BEYOND TRAINING LENGTHS: CONDITIONS FOR GENERALIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

A key challenge in learning-to-reason tasks is that models trained on shorter sequences often fail to handle longer instances at test time. This phenomenon, known as *length generalization* (LG), requires extrapolation beyond the training length, whereas most learning paradigms are designed for interpolation under the i.i.d. assumption. For this reason, LG is also referred to as *length extrapolation*. Despite substantial empirical progress in recent years, LG remains largely unresolved. While prior theoretical work has examined learning to reason, relatively little attention has been paid to reasoning under LG. In this paper, we study *necessary* and *sufficient* conditions for achieving LG in a manner that is independent of specific learning algorithms or model architectures. This perspective is important, as it avoids limiting the analysis to the capabilities and constraints of existing learning algorithms. Finally, we analyze two empirically successful LG approaches based on the proposed necessary and sufficient conditions, providing initial evidence for the relevance of our theoretical framework.

1 INTRODUCTION

Despite their impressive performance on complex reasoning tasks, large language models (LLMs) struggle with basic operations such as adding or multiplying large numbers (Nogueira et al., 2021; Qian et al., 2022; He et al., 2024). This is often due to a failure in *length generalization* (**LG**), where a model trained on smaller problem instances (e.g., 5-digit addition) has difficulty to solve larger instances (e.g., addition with more than 5 digits) (Abbe et al., 2023; Anil et al., 2022; Kazemnejad et al., 2023; Zhang et al., 2022; Zhou et al., 2024a; Ben-Kish et al., 2024). The key issue is that the current learning paradigm mainly learns to interpolate. Thus, the learned models struggle when extrapolation is required to solve some problems, which is why LG is also called *length extrapolation*.

Many methods have been proposed to deal with LG (Golowich et al., 2025; Hou et al., 2025; Marsden et al., 2025). One important method is Scratchpad (Nye et al., 2021) or Chain of Thought (CoT) (Wei et al., 2022), where models are trained on data that includes intermediate reasoning steps. For instance, `'2+3×2+1'` may be presented as `'2+3×2+1=2+6+1=8+1=9'` rather than just `'2+3×2+1=9'`. It has been shown that using Scatchpad or CoT can improve LG (Anil et al., 2022; Lee et al., 2023). However, a significant limitation persists: even with detailed CoT training, models still fail to generalize beyond the scope of their training data (Dziri et al., 2023; Kazemnejad et al., 2023). To further improve LG, considerable effort has focused on modifying the positional encoding mechanisms of Transformers (see the survey (Zhao et al., 2024)). However, such modifications alone are insufficient to solve the LG problem, and additional inductive biases are needed (Xiao & Liu, 2025). Perhaps, one of the most effective approaches is the one that uses tags to provide additional position information or bias to the learner (Cho et al., 2025; Xiao & Liu, 2025).

On the theoretical front, it has been shown that with CoT, reasoning is learnable but without it, it is not learnable (Feng et al., 2023; Wies et al., 2023). However, these studies are under a given length L, i.e., both training and testing involves only input sequences of length less than or equal to L. They are not applicable to LG. The work in Golowich et al. (2025) proved that under some assumptions in learning using Transformers, LG occurs as long as each predicted token depends on a small and fixed number of previous tokens. Their model justifies some techniques that modify positional embeddings to improve LG. It was conjectured in (Zhou et al., 2024a) that Transformers can achieve LG on a problem if a short RASP program Weiss et al. (2021) exists for all input lengths—a proposition known

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

as the RASP-L conjecture. The work in (Huang et al., 2025; Ahuja & Mansouri, 2024) provided formal proofs for simplified versions of this conjecture. However, the empirical support remains weak, particularly for problems like parity and addition, which are relatively easy for tagging-based methods in (Cho et al., 2024; Xiao & Liu, 2025). So far conditions for LG is still not well understood.

Xiao & Liu (2025) introduced a sufficient condition for LG, termed (n, r)-*consistency*, which states that if a problem's CoT scheme satisfies the condition, a Transformer can achieve LG. Strong results were reported to support the theoretical claim. Cho et al. (2024; 2025) proposed some problem-specific sufficient conditions on embedding dimensions with regard to decoder-only Transformers for solving certain size addition or $N \times 2$ multiplication problems, respectively (see Sec. 5).

However, these prior works have neither identified a necessary condition for LG nor moved beyond analyses tied to the Transformer architecture. In this paper, we investigate the *necessary* and *sufficient* conditions for achieving LG in a manner that is independent of specific learning algorithms or model architectures. This perspective is important because it avoids restricting the analysis to the capabilities and limitations of existing methods, which remain inadequate for robust LG.

Specifically, we propose a Markovian ***necessary*** condition, requiring that the transition from one step to the next in a chain-of-thought (CoT) process satisfies the *Markov property*. We further show that this condition is also ***functionally sufficient***: if a CoT scheme satisfies it, then a valid solution exists. However, functional sufficiency does not guarantee that a given network architecture can learn such a solution in practice. This highlights the core difficulty of LG: success depends not only on the correctness of the formal objective, but also on the presence of inductive biases that guide learning toward solutions with the required structure.

Since current learning algorithms by themselves remain inadequate for robust LG, we analyze two empirically successful LG approaches, (n, r)-*consistency* (Xiao & Liu, 2025) and *position coupling* (Cho et al., 2025) based on the proposed condition. Both methods introduce additional constraints (i.e., tags) as inductive biases to enable more effective learning with Transformers. Our work thus represents an important step toward a more rigorous theory of LG in reasoning tasks.

2 Related Work

Theoretical studies of learning to reason. Existing literature primarily addresses reasoning within an interpolation framework, which is not applicable to extrapolation inherent to LG. Feng et al. (2023) demonstrated that CoT is effective for problems decomposable into sub-problems and proved that such problems are not directly learnable without it. Wies et al. (2023) proved that, given CoT, a neural network can efficiently learn any function in the complexity class P, but without CoT, certain functions in P cannot be learned by any polynomial-time algorithm. A similar work is also done in Malach (2023). However, this line of research is predicated on a given problem size L and thus do not address LG. Hu et al. (2024) also shows that LLMs perform reasoning in a case-based manner rather than by applying learned, generalizable rules.

Zhou et al. (2024a) conjectured that Transformers tend to learn a length-generalizing solution if there exists a short RASP-L (a faction of the RASP language (Weiss et al., 2021)) program that works for all input lengths. However, this work is mainly empirical. Ahuja & Mansouri (2024) also provided proofs for simplified versions of the conjecture. Huang et al. (2025), Chen et al. (2025a), Izzo et al. (2026) and Yang et al. studied the generalization bounds. No study on conditions for LG were made.

Xiao & Liu (2025) proposes a *sufficient condition* for Transformers to achieve LG, called (n, r)-*consistency*. However, it did not identify an necessary condition. Several theoretical analyses were also done to justify specific LG methods. For example, Golowich et al. (2025) showed that under certain assumptions, LG occurs as long as each predicted token depends on a small and fixed number of previous tokens. Their theoretical model justifies why some specific positional embedding methods, such as position coupling Cho et al. (2024); McLeish et al. (2024), can improve LG. Chen et al. (2025b) showed theoretically that LG is unattainable without appropriate inductive bias. It further showed why CoT and relative position embedding can help improve LG. In a different vein, Marsden et al. (2025) conducted a theoretical study of LG for sequence prediction (e.g., text generation).

Empirical studies on LG. Many methods have been proposed to solve LG using Transformers, primarily by modifying positional encodings (Zhao et al., 2024; Kazemnejad et al., 2023; Zheng et al., 2024). Duan & Shi (2023); Zhou et al. (2024a); He et al. (2024); Hou et al. (2025) also proposed

methods involving bias calibration, data formatting, and novel encodings. Despite these efforts, their approaches still fail to solve multi-digit addition and cannot deal with multi-digit multiplication.

