). In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 77–90, Torino, Italia. ELRA and ICCL.
- Amirhossein Kazemnejad, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Payel Das, and Siva Reddy. 2023. [The impact of positional encoding on length generalization in transformers](#). In *Advances in Neural Information Processing Systems*, volume 36, pages 24892–24928. Curran Associates, Inc.
- Mikail Khona, Maya Okawa, Rahul Ramesh, Kento Nishi, Robert P. Dick, Ekdeep Singh Lubana, and Hidenori Tanaka. 2024. [Toward a mechanistic understanding of stepwise inference in transformers: A synthetic graph navigation model](#).
- Juno Kim and Taiji Suzuki. 2025. [Transformers provably solve parity efficiently with chain of thought](#). In *The Thirteenth International Conference on Learning Representations*.
- Ouail Kitouni, Niklas Nolte, Adina Williams, Michael Rabbat, Diane Bouchacourt, and Mark Ibrahim. 2024. [The factorization curse: Which tokens you predict underlie the reversal curse and more](#). In *The Thirtieth Annual Conference on Neural Information Processing Systems*.
- Shun Kiyono, Sosuke Kobayashi, Jun Suzuki, and Kentaro Inui. 2021. [SHAPE: Shifted absolute position embedding for transformers](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3309–3321, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. [Large language models are zero-shot reasoners](#). *Advances in neural information processing systems*, 35:22199–22213.
- Jason Lee, Elman Mansimov, and Kyunghyun Cho. 2018. [Deterministic non-autoregressive interpreting gpt: the logit lens](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1173–1182, Brussels, Belgium. Association for Computational Linguistics.
- Jooyoung Lee, Fan Yang, Thanh Tran, Qian Hu, Emre Barut, and Kai-Wei Chang. 2024. [Can small language models help large language models reason better?: LM-guided chain-of-thought](#). In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 2835–2843, Torino, Italia. ELRA and ICCL.
- Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. [Fast inference from transformers via speculative decoding](#). In *International Conference on Machine Learning*, pages 19274–19286. PMLR.
- Shanda Li, Chong You, Guru Guruganesh, Joshua Ainslie, Santiago Ontanon, Manzil Zaheer, Sumit Sanghai, Yiming Yang, Sanjiv Kumar, and Srinadh Bhojanapalli. 2024a. [Functional interpolation for relative positions improves long context transformers](#). In *The Twelfth International Conference on Learning Representations*.



- Yuhan Li, Peisong Wang, Xiao Zhu, Aochuan Chen, Haiyun Jiang, Deng Cai, Victor Wai Kin Chan, and Jia Li. 2024b. [GLBench: A comprehensive benchmark for graph with large language models](#). In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.
- Zhiyuan Li, Hong Liu, Denny Zhou, and Tengyu Ma. 2024c. [Chain of thought empowers transformers to solve inherently serial problems](#). In *The Twelfth International Conference on Learning Representations*.
- Yi Liao, Xin Jiang, and Qun Liu. 2020. [Probabilistically masked language model capable of autoregressive generation in arbitrary word order](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 263–274, Online. Association for Computational Linguistics.
- Pengxiao Lin, Zhongwang Zhang, and Zhi-Qin John Xu. 2025a. Reasoning bias of next token prediction training. *arXiv preprint arXiv:2502.02007*.
- Tianhe Lin, Jian Xie, Siyu Yuan, and Deqing Yang. 2025b. [Implicit reasoning in transformers is reasoning through shortcuts](#). *arXiv preprint arXiv:2503.07604*.
- Zhengkai Lin, Zhihang Fu, Kai Liu, Liang Xie, Binbin Lin, Wenxiao Wang, Deng Cai, Yue Wu, and Jieping Ye. 2024. [Delving into the reversal curse: How far can large language models generalize?](#) In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- David Lindner, Janos Kramar, Sebastian Farquhar, Matthew Rahtz, Tom McGrath, and Vladimir Mikulik. 2023. [Tracr: Compiled transformers as a laboratory for interpretability](#). In *Advances in Neural Information Processing Systems*, volume 36, pages 37876–37899. Curran Associates, Inc.
- Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. 2024a. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.
- Bingbin Liu, Jordan T. Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. 2023. [Transformers learn shortcuts to automata](#). In *The Eleventh International Conference on Learning Representations*.
- Chang Liu and Bo Wu. 2023. Evaluating large language models on graphs: Performance insights and comparative analysis. *arXiv preprint arXiv:2308.11224*.
- Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024b. [Lost in the middle: How language models use long contexts](#). *Transactions of the Association for Computational Linguistics*, 12:157–173.
- Sheng Lu, Hendrik Schuff, and Iryna Gurevych. 2024. [How are prompts different in terms of sensitivity?](#) In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5833–5856, Mexico City, Mexico. Association for Computational Linguistics.
- Zihan Luo, Xiran Song, Hong Huang, Jianxun Lian, Chenhao Zhang, Jinqi Jiang, and Xing Xie. 2024. Graphinstruct: Empowering large language models with graph understanding and reasoning capability. *arXiv preprint arXiv:2403.04483*.
- Ang Lv, Kaiyi Zhang, Shufang Xie, Quan Tu, Yuhan Chen, Ji-Rong Wen, and Rui Yan. 2024. [An analysis and mitigation of the reversal curse](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 13603–13615, Miami, Florida, USA. Association for Computational Linguistics.
- Jun-Yu Ma, Jia-Chen Gu, Zhen-Hua Ling, Quan Liu, and Cong Liu. 2023. Untying the reversal curse via bidirectional language model editing. *arXiv preprint arXiv:2310.10322*.
- Aman Madaan, Dheeraj Rajagopal, Niket Tandon, Yiming Yang, and Eduard Hovy. 2021. [Could you give me a hint? generating inference graphs for defeasible reasoning](#). In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 5138–5147, Online. Association for Computational Linguistics.
- Aman Madaan, Shuyan Zhou, Uri Alon, Yiming Yang, and Graham Neubig. 2022. [Language models of code are few-shot commonsense learners](#). In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1384–1403, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Andrea Matarazzo and Riccardo Torlone. 2025. A survey on large language models with some insights on their capabilities and limitations. *arXiv preprint arXiv:2501.04040*.
- Sean Michael McLeish, Arpit Bansal, Alex Stein, Neel Jain, John Kirchenbauer, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, Jonas Geiping, Avi Schwarzschild, and Tom Goldstein. 2024. [Transformers can do arithmetic with the right embeddings](#). In *The 4th Workshop on Mathematical Reasoning and AI at NeurIPS'24*.
- Tianyi Men, Pengfei Cao, Zhuoran Jin, Yubo Chen, Kang Liu, and Jun Zhao. 2024. [Unlocking the future: Exploring look-ahead planning mechanistic interpretability in large language models](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 7713–7724, Miami, Florida, USA. Association for Computational Linguistics.
- William Merrill and Ashish Sabharwal. 2023. [A logic for expressing log-precision transformers](#). In *Thirty-seventh Conference on Neural Information Processing Systems*.

- William Merrill and Ashish Sabharwal. 2024. [The expressive power of transformers with chain of thought](#). In *The Twelfth International Conference on Learning Representations*.
- Tomas Mikolov. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 3781.
- Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. 2024. Large language models: A survey. *arXiv preprint arXiv:2402.06196*.
- Seyed Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. 2025. [GSM-symbolic: Understanding the limitations of mathematical reasoning in large language models](#). In *The Thirteenth International Conference on Learning Representations*.
- Giovanni Monea, Armand Joulin, and Edouard Grave. 2023. Pass: Parallel speculative sampling. *arXiv preprint arXiv:2311.13581*.
- nostalgebraist. 2020. interpreting gpt: the logit lens. <https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens>. Accessed: 2024-12-18.
- 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. 2022. [Show your work: Scratchpads for intermediate computation with language models](#). In *Deep Learning for Code Workshop*.
- OpenAI. 2024. Openai o1 system card. <https://openai.com/index/openai-o1-system-card/>. Accessed: 2024-12-18.
- Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. *arXiv preprint arXiv:1904.01038*.
- Koyena Pal, Jiuding Sun, Andrew Yuan, Byron Wallace, and David Bau. 2023. [Future lens: Anticipating subsequent tokens from a single hidden state](#). In *Proceedings of the 27th Conference on Computational Natural Language Learning (CoNLL)*, pages 548–560, Singapore. Association for Computational Linguistics.
- Vassilis Papadopoulos, Jérémie Wenger, and Clément Hongler. 2024. [Arrows of time for large language models](#). In *Forty-first International Conference on Machine Learning*.
- Bryan Perozzi, Bahare Fatemi, Dustin Zelle, Anton Tsitsulin, Mehran Kazemi, Rami Al-Rfou, and Jonathan Halcrow. 2024. Let your graph do the talking: Encoding structured data for llms. *arXiv preprint arXiv:2402.05862*.
- Jacob Pfau, William Merrill, and Samuel R. Bowman. 2024. [Let’s think dot by dot: Hidden computation in transformer language models](#). In *First Conference on Language Modeling*.
- Aske Plaat, Annie Wong, Suzan Verberne, Joost Broekens, Niki van Stein, and Thomas Back. 2024. Reasoning with large language models, a survey. *arXiv preprint arXiv:2407.11511*.
- Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. 2023. [Measuring and narrowing the compositionality gap in language models](#). In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711, Singapore. Association for Computational Linguistics.
- Weizhen Qi, Yu Yan, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. 2020. [ProphetNet: Predicting future n-gram for sequence-to-Sequence Pre-training](#). In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2401–2410, Online. Association for Computational Linguistics.
- Junlang Qian, Zixiao Zhu, Hanzhang Zhou, Zijian Feng, Zepeng Zhai, and Kezhi Mao. 2025. [Beyond the next token: Towards prompt-robust zero-shot classification via efficient multi-token prediction](#). In *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)*, pages 7093–7115, Albuquerque, New Mexico. Association for Computational Linguistics.
- Shuofei Qiao, Yixin Ou, Ningyu Zhang, Xiang Chen, Yunzhi Yao, Shumin Deng, Chuanqi Tan, Fei Huang, and Huajun Chen. 2023. [Reasoning with language model prompting: A survey](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5368–5393, Toronto, Canada. Association for Computational Linguistics.
- Markus Norman Rabe, Dennis Lee, Kshitij Bansal, and Christian Szegedy. 2021. [Mathematical reasoning via self-supervised skip-tree training](#). In *International Conference on Learning Representations*.
- Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. 2021. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*.
- Xubin Ren, Jiabin Tang, Dawei Yin, Nitesh Chawla, and Chao Huang. 2024. A survey of large language models for graphs. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 6616–6626.
- Laura Eline Ruis, Akbir Khan, Stella Biderman, Sara Hooker, Tim Rocktäschel, and Edward Grefenstette. 2023. [Large language models are not zero-shot communicators](#).

- Anian Ruoss, Grégoire Delétang, Tim Genewein, Jordi Grau-Moya, Róbert Csordás, Mehdi Bennani, Shane Legg, and Joel Veness. 2023. [Randomized positional encodings boost length generalization of transformers](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1889–1903, Toronto, Canada. Association for Computational Linguistics.
- Swarnadeep Saha, Prateek Yadav, Lisa Bauer, and Mohit Bansal. 2021. [ExplaGraphs: An explanation graph generation task for structured commonsense reasoning](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7716–7740, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Keisuke Sakaguchi, Chandra Bhagavatula, Ronan Le Bras, Niket Tandon, Peter Clark, and Yejin Choi. 2021. [proScript: Partially ordered scripts generation](#). In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2138–2149, Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Clayton Sanford, Bahare Fatemi, Ethan Hall, Anton Tsitsulin, Mehran Kazemi, Jonathan Halcrow, Bryan Perozzi, and Vahab Mirrokni. 2024a. [Understanding transformer reasoning capabilities via graph algorithms](#). In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- Clayton Sanford, Daniel Hsu, and Matus Telgarsky. 2024b. [Transformers, parallel computation, and logarithmic depth](#). In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 43276–43327. PMLR.
- Clayton Sanford, Daniel J Hsu, and Matus Telgarsky. 2023. [Representational strengths and limitations of transformers](#). In *Advances in Neural Information Processing Systems*, volume 36, pages 36677–36707. Curran Associates, Inc.
- Abulhair Saparov, Srushti Ajay Pawar, Shreyas Pimpalgaonkar, Nitish Joshi, Richard Yuanzhe Pang, Vishakh Padmakumar, Mehran Kazemi, Najoung Kim, and He He. 2025. [Transformers struggle to learn to search](#). In *The Thirteenth International Conference on Learning Representations*.
- Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. [The graph neural network model](#). *IEEE Transactions on Neural Networks*, 20(1):61–80.
- Rylan Schaeffer, Brando Miranda, and Sanmi Koyejo. 2023. [Are emergent abilities of large language models a mirage?](#) In *Thirty-seventh Conference on Neural Information Processing Systems*.
- Sander Schulhoff, Michael Ilie, Nishant Balepur, Konstantine Kahadze, Amanda Liu, Chenglei Si, Yin-heng Li, Aayush Gupta, HyoJung Han, Sevien Schulhoff, et al. 2024. [The prompt report: A systematic survey of prompting techniques](#). *arXiv preprint arXiv:2406.06608*.
- Kulin Shah, Nishanth Dikkala, Xin Wang, and Rina Panigrahy. 2024. [Causal language modeling can elicit search and reasoning capabilities on logic puzzles](#). In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H. Chi, Nathanael Schärli, and Denny Zhou. 2023. [Large language models can be easily distracted by irrelevant context](#). In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 31210–31227. PMLR.
- Rui Song, Yingji Li, Fausto Giunchiglia, and Hao Xu. 2024. [Shortcut learning in in-context learning: A survey](#). *arXiv preprint arXiv:2411.02018*.
- Zayne Rea Sprague, Fangcong Yin, Juan Diego Rodriguez, Dongwei Jiang, Manya Wadhwa, Prasann Singhal, Xinyu Zhao, Xi Ye, Kyle Mahowald, and Greg Durrett. 2025. [To cot or not to cot? chain-of-thought helps mainly on math and symbolic reasoning](#). In *The Thirteenth International Conference on Learning Representations*.
- Joe Stacey, Pasquale Minervini, Haim Dubossarsky, and Marek Rei. 2022. [Logical reasoning with span-level predictions for interpretable and robust NLI models](#). In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3809–3823, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Kaya Stechly, Karthik Valmeekam, and Subbarao Kambhampati. 2025. [On the self-verification limitations of large language models on reasoning and planning tasks](#). In *The Thirteenth International Conference on Learning Representations*.
- David Steinmann, Felix Divo, Maurice Kraus, Antonia Wüst, Lukas Struppek, Felix Friedrich, and Kristian Kersting. 2024. [Navigating shortcuts, spurious correlations, and confounders: From origins via detection to mitigation](#). *arXiv preprint arXiv:2412.05152*.
- Lena Strobl, Dana Angluin, David Chiang, Jonathan Rawski, and Ashish Sabharwal. 2024a. [Transformers as transducers](#). *arXiv preprint arXiv:2404.02040*.
- Lena Strobl, William Merrill, Gail Weiss, David Chiang, and Dana Angluin. 2024b. [What formal languages can transformers express? a survey](#). *Transactions of the Association for Computational Linguistics*, 12:543–561.
- Anej Svete, Nadav Borenstein, Mike Zhou, Isabelle Augenstein, and Ryan Cotterell. 2024. [Can transformers learn  \$n\$ -gram language models?](#) In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 9851–9867, Miami, Florida, USA. Association for Computational Linguistics.