Scatchpad (Nye et al., 2021) and position coupling (Cho et al., 2024; McLeish et al., 2024) were combined for LG in (Cho et al., 2025). Position coupling groups tokens in an input sequence and assigns the tokens in each group a sequence of consecutive integers as position IDs. It enabled Transformers to better solve addition and n-digit $\times$ 2-digit multiplications. However, it requires a manually designed Scratchpad and tailored position coupling for each problem, making it difficult to scale. Sabbaghi et al. (2024) proposed some problem specific data formats and positional encodings for solving addition and multiplication with LG. Jelassi et al. (2023) also found that supplementing training data with a small number of long sequences offered some benefit, but still failed on the multiplication problem. Chi et al. (2023) introduced a novel architecture with weight-sharing and a working memory for regular languages, yet it remained unable to handle arithmetic tasks like multiplication or addition. Zhou et al. (2024b) achieved better results on addition by combining specific data formatting with positional encoding, but the generality of this method across other reasoning problems remains unproven. A review of positional encoding based methods can be found in (Zhao et al., 2024). Butoi et al. (2025) studied using Transoformer to recognize formal languages.

Fan et al. (2025) employed looped transformers for LG. These architectures can dynamically adjust the number of processing steps based on a problem's computational complexity. However, the method was evaluated only on binary versions of arithmetic tasks, not on traditional addition or multiplication.

Different attentions and architectures were proposed in (Nangia & Bowman, 2018; Bowman et al., 2015; Tay et al., 2021; Chowdhury & Caragea, 2023a;b; Opper et al., 2025). Their methods work on various text and list operations but still cannot solve them. LG has also been studied in text generation (Sun et al., 2023; Press et al., 2022; Ruoss et al., 2023; Han et al., 2023; Zheng et al., 2024). Further, Ruiz & Gu (2025) investigated LG in recurrent models. However, this body of work focuses on generic sequence prediction and is distinct from LG in reasoning tasks.

3 Main Results

Since the CoT process underlying a reasoning task can be naturally modeled as a stochastic process, this section begins by formulating the CoT process as a general stochastic process and then introduce a quantitative measure of non-Markovianity. This measure enables us to characterize deviations from the Markov property and, in turn, to deduce a necessary condition for achieving LG.

Denote a stochastic process by $(\mathcal{S}, P)$.[1] Consider a stochastic process $\{S_t\}_{t\in\mathbb{N}}$ with $\mathcal{S} = \{s\}$ as the state space and $S_t \in \mathcal{S}$ as the random variable at time step t. Denote $\Delta(\mathcal{S})$ as the probability simplex of $\mathcal{S}$. A stochastic process has the dynamic function $P : \mathcal{S}^\infty \to \Delta(\mathcal{S})$, $S_t \sim P(S_t \mid S_{t-1}, \ldots, S_0)$. Let $S_t \sim p_t$, $p_t \in \Delta(\mathcal{S})$, represent the probability distribution of S_t. When we write $(p_0, \ldots, p_t)$, we mean the initial distribution is p_0 and the t'th step distribution is p_t.

When a stochastic process is Markov, the dynamic function $P(S_t \mid S_{t-1}, \ldots, S_0) = P(S_t \mid S_{t-1})$, i.e., there exists a transition matrix $T = \{t_{i,j} \mid t_{i,j} \geq 0, \sum_i t_{i,j} = 1\}_{1\leq i,j\leq|\mathcal{S}|}$ s.t. $p_t = Tp_{t-1}$, where we regard p_t as a column vector of length $|\mathcal{S}|$ in matrix calculations. A Markov process is said to be memoryless.

3.1 Non-Markovianity Measure

Now we propose a measure $N(\mathcal{S}, P)$ to quantify how much non-Markovianity the stochastic process $(\mathcal{S}, P)$ is. Denote d be the distance function between the distributions in $\Delta(\mathcal{S})$. Inspired by the Breuer-Laine-Piilo (BLP) measure of quantum processes (Laine et al., 2010), define

$$\dot{d}_t(p, q) = d(p_t, q_t) - d(p_{t-1}, q_{t-1}), \tag{1}$$

$$N_0(p_0, q_0, \mathcal{S}, P) = \sum_{t>0} \dot{d}_t(p, q) \cdot \mathbf{1}_{\dot{d}_t(p,q)\geq 0}, \tag{2}$$

$$N(\mathcal{S}, P) = \max_{p_0, q_0 \in \Delta(\mathcal{S})} N_0(p_0, q_0, \mathcal{S}, P), \tag{3}$$

where $\{p_t\}_{t\in\mathbb{N}}$ and $\{q_t\}_{t\in\mathbb{N}}$ are stochastic processes of $(\mathcal{S}, P)$ with initial distributions p_0 and q_0.

[1] Note that we ignore the filtration and the mapping between the sample space and the state space for brevity.

The intuition is that $\dot{d}_t(p, q)$ measures how much *memory effect* occurs throughout the process. For a Markov process, the distribution p_t (or q_t) depends only on the distribution p_{t-1} (respectively, q_{t-1}), so the distance between p_t and q_t should not increase. If the distance increases, the *memory effect* must occur and current state depends not only on the immediate previous state but also on some older states. Therefore, the summation of the increase of distances is a natural measure of the *memory effect*, i.e. the non-Markovianity.

This intuition holds for the L^1 distance. Let

$$d(p, q) \triangleq ||p - q||_{L^1} = \sum_{1 \le i \le |\mathcal{S}|} |p_i - q_i|. \tag{4}$$

Lemma 3.1. *If $(\mathcal{S}, P)$ is Markov and the distance is L^1, then $\forall\, p_0, q_0 \in \Delta(\mathcal{S})$, $\forall\, t > 0$, $\dot{d}_t(p, q) \le 0$.*

When $(\mathcal{S}, P)$ is a Markov process, Lemma 3.1 shows $N(\mathcal{S}, P) = 0$, so $N(\mathcal{S}, P)$ is a proper measure of non-Markovian. The formal proof is given in Appendix A.

3.2 One-step Dynamic Function

Now we discuss approximating $(\mathcal{S}, P)$ by a one-step dynamic function. A one-step dynamic function is a function that describes how a system's state evolves from the current time step to the next, using only information available at the current step. It thus indicates the Markov property. Note that we do not discuss k-step dynamic function, as it is the same as stacking consecutive k states.

Define $\mathcal{H}$ to be the hypothesis space. Let $f \in \mathcal{H} : \mathcal{S} \to \Delta(\mathcal{S})$ approximates the dynamics of the stochastic process $(\mathcal{S}, P)$. For example, given a process $(S_0, \ldots, S_t)$, f is optimized s.t. $d(f(S_{t-1}), p_t) \to 0$.

Define $\bar{f} : \Delta(\mathcal{S}) \to \Delta(\mathcal{S})$ to be the expected output when the input state is probabilistically sampled,

$$\bar{f}(p) = \int_{s \in \mathcal{S}} f(s) p(s) ds,\ s \in \mathcal{S},\ p \in \Delta(\mathcal{S}). \tag{5}$$

Since $f(s) = \bar{f}(\delta_s)$, where δ_s is the Delta-Dirac distribution on s, with a slight abuse of notations, we write $\bar{f}$ as f. For example, given a process $(p_0, \ldots, p_t)$, f is optimized s.t. $d(f(p_{t-1}), p_t) \to 0$.

We define $\mu : \mathcal{H} \to \mathbb{R}_+$ to measure how far f approximates the ground truth dynamics of $(\mathcal{S}, P)$:

$$\begin{aligned} \mu(f) = \int_{p_0 \in \Delta(\mathcal{S})} \sum_{t>0} d(f(p_{t-1}), p_t) u(p_0) dp_0, \\ p_0 \in \Delta(\mathcal{S}),\ u \in \Delta(\Delta(\mathcal{S})), \end{aligned} \tag{6}$$

where $u \in \Delta(\Delta(\mathcal{S}))$ is a uniform probabilistic measure of $\Delta(\mathcal{S})$. Note that when $\mu(f) = 0$, f **completely represents** the dynamic of $(\mathcal{S}, P)$.

The hypothesis space $\mathcal{H}$ is ***time-independent***, because $f \in \mathcal{H}$ is independent of t,

$$f(p_t) = f(q_{t'}),\ \forall\, t, t' > 0,\ \forall\, p_t = q_{t'} \in \Delta(\mathcal{S}). \tag{7}$$

The ***time-dependency*** is a natural generalization of *time-independency*. Denote the *time-dependent* hypothesis space as $\mathcal{H}^{\mathbb{N}}$. We reuse f, with a slight abuse of notations, to represent the time-dependent one-step dynamic function $f \in \mathcal{H}^{\mathbb{N}} : \mathbb{N} \times \Delta(\mathcal{S}) \to \Delta(\mathcal{S})$ to approximate $p_{t-1} \to p_t$, i.e. $d(f(t, p_{t-1}), p_t) \to 0$. The *time-independency* is a special case of *time-dependency* as

$$\mathcal{H} = \{f \in \mathcal{H}^{\mathbb{N}} | f(t, p) = f(t', p), \forall\, t, t' > 0,\ \forall\, p \in \Delta(\mathcal{S})\}. \tag{8}$$

The measure of the approximation quality is

$$\mu(f) = \int_{p_0 \in \Delta(\mathcal{S})} \sum_{t>0} d(f(t, p_{t-1}), p_t) u(p_0) dp_0, p_0 \in \Delta(\mathcal{S}). \tag{9}$$

We say $f \in \mathcal{H}^{\mathbb{N}}$ is ***time-inconsistent*** if

$$\exists\, t, t' > 0, f(t, \cdot) \neq f(t', \cdot). \tag{10}$$

It's worth noting that $f(t, \cdot) \neq f(t', \cdot)$ happens when (i) Domain$(f(t, \cdot)) \neq$ Domain$(f(t', \cdot))$, or (ii) $\exists\, p_t = q_{t'}$, $f(t, p_t) \neq f(t', q_{t'})$. We say $f \in \mathcal{H}^{\mathbb{N}}$ is *time-consistent* if f is not *time-inconsistent*. Specifically, $\forall\, f \in \mathcal{H}$ is *time-consistent*.

3.3 NECESSITY

This section shows that $N(\mathcal{S}, P) = 0$ is *necessary* for the existence of f s.t. $\mu(f) = 0$, for time-dependent and time-independent hypothesis space.

Theorem 3.2. *For a stochastic process $(\mathcal{S}, P)$, if $N(\mathcal{S}, P) > 0$, then $\forall\ \mathcal{H}^{\mathbb{N}}$ to be compact, $\min_{f \in \mathcal{H}^{\mathbb{N}}} \mu(f) > 0$.*

Theorem 3.2 shows that $N(\mathcal{S}, P) = 0$ is necessary to find $f \in \mathcal{H}^{\mathbb{N}}$ s.t. $\mu(f) = 0$. Let's intuitively prove Theorem 3.2. When $N(\mathcal{S}, P) > 0$, $\exists\ p_0, q_0 \in \Delta(\mathcal{S})$, $\exists\ t > 0$, $d(p_t, q_t) > d(p_{t-1}, q_{t-1})$, which means p_{t-1} and q_{t-1} are similar, but p_t and q_t are quite different. Intuitively, this phenomenon happens because more historical information except for the $(t-1)$'th step is involved in the dynamics for deriving the t'th step, where the *memory effect* occurs. For the most extreme case $p_{t-1} = q_{t-1}$ and $p_t \neq q_t$, it is impossible to define a function f s.t. $f(p_{t-1}) = p_t \neq q_t = f(q_{t-1})$. Therefore, a partial input, the $(t-1)$'th step, is insufficient to recover the dynamics, i.e. $\mu(f) > 0$, $\forall\ f$, and this failure is independent of the choice of the hypothesis space. The formal proof is in Appendix A.

In practice, when $N(\mathcal{S}, P) > 0$, it means that the t'th CoT step contains more information than the $(t-1)$'th step's. So the t'th CoT step cannot be deduced solely from the $(t-1)$'th step. This failure cannot be eliminated through any approximate approaches, because it is an inherent functional problem. Again, stacking consecutive states by letting $\mathcal{S} \leftarrow \mathcal{S}^k$ may help $N(\mathcal{S}, P) \to 0$, if the memory effect only requires information from a finite range of historical steps.

Corollary 3.3. *For a stochastic process $(\mathcal{S}, P)$, if $N(\mathcal{S}, P) > 0$, then $\forall\ \mathcal{H}$, $\min_{f \in \mathcal{H}} \mu(f) > 0$.*

Corollary 3.3 shows that $N(\mathcal{S}, P) = 0$ is necessary to find $f \in \mathcal{H}$ s.t. $\mu(f) = 0$. The *necessity* showed by Theorem 3.2 is strong, as it proves necessity for time-dependent hypothesis space $\mathcal{H}^{\mathbb{N}}$. As a special case, the conclusion for the time-independent hypothesis space $\mathcal{H}$ also holds.

In practice, Corollary 3.3 is usually more useful, though Theorem 3.2 is stronger. Especially when dealing with the length generalization (LG) problem, we expect the hypothesis space to be time-independent, so that the learned dynamic function could be applied to extrapolate lengths.

Summary: $N(\mathcal{S}, P)$ measures non-markovianity. A Markov process $(\mathcal{S}, P)$ satisfies $N(\mathcal{S}, P) = 0$. $N(\mathcal{S}, P) = 0$ is *necessary* for the existence of both time-dependent and time-independent one-step dynamic function. Thus, $N(\mathcal{S}, P) = 0$ is *necessary* for LG that is independent of specific learning algorithms or model architectures.

3.4 SUFFICIENCY

This section shows that $N(\mathcal{S}, P) = 0$ is also *sufficient* for the existence of f s.t. $\mu(f) = 0$ for the time-dependent hypothesis space (Theorem 3.4), which we call *functionally sufficient*. This is the most general case. However, $N(\mathcal{S}, P) = 0$ is *not sufficient* for the existence of f s.t. $\mu(f) = 0$ for the time-independent hypothesis space (Theorem 3.5).

Theorem 3.4. *For a stochastic process $(\mathcal{S}, P)$, if $N(\mathcal{S}, P) = 0$, then $\exists\ \mathcal{H}^{\mathbb{N}}$, $\min_{f \in \mathcal{H}^{\mathbb{N}}} \mu(f) = 0$.*

See Appendix A for the proof. Theorem 3.4 shows that $N(\mathcal{S}, P) = 0$ is sufficient to find $f \in \mathcal{H}^{\mathbb{N}}$ s.t. $\mu(f) = 0$. When $N(\mathcal{S}, P) = 0$, $\forall\ p_0, q_0 \in \Delta(\mathcal{S})$, $\forall\ t > 0$, $d(p_t, q_t) \leq d(p_{t-1}, q_{t-1})$. Intuitively, the t'th step doesn't have more information than $(t-1)$'th step's. Therefore, this is injective and it's feasible to define a function f that maps from the $(t-1)$'th step to the t'th step.

Therefore, we expect $N(\mathcal{S}, P) = 0$ to be sufficient to define time-independent $f \in \mathcal{H}$ s.t. $\mu(f) = 0$. However, this is not correct. Theorem 3.5 states that. The proof is given Appendix A.