- Anej Svete and Ryan Cotterell. 2024. [Transformers can represent  \$n\$ -gram language models](#). In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6845–6881, Mexico City, Mexico. Association for Computational Linguistics.
- Niket Tandon, Bhavana Dalvi, Keisuke Sakaguchi, Peter Clark, and Antoine Bosselut. 2019. [WIQA: A dataset for “what if...” reasoning over procedural text](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6076–6085, Hong Kong, China. Association for Computational Linguistics.
- Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Lixin Su, Suqi Cheng, Dawei Yin, and Chao Huang. 2024. [Graphgpt: Graph instruction tuning for large language models](#). In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR ’24*, page 491–500, New York, NY, USA. Association for Computing Machinery.
- Jianheng Tang, Qifan Zhang, Yuhan Li, Nuo Chen, and Jia Li. 2025. [Grapharena: Evaluating and exploring large language models on graph computation](#). In *The Thirteenth International Conference on Learning Representations*.
- Yao-Hung Hubert Tsai, Shaojie Bai, Makoto Yamada, Louis-Philippe Morency, and Ruslan Salakhutdinov. 2019. [Transformer dissection: An unified understanding for transformer’s attention via the lens of kernel](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4344–4353, Hong Kong, China. Association for Computational Linguistics.
- Lifu Tu, Garima Lalwani, Spandana Gella, and He He. 2020. [An empirical study on robustness to spurious correlations using pre-trained language models](#). *Transactions of the Association for Computational Linguistics*, 8:621–633.
- Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. 2023a. [Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change](#). In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.
- Karthik Valmeekam, Matthew Marquez, Sarath Sreedharan, and Subbarao Kambhampati. 2023b. [On the planning abilities of large language models - a critical investigation](#). In *Thirty-seventh Conference on Neural Information Processing Systems*.
- Karthik Valmeekam, Kaya Stechly, and Subbarao Kambhampati. 2024. [Llms still can’t plan; can lrms? a preliminary evaluation of openai’s o1 on planbench](#). *arXiv preprint arXiv:2409.13373*.
- Bhavya Vasudeva, Deqing Fu, Tianyi Zhou, Elliott Kau, Youqi Huang, and Vatsal Sharan. 2025. [Transformers learn low sensitivity functions: Investigations and implications](#). In *The Thirteenth International Conference on Learning Representations*.
- Boshi Wang, Sewon Min, Xiang Deng, Jiaming Shen, You Wu, Luke Zettlemoyer, and Huan Sun. 2023a. [Towards understanding chain-of-thought prompting: An empirical study of what matters](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2717–2739, Toronto, Canada. Association for Computational Linguistics.
- Heng Wang, Shangbin Feng, Tianxing He, Zhaoxuan Tan, Xiaochuang Han, and Yulia Tsvetkov. 2023b. [Can language models solve graph problems in natural language?](#) In *Thirty-seventh Conference on Neural Information Processing Systems*.
- Jianing Wang, Junda Wu, Yupeng Hou, Yao Liu, Ming Gao, and Julian McAuley. 2024a. [InstructGraph: Boosting large language models via graph-centric instruction tuning and preference alignment](#). In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 13492–13510, Bangkok, Thailand. Association for Computational Linguistics.
- Jie Wang, Tao Ji, Yuanbin Wu, Hang Yan, Tao Gui, Qi Zhang, Xuanjing Huang, and Xiaoling Wang. 2024b. [Length generalization of causal transformers without position encoding](#). In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 14024–14040, Bangkok, Thailand. Association for Computational Linguistics.
- Xuezhi Wang and Denny Zhou. 2024. [Chain-of-thought reasoning without prompting](#). In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. [Chain-of-thought prompting elicits reasoning in large language models](#). *Advances in neural information processing systems*, 35:24824–24837.
- Gail Weiss, Yoav Goldberg, and Eran Yahav. 2021. [Thinking like transformers](#). In *International Conference on Machine Learning*, pages 11080–11090. PMLR.
- Noam Wies, Yoav Levine, and Amnon Shashua. 2023. [Sub-task decomposition enables learning in sequence to sequence tasks](#). In *The Eleventh International Conference on Learning Representations*.
- Qiming Wu, Zichen Chen, Will Corcoran, Misha Sra, and Ambuj K. Singh. 2024a. [Grapheval2000: Benchmarking and improving large language models on graph datasets](#). *CoRR*, abs/2406.16176.

- Wilson Wu, John Xavier Morris, and Lionel Levine. 2024b. [Do language models plan ahead for future tokens?](#) In *First Conference on Language Modeling*.
- Xixi Wu, Yifei Shen, Caihua Shan, Kaitao Song, Siwei Wang, Bohang Zhang, Jiarui Feng, Hong Cheng, Wei Chen, Yun Xiong, et al. 2024c. Can graph learning improve planning in llm-based agents? In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- Heming Xia, Tao Ge, Si-Qing Chen, Furu Wei, and Zhifang Sui. 2022. [Speculative decoding: Lossless speedup of autoregressive translation](#).
- Changnan Xiao and Bing Liu. 2024. A theory for length generalization in learning to reason. *arXiv preprint arXiv:2404.00560*.
- Changnan Xiao and Bing Liu. 2025. [Generalizing reasoning problems to longer lengths](#). In *The Thirteenth International Conference on Learning Representations*.
- Yudong Xu, Elias B. Khalil, and Scott Sanner. 2023. [Graphs, constraints, and search for the abstraction and reasoning corpus](#). *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(4):4115–4122.
- Andy Yang, Lena Strobl, David Chiang, and Dana Angluin. 2024. Simulating hard attention using soft attention. *arXiv preprint arXiv:2412.09925*.
- Bowen Yang, Bharat Venkitesh, Dwarak Talupuru, Hangyu Lin, David Cairuz, Phil Blunsom, and Acyr Locatelli. 2025. Rope to nope and back again: A new hybrid attention strategy. *arXiv preprint arXiv:2501.18795*.
- Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.
- Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik R Narasimhan. 2023. [Tree of thoughts: Deliberate problem solving with large language models](#). In *Thirty-seventh Conference on Neural Information Processing Systems*.
- Jiacheng Ye, Jiahui Gao, Shansan Gong, Lin Zheng, Xin Jiang, Zhenguo Li, and Lingpeng Kong. 2025a. [Beyond autoregression: Discrete diffusion for complex reasoning and planning](#). In *The Thirteenth International Conference on Learning Representations*.
- Jiacheng Ye, Zhenyu Wu, Jiahui Gao, Zhiyong Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. 2025b. [Implicit search via discrete diffusion: A study on chess](#). In *The Thirteenth International Conference on Learning Representations*.
- Yongjing Yin, Junran Ding, Kai Song, and Yue Zhang. 2024. [Semformer: Transformer language models with semantic planning](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 18669–18680, Miami, Florida, USA. Association for Computational Linguistics.
- Alexander Yom Din, Taelin Karidi, Leshem Choshen, and Mor Geva. 2024. [Jump to conclusions: Short-cutting transformers with linear transformations](#). In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 9615–9625, Torino, Italia. ELRA and ICCL.
- Zike Yuan, Ming Liu, Hui Wang, and Bing Qin. 2024. Gracore: Benchmarking graph comprehension and complex reasoning in large language models. *arXiv preprint arXiv:2407.02936*.
- Chulhee Yun, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank Reddi, and Sanjiv Kumar. 2020. [Are transformers universal approximators of sequence-to-sequence functions?](#) In *International Conference on Learning Representations*.
- Daoguang Zan, Bei Chen, Fengji Zhang, Dianjie Lu, Bingchao Wu, Bei Guan, Wang Yongji, and Jian-Guang Lou. 2023. [Large language models meet NL2Code: A survey](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7443–7464, Toronto, Canada. Association for Computational Linguistics.
- Li Zhang, Hainiu Xu, Yue Yang, Shuyan Zhou, Weiqiu You, Manni Arora, and Chris Callison-Burch. 2023a. [Causal reasoning of entities and events in procedural texts](#). In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 415–431, Dubrovnik, Croatia. Association for Computational Linguistics.
- Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, et al. 2023b. Instruction tuning for large language models: A survey. *arXiv preprint arXiv:2308.10792*.
- Yi Zhang, Arturs Backurs, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, and Tal Wagner. 2022. Unveiling transformers with lego: a synthetic reasoning task. *arXiv preprint arXiv:2206.04301*.
- Yizhuo Zhang, Heng Wang, Shangbin Feng, Zhaoxuan Tan, Xiaochuang Han, Tianxing He, and Yulia Tsvetkov. 2024. [Can LLM graph reasoning generalize beyond pattern memorization?](#) In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 2289–2305, Miami, Florida, USA. Association for Computational Linguistics.
- Yu Zhao, Huifeng Yin, Bo Zeng, Hao Wang, Tianqi Shi, Chenyang Lyu, Longyue Wang, Weihua Luo, and Kaifu Zhang. 2024. Marco-o1: Towards open

reasoning models for open-ended solutions. *arXiv preprint arXiv:2411.14405*.

Zirui Zhao, Wee Sun Lee, and David Hsu. 2023. [Large language models as commonsense knowledge for large-scale task planning](#). In *Thirty-seventh Conference on Neural Information Processing Systems*.

Hattie Zhou, Arwen Bradley, Etai Littwin, Noam Razin, Omid Saremi, Joshua M. Susskind, Samy Bengio, and Preetum Nakkiran. 2024a. [What algorithms can transformers learn? a study in length generalization](#). In *The Twelfth International Conference on Learning Representations*.

Yongchao Zhou, Uri Alon, Xinyun Chen, Xuezhi Wang, Rishabh Agarwal, and Denny Zhou. 2024b. [Transformers can achieve length generalization but not robustly](#). In *ICLR 2024 Workshop on Mathematical and Empirical Understanding of Foundation Models*.

Yuqing Zhou, Ruixiang Tang, Ziyu Yao, and Ziwei Zhu. 2024c. [Navigating the shortcut maze: A comprehensive analysis of shortcut learning in text classification by language models](#). In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 2586–2614, Miami, Florida, USA. Association for Computational Linguistics.

Kaijie Zhu, Jiaao Chen, Jindong Wang, Neil Zhenqiang Gong, Diyi Yang, and Xing Xie. 2024. [Dyval: Dynamic evaluation of large language models for reasoning tasks](#). In *The Twelfth International Conference on Learning Representations*.

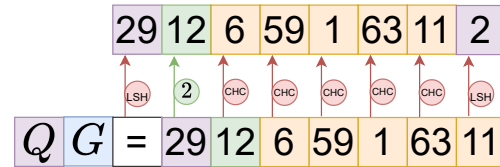
Chunsheng Zuo, Pavel Guerzhoy, and Michael Guerzhoy. 2025. [Position information emerges in causal transformers without positional encodings via similarity of nearby embeddings](#). In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 9418–9430, Abu Dhabi, UAE. Association for Computational Linguistics.

## A Relationships Between Methods

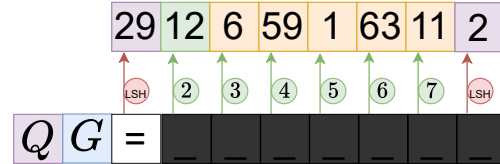
This section provides a summary of the methods introduced and the various relations between the methods, focusing on how each induces subtask decomposition. We also outline how each method either prevents, alleviates, or just ignores the CHC. Note that, except for the alternative sequential distributions, all methods use standard teacher-forcing and next-token prediction.

### A.1 The Different Forms of Masking

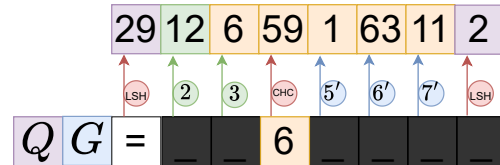
A comparison between fully observed, fully masked, partially masked, and alternative sequential distributions is given in Fig. 12. These are all target-side modifications to the task. Masking modifies the target-side input, while the alternative sequential distributions modify the target-side targets. However, this can also be viewed as inducing



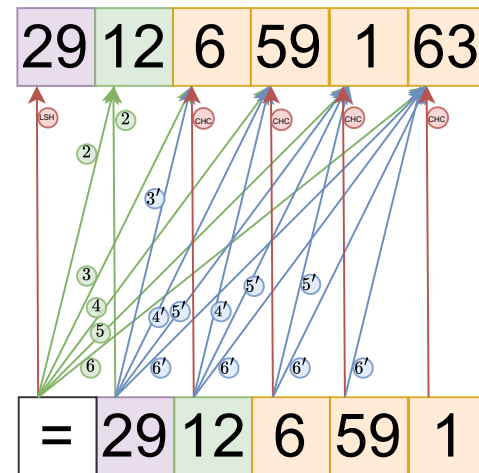
(a) The original task with **fully observed** teacher-forced input induces learning the CHC for all but the leading node. Note, we have extended the length of the arm from Fig. 1.



(b) An example of a **fully masked** sequence (equivalent to the teacher-less model or NAR in terms of masking). All five predictions are conditioned via different masked tokens (at unique positions). Each prediction must use the backward algorithm to determine the correct target.



(c) A **partially masked** sequence. Unmasked tokens cause the CHC to be learnt for the corresponding prediction. Predictions after an unmasked token *may* use the forward algorithm to determine the correct target (indicated with primes).



(d) **Alternative sequential distributions** create dense decomposition supervision. The CHC is learnt for one prediction at all time-steps, except for the leading node (same as in the original task with no masking). In the first time-step, all predictions corresponding to the fully masked must be learnt; the difference between this and Fig. 12b is that all predictions are conditioned from a single hidden-state instead of being separated into five temporally distinct hidden-states created via different masked token inputs. Then, for all other future predictions, the forward algorithm *may* be used to determine the correct targets since the model conditions on ground-truth tokens on those time-steps. Also, some of these predictions may be repeated at different time-steps. Nodes ‘11’ and ‘2’ are omitted for visibility.

Figure 12: A comparison of subtask decomposition between masking and alternative sequential distributions.

the same masking modification on the target-side inputs, except that this is done implicitly instead of explicitly via masked tokens.

We illustrate teacher-forced inputs and corresponding targets while also labeling how the model achieves each prediction in Fig. 12. Predictions learnt by the CHC are indicated with ‘CHC’ red labels. Because the path-star task uses graphs of the same size, the start and target nodes can be trivially predicted via *length shortcuts* as they are both included in  $Q$  (see Fig. 16). These are indicated with ‘LSH’ red labels. Predictions for the second through seventh target tokens that use the desired backward algorithm are indicated with green labels ‘2’ through ‘7’. Those that may use the forward algorithm are indicated with primes and blue labels. Consider again Fig. 5 in Sec. 3.1.2 for a more detailed illustration of how subtask decomposition is achieved via masking.

We illustrate the original adulterated task in Fig. 12a, where the CHC is learnt for all predictions except for the leading node. The CHC was the mechanism originally attributed to causing the task to be unlearnable by Bachmann and Nagarajan (2024). This motivated the use of a ‘teacher-less’ model, which fully masks the target-side inputs.<sup>12</sup> We illustrate the effect of full masking in Fig. 12b. This prevents the CHC and induces learning of the desired backward algorithm for all predictions.<sup>13</sup> This is in contrast to partial or sampled masking, illustrated in Fig. 12c, which still allows the CHC to be learnt, hence only alleviating the issue.<sup>14,15</sup>

<sup>12</sup>This has a technical difference with our masking method. Our masking keeps true autoregressive conditioning at all steps while ‘teacher-less’ models only condition on a single mask token per time-step and, hence, can predict all masked positions in parallel (like a non-autoregressive model). This difference only matters in that our masking method is a standard method to use with language models, while the teacher-less method is not, and this then helps to support our argument that standard methods are sufficient to solve the task and that alternative models are not needed.

<sup>13</sup>Thus, the only methods that completely prevent learning the CHC are full masking, the teacher-less model, and ‘fully’ non-autoregressive models (Frydenlund, 2024).

<sup>14</sup>This also means that the iterative autoregressive models and the discrete diffusion models only alleviating the issue (see Appx. C.10).

<sup>15</sup>Bachmann and Nagarajan (2024) introduced the teacher-less model with masking as a direct anti-CHC mechanism on the assumption that it was the mechanism causing the task to be unlearnable and that its prevention is necessary to make the task learnable. However, we show that masking also introduces subtask decomposition. **This raises the question: what is the critical aspect of this method that allows for the task to be learnt? We argue that decomposition is the critical aspect, not CHC-prevention.** This is held up by the fact that, as mentioned above, partial masking doesn’t actually fully pre-

Whereas full masking induces learning of the backward algorithm for predictions, partial masking means that predictions after any unmasked ground-truth inputs may use the forward algorithm. **The differences in algorithms may be important since, while they are very similar, the backward algorithm is required to be learnt to correctly predict the leading node, and as the forward algorithm is different, this means it is not an exact subtask decomposition. Assuming that learning the forward algorithm supports learning the backward algorithm, this suggests that subtask decomposition does not need to be exact mirror of the original or core task.** We also discuss this idea in relation to how tree topology induces decomposition in Sec. 3.4. Given the success of these methods, it seems like subtask decomposition does not need to be exact. This makes a formal definition of decomposition difficult (whereas in the exact case, we could just define a task recursively and treat each recursive step as a decomposition).