Theorem 3.5. *There exists a stochastic process $(\mathcal{S}, P)$, s.t. $N(\mathcal{S}, P) = 0$, but $\forall\ \mathcal{H}$, $\min_{f \in \mathcal{H}} \mu(f) > 0$.*

To explain this theorem, we need to discuss the property of the stochastic process $(\mathcal{S}, P)$ itself, which we have not done so far. For a stochastic process, the dynamics may differ between different time steps. For example, $p_{t-1} \to p_t$ and $q_{t'-1} \to q_{t'}$ may satisfies $p_{t-1} = q_{t'-1}$ but $p_t \neq q_{t'}$, because the dynamics of step t and t' are simply different. This means that $(\mathcal{S}, P)$ can be *time-dependent* or *time-independent* as well, corresponding to those of the one-step dynamic function f, respectively.

Theorem 3.5 shows that $N(\mathcal{S}, P) = 0$ is insufficient for finding $f \in \mathcal{H}$ s.t. $\mu(f) = 0$. The failure is because the dynamic process $(\mathcal{S}, P)$ can be time-dependent, which may introduce *time-inconsistency*.

Summary: We have two important conclusions: **(1)** $N(\mathcal{S}, P) = 0$ is ***sufficient*** for the existence of time-dependent one-step dynamic function, which is the most general case because time-independency is a special case of time-dependency. **(2)** $N(\mathcal{S}, P) = 0$ is also ***sufficient*** for the existence of a time-independent one-step dynamic function if the dynamic process $(\mathcal{S}, P)$ itself is time-independent.

In practice, (2) is usually more useful, though (1) is stronger and general because for most reasoning problems, we expect the hypothesis space to be time-independent, so that the learned dynamic function could be applied to extrapolate lengths. The time-dependent hypothesis space may suffer from time-inconsistency, which is usually undesirable.

Learnability: *Sufficiency* does not mean learnability. For (1) above, in practice, a time-dependent dynamic function may fail to extrapolate due to two reasons: (i) the function is *time-inconsistent*, which leads to extrapolation failure; or (ii) the dynamic function is *time-consistent*, but is not successfully learned due to limitations of the learning algorithm. For (2), even a time-independent function exists, it may not be learned also because of limitations of the learning algorithm.

In the next two sections, we analyze two empirically successful LG methods based on the proposed necessary and sufficient conditions. We do not empirically investigate *sufficiency* as it is well known that current learning algorithms (e.g., Transformers and RNNs) are by themselves not sufficiently expressive or powerful to achieve LG for arbitrary CoT (dynamic) processes, and additional structural constraints as inductive biases are required (see the two sections below). For our analysis, we consider two reasoning problems—*addition* of two numbers and *parity*—since their CoT processes are both time-independent. In contrast, a simple example of a time-dependent dynamic process is the sequence of natural numbers $1, 2, 3, 4, \ldots$, where the underlying one-step function differs across regimes, e.g., '1 to 9' versus '9 to 99,' because from the token level perspective, '9 to 10' has a completely different dynamic function than 't to $t + 1$' for $t < 9$.

4 Analysis of (n, r)-Consistency

Xiao & Liu (2025) proposed (n, r)-*consistency* as a sufficient condition for LG on Transformers. Specifically, the authors defined a *problem class* whose problems can have CoT schemes satisfying (n, r)-*consistency* and proved that LG can be achieved with a Transformer for this class of problems. The key idea of (n, r)-consistency is that the CoT scheme of a problem should provide sufficient bias for the learner to learn and predict the same token in the same context with no conflict or uncertainty.

This section shows (1) that (n, r)-consistency satisfies the proposed *necessary* condition following some examples and (2) why (n, r)-consistency guarantees sufficiency for the existence of a Transformer solution and Theorem 3.5 fails.

4.1 Introduction to (n, r)-Consistency

The formal definition of (n, r)-consistency is involved and we include it in Appendix B. For easy understanding, here we use the integer addition example `'123+567'` in Xiao & Liu (2025) to illustrate. Let us consider a particular CoT step with the **input** $S^0 =$ `'123+567= $0'` and **output** $S^1 =$ `'123+567=?90'`, where `?` denotes that 0 is carried and `$` denotes that 1 is carried. To ensure that the input and the output are of the same length, a blank space (or empty) token is added after `'='` in S^0. This CoT scheme of addition is called ***addition*-[1]**.

This CoT step is for predicting the 10's token `9` in the output S^1, i.e., $S^1[10] = 9$. The 10's position of S^0 is `$`. Let us use the three token ***interval*** or subsequence `' $0'` in S^0 (i.e., $r = 3$ and `$` is the token position to be predicted, called the ***central token*** of the interval) as the context to predict `9` in the 10's position of S^1. This is clearly insufficient as `9` can only be calculated by using `2`, `6`, and `$` in S^0. This means that we need at least three intervals ($n = 3$) in S^0 as a context to predict `9` in S^1, i.e., `'123'`, `'567'`, and `' $0'`, where `2`, `6`, and `$` are the central tokens in the intervals respectively. `' $0'` is called the ***anchor interval*** as its central token `$` position is the one that we want to predict in S^1 with this set of three intervals, which is called a $(3, 3)$-***context***, representing three 3-length intervals.

The **concept of** (n, r)***-consistency***, i.e., $(3, 3)$-consistency in this case, means that `9` should be predicted for the position of the `'$'` **whenever** the three intervals (`'123'`, `'567'`, `' $0'`) appear in an input (instance of *addition*).

This $(3,3)$-context is ***not consistent*** for the addition-[1] problem. We can easily find another problem instance or input that contains the $(3,3)$-context but does not predict `'9'` for the position of the central token of the anchor interval. For example, the input S^0 = `'12342+45678= $0'` and output S^1 = `'12342+45678=$20'`. Obviously, S^0 here contains the above $(3,3)$-context, but the prediction for the central token '$' of the anchor interval in the input S^0 should be **2** in the output S^1, not 9. These two scenarios result in a **conflict**. Thus, addition-[1] is not $(3,3)$-*consistent* for $n=3$ and $r=3$. Thus, the **intuition of** (n,r)***-consistency*** is that we want to learn and predict the same token value in the same context with no conflict or uncertainty.

Below, we give the CoT schemes of the addition and parity problems that satisfy (n,r)-consistency, and show that both **their CoT schemes** and (n,r)**-consistency in general** satisfy our *necessary* condition. Comparing addition-[1] with addition-[2], we also show that functional sufficiency does not necessarily mean practical sufficiency (with the condition, any learner can learn to achieve LG).

4.2 Addition

4.2.1 CoT Scheme for Addition

The CoT for addition in (Xiao & Liu, 2025) uses two lines to achieve $(3,3)$-*consistent*, called ***addition*-[2]**. It uses tags to indicate the tokens involved in the next step calculation. The two CoT examples above become those in Figure 1a, where I and J tag the digits to be added next and K tags the position in the output. In this case, each token has 2 dimensions and when the second dimension is not I, J, or K, it is an empty element, which is also significant. The $(3,3)$-context of `'123+567= $0'` becomes that in Figure 1b.

Clearly, this $(3,3)$-context is no longer contained in the above CoT step of `'12342+45678'`, or any other possible input (or problem instance) whose central tokens are not the tokens to be calculated next because I, J, and K indicate the tokens to be calculated next. Thus, *addition*-[2] is $(3,3)$-*consistent*, meaning that for any input (instance of the *addition* problem) if it contains the $(3,3)$-context in Figure 1b, the output token at position K will always be 9.

$$123+567=\$0 \Rightarrow \begin{pmatrix} 1&2&3&+&5&6&7&=&\$&0 \\ I & & & & J & & & & K & \end{pmatrix}$$

$$12342+45678=\$0 \Rightarrow \begin{pmatrix} 1&2&3&4&2&+&4&5&6&7&8&=&\$&0 \\ & & I & & & & J & & & & & & K & \end{pmatrix}$$

(a) Transforming addition-[1] to addition-[2].

$$\begin{pmatrix} \text{`1 2 3'} & \text{`5 6 7'} & \text{` \$ 0'} \\ I & J & K \end{pmatrix}$$

(b) A $(3,3)$-context for addition-[2].

Figure 1: An example for Addition-[2]

4.2.2 Necessary and Sufficient Condition for Addition-[2] and (n,r)-consistency

By the definition of $(3,3)$-consistency, the next step token on the central position of the anchor interval is uniquely determined by the $(3,3)$-context. Thus, it defines a function $g: V^{3\times 3} \to V$. Denote the state space $\mathcal{S}$ to be the set of all inputs, e.g., S^0 = `'123+567= $0'` $\in \mathcal{S}$. Then apply g to scan the input by letting every position of input to be the central of an anchor interval. For example, we can apply g to scan the input S^0 = `'123+567= $0'` by letting the anchor interval scan from `' 12'` to `'$0 '`. This defines a deterministic function $f: V^\infty \to V^\infty$ that only depends on the input. This CoT process is Markov, so $N(\mathcal{S}, P) = 0$.

In general, the (n,r)**-consistent process is Markov** (see the definition in Appendix B) because when the problem class is (n,r)-consistent, we could always define a deterministic dynamic function $f: V^\infty \to V^\infty$ in the same way as that in addition-[2] with $(3,3)$-consistency above. Thus the next CoT step can be decided by the current CoT step, which is Markov.

Xiao & Liu (2025) proved that satisfying (n,r)-consistency is sufficient for the existence of a Transformer to perfectly solve the problem. The key reason is that (n,r)-consistency requires the problem class to satisfy: whenever a (n,r)-context appears in any input of any step, the next step (or output) token in the central position of the anchor interval is always the same token. This requirement guarantees the *time-consistency*. So (n,r)-consistency is actually functionally sufficient for the existence of a time-independent solution. The inductive bias induced by (n,r)-consistency has also been shown empirically to yield strong learnability for Transformer models.

In summary, addition-[2] is (n, r)-consistent, so it satisfies our necessary condition $N(\mathcal{S}, P) = 0$. Although $N(\mathcal{S}, P) = 0$ is insufficient for the existence of a time-independent solution, (n, r)-consistency is sufficient for the existence of a Transformer solution, which is time-independent. Empirical results showed addition-[2] provides enough bias for learning for LG almost perfectly.

4.2.3 Necessary and Sufficient Condition for Addition-[1]

Xiao & Liu (2025) showed addition-[1] is not (n, r)-consistent. However, addition-[1] is still Markov. Define the set of all inputs as the state space $\mathcal{S}$. For example, $S^1 = \texttt{`123+567=?90'} \in \mathcal{S}$ is uniquely determined by $S^0 = \texttt{`123+567= \$0'} \in \mathcal{S}$. It's obvious that there exists $f \in \mathcal{H} : V^\infty \to V^\infty, \forall\, S^0 \in \mathcal{S}$, $S^1 = f(S^0)$. For example, f could be defined by a dynamic programming program. So addition-[1] is Markov, $N(\mathcal{S}, P) = 0$, which satisfies our necessary condition.

By Theorem 3.4, addition-[1] is functional sufficient for time-dependent hypothesis space. But it's functional insufficient for time-independent hypothesis space. The reason is the *time-inconsistency*. The longer OOD input, that is not in the domain of shorter inputs, always exists and the domain is not consistent. Empirically, addition-[1] trained with Transformers shows poor results for LG Xiao & Liu (2025). Compared with addition-[2], which provides more bias and only requires $(3, 3)$-context for arbitrary length inputs (note that $(3, 3)$-context is not OOD), addition-[1] is more fragile to learn.

4.3 Parity

CoT Scheme for Parity: Parity is the problem of deciding whether there are an even or odd number of 1's in a sequence of 0's and 1's. Its CoT is structured in a two-line format, denoted as parity-[2]. As illustrated in Figure 2 for the query sequence 1011 (where each bit is a token), the first line (in the input or output) always contains the original query. The second line tracks the running parity state (odd or even) of the sequence up to each corresponding bit, with 1 representing odd and 0 representing even. The symbol '?' indicates the current CoT step. Note this figure only shows one CoT step. The output of this step will be the input of the next CoT step. This CoT scheme is $(1, 2)$-consistent. See (Xiao & Liu, 2025) for the detailed reason.

Input: 1 0 1 1
1 1 ?

Output: 1 0 1 1
1 1 0 ?

Figure 2: Input and output of a CoT step for the parity query 1011.

Necessary and Sufficient Conditions: They can be similarly analyzed as for addition-[2] above.

5 Analysis of Position Coupling

Position coupling is a technique that aligns tokens by marking their corresponding positions at each step of a calculation Cho et al. (2024; 2025). It makes use of Scratchpad to specify the intermediate reasoning steps, closely resembling CoT. A distinct position coupling scheme is designed for each type of reasoning problem. The authors also showed that with a proper input format, Scratchpad, and Position Coupling, a 1-layer (or 2-layer) decoder-only Transformer with certain choice of the embedding dimension is *sufficient* to solve addition (respectively, $N \times 2$ multiplication). However, no general theory or general sufficient condition was proposed for LG in Cho et al. (2024; 2025).

Below, again using *addition* and *parity*, we present the coupling scheme for each problem, demonstrating that the scheme satisfies the proposed necessary condition and analyze its sufficiency. These two problems have been shown to work well on LG (Cho et al., 2024; 2025). We will not study $N \times 2$ multiplication here as it is too complex, but it is easy to check that it also satisfies our necessary condition. Due to space limits, we are only able to discuss addition. Parity is given in Appendix C.

5.1 Addition

5.1.1 Position Coupling Scheme for Addition

The position coupling method for addition is illustrated using the example `'653+49=702'` in Figure 3. Each decimal digit (0–9) is treated as a single token. The response (or answer) is reversed, zero-padded to match operand lengths, and wrapped with BOS/EOS tokens `$`. For the example, `'653+49=702'` becomes `'653+049=207'`.

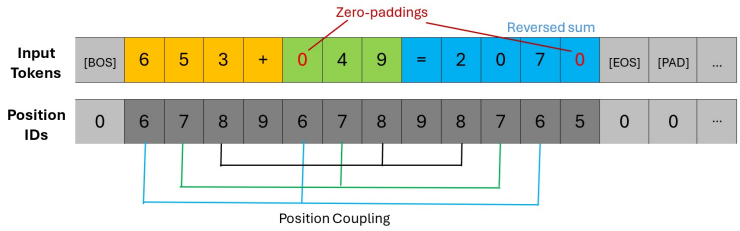


Figure 3: Positional coupling for $(653 + 49 = 702)$. The initial position ID '6' is randomly assigned

The sequence is partitioned into three logical groups: (1) First operand & '+', (2) Second operand, and (3) '=' & response (called *sum*). Digits in corresponding positions across groups share the same significance (ones, tens, hundreds, etc.), which induces a natural one-to-one correspondence between most tokens in groups (1), (2), and (3). '=' is grouped with the sum as these are the positions where next-token predictions are made.

Assigning position IDs. Digits of the same significance are assigned the same position ID. Starting from an arbitrary initial ID (e.g., 6), consecutive IDs are assigned to tokens in the operands. For the sum, which is reversed, IDs are assigned in descending order of significance to maintain alignment.

For `'653+049=207'`, operands (groups (1) and (2)): 6, 7, 8 (assigned left to right), and Sum (group (3)): 5, 6, 7, 8 (assigned right to left, matching significance). '+' and '=' tokens are assigned ID 9, as they follow the digit with the highest-ID (which is 8) in the operands. The most significant digit of the sum (which is a padded zero in this case) receives ID 5, placed before the next matching digit (ID 6, on the left), even though no operand digit corresponds to it.