## A.2 Masking and Alternative Sequential Distributions

Comparing masking to alternative sequential distributions described in Sec. 3.2, **alternative sequential distributions act like a combination of all three of the described masking types – fully observed (no mask), fully masked, and partially masked – all at once.** This is achieved due to having multiple targets per time-step (i.e., being multi-token prediction methods), which leads to a greater density of target supervision (and hence predictions). Or to put another way, more targets are given, and a subset of these targets induce predictions like the fully observed task, a subset induces predictions like the fully masked task, and the remaining subset induces predictions like the partially masked task.

This is illustrated in Fig. 12d. Here, teacher-forcing with fully observed input is used. This means that the same predictions in Fig. 12a are also learnt, thus, the CHC is still learnt. However, by predicting all future tokens from the initial generic task token ‘=’ on the first time-step, no actual instance-specific supervision is being provided to these predictions via teacher-forcing, i.e., there

vent the CHC and still learns the task, and that other methods like alternative sequence distributions, general queries, and general lengths, a), learn the task, b), do not prevent the CHC at all, but, c), do induce decomposition. Hence, decomposition is necessary to learn the task, but preventing the CHC is not, and so, this must be the critical aspect of why masking works.

is no conditioning on the correct arm. Thus, these predictions behave exactly like the fully masked predictions in Fig. 12b. Then, partial ground-truths are observed for future predictions for all other time-steps, and these predictions behave exactly like the partially masked prediction in Fig. 12c. Note that these are repeated multiple times.

Because alternative sequential distributions act like a combination of masking methods, they induce multihop arm reconstruction and subtask decomposition in the same way as masking. The only difference here is that multiple predictions are conditioned from a single hidden-state instead of being conditioned on five temporally distinct hidden-states (each of which is made unique by using a new masked token and positional embeddings), however, this technical difference does not affect the induced decomposition. This is also why one can view this as implicitly modifying the target-side input with masked tokens (and the galaxy-brain view is that we have converted target-side input supervision to target-side target supervision.).

See Gerontopoulos et al. (2025) for an example of a method which helps bridge the masking method that we use with the alternative sequential distributions to help demonstrate this idea. They introduced extra tokens as input into the model to create new hidden-states from which they can make future predictions. These extra tokens can be compared to masked tokens, except they do not replace exiting tokens. Then this method can be compared to our alternative sequential distributions, where the former creates new hidden-states via modifying the input to the model, whereas the latter reuses hidden-states to make multi-token predictions. This then shows how one can convert input supervision into target supervision using a structured loss.<sup>16</sup>

Finally, alternative sequential distributions also create a unique or novel form of subtask decomposition since the same target is reused at different time-steps. For example, it requires a subset of the steps needed to predict a target at step 6 from step 3, compared to predicting the target at step 6 from step 2 (one less step to be precise). Consider the four (teacher-forced) predictions for target ‘63’ indicated with the label 6’ in Fig. 12d. To predict ‘63’ from ‘59’ requires one hop through ‘1’. To predict ‘63’ from ‘6’ requires two hops; one through ‘59’ and then the same one through ‘1’. To predict ‘63’

<sup>16</sup>See Appx. C.11 for a discussion of their work – we highly recommend it to readers who have made it this.

from ‘12’ requires three hops; one through ‘6’ and then the same two through ‘1’ and ‘59’ etc.

**Because teacher-forcing is still being used for the alternative sequential distributions, the only non-standard aspect of these methods is that they are not strictly using the next-token prediction paradigm but rather a generalization of it – next-tokens prediction, if you will.**<sup>17</sup> This distinction is important for this work since the core research question of this work is whether next-token prediction is sufficient to learn the task. Thus, these methods are not directly relevant to that research question. However, they do provide another example of how subtask decomposition makes the task solvable, and this ties back to the core research question, as we show that next-token prediction is sufficient to learn the task, so long as some kind of subtask decomposition is given as supervision and not adulterated.<sup>18</sup>

### A.3 BoW Label Smoothing and Scratchpad

As mentioned in Sec. 3.3, the BoW scratchpad unifies the alternative sequential BoW distribution with a single next-token distribution since the next  $M$  scratchpad tokens are the BoW. **That is, we can treat the multi-target BoW prediction task as a next-token sequential task via intermediate scratchpad predictions.** This is done by sampling a permutation of the target arm sequence and treating that as the ground-truth sequence for both the inputs and the targets of the scratchpad.

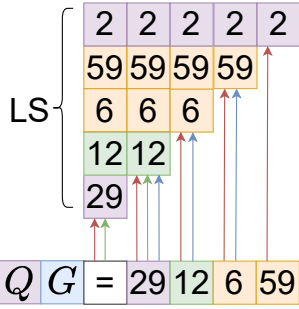
We apply label smoothing to the scratchpad targets. In theory, this does not need to be done, provided enough BoW orders are sampled. However, given the issue concerning the large sample space of graphs, this would be a bad strategy in practice (especially since the sample space is made even larger when considering sampling a scratchpad in addition to the graph and target arm). By sampling an input order for the scratchpad, we have incorporated order permutation into learning the task.<sup>19</sup> **Even though the scratchpad has multi-**

<sup>17</sup>This is more commonly called multi-token prediction, but that also includes NAR/IAR/discrete diffusion models that can predict anywhere in the sequence.

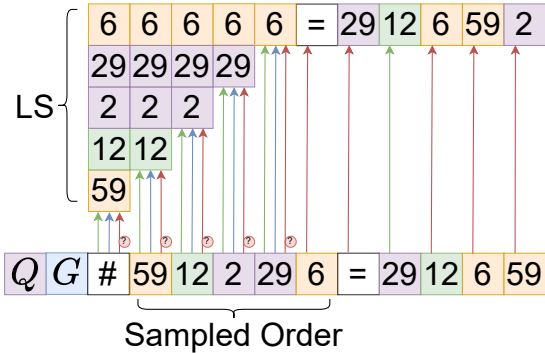
<sup>18</sup>We also included these methods in this work to explain the success of other multi-token works which consider the path-star task.

<sup>19</sup>This relates to encoder-only non- and iterative-autoregressive models, except that these also take it one step further to create true permutation invariance via removing the causal constraint, where this attempts to learn permutation invariance via sampling permuted input sequences but keeps the causal constraint of decoder-only models.





(a) **Alternative sequential distribution using uniform label smoothing.** This is the same as Fig. 12d except the targets are stacked and only prediction types, but not individual predictions, are illustrated for visibility. In each step, ‘2’ and ‘29’ are trivially predicted via the length shortcuts and a token is trivially predicted by the CHC for all but the second time-step. The prediction of the leading node ‘12’ must always be achieved using the forward algorithm.



(b) We sample a BoW order to be used as sequential teacher-forced input into the model. Multiple targets are valid since any unsampled node in the arm can be the next token. Despite having multiple targets, this is a true next-token distribution.

Figure 13: BoW alternative sequential distribution and an equivalent BoW scratchpad.

ple targets, this is not an alternative sequential distribution but a true next-token distribution since having multiple targets is a consequence of the order permutation, which allows for any unsampled node in the arm to be the next token. One can see how this becomes less unified if we weighted the labels like we do with the other alternative sequential distributions, which would result in preferring specific permutations over others.

This is illustrated in Fig. 13, which shows the BoW label smoothing alternative distribution in Fig. 13a and how this is equivalent to a BoW scratchpad in Fig. 13b. Because we allow the BoW scratchpad to be generated in any order, an easy shortcut the model could learn would be to just learn to generate the BoW as the arm in reverse order. We know this shortcut is not learnt since it would allow the model to trivially solve the task (see Tbl. 5).

Now we explain how the BoW scratchpad induces subtask decomposition. First, determining which nodes can be in the BoW requires selecting which nodes belong to the target arm, and this necessitates arm reconstruction. Second, since the trivial reverse order is not learnt, this means that the CHC can not be solely used to achieve this, and that multihop arm reconstruction must be learnt. To see why this is, consider when two nodes,  $u$  and  $v$  are adjacent in the BoW but not in the arm, then there is no edge,  $(u, v)$ , and to predict  $v$  as the next node after  $u$  requires going through at least one intermediate node. Then this will induce subtask decomposition in the same way as masking. Fig. 14 illustrates how the next token prediction of the BoW scratchpad is related to masked predictions, where, like the alternative sequence distributions, the masked input is implicit.

#### A.4 Sorting Scratchpad and Graph-Reconstruction Scratchpads

A sorted scratchpad induces subtask decomposition in the same way as the BoW scratchpad as it is also a randomly ordered scratchpad like the BoW scratchpad. This is because the node IDs/values are randomly sampled; therefore, even though the input is in a canonical sorted order, it is still random with respect to the arm’s true order. **Thus, masking (including non-autoregressive, iterative-autoregressive, and discrete diffusion models), alternative sequential distributions, the BoW scratchpad, and the sorted scratchpad methods all induce subtask decomposition dispute in the same way, being very different meth-**

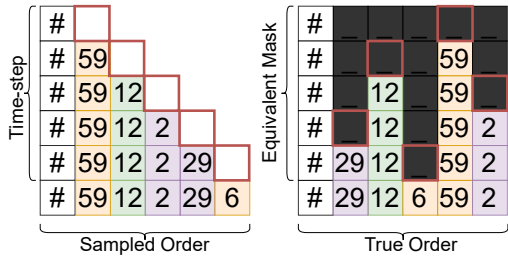


Figure 14: On the left, we show a sampled BoW order. This is a permutation of the target sequence, and we show this across all time-steps when using autoregressive next-token prediction. On the right, we show the sampled masks over the ground-truth sequence that are equivalent to a permuted sequence with next-token prediction at each step. Single-target predictions are indicated with red boxes, since this is true of either single- or multi-target predictions.

### ods in terms of implementation and motivations.

Sorting by itself does not induce subtask decomposition. This is demonstrated with the graph reconstruction scratchpad which learn to either sort the leading or target nodes but fail to learn the task. Here, the scratchpads do not require learning to select which nodes belong in the target arm, and hence there is no decomposition of learning to reconstruct the target arm. Also, the sorting of either the leading or target nodes only works because these can be trivially identified using shortcuts (the leading nodes by their adjacency to the start node and the target nodes by their degree of 1).

Sorting induces node semantics. This is because the model now learns a relationship between all the nodes ( $1 < 2 < \dots < |V|$ ) based solely on their values and not the graph structure. This is in contrast with the other versions of the task, where there is no relationship between nodes outside of the graph structure, making the nodes – and overall graphs – for these versions of the task *semanticsless*.

### A.5 Source-Side Methods

So far, the above methods, masking, alternative distributions, and scratchpads, have all been target-side modifications. Trees (or in graph topology in general), general queries, and general lengths<sup>20</sup> are all source-side modifications.

We illustrated how trees induce subtask decomposition in Fig. 8 and how tree topology alleviates the CHC by creating new positions in the sequence which avoid the CHC in Fig. 7. Here, the CHC is

<sup>20</sup>Note that general lengths are actually a graph topology modification as well, just a minor one.

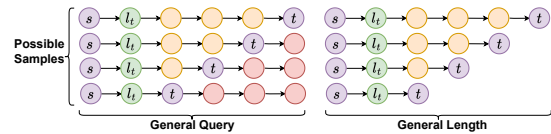


Figure 15: We show the possible general queries that can be sampled on the right and the corresponding general lengths on the left. Red nodes succeed the given target node and present the only difference between the two versions of the task.

not completely prevented. This makes sense since, in the original task, the leading node is immune to the CHC, but all others are not. Then, by inducing subtasks that mirror the original task, we are creating new subtask leading nodes (indicated with primes) that are also immune to the CHC, but any non-leading node in the subtask is still affected by the CHC. The general query and length methods still only have the leading node being immune, thus alleviating the CHC even less.

The success of autoregressive language models is due to their sample-efficient training procedure, which use the chain-rule to factorize the probability of a  $T$ -length sequence into  $T$  conditional distributions, which are efficiently trained together in parallel for transformers. One could imagine a poor training procedure for language models, which samples a single conditional distribution to train on per training sample. **Like masking, the subtask decomposition of trees is achieved across the sequential dimension, i.e., within a given training sample. This contrasts with general queries and general lengths, which achieve decomposition across training samples, i.e., intra vs. inter sample decomposition. Since the CHC applies to all targets but the leading node, this is effectively sampling a single conditional distribution per training sample – exactly like our imagined poor training procedure. This will make general queries and general lengths less sample efficient than the other methods and may explain their limited success.**

We illustrate the possible samples in Fig. 15. **General queries and general induce the same possible samples, and hence, the same subtask decomposition.** The only difference between them is that the task using general queries also requires the model to generate the full arm with tokens coming after the target token, while general lengths do not. However, this is irrelevant since tokens after the target can trivially be predicted via the CHC.

## B Experiments

We report the results of  $n = 5$  differently seeded trials for each experiment (except for a few where some trials prematurely stopped due to issues with our GPU cluster). **We find a high variance for the number of iterations needed to solve the task between trials.**<sup>21</sup> This makes considering multiple trials important when considering if a given experiment is learnable or not. Note that when we say ‘unlearnable’, this does not mean the task is provably unlearnable, but rather a shorthand for ‘not found to be empirically learnt given 100 epochs’. We abuse the term ‘epoch’ to mean 1M samples for reporting results as there are no true epochs when using online datasets. In the original setting using an offline dataset, there are 1M sampled examples, and the models are trained for 100 true epochs.

Each trial can be run on a single small GPU (a 12GB Titan X works) and, at max, takes about a day to finish (if the model does not converge and runs for all 100 epochs, and depending on the specific GPU used). The main computational cost is reporting results for 1220 trials. So, while each experiment is cheap and easy to run on moderately sized hardware, the entire set of experiments is not.

We implemented our experiments using Fairseq (Ott et al., 2019). We perform greedy decoding via Fairseq’s beamsearch with a single beam. Prefixes up to and including the special start-of-targets, ‘=’ (or start-of-scratchpad, ‘#’) are force-generated. Temperature is set to 1.0, and no length penalty is applied. We generate for a max length of 20 tokens over the ground-truth sequence length. Despite common knowledge that beamsearch with a single beam is equivalent to greedy search, we were unsure of this due to implementation details about beamsearch, which is complex (Kasai et al., 2024). Both the vanilla and first-come-first-serve (used by Fairseq) variants should be equivalent with a beam-size of 1 and hence equivalent to greedy search.<sup>22</sup> This was important to verify, as we did not want the model to ‘cheat’ by using post-hoc inference search methods in place of reasoning.

### B.1 Baseline Results

Tbl. 1 provides baseline results of the path-star task (PST) using both edge- and causal-wise shuffling of

<sup>21</sup>This was also independently observed by Saparov et al. (2025), see Appx. C.2.

<sup>22</sup>As a side note, interestingly, this no longer becomes true if employing the patience hyperparameter ( $> 1$ ) proposed in Kasai et al. (2024).

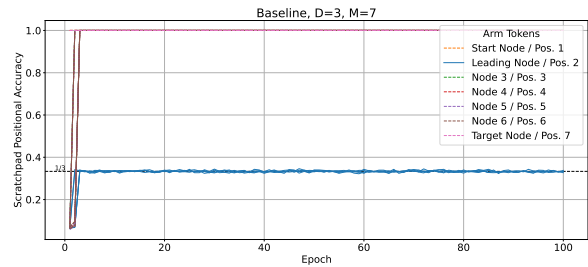


Figure 16: A baseline demonstrating multiple shortcuts used to learn all nodes except the leading node. The start and target nodes can be immediately learnt by positional shortcuts, while nodes 3-6 are learnt by the bigram CHC. The leading node is only predicted at chance accuracy of  $1/D$ . These consider ‘teacher-forced’ inference which conditions on the correct sequence regardless of past inaccuracies. We use online training so each ‘epoch’ is 1M sampled examples. It is over five seeded trials.

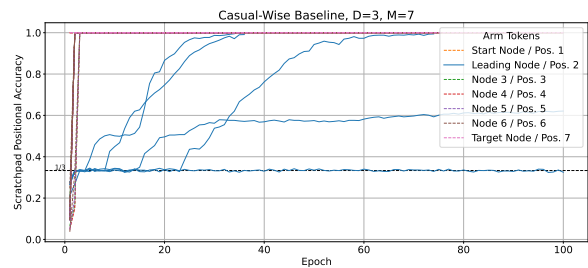


Figure 17: Using the causal-wise ordering of edges allows the task to be learnt on 3/5 trials with one run exceeding the  $1/D$  baseline but not learning the task to 95% sequential accuracy. **This is also an example showing the large variability on task success depending on the initial seed.** Note that for this experiment the only source of randomness is in graph generation.