5.1.2 Necessary and Sufficient Condition

For `'653+049=207'`, on step 1, the position IDs of '3', '9', '2' are the same, so '2' could be determined by '3' and '9'. For the next step, '53', '49' and '2' can determine the next token '0'. Here, we consider the previous token to determine whether carrying 1 happens. For all the remaining steps, predicting the token of position ID k only requires tokens with position IDs $k+1$ and k. Define the state space $\mathcal{S}$ to be the set of all sequences, e.g., `'653+049='`, `'653+049=2'` $\in \mathcal{S}$. Define a deterministic function $f : V^{\infty} \to V$ that only depends on tokens with position IDs $k+1$ and k when predicting token with ID k. This process is Markov, so $N(\mathcal{S}, P) = 0$.

Cho et al. (2024; 2025) show that a decoder-only Transformer is sufficient to solve the problem. The key reason is that the dynamic function is *time-consistent*, although OOD happens. This is a subtle situation and the length generalization result depends on the specific learning algorithm. OOD is obvious, because new position IDs would be met when increasing the length of summation. But it doesn't mean that the dynamic function is time-inconsistent. The dynamic function only has to do position matching for the position ID k and $k+1$. And such position matching is consistent for all k's and is possible to be learned within finite training lengths.

6 Conclusion

In this paper, we investigated the problem of length generalization (LG) in learning-to-reason tasks, a setting that fundamentally requires extrapolation beyond the training regime. Unlike standard learning scenarios that rely on interpolation under the i.i.d. assumption, LG poses unique challenges that are not well addressed by existing theoretical frameworks.

To bridge this gap, we proposed a theoretical analysis of LG through the lens of necessary and sufficient conditions that are independent of specific model architectures or learning algorithms. Based on the proposed necessary and sufficient condition, we examined two empirically successful LG approaches under this framework, which showed that these methods implicitly satisfy the proposed conditions, offering insight into why they perform well.

Limitations: (1) The work is primarily theoretical and explanatory, and does not directly propose a new learning algorithm that guarantees LG in practice. (2) The gap between existence and learnability remains uncharacterized. Our future work will focus on these.

AI USE STATEMENT

Generative AI tools were used solely to assist with language polishing.

ETHICS STATEMENT

This is a theoretical work that establishes necessary and sufficient conditions for learning with length generalization. It does not involve experimental data, human subjects, or sensitive information. We thus believe that there are no ethical risks associated with this work.

REFERENCES

Emmanuel Abbe, Samy Bengio, Aryo Lotfi, and Kevin Rizk. Generalization on the unseen, logic reasoning and degree curriculum. *arXiv preprint arXiv:2301.13105*, 2023.

Kartik Ahuja and Amin Mansouri. On provable length and compositional generalization. *arXiv preprint arXiv:2402.04875*, 2024.

Cem Anil, Yuhuai Wu, Anders Andreassen, Aitor Lewkowycz, Vedant Misra, Vinay Ramasesh, Ambrose Slone, Guy Gur-Ari, Ethan Dyer, and Behnam Neyshabur. Exploring length generalization in large language models. *Advances in Neural Information Processing Systems*, 35:38546–38556, 2022.

Assaf Ben-Kish, Itamar Zimerman, Shady Abu-Hussein, Nadav Cohen, Amir Globerson, Lior Wolf, and Raja Giryes. Decimamba: Exploring the length extrapolation potential of mamba. *arXiv preprint arXiv:2406.14528*, 2024.

Samuel R Bowman, Christopher D Manning, and Christopher Potts. Tree-structured composition in neural networks without tree-structured architectures. *arXiv preprint arXiv:1506.04834*, 2015.

Alexandra Butoi, Ghazal Khalighinejad, Anej Svete, Josef Valvoda, Ryan Cotterell, and Brian DuSell. Training neural networks as recognizers of formal languages, 2025. *ICLR*, 2025.

Thomas Chen, Tengyu Ma, and Zhiyuan Li. Non-asymptotic length generalization. *ICML-2025*, 2025a.

Yang Chen, Long Yang, Yitao Liang, and Zhouchen Lin. Low-dimension-to-high-dimension generalization and its implications for length generalization. *ICML*, 2025b.

Ta-Chung Chi, Ting-Han Fan, Alexander I Rudnicky, and Peter J Ramadge. Transformer working memory enables regular language reasoning and natural language length extrapolation. *arXiv preprint arXiv:2305.03796*, 2023.

Hanseul Cho, Jaeyoung Cha, Pranjal Awasthi, Srinadh Bhojanapalli, Anupam Gupta, and Chulhee Yun. Position coupling: Leveraging task structure for improved length generalization of transformers. In *First Workshop on Long-Context Foundation Models@ ICML 2024*, 2024.

Hanseul Cho, Jaeyoung Cha, Srinadh Bhojanapalli, and Chulhee Yun. Arithmetic transformers can length-generalize in both operand length and count. *ICLR*, 2025.

Jishnu Ray Chowdhury and Cornelia Caragea. Monotonic location attention for length generalization. *Proceedings of the 40th International Conference on Machine Learning (ICML-2023)*, 2023a.

Jishnu Ray Chowdhury and Cornelia Caragea. Recursion in recursion: Two-level nested recursion for length generalization with scalability. *Proceedings of 37th Conference on Neural Information Processing Systems (NeurIPS 2023)*, 2023b.

Shaoxiong Duan and Yining Shi. From interpolation to extrapolation: Complete length generalization for arithmetic transformers. *arXiv preprint arXiv:2310.11984*, 2023.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jian, Bill Yuchen Lin, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D Hwang, et al. Faith and fate: Limits of transformers on compositionality. *arXiv preprint arXiv:2305.18654*, 2023.

Ying Fan, Yilun Du, Kannan Ramchandran, and Kangwook Lee. Looped transformers for length generalization. *ICLR*, 2025.

Guhao Feng, Yuntian Gu, Bohang Zhang, Haotian Ye, Di He, and Liwei Wang. Towards revealing the mystery behind chain of thought: a theoretical perspective. *arXiv preprint arXiv:2305.15408*, 2023.

Noah Golowich, Samy Jelassi, David Brandfonbrener, Sham M Kakade, and Eran Malach. The role of sparsity for length generalization in llms. In *Proceedings of the Forty-second International Conference on Machine Learning (ICML-2025)*, 2025.

Chi Han, Qifan Wang, Wenhan Xiong, Yu Chen, Heng Ji, and Sinong Wang. Lm-infinite: Simple on-the-fly length generalization for large language models. *arXiv preprint arXiv:2308.16137*, 2023.

Zhenyu He, Guhao Feng, Shengjie Luo, Kai Yang, Liwei Wang, Jingjing Xu, Zhi Zhang, Hongxia Yang, and Di He. Two stones hit one bird: Bilevel positional encoding for better length extrapolation. *Proceedings of the 41st International Conference on Machine Learning*, 2024.

Kaiying Hou, David Brandfonbrener, Sham Kakade, Samy Jelassi, and Eran Malach. Universal length generalization with turing programs. *ICML-2025*, 2025.

Yi Hu, Xiaojuan Tang, Haotong Yang, and Muhan Zhang. Case-based or rule-based: How do transformers do the math? *Proceedings of the 41st International Conference on Machine Learning*, 2024.

Xinting Huang, Andy Yang, Satwik Bhattamishra, Yash Sarrof, Andreas Krebs, Hattie Zhou, Preetum Nakkiran, and Michael Hahn. A formal framework for understanding length generalization in transformers. *ICLR*, 2025.

Zachary Izzo, Eshaan Nichani, and Jason Lee. Quantitative bounds for length generalization in transformers. In *International Conference on Learning Representations*, volume 2026, pp. 44901–44939, 2026.