$G$ . We find that the PST is unlearnable, even when using online training, reducing the sample space by setting the vocabulary size to the graph size ( $|V| = |G|$ ), and tokenizing  $Q$  before  $G$  (Frydenlund, 2024). This is consistent with the results of Bachmann and Nagarajan (2024). Tbl. 2 provides a more fine-grained breakdown of the accuracy of the positional token accuracy for  $l_t$  and the following two nodes for each run. This shows that  $l_t$  is predicted at  $1/D$  chance, while the next two nodes are predicted with 100% accuracy due to the CHC. This behaviour is illustrated in Fig. 16. One exception is Run 3 of the exp. where  $D = 5$ ,  $M = 7$  which fails to learn the CHC, **demonstrating the CHC is not guaranteed to be learnt, despite its seeming simplicity.** The start and target nodes are learnt immediately due to positional shortcuts and not because of the CHC.

Casual-wise shuffling enforces that, given two

edges,  $(u, v)$  and  $(v, w)$ , the former always proceeds the latter. This avoids the issue of learning two separate routing rules for decoder-only models. See Fig. 18 for an illustration. This is achieved via a sampling procedure where an arm is sampled and the edge closest to  $s$  is taken without replacement. This sampling is done until no edges remain.

We find that using a causal-wise shuffle makes the PST learnable. **This indicates that the causal constraint accounts for some of the task’s difficulty.** However, once we consider arms with moderate length ( $M = 9$ ), the task is no longer perfectly learnt (at least within the 100 epochs provided). Fig. 17 shows  $l_t$  being fitted by in the causal-wise version of the task. This also shows that, even when the task is learnt, the CHC is still employed when solving the task, otherwise we would expect a change in the overall accuracy of the other nodes as they become predicted by the new algorithm.

Because the PST in its original form is unlearnable (for reasons not due to regular hyperparameters), it is impossible to hyperparameter tune it. **As such, we use the causal-wise version to determine valid hyperparameters.** The reported results for all experiments are after having found hyperparameters using the causal-wise version of the task and the baseline results which were redone for consistency and to rule-out improper hyperparameters causing the task to be unlearnable.

## B.2 Masking Results

We sample which tokens are masked or replaced using a either a uniform distribution over the sequence length or spanning masks. Using uniform sampled masks introduces no inductive bias since it is not dependent on any task-specific information. Uniform sampling is also consistent with the sampling method using in the iterative-autoregressive models used by Frydenlund (2024) and most discrete diffusion models. Spanning masks sample multiple contiguous tokens to be masked. This is achieved via sampling a span length from a geometric distribution parameterized by  $p$  (Joshi et al., 2020). We sample two different spanning distributions to discourage contiguous ground-truth tokens with  $p = \{.4, .5\}$  for the mask spans and  $p = .8$  for the ground-truth spans. The latter means that the majority of ground-truth spans will only be a single token, which means that the CHC will not be supported in these cases. We randomize if we start with a masking- or a ground-truth span so  $l_t$  is not

always masked (which we found was important). As we are sampling based on task specific information, this is an inductive bias. But note, if we sampled enough masks, we would expect to sample these spanning masks from the uniform distribution. In addition to which positions get masked, there is also how they get masked where we try both masking via a special ‘mask-token’ (dropout) or replaced by another node in  $G$  (where the choice of node is uniformly sampled with replacement).

Tbl. 3 provides results using uniform masking via token dropout and span masking via token dropout, token replacement, and a mixture of both. The results show that the difference between these methods are not very consistent. We suspect this is due to the amount of noise being added to the training procedure (and we do not ensure that the same nodes are noised in the same place across the different noise types to control for this). **We also show that mixing causal-wise shuffling with masking improves the results over using either in isolation, implying that they are helping to solve different underlying issues** (the causal constraint issue with the former and subtask decomposition issue with the latter).

## B.3 Alternative Distributions Results

The alternative sequential distributions have different semantics from next-token distributions and break the ‘distributional’ semantics of natural language (Mikolov, 2013; Emerson, 2020). Thus they may not apply to non-planning tasks.

For each alternative sequential distribution, we employ an auxiliary loss that is only used during training. Fig. 19 illustrates the extra target-side label supervision given to the model during training. **Note how this is just a replication of the original target labels with an alternative structure, thus no new information is given, but rather it is just provided in an alternative way.** The auxiliary loss is trained in conjunction with the main loss. Because of the change in semantics, we do not want to interfere with the main loss and the true next-token distribution. As such, we use an interior hidden-state as  $B$  instead of the final hidden-state, which supports the main loss as usual (we use the second last hidden-state). We increase the number of layers from  $L = 8$  to 9 to account for this. This allows the auxiliary distributions to perform the same number of hops as the baseline models (see RASP constructions in Frydenlund (2024)). We do not believe that the extra computation affects com-

Graph,  $G$  as a causal-wise shuffled list of all edges

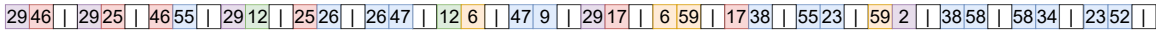


Figure 18: One potential causal-wise shuffle of the path-star graph of Fig. 1. The arms are not contiguous, but, the order of the edges is such that a given one is always further or of equal distance from  $s$  compared to all prior edges.

Experiment Description	$D$	$M$	Test-Force $R_t$		Test-Gen $R_t$	
			SR	ABB	SR	ABB
Edge-Wise $ V  =  G $ , Online Training, $Q$ Before $G$	2	5	0%	0%	0%	0%
	3	5	0%	0%	0%	0%
	4	5	0%	0%	0%	0%
	5	5	0%	0%	0%	0%
	2	7	0%	0%	0%	0%
	3	7	0%	0%	0%	0%
	4	7	0%	0%	0%	0%
	5	7	0%	0%	0%	0%
	2	5	100%	100%	100%	100%
	3	5	100%	100%	100%	100%
Causal-Wise	4	5	100%	100%	100%	100%
	5	5	60%	100%	60%	100%
	2	7	100%	100%	100%	100%
	3	7	60%	80%	60%	80%
	4	7	0%	20%	0%	20%
	5	7	0%	40%	0%	40%
	2	9	40%	100%	40%	100%
	3	9	0%	40%	0%	40%
	2	12	0%	80%	0%	80%
	3	12	0%	20%	0%	20%

Table 1: Full baseline experiment results. We report the *Success Rate* (SR) where the model predicts  $> 95\%$  sequential accuracy and *Above-Baseline* (ABB) where the model predicts  $> (100/D + 10)\%$  sequential accuracy. When this happens it indicates that the model can predict  $l_t$  in some cases. As such, when  $ABB > SR$ , it implies that the model would have learnt the task had it been provided with more training time in these cases. We report on the test partition using both ‘teacher-forced inference’ which conditions on the correct sequence regardless of past inaccuracies (Test-Force  $R_t$ ) as well as true auto-regressive generation (Test-Gen  $R_t$ ). In general, these provide the same results, since  $l_t$  will either be learnt at  $> 95\%$  accuracy or not in both cases, leading to the same overall sequential accuracy. Results are reported after 100 epochs i.e. 100M training samples.

$D$	$M$	Run 1			Run 2			Run 3			Run 4			Run 5		
		$l_t$	pos. 2	pos. 3	$l_t$	2	3	$l_t$	2	3	$l_t$	2	3	$l_t$	2	3
2	5	50%			50%			50%			50%			50%		
3	5	33%			33%			33%			33%			33%		
4	5	25%			25%			25%			25%			25%		
5	5	20%			20%			20%			20%			20%		
2	7	50%			50%			50%			50%			50%		
3	7	33%			33%			33%			33%			33%		
4	7	25%			25%			25%			25%			25%		
5	7	20%			20%			4%	4%	4%	20%			20%		

Table 2: Training positional accuracy for  $l_t$ , pos. 2, and pos. 3 for the edge-wise baseline results in Tbl. 1. indicates 100% accuracy. In all but a single run, the CHC is learnt for pos. 2 and 3 (and all other non-leading nodes) and  $l_t$  is predicted at  $1/D$  chance. ‘pos.’ is redacted for space for trials 2-5.

Experiment Description	$D$	$M$	Test-Force $R_t$		Test-Gen $R_t$		
			SR	ABB	SR	ABB	
Uniform Token Dropout	2	5	60%	60%	60%	60%	
	3	5	100%	100%	100%	100%	
	4	5	100%	100%	100%	100%	
	5	5	80%	80%	80%	80%	
	2	7	60%	80%	60%	80%	
	3	7	80%	80%	80%	80%	
	4	7	40%	60%	40%	60%	
	5	7	0%	20%	0%	20%	
	2	9	40%	60%	40%	60%	
	3	9	0%	20%	0%	20%	
	4	9	0%	0%	0%	0%	
	5	9	0%	0%	0%	0%	
	Span Token Dropout	2	5	100%	100%	100%	100%
		3	5	100%	100%	100%	100%
		4	5	100%	100%	100%	100%
5		5	60%	80%	60%	80%	
2		7	80%	80%	80%	80%	
3		7	40%	60%	40%	60%	
4		7	20%	20%	20%	20%	
5		7	40%	40%	40%	40%	
2		9	40%	40%	40%	40%	
3		9	0%	0%	0%	0%	
4		9	0%	0%	0%	0%	
5		9	0%	0%	0%	0%	
2		12	0%	0%	0%	0%	
Causal-Wise Span Token Dropout		5	9	60%	60%	60%	100%
		5	12	0%	66%	0%	66%
Span Token Replacement	2	5	60%	60%	60%	60%	
	3	5	80%	80%	80%	80%	
	2	7	100%	100%	100%	100%	
	3	7	20%	40%	20%	40%	
Span Mixed Token Dropout and Replacement	2	5	80%	80%	80%	80%	
	3	5	80%	80%	80%	80%	
	4	5	100%	100%	100%	100%	
	5	5	80%	80%	80%	80%	
	2	7	100%	100%	100%	100%	
	3	7	100%	100%	100%	100%	
	4	7	0%	20%	0%	20%	
	5	7	0%	20%	0%	20%	
	2	9	60%	60%	60%	60%	
	3	9	80%	80%	80%	80%	
	2	12	20%	20%	20%	20%	
	3	12	0%	0%	0%	0%	

Table 3: Full masking experiment results.

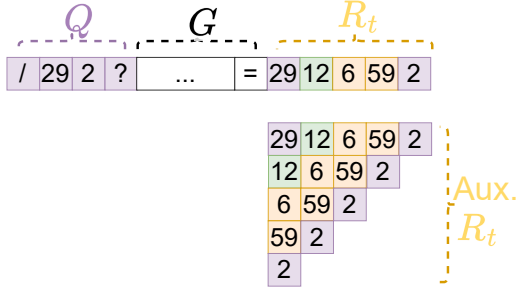


Figure 19: Auxiliary targets, Aux.  $R_t$ , provided for training the BoW, LS, and RITF auxiliary losses. Tokens from prior steps are removed from consideration. Here  $R_t$  provides a singular ground-truth at each step, while aux.  $R_t$  provides multiple for each step but the last.

parisons between these results and other methods that use  $L = 8$ .

We use a monotonically decreasing stepped weighting for LS where the value between each consecutive weight is the same. Thus the actual weighting dynamically changes depending on  $M$  and the current step. In Fig. 19 this is applied on each column of aux.  $R_t$  (at each step) individually.

For RITF, we provided a partial implementation of the hinge-wise loss in Eq. 2. This corresponds with ranking the elements in each column in Fig. 19 from highest to lowest (equivalent to the sequential order of the arm).<sup>23</sup> We used a hinge of  $h = 1$  and did not experiment with other values. Note that Eq. 2 is slightly ill-defined since we used score indices over the sequence length where these need to be translated to vocabulary indices in the range of  $|V|$  plus the number of special tokens. Including this would have complicated the equation to little benefit to the reader.

In addition ranking nodes into the future, we also rank any node in  $G$  not in  $R_t$  lower than any node in  $R_t$ . We consider the entire vocabulary in practice because it is easier to calculate, however, the intuition of the inductive bias concerns nodes in  $G$ . This can be done using the same calculation,

$$L_B = \sum_{i=1}^M \sum_j \sum_k \max(0, 1 - (\sigma_i[j] - \sigma_i[k])), \quad (3)$$

except where  $j \in R_t$  and  $k \in V - R_t$  i.e. over differently selected pairs from Eq. 2. Thus the overall

<sup>23</sup>By removing the prior tokens from consideration, there are superficial similarities to the exclusionary procedure of Plackett-Luce, however, this is only superficial because the logits or scores change at every step here.

ranking loss factorizes as two disjoint losses, one for each inductive bias being modelled.

Tbl. 4 provides results using the alternative distributions. We find poor results for LS in particular. We strongly suspect this is because each LS weight also functions as a weight on the corresponding loss term. This means that far-future tokens will have tiny contributions to the overall loss. Although we allow for scaling of the monotonically decreasing terms (via a temperature hyperparameter), we do not experiment with this. It may be that doing so will result in better performance, but, we argue that using RITF instead avoids this complication.

#### B.4 Arm Reconstruction Scratchpads Results

Tbl. 5 shows the results for arm reconstruction scratchpads. These are the first results that see differences between autoregressive inference and teacher-forced inference. Sequence accuracy is evaluated independently for  $R_t$  and the scratchpad ( $S$ ) in order to support better analysis of what the model is learning. This is why the model can get 100% sequence accuracy for  $R_t$  but less than that for the SP in the teacher-forced setting and why the autoregressive inference can differ from teacher-forced inference results. As such, ‘Test-Gen  $R_t$ ’ is the statistic one should consider when determining if a given experiment actually learnt the task.

**Note that while the reverse ‘solution’ is always 100% accurate, it is only so because of the use of shortcuts were we can learn the reverse output via CHC, which then can be reversed. However, this just allows the model to bypass learning any planning or graph reconstruction. Thus while the task is ‘solved’, it is for the wrong reasons.**

In addition to the SPs described above, we tried one that predicts  $R_t$  twice in a row to compare with the reverse SP i.e. forward-forward instead of reverse-forward. To do this we also used masking with token replacement on the SP. This is the only experiment where we consider adding masking noise to the SP and, hence, is slightly incompatible with the others.

#### B.5 Graph Reconstruction Scratchpads Results

Given the path-star graph in Fig. 21a, Figs., 21b, 21d, 21c, and 21e illustrate the tokenization of the graph reconstruction scratchpads for all four combinations of leading-to-target or target-to-leading and either sorting by leading or target node values.

Experiment Description	$D$	$M$	Test-Force $R_t$		Test-Gen $R_t$		
			SR	ABB	SR	ABB	
Uniform Label Smoothing (BoW)	2	5	100%	100%	100%	100%	
	3	5	100%	100%	100%	100%	
	4	5	20%	60%	20%	60%	
	5	5	20%	60%	20%	60%	
	2	7	60%	60%	60%	60%	
	3	7	100%	100%	100%	100%	
	4	7	0%	0%	0%	0%	
	5	7	0%	0%	0%	0%	
	2	9	100%	100%	100%	100%	
	3	9	100%	100%	100%	100%	
	4	9	20%	60%	20%	60%	
	5	9	0%	0%	0%	0%	
	2	12	0%	20%	0%	20%	
	3	12	0%	0%	0%	0%	
	4	12	0%	0%	0%	0%	
	5	12	0%	0%	0%	0%	
	Monotonically Decreasing Label Smoothing (LS)	2	5	100%	100%	100%	100%
		3	5	100%	100%	100%	100%
		4	5	100%	100%	100%	100%
		5	5	100%	100%	100%	100%
2		7	100%	100%	100%	100%	
3		7	100%	100%	100%	100%	
4		7	40%	40%	40%	40%	
5		7	0%	0%	0%	0%	
2		9	60%	60%	60%	60%	
3		9	0%	0%	0%	0%	
4		9	0%	0%	0%	0%	
5		9	0%	0%	0%	0%	
Ranking into the Future (RITF)		2	5	100%	100%	100%	100%
		3	5	100%	100%	100%	100%
		4	5	100%	100%	100%	100%
	5	5	80%	80%	80%	80%	
	2	7	100%	100%	100%	100%	
	3	7	100%	100%	100%	100%	
	4	7	80%	80%	80%	80%	
	5	7	60%	60%	60%	60%	
	2	9	100%	100%	100%	100%	
	3	9	100%	100%	100%	100%	
	4	9	100%	100%	100%	100%	
	5	9	60%	100%	60%	100%	
	2	12	100%	100%	100%	100%	
	3	12	100%	100%	100%	100%	
	4	12	60%	100%	60%	100%	
	5	12	0%	100%	0%	100%	
	2	15	60%	100%	60%	100%	
	3	15	0%	100%	0%	100%	

Table 4: Alternative sequential (future) distribution results.