Samy Jelassi, Stéphane d'Ascoli, Carles Domingo-Enrich, Yuhuai Wu, Yuanzhi Li, and François Charton. Length generalization in arithmetic transformers. *arXiv preprint arXiv:2306.15400*, 2023.

Amirhossein Kazemnejad, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Payel Das, and Siva Reddy. The impact of positional encoding on length generalization in transformers. *37th Conference on Neural Information Processing Systems (NeurIPS 2023)*, 2023.

Elsi-Mari Laine, Jyrki Piilo, and Heinz-Peter Breuer. Measure for the non-markovianity of quantum processes. *Physical Review A—Atomic, Molecular, and Optical Physics*, 81(6):062115, 2010.

Nayoung Lee, Kartik Sreenivasan, Jason D Lee, Kangwook Lee, and Dimitris Papailiopoulos. Teaching arithmetic to small transformers. *arXiv preprint arXiv:2307.03381*, 2023.

Eran Malach. Auto-regressive next-token predictors are universal learners. *arXiv preprint arXiv:2309.06979*, 2023.

Annie Marsden, Evan Dogariu, Naman Agarwal, Xinyi Chen, Daniel Suo, and Elad Hazan. Provable length generalization in sequence prediction via spectral filtering. *ICML*, 2025.

Sean McLeish, Arpit Bansal, Alex Stein, Neel Jain, John Kirchenbauer, Brian Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, Jonas Geiping, Avi Schwarzschild, et al. Transformers can do arithmetic with the right embeddings. *Advances in Neural Information Processing Systems*, 37: 108012–108041, 2024.

Nikita Nangia and Samuel R Bowman. Listops: A diagnostic dataset for latent tree learning. *arXiv preprint arXiv:1804.06028*, 2018.

Rodrigo Nogueira, Zhiying Jiang, and Jimmy Lin. Investigating the limitations of transformers with simple arithmetic tasks. *arXiv preprint arXiv:2102.13019*, 2021.

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

Mattia Opper, Roland Fernandez, Paul Smolensky, and Jianfeng Gao. Tra: Better length generalisation with threshold relative attention. *arXiv preprint arXiv:2503.23174*, 2025.

Ofir Press, Noah A Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. *ICLR-2022*, 2022.

Jing Qian, Hong Wang, Zekun Li, Shiyang Li, and Xifeng Yan. Limitations of language models in arithmetic and symbolic induction. *arXiv preprint arXiv:2208.05051*, 2022.

Ricardo Buitrago Ruiz and Albert Gu. Understanding and improving length generalization in recurrent models. *ICML-2025*, 2025.

Anian Ruoss, Grégoire Delétang, Tim Genewein, Jordi Grau-Moya, Róbert Csordás, Mehdi Bennani, Shane Legg, and Joel Veness. Randomized positional encodings boost length generalization of transformers. *arXiv preprint arXiv:2305.16843*, 2023.

Mahdi Sabbaghi, George Pappas, Hamed Hassani, and Surbhi Goel. Explicitly encoding structural symmetry is key to length generalization in arithmetic tasks. *arXiv preprint arXiv:2406.01895*, 2024.

Yutao Sun, Li Dong, Barun Patra, Shuming Ma, Shaohan Huang, Alon Benhaim, Vishrav Chaudhary, Xia Song, and Furu Wei. A length-extrapolatable transformer. *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, 2023.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. Long range arena: A benchmark for efficient transformers. *Proceedings of ICLR2021*, 2021.

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

Gail Weiss, Yoav Goldberg, and Eran Yahav. Thinking like transformers. In *Proceedings of International Conference on Machine Learning (ICML)*, pp. 11080–11090, 2021.

Noam Wies, Yoav Levine, and Amnon Shashua. Sub-task decomposition enables learning in sequence to sequence tasks. *Proceddings of International Conference on Learning Representations (ICLR-2023)*, 2023.

Changnan Xiao and Bing Liu. Generalizing reasoning problems to longer lengths. In *ICLR*, 2025.

Andy Yang, Pascal Bergsträßer, Georg Zetzsche, David Chiang, and Anthony W Lin. Length generalization bounds for transformers. *ICML-2026, year=2026*.

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

Liang Zhao, Xiachong Feng, Xiaocheng Feng, Weihong Zhong, Dongliang Xu, Qing Yang, Hongtao Liu, Bing Qin, and Ting Liu. Length extrapolation of transformers: A survey from the perspective of positional encoding. *Findings of EMNLP-2024*, 2024.

Chuanyang Zheng, Yihang Gao, Han Shi, Minbin Huang, Jingyao Li, Jing Xiong, Xiaozhe Ren, Michael Ng, Xin Jiang, Zhenguo Li, et al. Dape: Data-adaptive positional encoding for length extrapolation. *Advances in Neural Information Processing Systems*, 37:26659–26700, 2024.

Hattie Zhou, Arwen Bradley, Etai Littwin, Noam Razin, Omid Saremi, Josh Susskind, Samy Bengio, and Preetum Nakkiran. What algorithms can transformers learn? a study in length generalization. *Proceedings of International Conference on Learning Representations (ICLR-2024)*, 2024a.

Yongchao Zhou, Uri Alon, Xinyun Chen, Xuezhi Wang, Rishabh Agarwal, and Denny Zhou. Transformers can achieve length generalization but not robustly, 2024. *URL https://arxiv.org/abs/2402.09371*, 2, 2024b.

A PROOFS

Lemma A.1 (Lemma 3.1)**.** *If $(\mathcal{S}, P)$ is Markov, then $\forall\, p_0, q_0 \in \Delta(\mathcal{S})$, $\forall\, t > 0$, $\dot{d}_t(p, q) \leq 0$.*

Proof. Since $(\mathcal{S}, P)$ is Markov, denote the transition matrix $T = \{t_{i,j} \mid t_{i,j} \geq 0, \sum_i t_{i,j} = 1\}_{1 \leq i,j \leq |\mathcal{S}|}$.

For $\forall\, p, q \in \Delta(\mathcal{S})$, denote $p = (p_1, \ldots, p_n)^\top, q = (q_1, \ldots, q_n)^\top$.

$$||Tp - Tq||_{L^1} = |||T(p-q)||_{L^1} = \sum_i |\sum_j t_{i,j}(p_j - q_j)| \leq \sum_i \sum_j t_{i,j}|p_j - q_j| = \sum_j |p_j - q_j| = ||p-q||_{L^1}.$$

Let $p = p_{t-1}$ and $q = q_{t-1}$, then $\dot{d}_t(p, q) = d(p_t, q_t) - d(p_{t-1}, q_{t-1}) = ||Tp - Tq||_{L^1} - ||p-q||_{L^1} \leq 0$. □

Theorem A.2 (Theorem 3.2)**.** *For a stochastic process $(\mathcal{S}, P)$, if $N(\mathcal{S}, P) > 0$, then $\forall\, \mathcal{H}^{\mathbb{N}}$ to be compact, $\min_{f \in \mathcal{H}^{\mathbb{N}}} \mu(f) > 0$.*

Proof. (Step 1) The distribution at step t is linear w.r.t the initial distribution. We write the process $(\mathcal{S}, P)$ as $p_t(\cdot) = \int_{\mathcal{S}} T_t(s, \cdot) d(p_0(s))$, where $T_t(\cdot, \cdot)$ is the transition kernel from 0 to t (note not assumed Markov). Then p_t is linear w.r.t p_0, i.e. $\forall\, p_0^1, p_0^2 \in \Delta(\mathcal{S})$,

$$\begin{aligned}(\alpha p_0^1 + (1-\alpha) p_0^2)_t &= \int_{\mathcal{S}} T_t(s, \cdot) d((\alpha p_0^1 + (1-\alpha) p_0^2)(s)) \\ &= \alpha \int_{\mathcal{S}} T_t(s, \cdot) d(p_0^1(s)) + (1-\alpha) \int_{\mathcal{S}} T_t(s, \cdot) d(p_0^2(s)) \\ &= \alpha p_t^1 + (1-\alpha) p_t^2.\end{aligned}$$