Exp. Desc.	$D$	$M$	Test-Force $R_t$		Test-Force $S$		Test-Gen $R_t$		Test-Gen $S$		
			SR	ABB	SR	ABB	SR	ABB	SR	ABB	
Reverse	5	5	100%	100%	100%	100%	100%	100%	100%	100%	
	5	7	100%	100%	100%	100%	100%	100%	100%	100%	
	5	9	100%	100%	100%	100%	100%	100%	100%	100%	
	5	12	100%	100%	100%	100%	100%	100%	100%	100%	
BoW	2	5	100%	100%	100%	100%	100%	100%	NA	NA	
	3	5	100%	100%	100%	100%	100%	100%	NA	NA	
	4	5	100%	100%	100%	100%	100%	100%	NA	NA	
	5	5	100%	100%	100%	100%	100%	100%	NA	NA	
	2	7	100%	100%	100%	100%	100%	100%	NA	NA	
	3	7	100%	100%	100%	100%	100%	100%	NA	NA	
	4	7	100%	100%	100%	100%	100%	100%	NA	NA	
	5	7	60%	60%	60%	60%	60%	60%	NA	NA	
	2	9	100%	100%	100%	100%	100%	100%	NA	NA	
	3	9	80%	80%	80%	80%	40%	80%	NA	NA	
	4	9	40%	60%	60%	60%	0%	40%	NA	NA	
	5	9	40%	40%	40%	40%	0%	40%	NA	NA	
	Sorted Arm	2	5	100%	100%	100%	100%	100%	100%	100%	100%
		3	5	100%	100%	100%	100%	100%	100%	100%	100%
		4	5	100%	100%	100%	100%	100%	100%	100%	100%
		5	5	100%	100%	100%	100%	100%	100%	100%	100%
2		7	100%	100%	100%	100%	100%	100%	100%	100%	
3		7	100%	100%	100%	100%	100%	100%	100%	100%	
4		7	100%	100%	60%	100%	60%	100%	60%	100%	
5		7	100%	100%	40%	100%	40%	100%	40%	100%	
2		9	100%	100%	100%	100%	100%	100%	100%	100%	
3		9	100%	100%	80%	100%	100%	100%	80%	100%	
4		9	100%	100%	0%	100%	0%	100%	0%	100%	
5		9	60%	60%	20%	20%	20%	20%	20%	20%	
2		12	100%	100%	40%	80%	40%	80%	40%	80%	
3		12	100%	100%	0%	40%	0%	40%	0%	40%	
4		12	0%	20%	0%	0%	0%	0%	0%	0%	
5		12	0%	0%	0%	0%	0%	0%	0%	0%	
Forward	2	5	100%	100%	80%	80%	100%	80%	100%	80%	
	3	5	100%	100%	100%	100%	100%	100%	100%	100%	
	4	5	100%	100%	100%	100%	100%	100%	100%	100%	
	5	5	100%	100%	100%	100%	100%	100%	100%	100%	
	2	7	100%	100%	75%	75%	75%	75%	75%	75%	
	3	7	20%	100%	20%	20%	20%	20%	20%	20%	
	4	7	0%	0%	0%	0%	0%	0%	0%	0%	
	5	7	0%	0%	0%	0%	0%	0%	0%	0%	
	2	9	60%	100%	40%	60%	40%	60%	40%	80%	
	3	9	0%	0%	0%	0%	0%	0%	0%	0%	
	4	9	0%	0%	0%	0%	0%	0%	0%	0%	
	5	9	0%	0%	0%	0%	0%	0%	0%	0%	

Table 5: Results for arm reconstruction scratchpad. Sequence accuracy is evaluated independently for  $R_t$  and the scratchpad ( $S$ ) We also achieved the same results for the reverse scratchpad for  $D \in \{2, 3, 4\} \times M \in \{5, 7, 9\}$ . For the BoW experiments, there are multiple correct values for each predictive scratchpad step (except for the last) and we did not implement multi-value accuracy for scratchpad generation (hence reporting ‘NA’).

Exp. Desc.	$D$	$M$	Test-Force $R_t$		Test-Force $S$		Test-Gen $R_t$		Test-Gen $S$	
			SR	ABB	SR	ABB	SR	ABB	SR	ABB
$S_{L \rightarrow T, \prec L}$	2	5	100%	100%	40%	40%	40%	40%	40%	40%
	3	5	100%	100%	0%	0%	0%	0%	0%	0%
	4	5	100%	100%	0%	0%	0%	0%	0%	0%
	5	5	100%	100%	0%	0%	0%	0%	0%	0%
$S_{T \rightarrow L, \prec L}$	2	5	100%	100%	20%	20%	20%	20%	20%	20%
	3	5	100%	100%	0%	0%	0%	0%	0%	0%
	4	5	100%	100%	0%	0%	0%	0%	0%	0%
	5	5	100%	100%	0%	0%	0%	0%	0%	0%
$S_{L \rightarrow T, \prec T}$	2	5	100%	100%	0%	0%	0%	0%	0%	0%
	3	5	100%	100%	0%	0%	0%	0%	0%	0%
	4	5	100%	100%	0%	0%	0%	0%	0%	0%
	5	5	100%	100%	0%	0%	0%	0%	0%	0%
$S_{T \rightarrow L, \prec T}$	2	5	100%	100%	20%	20%	20%	20%	20%	20%
	3	5	100%	100%	0%	0%	0%	0%	0%	0%
	4	5	100%	100%	0%	0%	0%	0%	0%	0%
	5	5	100%	100%	0%	0%	0%	0%	0%	0%

Table 6: Results for graph-reconstruction scratchpads.

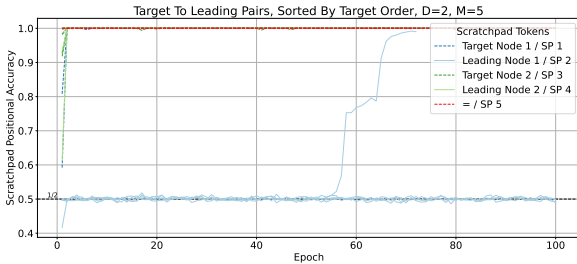


Figure 20: A  $D = 2$  graph-reconstruction experiment where one of the trials successfully learnt the task.

Tbl. 6 shows the results for graph reconstruction scratchpads. This shows that only 4 trials succeeded in learning the task (again, ‘Test-Gen  $R_t$ ’ is the statistic to consider for success).

Results for the experiments where  $D = 4$  and  $M = 5$  are plotted across training in Figs. 22a, 22b, 22c, and 22d. Figs. 22a and 22b demonstrate that sorting by leading node leads to all leading nodes being correctly predicted, regardless of whether the leading node precedes or succeeds the corresponding target node. Figs. 22c and 22d show the inverse of this where sorting on the target node leads to all target nodes being correctly predicted. Here we see a consistent pattern where the corresponding non-sorted nodes fail to be predicted above chance. Note how once the model conditions on one of these nodes, it removes it from consideration in the next prediction, hence the first one fails at  $1/D = 1/4$  chance, and the next at

$1/(D - 1) = 1/3$  chance etc.

Consider the two cases where the leading node precedes the target node and the arms are sorted by leading node value in Fig. 22a and where the target node precedes the leading node and the arms are sorted by target node value in Fig. 22d. **This indicates that the model can correctly predict and thus condition on the correct preceding node but fails to predict the corresponding target or leading node in the next prediction even though the path between them is deterministic. This is also an instance where, in trying to solve the problem, we introduce alternative shortcuts which also (seem to) prevent learning and shows that one needs to be careful when adding extra supervision via scratchpads to avoid adulteration.**

Note that had we given the full arms instead of just pairs of nodes as the scratchpad, the task should become learnable, though for incorrect reasons, where instead of learning planning or arm reconstruction, the model should be able to use the sorting shortcuts in conjunction with the CHC to get 100% accuracy. That is, the version we present here is designed to avoid adulteration, even if that would actually lead to ‘succeeding’ on the task.

## B.6 Query Results

For the query subsets method, we use a random subset of  $R_t$  as the query nodes in addition to the start node. These are in random order. The query is

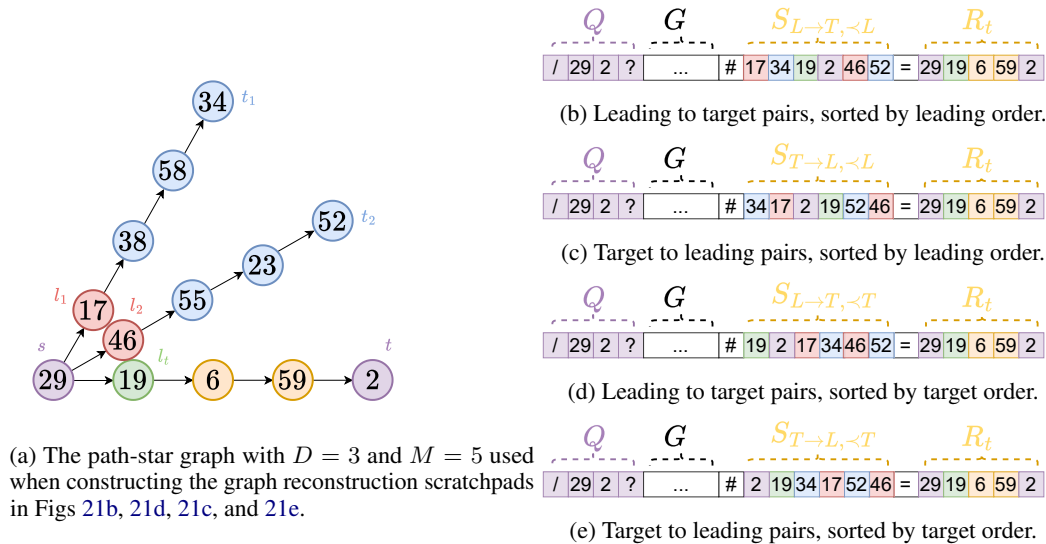


Figure 21: Illustration of graph reconstruction scratchpad. Note this is slightly different from the above graph. This is done to have more illustrative combinations of leading and target nodes after sorting.

then padded out to be of length  $M$  with the padding tokens coming after the observed query nodes. This is to avoid introducing dynamic sequence lengths which would be a confounding factor when comparing against the original single target query results. During evaluation, only the final node  $t$  is given but the query is still padded to length  $M$ .

For the general single target method, all nodes in  $R_t$ , with the exception of  $s$ , are considered with uniform probability. Again only the final node is used during evaluation.

Tbl. 7 shows the results of using general queries. Only 3/20 trials succeeded in learning the task with a general target in the original setting (which uses  $|V|=100$ , Offline Training, and placing  $Q$  after  $G$ ) compared 16/20 successful corresponding trials in the new setting (which uses  $|V| = |G|$ , online Training, and placing  $Q$  before  $G$ ). This indicates that such a finding may be easy to miss. However, as discussed above there is also the issue of hyperparameter-tuning a model which fails to learn a task. One can not hyperparameter tune models on the unadulterated path-star task in its original form as it doesn't learn above chance. **Thus another explanation for why this result might be hard to find, is that, without first finding working models (in our case using the causal-wise shuffling), we may not have properly set the hyperparameters needed for finding successful trials.**

## B.7 Tree Results

Tbl. 8 shows the results of the tree experiments. During training, we intermix sampling trees and path-star graphs with the latter being 10% of the training examples. This was done due to a length issue where trees can only make strictly shorter paths than the original path task.

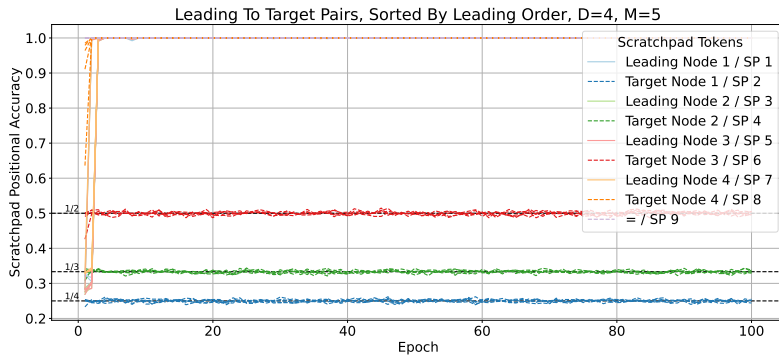
We generate  $D$ -ary trees by considering branching at probabilities 0.3, 0.4, 0.2, and 0.1 for no branching, branching with 2 children, 3 children, and 4 children respectively. In any branching case, the remaining nodes are equally divided into each new subtree. This is repeated recursively until all nodes in  $R_t$  are consumed. We generate split trees using a 0.5 split probability and the remaining nodes are equally divided into each new subtree. The 'left' subtree is just a path while the 'right' subtree repeats this process recursively.

## B.8 Training on Multiple Lengths and/or Degrees Results

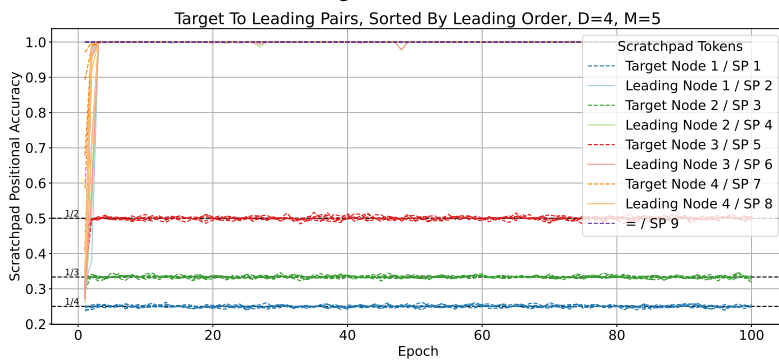
Tbl. 9 shows the results for training using a sampled  $M$  and/or sampled  $D$ . All values are uniformly sampled.

## C Related Work

There are extensive prior works given that, a), the path-star task questions the fundamental sufficiency of next-token prediction for planning tasks, b), the presented solutions vary widely in terms of



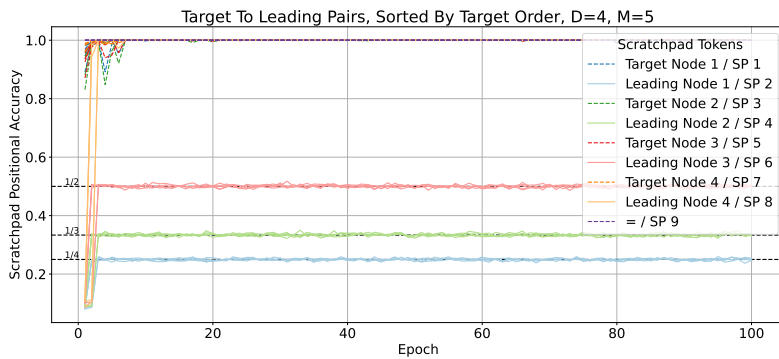
(a) Leading nodes are predictable when sorting by leading order. However, the targets corresponding to leading nodes can not be predicted even when conditioning on the correct corresponding leading node. These then get guessed at  $1/4$ ,  $1/3$ , and  $1/2$  accuracy, with the last being correctly predicted as the only remaining target. Each plot consists of 5 differently seeded experiments. Note that colours correspond to leading/target index and not scratchpad (SP) index i.e. the sort order not the sequential order. Thus the colours are consistent across figures.



(b) Leading nodes are still predictable when sorting by leading order, even when following incorrect target nodes in the scratchpad.



(c) Target nodes are predictable when sorting by target order, even when following incorrect leading nodes in the scratchpad.



(d) Target nodes are still predictable when sorting by target order, however, the correct leading node can not be predicted even when conditioning on the correct corresponding target node.

Figure 22: Validation set accuracy of the scratch pad tokens across training. These results consider ‘teacher-forced’ inference which conditions on the correct sequence regardless of past inaccuracies.

Experiment Description	$D$	$M$	Test-Force $R_t$		Test-Gen $R_t$	
			SR	ABB	SR	ABB
	2	5	100%	100%	100%	100%
	3	5	75%	75%	75%	75%
	4	5	40%	40%	40%	40%
	5	5	80%	80%	80%	80%
	5	5	80%	80%	80%	80%
Query Subset (Padded)	2	7	60%	100%	60%	100%
	3	7	0%	0%	0%	0%
	4	7	0%	0%	0%	0%
	5	7	0%	0%	0%	0%
	5	7	0%	0%	0%	0%
General Single Target	2	9	20%	100%	20%	100%
	2	5	100%	100%	100%	100%
	3	5	100%	100%	100%	100%
	4	5	80%	80%	80%	80%
	5	5	40%	80%	40%	80%
	2	7	100%	100%	100%	100%
	3	7	80%	100%	80%	100%
	4	7	20%	60%	20%	60%
	5	7	40%	40%	40%	40%
	5	7	40%	40%	40%	40%
General Single Target (Original Setting) $ V =100$ , Offline Training, $Q$ After $G$	2	5	20%	20%	20%	20%
	3	5	20%	20%	20%	20%
	4	5	20%	60%	20%	60%
	5	5	0%	0%	0%	0%

Table 7: Results for alternative query methods. All results are evaluated with the query being the final node only.

methodology, and, c), we provide theoretical insights into the task. As such, this is not an exhaustive review (and still reads like ‘A House of Leaves’ (Danielewski, 2000))<sup>24</sup>. We also point the reader to the substantial review given in Bachmann and Nagarajan (2024).

### C.1 Large Language Model (LLMs)

LLMs have become the ubiquitous model for solving NLP tasks (Brown et al., 2020; Minaee et al., 2024; Matarazzo and Torlone, 2025). Their abilities are assessed under various settings and methods. Zero-shot evaluation queries an LLM with a single direct question. This can be enhanced with various prompting methods which prepend additional text to the query (Qiao et al., 2023; Schulhoff et al., 2024). This leads to few-shot prompting which uses supervised exemplars of question-answer pairs, enabling in-context learning (Brown et al., 2020; Dong et al., 2024). These methods are training-free as they do not modify the model’s

parameters. Alternatively, fine-tuning can be performed (Han et al., 2024; Zhang et al., 2023b).

Methods like chain-of-thought (CoT) and scratchpads elicit LLMs to generate multiple reasoning steps before generating an answer (Nye et al., 2022; Wei et al., 2022; Chu et al., 2024). This is achieved via additional prompt supervision. CoT can be done in the zero-shot setting with the generic prompt ‘Let’s think step by step’ (Kojima et al., 2022). For graphical tasks, zero-shot prompts include ‘Let’s construct a graph with the nodes and edges first’ and ‘We can use a Depth-First Search (DFS) algorithm’ (Wang et al., 2023b). In the few-shot setting, prompts can provide a step-by-step decomposition of the task. For example ‘Let’s run depth-first search (DFS) step by step. Visit node 0. Neighbors of node 0: [3, 6]. Visit node 6. Neighbors of node 6: [3, 0]. ...’ (sic., Luo et al., 2024).<sup>25</sup>

Sprague et al. (2025) found that CoT is most beneficial for symbolic tasks. Wang and Zhou (2024) showed that unprompted LLMs still perform a CoT-

<sup>24</sup>See ‘The House Is Turing Complete Under Assumptions’ in Transactions of House Mathematics, 2035). To be slightly more serious, this bloated related works was done for our thesis and it would be nice if more people than just our committee ever saw it, so we included it in case others find it useful.

<sup>25</sup>CoT and scratchpads use similar methods and were introduced simultaneously. ‘CoT’ is most commonly used in the training-free setting, whereas ‘scratchpads’ generally implies a training setting and supervised decomposition.

Experiment Description	$D$	$M$	Test-Force $R_t$		Test-Gen $R_t$		
			SR	ABB	SR	ABB	
<i>D</i> -ary Trees	2	5	100%	100%	100%	100%	
	3	5	60%	100%	60%	100%	
	4	5	0%	100%	0%	100%	
	5	5	0%	100%	0%	100%	
	2	7	0%	80%	0%	80%	
	3	7	20%	100%	20%	100%	
	4	7	0%	100%	0%	100%	
	5	7	0%	60%	0%	60%	
	Split Trees	2	5	100%	100%	100%	100%
		3	5	100%	100%	100%	100%
4		5	100%	100%	100%	100%	
5		5	100%	100%	100%	100%	
2		7	60%	100%	60%	100%	
3		7	100%	100%	100%	100%	
4		7	20%	100%	20%	100%	
5		7	60%	100%	60%	100%	
2		9	80%	100%	80%	100%	
3		9	0%	40%	0%	40%	
4		9	0%	20%	0%	20%	
5		9	0%	0%	0%	0%	
2		12	0%	20%	0%	20%	
3		12	0%	20%	0%	20%	
4		12	0%	0%	0%	0%	
5	12	0%	0%	0%	0%		
2	15	0%	20%	0%	20%		

Table 8: Results for tree methods.

Experiment Description	Trained On		Test-Force $R_t$		Test-Gen $R_t$		
	$D$	$M$	SR	ABB	SR	ABB	
Multi. $M$	2	[2-5]	100%	100%	100%	100%	
	3	[2-5]	100%	100%	100%	100%	
	4	[2-5]	100%	100%	100%	100%	
	5	[2-5]	100%	100%	100%	100%	
	2	[2-7]	60%	60%	60%	60%	
	3	[2-7]	20%	80%	20%	80%	
	4	[2-7]	0%	60%	0%	60%	
	5	[2-7]	0%	100%	0%	100%	
	2	[2-9]	40%	40%	40%	40%	
	3	[2-9]	20%	40%	20%	40%	
	4	[2-9]	0%	20%	0%	20%	
	5	[2-9]	0%	20%	0%	20%	
	2	[2-12]	0%	20%	0%	20%	
	Multi. $D$	[2-3]	5	0%	NA	0%	NA
		[2-4]	5	0%	NA	0%	NA
[2-5]		5	0%	NA	0%	NA	
Multi. $M$ with Multi. $D$	[2-3]	[2-9]	60%	NA	60%	NA	
	[2-4]	[2-9]	20%	NA	20%	NA	
	[2-5]	[2-9]	0%	NA	0%	NA	
	[2-3]	[2-12]	0%	NA	0%	NA	
Multi. $M$ with General Single Target	2	[2-9]	60%	100%	60%	100%	
	3	[2-9]	0%	100%	0%	100%	
	4	[2-9]	20%	80%	20%	80%	
	5	[2-9]	80%	100%	80%	100%	
	3	[2-12]	20%	80%	20%	80%	

Table 9: All results are evaluated with the query being only the final node in the arm. We sample both  $M$  and  $D$  during the evaluation (where applicable). The above baseline (ABB) statistic does not work when considering multiple  $D$  values as it depends on a single  $D$  value (hence ‘NA’).

like reasoning in non-top scoring beams during beamsearch which implies task decomposition is done by LLMs given a proper search method.

### C.1.1 Reasoning and Planning

While LLMs were originally designed for use on natural language tasks, it has become common to use LLMs as general predictive computation models and to apply them to reasoning tasks (Huang and Chang, 2023; Bubeck et al., 2023; Zhao et al., 2023; OpenAI, 2024), including math (Rabe et al., 2021; Zhang et al., 2022), puzzles (Shah et al., 2024; Stechly et al., 2025), code generation (Zan et al., 2023; Jiang et al., 2024b), question answering (Geva et al., 2021; Kamaloo et al., 2023; Ding et al., 2024), abstract pattern matching (Chollet et al., 2024; Chollet, 2024), graphs (see Appx. C.1.2), and planning (Zhao et al., 2023; Valmeekam et al., 2023b; Plaat et al., 2024; Stechly et al., 2025; Kang et al., 2024). Planning and reasoning are closely linked, with planning being a kind of reasoning that achieves a desired goal after a series of actions thus requiring sequential decision-making (Kang et al., 2024).

It has been found that LLMs struggle to solve various reasoning tasks (Rae et al., 2021; Han et al., 2022; Zhang et al., 2023a; Ruis et al., 2023; Creswell et al., 2023; Balepur et al., 2024; Mirzadeh et al., 2025; Jiang et al., 2024a; Bian et al., 2024) including planning (Bubeck et al., 2023; Valmeekam et al., 2023a,b; Stechly et al., 2025; Plaat et al., 2024; Kambhampati et al., 2024). Huang et al. (2024b) found that fine-tuning on planning tasks does not lead to good out-of-distribution (OOD) performance. These results can be improved using various heuristics and strong search methods (Yao et al., 2023; Valmeekam et al., 2023b; Creswell et al., 2023; Stechly et al., 2025; Plaat et al., 2024; Huang et al., 2024b). Hao et al. (2023) explored the need for LLMs to represent planning states explicitly. They experiment with both easy and hard problems after observing that LLMs can fail on tasks that humans view as easy. Kambhampati et al. (2024) argued that LLMs by themselves can not plan, but can when provided with auxiliary models which verify generated plans. This poor performance has led to LLMs being pretrained specifically for reasoning tasks (OpenAI, 2024)<sup>26</sup>, which have been shown to outperform other LLMs on reasoning and graphical tasks

<sup>26</sup>Marketed under the name ‘Large Reasoning Models’ (Valmeekam et al., 2024; Zhao et al., 2024).

(Valmeekam et al., 2024; Tang et al., 2025).

For the path-star task, the reasoning task is choosing the correct leading node, and this requires planning to achieve. Bachmann and Nagarajan (2024) put forth the argument that LMs failing to learn the path-star task indicates a fundamental inability to learn simple planning tasks via next-token prediction, implying that the poor planning abilities of LLMs may stem from being trained via next-token prediction. **We find that the core difficulty of the path-star task does not concern planning.** Note, while we argue that planning is not the core difficulty, planning and reasoning often require multihop reasoning. This is highly related to task decomposition where each hop is the same operation. Thus our decomposition findings may be of relevance to other reasoning tasks.

**We also believe that the kinds of adulteration we have described would have a small impact on the above LLMs. However, any symbolic tasks where next-tokens can be directly inferred via prior tokens, and are trained to do so, will be at risk of adulteration. This issue may become more common due to the recent interest in pretraining models on reasoning tasks.** We discuss this further in Appx. C.1.2. It is unclear if in-context learning will induce the same kind of shortcuts like CHC as training, however, Khona et al. (2024) showed a simplicity bias for in-context learning, which they point out is related to shortcut learning (see Appx. C.3).

We use small LMs. The reasoning abilities of LLMs are considered an emergent property (Huang and Chang, 2023), though this may be an artifact of using discontinuous evaluations (Schaeffer et al., 2023). Bi et al. (2024) used knowledge distillation to generate chain-of-thought/scratchpad supervision to fine-tune small language models. Lee et al. (2024) did the opposite of this where a small LM was used to guide the generation of a large one.

Lin et al. (2025a) studies the effect of restricting training to just predicting ‘critical tokens’ instead of using full next-token prediction on reasoning tasks. They find that full next-token prediction works better for pertaining but restricted training can be more efficient for finetuning. **Interestingly, the training procedure of the path-star task can be viewed as such a restriction since next-token prediction is only performed on the target-side.** This is because next-token prediction on  $G$  and  $Q$  is invalid as both must be given information, i.e., you can not predict the next token in the graph



without first knowing the graph.

### C.1.2 LLMs on Graphs

Reasoning tasks have an implicit graphical structure (Dziri et al., 2023; Creswell et al., 2023; Xu et al., 2023; Hao et al., 2023; Zhao et al., 2023; Wu et al., 2024c; Khona et al., 2024; Zhu et al., 2024; Kang et al., 2024; Stechly et al., 2025; Han et al., 2025, *inter alia*). In general, the outputs of any deterministic algorithm decompose into a series of reasoning/computation steps forming a DAG (Dziri et al., 2023; Khona et al., 2024).

These tasks can be specified in natural language (Tandon et al., 2019; Madaan et al., 2021; Saha et al., 2021; Sakaguchi et al., 2021; Huang et al., 2022; Valmeekam et al., 2023b; Zhang et al., 2023a; Ding et al., 2024; Huang et al., 2024a, *inter alia*). This introduces a subtask of mapping language to graph (Wang et al., 2023b; Fatemi et al., 2024). Madaan et al. (2022) found that LLMs that generate reasoning as code instead of natural language are better reasoners, i.e., mapping to a symbolic language may offer better predictive performance.

The implicit graphical nature of reasoning tasks has motivated evaluating LLMs on explicit graphical tasks isolated from various confounding complexities that these reasoning tasks often introduce. **This assumes that the minimized graphical tasks act as a surrogate to the original reasoning tasks and that this isolates aspects that make the original tasks difficult without introducing new difficulties.** To this end, many graph benchmarks and datasets have recently been introduced using synthetic data (Wang et al., 2023b; Liu and Wu, 2023; Fatemi et al., 2024; Luo et al., 2024; Chen et al., 2024b; Dai et al., 2024b,a; Fan et al., 2024) and real-world data (Guo et al., 2023; Wang et al., 2024a; Zhang et al., 2024; Wu et al., 2024a; Yuan et al., 2024; Li et al., 2024b; Tang et al., 2025). Tang et al. (2025); Fan et al. (2024) group the task by difficulty according to its complexity class (which relates to expressibility, Appx. C.5).

LLMs struggle to solve graphical tasks (Wang et al., 2023b; Liu and Wu, 2023; Fatemi et al., 2024; Ge et al., 2024; Guo et al., 2023; Dai et al., 2024b; Perozzi et al., 2024; Tang et al., 2025). Zhang et al. (2024) showed poor performance on out-of-domain tasks and that performance on synthetic data does not generalize to real-world data.

Various things have been attributed to this poor performance. Fatemi et al. (2024) demonstrated that the way the graph is encoded in natural lan-

guage for the LLM has a large impact on performance. Ge et al. (2024) found that this can be alleviated by pre-processing using some deterministic ordering such as depth- or breadth-first-search. Yuan et al. (2024) found similar results with a random ordering of the graphs and showed that sorting can help. (i.e., that order matters, Appx. C.9)

Dai et al. (2024b) showed how task difficulty does not just scale with graph size but also the topology of graphs being evaluated. **The path-star task is a powerful example of this, where the type of graph makes it very difficult even at small sizes, however, this isn't an inherent property of the topology but a pathological relation between topology and training method.** They also found that LLMs may apply different algorithms to various tasks and that this is sensitive to input, indicating that the LLM may be using shortcuts. Other works have also identified spurious correlations as an issue (Wang et al., 2023b, see Appx. C.3). Fatemi et al. (2024) evaluated the performance of LLMs on various graph tasks on star-shaped graphs. They found that a) the topology of graph strongly affects performance, and, b) LLMs generally do better on star-shape graphs than other types of graphs. Hallucinations have also been found to be an issue that relates to model scale and graph scale (Tang et al., 2025).

Various methods have been proposed to improve graphical reasoning: graph-specific zero-shot CoT prompts (Wang et al., 2023b, described in Appx. C.1), alternative algorithmic prompts (Dai et al., 2024b), self-prompting (Guo et al., 2023), soft-prompting (Perozzi et al., 2024), instruction-tuning (Chen et al., 2024b; Wang et al., 2024a) and instruction-tuning in conjunction with masking (Luo et al., 2024, see Appx. C.4), preference alignment, (Zhang et al., 2024; Wang et al., 2024a; Chen et al., 2024b), and re-framing the task as code for code-aware LLMs (Zhang et al., 2024; Wu et al., 2024a), which has been shown to help for other reasoning tasks (Madaan et al., 2022).

Another proposed method is to modify the underlying neural architecture by incorporating graph neural nets into the LLM (Scarselli et al., 2009; Tang et al., 2024; Chai et al., 2023; Wu et al., 2024c; Ren et al., 2024; Jin et al., 2024). **Given adulterated supervision, the CHC prevents learning about multi-edge relations which require considering more than two nodes at once. This is partially caused by the attention mechanism of the transformer which is limited to pair-wise itera-**

**tions. Thus modifications that consider triplet interactions may also be useful for graphical tasks (Hussain et al., 2024).**

As we use synthetic data, we consider this in more detail. Wang et al. (2023b) introduced the NLGraph benchmark which contains 8 graph-based tasks with 29,370 examples, partitioned into three difficulties. They stated that they ‘employ a general-purpose random graph generator to generate base graphs while using the number of nodes and graph destiny to control for complexity’.<sup>27</sup> **Random graph construction is complex** and one generation process may lack diversity. As such, Fatemi et al. (2024) used seven generation process, including Erdős–Rényi (Erdős and Rényi, 1959), scale-free networks (Barabási and Albert, 1999), Barabási-Albert (Albert and Barabási, 2002), and stochastic block model (Holland et al., 1983), and star-shaped graphs. **Random tree construction is also complex and we do a poor job of generating trees that would better support the task. However, we believe this is best left to future work which considers search on general graph structures.**

Out-of-domain evaluation has also been considered. Luo et al. (2024) introduced GraphInstruct, which contains 21 graph-based tasks with 4 tasks being reserved as out-of-domain tasks that are not included in fine-tuning. Each in-domain task has 800 training examples. They used three different graph generation processes. Zhang et al. (2024) introduced NLGIFT, which included out-of-domain testing. This includes an experimental setup for fine-tuning on synthetic data and testing on real-world data. They used two different graph generation processes for the syntactic data. It has been shown that graph construction has a large impact on learnability (Saparov et al., 2025, see Appx. C.2).

**We believe our work has several implications for graph benchmarks of LLMs.** These works and ours have different goals and hence different research questions; these they are asking ‘*how well do pretrained LLMs perform on a suit of graphical tasks?*’ and then often with the secondary questions ‘*why do they struggle to perform well?*’ and ‘*how can we improve performance post-hoc?*’, whereas we are asking ‘*why is learning graphical tasks hard?*’.<sup>28</sup> The former concerns performance while the latter is a question of learnability. From

<sup>27</sup>This process was Erdős–Rényi (Fatemi et al., 2024).

<sup>28</sup>This is under the assumption that the path-star task is a minimal example of search on graphs, however, as we found, task-specific issues contribute to its difficulty.

these stems the question: *can the poor performance of LLMs on graphs be attributable to the same difficulties that hinder learning the path-star task?* **As mentioned above, we believe that adulteration will have a small impact on LLMs. However, the issues we present will become more applicable as people move to pretraining LLMs for reasoning tasks. These issues may also affect finetuning. Thus our work motivates the careful design of graph tasks when training or finetuning models.** We leave it to future work to see if our methods can be used to improve the performance of LLMs.

Because these works concern evaluating LLMs, they used small datasets. Frydenlund (2024) found that randomly sampled graphs can easily lead to spurious correlations due to the size of the sample space. We solved this using an online dataset. However, such a solution will be less useful for LLMs which are generally not trained on multiple epochs. Regardless of this, **we strongly urge the move to online datasets, which, for synthetic datasets, should be as easy as exposing the original data generation process. This should be done during both training and evaluation, where data contamination is and will become a bigger issue for evaluating LLMs (Zhu et al., 2024).**

## C.2 Learnability of Graphs

Unlike the above works evaluating LLMs, we are concerned with the learnability of graph algorithms on decoder-only transformer language models. Saparov et al. (2025) is the most closely related work (outside of Bachmann and Nagarajan (2024)). They consider finding the shortest path given a graph. As with our experimental setup, they provide a query with a start and end node, and the graph is encoded as a list of shuffled edges. The graph is also randomly generated and semanticless.

Their first finding is that graph topology highly affects performance (especially in out-of-domain evaluation across topologies). This was also observed by Dai et al. (2024b). They find that a ‘balanced’ graph topology works the best. These are graphs sampled from a generative process which creates a graph with a uniform distribution over the number of ‘lookaheads’ (path length) required to solve the task. Note that, while we discussed these as ‘more general graphs’ in the main text, they are very closely related to path-star graphs.

As our work was nearly completed before we became aware of their work, we do not do direct comparisons. There are several differences: 1) Most of

their experiments use encoder-only models. They did not evaluate path-star graphs using decoder-only models (only using encoder-only models, as in [Frydenlund \(2024\)](#)). 2) They employed a slight architecture modifications that concatenates the token and position embeddings. 3) They used rotary positional embeddings in their decoder-only experiments (again, only on balanced graphs). 4) They used an approximate second-order optimizer, Sophia (where we used Adam). 5) Their best models were also trained for 883M samples (where we used 100M). We suspect that all of these may contribute to differences in performance.

**However, even given these differences, we observe similar scalability issues (and these may have increased scientific value as they were observed independently).** We both find that, as graph size increases, trials become less likely to converge, i.e., successfully learn the task to high sequence accuracy. We also both find that there is a high variability in this convergence across seeds. This is (implicitly) shown in our tables where we show that many trials are unsuccessful but are still learn above the baseline (ABB > SR). [Saparov et al. \(2025\)](#) reported these results in their Fig. 6, which shows the fraction of converged seeds on graphs of various sizes. This shows a less than 20% convergence rate for balanced graphs when  $|V| > 40$ .<sup>29</sup>

Finally, they also show that using depth-first-search or section-inference scratchpads which explicitly decompose the task into intermediate steps does not solve these scaling issues. This leads them to conclude that transformers struggle to learn to search over graphs as the size of the graph grows.

[Khona et al. \(2024\)](#) studied the behavioral difference of 2-layer LM on graph tasks with and without in-context examples in order to explicitly limit the model to only reason via in-context learning. They demonstrated a performance gap between the two as well as showed that in-context exemplars allow for compositional generalization on OOD data but this does not apply to length generalization. [Cohen et al. \(2025\)](#) demonstrated that 2-layer decoder-only models can learn shortest-path representations on small graphs where the learnt embeddings correlate with the spectral decomposition of the graph.

[Wu et al. \(2024c\)](#) considered if learning graph tasks leads to improved planning abilities. They put forth a related argument to the one given by [Bach-](#)

[mann and Nagarajan \(2024\)](#) that that next-token prediction is potentially problematic for learning planning graph tasks due to learning spurious correlations. We do not fully appreciate the pertinence of their Theorem 2 to support a broader insufficiency claim, which we feel is being implicitly made. In particular, they assume that the next-token logits are determined by the target and the current node, however, logits given by real models are formed as a function of the entire graph. Their Example 1. seems to be empirically contradicted by [Saparov et al. \(2025\)](#) and our work. Indeed, when the CHC causes the logits to become a function of only the current node, they converge to be  $1/D$ . As far as we can tell, there is no empirical investigation into LM’s performance being impeded by these specific conjectured spurious correlations.

### C.3 Spurious Correlations and Shortcuts

Spurious correlations in LLMs generally concern OOD performance along with related topics like adversarial attacks and fairness ([Geirhos et al., 2020](#); [Du et al., 2023](#); [Song et al., 2024](#); [Zhou et al., 2024c](#); [Steinmann et al., 2024](#)). [Steinmann et al. \(2024\)](#) provided a literature review and taxonomy of shortcut learning where they define a shortcut as ‘when a model used a spurious correlation as the basis for its decision making’. They also considered why models learn shortcuts and considered that one reason is that ‘a model’s task is generally not precisely defined’ while citing [Bachmann and Nagarajan \(2024\)](#) (and hence commenting on the path-star task). They then followed this with ‘The broad task definitions do not specify how the task should be solved, thus enabling the model to rely on shortcuts rather than relevant features’. **This statement is consistent with our description that the original task setup supports learning two different tasks: the desired path-star task and the undesired edge-following task. What is also interesting about the path-star task is that the features used in learning the shortcut are not irrelevant features but rather relevant features used in the wrong way.**

[Wang et al. \(2023b\)](#) showed that spurious correlations affect the performance of LLMs on graph tasks. In particular, they design two special types of graphs; a ‘chain’ which is just a very long path and a ‘clique’ which has a high edge density. They found that LLMs fail to solve a connectivity task at high rates on these graphs compared to other general graphs. This implies that the underlying al-

<sup>29</sup>Note that the accuracies reported in their Fig. 2 used a best model and not are the average rates over all trials.

gorithm is not learnt (or being applied consistently across different graph types) and thus a shortcut is being employed. Jiang et al. (2024a); Mirzadeh et al. (2025) showed similar results for reasoning tasks where they argue that the model is learning in-domain spurious correlations and thus only learning a superficial pattern matching instead of true reasoning. Press et al. (2023) showed that LLMs can often correctly solve multi-hop subtasks without getting the overall or final answer correct, which they attribute to fact memorization which can be considered as a spurious correlation or undesired shortcut.

Addition is a surprisingly hard task for LMs due to the left-to-right ordering of next-token prediction not matching the order of addition carry-overs, thus requiring that models plan  $n$ -digits ahead. Baeumel et al. (2025) showed how LLMs use a single-lookahead shortcut to perform integer addition (for three-digit numbers). They demonstrated that this shortcut works well – but not perfectly – for two operands, but fails as the number of operands increases. Lin et al. (2025b) showed that LLMs use shortcuts for implicit math reasoning and that, while these work well in-domain, they often fail to solve out-of-domain reasoning tasks.

Liu et al. (2023) demonstrated that automata on sequenced of length  $T$  can be simulated with transformers of  $\log(T)$ -depth via algorithmic ‘shortcuts’ and that these are not robust to OOD data (so being true shortcuts in the above sense).

**The path-star task is unique in that the induced shortcut failure is in-domain where the shortcut actually absorbs supervision and so prevents learning the primary task instead of just compromising performance OOD.** Frydenlund (2024) identified spurious correlation in the original experimental set-up of Bachmann and Nagarajan (2024). This was partially resolved with structured samples. We fully resolve the issue by using an online dataset. We believe that the learnt shortcuts induced by the path-star task are not shortcuts that will appear in natural language – or at least affect the task so potentially as they do symbolic tasks (Tu et al., 2020; Zhou et al., 2024c).

#### C.4 Masking

Masking is often done to avoid spurious correlations and overfitting. Masks can be crafted or structured depending on the task via inductive biases that mask specifically linked tokens. Deng et al. (2021) used constructed query-evidence data

pairs and a masked spanning objective that masks parts of the query that are supported by evidence, thus inducing the model to learn a connection between the evidence and the query. Span selection is also needed in other ways of supervised training of reasoning tasks (Stacey et al., 2022). Rabe et al. (2021) used masking for math reasoning by masking specific sub-expressions. This used an inductive bias which masks all occurrences of such sub-expression (thus the mask is structured with the task). Chen et al. (2024a) showed that masking tokens within the CoT improved their effectiveness for fine-tuning and that the placement of the masking is important.

Luo et al. (2024) used masking over the fine-tuning instructions for graph-based tasks. These were selected by choosing ‘unimportant’ words and, hence, employed an inductive bias for selecting the masks. Given that it only masked unimportant words, we suspect this did not mask graph information and so would not prevent adulteration.

#### C.5 Expressivity and Learnability

Various works have considered the computational limits or expressivity of transformers, i.e., ‘can a transformer actually solve this problem’ (given a particular capacity in terms of hidden-state size or number of layers etc.) (Yun et al., 2020). Various computational models are used to prove expressibility, such as formal logic (Merrill and Sabharwal, 2023), formal languages (Hao et al., 2022; Strobl et al., 2024b), massively parallel computation (Sanford et al., 2024b), or declarative programming languages such as RASP (Restricted Access Sequence Programming)(Weiss et al., 2021).

Weiss et al. (2021) demonstrated that RASP programs upper-bound the difficulty/complexity of a task (for a transformer) in terms of the number of required layers (and attention heads) required to solve the task. It employs a limited computational model of transformers that are restricted to performing uniform attention over a subset of queries (i.e., average-hard attention (Strobl et al., 2024b)).<sup>30</sup> While this excludes RASP’s use on numerical tasks, it does make it easy to model symbolic tasks such as path-star. Zhou et al. (2024a) extended RASP to causal attention and conjecture that short RASP programs lead to length-generalizability.<sup>31</sup> Huang

<sup>30</sup>See Yang et al. (2024) who consider when soft attention can simulate various kinds of hard attention.

<sup>31</sup>They also wrote RASP in Numpy, making it an easy tool for NLP/ML practitioners.

et al. (2025) formalized this conjecture, showing why certain problems have poor length generalization while also showing that a certain class of tasks have guaranteed length generalization. Strobl et al. (2024a) extended RASP to model transformers as transducers, which requires accounting for non-length preserving transitions. RASP programs can be compiled into actual transformers and the reverse (Friedman et al., 2023; Lindner et al., 2023).

Transformer can not learn distributions for next-token prediction for some regular and context-sensitive languages and so expressibility does not match the Chomsky hierarchy (Strobl et al., 2024b; Hu et al., 2025b). The expressibility of RNNs/state space models and transformers is different (Sanford et al., 2024b; Bhattamishra et al., 2024; Sanford et al., 2023; Jelassi et al., 2024). Thus RNNs/Mamba and transformers may not behave the same on the path-star task.

de Luca and Fountoulakis (2024) showed that looped transformers can express various graph algorithms with a constant number of layers. They used a modified transformer architecture which allows for encoding a graph as an adjacency matrix, with a special attention mechanism over this matrix. They made significant note of the need to limit numerical errors through various methods like using hard attention and careful choice of positional embeddings (see Appx. C.7). Sanford et al. (2024a) developed a representational hierarchy of problem classes for transformers on graph problems. **Path-star falls under the ‘parallelizable tasks’ class, in particular, those solvable with logarithmic depth. Frydenlund (2024) showed that transformers can express the path-star task via RASP for encoder- and decoder-only models.**

Expressibility is not to be confused with learnability, i.e., ‘can standard learning methods be used to train a transformer to solve this problem’ (Allen-Zhu and Li, 2023; Deletang et al., 2023; Sanford et al., 2023, 2024b).<sup>32</sup>

Going back to RASP, Zhou et al. (2024a) modified RASP to better model numerical representation and align RASP with empirical results about learnability. This included only allowing single increment indexing. Chang and Bisk (2025) pointed out that transformers fail to count inductively and as

<sup>32</sup>See Svete and Cotterell (2024) and Svete et al. (2024) for a case study, where the former considered the expressibility of transformers to model  $N$ -gram language models, and develop various computational models either using  $N-1$  layers or  $N-1$  attention heads in combination with hard/sparse attention, while the later then considered the learnability of such models.

such, such abilities should not be inherent abilities in the computational model. This was also an argument stemming from empirical learnability results. This demonstrates an inherent divide between the models used for expressibility and learnability.

**Learnability is the core question of this work, i.e., can decoder-only transformers learn the path-star task?** We show this empirically as well as provide theoretical explanations for why adulteration or lack of decomposition causes the task to be unlearnable.

de Luca and Fountoulakis (2024) also considered a small number of learnability experiments using the CLRS dataset. Here they train on 16 node graphs and evaluate on 64 node graphs as a form of length generalization (see Appx. C.7). They highlighted how learnability is much more difficult than expressibility where ‘despite demonstrating the existence of parameters capable of [graph] simulation, discovering them through gradient-based training is challenging.’

## C.6 Sensitivity

A specific and highly relevant case of transformer expressivity and learnability is for parity due to it being a (maximally) sensitive function (Hahn et al., 2021; Bhattamishra et al., 2023; Hahn and Rofin, 2024).<sup>33</sup> The sensitivity of a discrete function on an input sequence  $x$  describes the number of disjoint subsets of  $x$  which, when changed, cause changes to the output. Thus functions with low sensitivity contain redundant information across  $x$ , whereas functions with high sensitivity have tokens that isolate important information. The path-star task completely changes its output based on a single target token provided in the query. Another view of sensitivity is as an analog to the smoothness of continuous functions, where path-star is not smooth with respect to a change in target.

Chiang and Cholak (2022) showed that small model details (layer normalization) can have a big impact on the empirical results of learning sensitive functions. Hahn and Rofin (2024) described the interaction of cross-entropy training with transformers on sensitive functions and found that these transformers inhabit only a small volume of parameter space. Vasudeva et al. (2025) considered the sensitivity of non-boolean functions and found that lower sensitivity correlates with better robustness and flatter minima in the loss landscape.

<sup>33</sup>Again, see Hu et al. (2025a), for a connection between parity and the path-star task.

Sensitivity issues can also appear in more complex NLP tasks (Hahn et al., 2021; Chen et al., 2023b; Chakraborty et al., 2023; Lu et al., 2024; Vasudeva et al., 2025) as well as reasoning tasks, where small changes to the task input can cause large variances in reasoning abilities (Shi et al., 2023; Jiang et al., 2024a; Mirzadeh et al., 2025).<sup>34</sup> Such sensitivity issues can often be attributed to learning spurious correlations or shortcuts.

### C.7 Length Generalization, Task Decomposition, and Scratchpads

The effect of task decomposition on learnability has been studied (Wies et al., 2023; Dziri et al., 2023; Abbe et al., 2024b). This is often studied in the context of length generalization. This is a specific kind of OOD generalization of great importance to LMs due to their sequential nature (Anil et al., 2022; Zhou et al., 2024a,b). Length generalization relates to reasoning tasks that scale to the number of required reasoning steps (hops) (Dziri et al., 2023; Abbe et al., 2024a; Xiao and Liu, 2024, 2025; Mirzadeh et al., 2025). One of the main concerns about the difficulty for transformers to learn parity is that, when they do learn the task, this does not generalize to unseen sequence lengths (Bhatamishra et al., 2020; Hahn and Rofin, 2024). This betrays the fact that the underlying algorithm has not been learnt by the model, i.e., “the failure in length generalization corroborates the models’ fundamental limitation that they may not genuinely understand the task solving algorithm but may rely on short-cut learning that is only applicable to sequences of trained length” (Cho et al., 2024b). In addition to parity, integer addition is often used as a test-bed for length generalization (McLeish et al., 2024; Cho et al., 2024b). Chang and Bisk (2025) showed that transformers show poor OOD performance for the simple task of counting (including task variants). Deletang et al. (2023) showed similar results for a series of more challenging tasks based on formal languages grouped within the Chomsky hierarchy.

Naively applying LMs to these tasks results in poor length generalization. This motivated the use of scratchpads (Nye et al., 2022) which are necessary to (efficiently) solve parity with transformers (Wies et al., 2023; Hahn and Rofin, 2024; Abbe et al., 2024b; Kim and Suzuki, 2025). Wies et al.

<sup>34</sup>Often sensitivity is not formally defined in these tasks compared to parity. This is due to the inherent difficulty of formal definitions of sensitivity for complex tasks.

(2023) showed that there is a large gap in learnability between RNN models that are given scratchpad supervision and those that are not. This class of tasks includes parity. Kim and Suzuki (2025) established similar results for transformer models. They are also necessary for arithmetic (Kazemnejad et al., 2023; Cho et al., 2024b). Scratchpads/CoT are also used for other reasoning tasks (Shah et al., 2024).

**The reason why scratchpads are critical is because they provide extra computation via autoregressive generation in conjunction with extra training supervision, allowing tasks to be decomposed into subtasks via intermediate supervision (Wies et al., 2023).**<sup>35</sup> This can also be framed as a simplification of next-token prediction tasks (Zhou et al., 2024a). Dziri et al. (2023) examined the ability of LLMs to decompose tasks into subtasks via framing tasks as computational graphs. This allowed them to quantize task difficulty/complexity. They showed that OOD performance is poor, attributing such behaviour to learnt shortcuts and that, while scratchpads help, they may not for highly difficult tasks. Abbe et al. (2024b) conjectured that (set-sized) transformers can weakly solve problems that only require ‘local’ information (a small subset of input tokens) but that a ‘locality barrier’ exists which prevents solving problems that require global information. They then showed how to overcome this via scratchpads.

**Like the path-star task, integer arithmetic is a simple task with a simple underlying algorithm that LMs fail on. Also, a trivial reverse solution exists.** Zhou et al. (2024b) shows that the reverse solution to arithmetic is more robust in terms of length generalization. **Note that both these reverse solutions work because they do not need to think multiple steps ahead. For the path-star task, this betrays a lack of reasoning, however, for addition, this conforms with how humans perform arithmetic and thus feels like a more valid solution despite not requiring multi-step reasoning.** The more interesting question is if models can learn to find the trivial order (which we find does not happen for the BoW experiments). See Appx. C.9 for order considerations.

Scratchpads require supervised targets. To avoid this, *thinking tokens* have been introduced which

<sup>35</sup>Note that the extra computation increases expressibility (Feng et al., 2023; Merrill and Sabharwal, 2024; Li et al., 2024c), while the intermediate supervision increases learnability.

are special tokens inserted as input at specified times without any corresponding targets (Herel and Mikolov, 2023; Goyal et al., 2024). This increases the available sequential computation – and hence expressibility. Note this also may affect learnability just via the ability to learn different functions which require additional computation. Pfau et al. (2024) showed that increasing the expressive computation capability using thinking tokens does not mean it is easy to learn to use this capacity.

Yin et al. (2024) tried thinking tokens for the path-star problem to negative results.<sup>36</sup> This is interesting because using  $M$  thinking tokens can, in theory, provide a trivial solution by computing the reverse arm with the thinking token and then the forward arm from the reverse solution (similar to the BoW experiments, just with even less supervision). **We conjecture that thinking tokens do not work for the path-star task as they do not provide additional decomposition supervision.**<sup>37</sup>

### C.8 Positional Embeddings

With length generalization comes the need to have positional embeddings that allow for exact matching across long lengths and generalize to unseen positions (Kiyono et al., 2021; Kazemnejad et al., 2023; Ruoss et al., 2023; Li et al., 2024a; McLeish et al., 2024). Chang and Bisk (2025) showed that different embedding types generalize differently to different counting tasks. **Such methods will be important considerations for graph-based tasks when scaling up the size of graphs and considering length generalization.** This can also lead to some unexpected results like using no positional embeddings (NoPE) being possible for decoder-only models (Irie et al., 2019; Tsai et al., 2019; Haviv et al., 2022; Chi et al., 2023; Kazemnejad et al., 2023; Wang et al., 2024b; Irie, 2024; Zuo et al., 2025) and can lead to better length generalization for symbolic reasoning tasks (Kazemnejad et al., 2023). However, Wang et al. (2024b) showed that NoPE fails to generalize due to a collapse in the attention head distribution as the context size increases. Yang et al. (2025) followed this up with a hybrid strategy that combines NoPE, for its strong token retrieval and RoPE for its inductive biases. Frydenlund (2024) found that the choice of posi-

tional embedding mattered for the path-star task and that NoPE worked when using decoder-only models.

Modifying how the task is represented can improve the behaviour of the positional embeddings. For example, McLeish et al. (2024); Cho et al. (2024a,b) coupled or reused positional embeddings at similar positions for both operands for the task of addition, leading to better length generalization. This is an example of symbolic tasks being brittle to how the task is represented and requiring a strong task-specific inductive bias to overcome.

### C.9 Order Matters and Reversal Curse

Prior work has explored the impact of ordering source-side information in LLMs for premise order on the complex tasks of reasoning (Wang et al., 2023a; Chen et al., 2024c; Allen-Zhu and Li, 2024; Shah et al., 2024) and proof generation (An et al., 2024). Liu et al. (2024b) has shown that LLMs struggle to retrieve relevant information in long contexts when that information is placed in the middle of the context compared to either the front or back. Frydenlund (2024) showed that **order matters for the path-star where they considered that the query should proceed the graph.**

Order matters on the target-side as well, since the path-star task becomes trivial when asked to generate the arm in reverse order. An asymmetry in LM predictive abilities coined the *reversal curse* is a recent but well-studied phenomenon (Berglund et al., 2024; Lin et al., 2024). An example of this is for an LM to be able to predict ‘A is B’ but not ‘B is A’. A common proposed solution is bidirectional training, incorporating bidirectional information, or a bidirectional model modification (Ma et al., 2023; Golovneva et al., 2024; Lv et al., 2024; Guo et al., 2024b,a). This is also the underlying idea of the Belief-State Transformer introduced by Hu et al. (2025a) for solving the path-star task.

Papadopoulos et al. (2024) discovered an asymmetry in perplexity between models trained either in the forward or reverse direction, with the forward having consistently lower perplexity. This is surprising given that both directions give a valid and theoretically equivalent factorization of the sequential probability.

Chen et al. (2023c); Fang et al. (2025) consider order invariance for few-shot in-context learning. The issue here is that the order of the exemplars should not matter. This requires considering how the attention or model is parameterized as well

<sup>36</sup>Note these are done under the original settings that allow for spurious correlations and overfitting, and thus may have failed due to other reasons.

<sup>37</sup>However, these experiments need to be redone for confirmation since they did not use an online dataset.

as the positional embeddings used. Fang et al. (2025) also considered that fully observed question-answer pairs lead to data leakage and shortcuts. **This ‘leakage’ can be framed as adulterated supervision. Order invariance is also very important for graphs represented as lists of edges since the order of this list should not matter. However, it will matter with decoder-only models due to the causal constraint.**

### C.10 Non-AR, Iterative-AR, and Discrete Diffusion Models

Given the perceived belief that left-to-right autoregressive models were incapable of solving the path-star task, a natural conclusion would be to use non-autoregressive models (NAR) (Gu et al., 2018; Gu and Kong, 2021) or iterative autoregressive models (IAR) (Lee et al., 2018; Ghazvininejad et al., 2019). There are two core aspects of NAR/IAR models. The first is that they use an any-order model parameterization which forgoes enforcing the causal constraint (achieved by not employing a causal mask in the attention mechanism). The second is that these are trained using a masking loss (MLM) (Devlin et al., 2019). In the NAR case, the targets are fully masked. This means each target token is modeled independently (at the classification layer) and all tokens are decoded in a single step during inference. This can lead to poor performance, motivating the use of IAR models trained using partial masks, thus allowing for partial dependencies. This allows for multiple decoding steps during inference. Both these aspects come together to allow the model to generate in any-order.

Bachmann and Nagarajan (2024) used a ‘teacher-less’ model which masks out all input tokens. Frydenlund (2024) connected this model to NAR models and also showed that the path-star task was solvable via an encoder-only model with NAR and IAR training. This was based on a modified version of the CMLM model (where the conditional ‘C’ part of the model is removed) (Ghazvininejad et al., 2019). Frydenlund (2024) incorrectly implied that the original ‘teacher-less’ model was non-causal when considering it as a NAR model. The model described by Monea et al. (2023) is meant to modify an autoregressive model post-hoc and thus is designed to keep the causal constant. However, in terms of independently modeling and predicting multiple tokens it behaves exactly like a NAR model (i.e., loss, training, and inference procedure).

**As we know the reverse ‘solution’ works, the any-order aspect of the NAR/IAR models potentially allows these models to learn the reverse solution without direct supervision. Our results show that the masking operation allows for task decomposition for the path-star task. We expect that this is the more important aspect of these model’s successes over the model’s parametrization, however, we also believe that parameterizations will matter due to the causal constraint making graph reconstruction more difficult.**

The connection between IAR models and discrete diffusion models was described by Austin et al. (2021). Kitouni et al. (2024) introduced an ‘MLM- $U$ ’ diffusion model which uses a uniform masking rate and applied it to the path-star task. They wrote ‘this approach can be implemented as a denoising process which recovers randomly masked tokens, like BERT, but with uniformly sampled masking rates. This key difference allows training a generative model with masked modeling.’ This key insight was first described by Ghazvininejad et al. (2019) with their IAR CMLM model. As mentioned, CLMC was used by Frydenlund (2024), however, the path-star experiments in Kitouni et al. (2024) were not described in enough detail to do a comparison with Frydenlund (2024).

Different masking strategies have been used; Lee et al. (2018) used token replacement from  $V$  while Ghazvininejad et al. (2019) used a special masked token. Other works have explored any-order LM parameterization outside of the NAR/IAR/diffusion framework (Yang et al., 2019; Liao et al., 2020)

### C.11 Future Token Prediction

Early works in future prediction designed models which could predict  $N$  tokens into the future by creating  $N$  separate hidden-states and training on each state with cross-entropy against a single future token (Goodman et al., 2020; Qi et al., 2020). Thus these are not truly belief-states directly, however, a belief-state must be present in the model in order to generate the  $N$  separate hidden-states. The general goal of this was for improved training by explicitly learning to predict future tokens and hence plan for future tokens, and was not for multi-token inference. Heo et al. (2024) also used multi-state prediction for future n-grams but also introduced a method to explicitly create representations that are compositional into the future. Gloeckle et al. (2024) proposed an efficient training method for  $N$  token prediction which repurposed  $N$  multi-head



attention to create the  $N$  hidden-states.<sup>38</sup> While they did consider how to use this to increase inference speed, their main goal was to demonstrate that training with multi-token prediction increases performance on downstream tasks even if this ability is removed during inference. Because of the success of training with multi-token prediction, it has as been adopted into foundational models such as DeepSeek3 (Liu et al., 2024a), which used sequential prediction method which requires introducing  $N$  multiple parameterized modules for  $N$  future tokens.

Gerontopoulos et al. (2025) also introduced a method for training a model with multi-token prediction which includes extra *register* tokens into the input sequence during training. These extra tokens induce creating extra hidden-states from which future tokens can be predicted. Thus, instead of modifying the model’s architecture by repurposing heads to create extra hidden-state, this simply modifies the input sequence. Since they only use this for training, they also prevent these register tokens from being attended to by regular tokens, which would result in a inference-time bias. They showed the effectiveness of this method compared to Gloeckle et al. (2024).<sup>39</sup> **Their method is similar to our RITF method, except whereas they make future predictions from extra hidden-states created by modifying the input to the model, we instead reuse hidden-states for multiple future predictions via using a structured loss function. This helps highlight the connection between inputs and targets for multi-token prediction.**

Cai et al. (2024) expanded on the multi-head method for multi-token prediction by incorporating it with speculative decoding faster inference (Xia et al., 2022; Chen et al., 2023a; Leviathan et al., 2023).<sup>40</sup> The parameterized modules of DeepSeek3 can also be retrained during inference and used for speculative decoding (Liu et al.,

2024a).

Pal et al. (2023) studied the extent to which the hidden-states of LMs contain predictive information about future tokens and hence act as belief states. They used *lens* to show that the hidden-states of models trained solely to predict the next token contain enough to predict up to three tokens into the future between 20-40% of the type (where, the type of lens and prompting method used had a large effect on the predictive ability (Hewitt and Manning, 2019; nostalgebraist, 2020; Belrose et al., 2023; Yom Din et al., 2024)). Men et al. (2024) also investigates the existence of belief-states in LLMs specifically for planning tasks.

Wu et al. (2024b) studied the mechanism for LMs to learn future information from a next-token prediction objective. They hypothesized that it could be due to two mechanisms; a deliberate *pre-caching* mechanism which computes features earlier than they are needed and an unintentional *breadcrumb* mechanism which considers that a LM learns features for predicting the next token and that these just happen to also be good features for predicting future tokens also. They construct a synthetic dataset and show that pre-caching is done and necessary for some planning tasks. However, they also show that pre-caching is less noticeable in a GPT2 model used for natural language (but also consider this might be less true as LMs scale up). Notably this mechanism will not help for the path-star task as future predictions can ignore both pre-caching and breadcrumb features due to the CHC. That is, any features used for planning will just be ignored. This also means that there will be no learning signal to reinforce learning such features.

**We design future distributions and associated losses to enhance this ability for the path-star task. Not only do these create an explicit belief-state that allows for planning, they also avoid adulteration by targeting tokens that require multiple edges or path-reconstruction to predict.** RITF is also designed to be efficient during train as it only requires a single parallelizable loss. This is in contrast to other losses on  $N$  tokens into the future, which scale linearly with  $N$  (Goodman et al., 2020; Qi et al., 2020; Gloeckle et al., 2024).

<sup>38</sup>Qian et al. (2025) considered the use of multi-token prediction to alleviate prompt sensitivity for LLMs. Interestingly, they mentioned that the four future tokens used by Gloeckle et al. (2024) are insufficient for their task. **This highlights a potential benefit of RITF, which may be able to scale to more distant future tokens since this is decoupled from any model parameterization.**

<sup>39</sup>They also show their method works on the path-star task using a pretrained GPT2 model.

<sup>40</sup>As an aside, speculative decoding was also the original purpose of the ‘teacher-less’ model (Monea et al., 2023) and, as with NAR/IAR, the main motivating factor for speculative decoding is improved inference speed.