For $s \in \mathcal{S}$, recall that δ_s is the Delta-Dirac distribution on s. Denote $p_t = (\delta_s)_t$ when $p_0 = \delta_s$. Therefore, by the linearity, $\{(\delta_s)_t\}_{s \in \mathcal{S}}$ is a basis of p_t, i.e.

$$p_t = \sum_{s \in \mathcal{S}} p_0(s) (\delta_s)_t.$$

(Step 2) Assume there exists a function $f \in \mathcal{H}^{\mathbb{N}} : \Delta(\mathcal{S}) \to \Delta(\mathcal{S})$ s.t. $\mu(f) = \int_{p_0 \in \Delta(\mathcal{S})} \sum_{t>0} d(f(t, p_{t-1}), p_t) u(p_0) dp_0 = 0$. Then we have

$$p_t = f(t, p_{t-1}),\ a.e.\ p_0 \in \Delta(\mathcal{S}),\ \forall\, t > 0.$$

Specifically, $(\delta_s)_t = f(t, (\delta_s)_{t-1}),\ s \in \mathcal{S}$. By linearity, we have

$$\begin{aligned}p_t &= \sum_{s \in \mathcal{S}} p_0(s) (\delta_s)_t = \sum_{s \in \mathcal{S}} p_0(s) f(t, (\delta_s)_{t-1}), \\ p_{t-1} &= \sum_{s \in \mathcal{S}} p_0(s) (\delta_s)_{t-1}.\end{aligned}$$

Since $p_t = f(t, p_{t-1})$, we have

$$f(t, \sum_{s \in \mathcal{S}} p_0(s) (\delta_s)_{t-1}) = f(t, p_{t-1}) = p_t = \sum_{s \in \mathcal{S}} p_0(s) f(t, (\delta_s)_{t-1}).$$

Note that $(\delta_s)_{t-1}$ is irrelevant to p_0, we could simply rewrite and replace p_0 by any weight $\alpha \in \Delta(\mathcal{S})$:

$$f(t, \sum_{s \in \mathcal{S}} \alpha(s) (\delta_s)_{t-1}) = \sum_{s \in \mathcal{S}} \alpha(s) f(t, (\delta_s)_{t-1}),\ \forall\, \alpha \in \Delta(\mathcal{S}).$$

Since $p_{t-1} = \sum_{s \in \mathcal{S}} p_0(s) (\delta_s)_{t-1}$, the space of p_{t-1} is convex, denoted as $C_{t-1} \subseteq \Delta(\mathcal{S})$. The vertices of $\Delta(\mathcal{S})$ is δ_s's, we could represent $(\delta_s)_{t-1}$ by linear combination of δ_s's with constraint on C_{t-1},

$$f(t, \sum_{s \in \mathcal{S}} \alpha(s) \delta_s) = \sum_{s \in \mathcal{S}} \alpha(s) f(t, \delta_s),\ \forall\, \alpha \in \Delta(\mathcal{S}),\ \forall\ \sum_{s \in \mathcal{S}} \alpha(s) \delta_s \in C_{t-1}.$$

Now we know f is a linear function on C_{t-1}. Define

$$T_t(s'|s) = f(t, \delta_s)(s'),$$

then $\forall\, p_{t-1} = \sum_{s\in\mathcal{S}} p_{t-1}(s)\delta_s \in C_{t-1}$, we have

$$p_t = f(t, p_{t-1}) = f(t, \sum_{s\in\mathcal{S}} p_{t-1}(s)\delta_s) = \sum_{s\in\mathcal{S}} p_{t-1}(s) f(t, \delta_s) = T_t p_{t-1}.$$

Therefore, similar to Lemma 3.1, $\forall\, p_{t-1}, q_{t-1} \in C_{t-1}$, we have $d(p_t, q_t) \le d(p_{t-1}, q_{t-1})$, i.e. $\dot{d}_t(p, q) \le 0$. Note that the inequality holds a.e. $p_0, q_0 \in \Delta(\mathcal{S}), \forall\, t > 0$.

However, $p_t = \sum_{s\in\mathcal{S}} p_0(s)(\delta_s)_t$ is a continues function of p_0, $d(p_t, q_t)$ is a continuous function of (p_0, q_0), $\dot{d}_t(p, q)$ is a continuous function of (p_0, q_0). When $N(\mathcal{S}, P) = \max_{p_0,q_0\in\Delta(\mathcal{S})} N_0(p_0, q_0, \mathcal{S}, P) > 0$, $\exists\ p_0^*, q_0^*, \exists\ t > 0$, $\exists$ a neighborhood $B(p_0^*, q_0^*, \delta)$ s.t. $\forall\, (p_0, q_0) \in B(p_0^*, q_0^*, \delta)$, $\dot{d}_t(p, q) > 0$. Contradiction. □

Theorem A.3 (Theorem 3.4)**.** *For a stochastic process $(\mathcal{S}, P)$, if $N(\mathcal{S}, P) = 0$, then $\exists\ \mathcal{H}^{\mathbb{N}}$, $\min_{f\in\mathcal{H}^{\mathbb{N}}} \mu(f) = 0$.*

Proof. We construct f by the following steps.

(Step 1) Since $N(\mathcal{S}, P) = 0, \forall\, p_0, q_0 \in \Delta(\mathcal{S}), \forall\, t > 0, \dot{d}_t(p, q) = 0$. For each t, f is well-defined on $\{(\delta_s)_{t-1} \mid s \in \mathcal{S}\}$ to satisfy

$$f(t, (\delta_s)_{t-1}) = (\delta_s)_t,$$

because whenever $d(p_{t-1}, q_{t-1}) = 0$, we always have $d(p_t, q_t) \le d(p_{t-1}, q_{t-1}) = 0$.

(Step 2) For each t, define f on $ConvexHull(\{(\delta_s)_{t-1} \mid s \in \mathcal{S}\})$ by

$$f(t, \sum_{s\in\mathcal{S}} \alpha(s)(\delta_s)_{t-1}) = \sum_{s\in\mathcal{S}} \alpha(s) f(t, (\delta_s)_{t-1}),\ \forall\, \alpha \in \Delta(\mathcal{S}).$$

Now we check the defined f satisfying $f(t, p_{t-1}) = p_t$. By linearity in the proof of Theorem 3.2,

$$p_{t-1} = \sum_{s\in\mathcal{S}} p_0(s)(\delta_s)_{t-1},\ p_t = \sum_{s\in\mathcal{S}} p_0(s)(\delta_s)_t.$$

Note that when $\alpha = p_0$, by definition in (Step 2), we have

$$\begin{aligned}
f(t, p_{t-1}) &= f(t, \sum_{s\in\mathcal{S}} p_0(s)(\delta_s)_{t-1}) && (linearity)\\
&= f(t, \sum_{s\in\mathcal{S}} \alpha(s)(\delta_s)_{t-1}) && (\alpha = p_0)\\
&= \sum_{s\in\mathcal{S}} \alpha(s) f(t, (\delta_s)_{t-1}) && (Step\ 2\ Definition)\\
&= \sum_{s\in\mathcal{S}} \alpha(s)(\delta_s)_t && (Step\ 1\ Definition)\\
&= \sum_{s\in\mathcal{S}} p_0(s)(\delta_s)_t && (\alpha = p_0)\\
&= p_t. && (linearity)
\end{aligned}$$

□

Theorem A.4 (Theorem 3.5)**.** *There exists a stochastic process $(\mathcal{S}, P)$, s.t. $N(\mathcal{S}, P) = 0$, but $\forall\, \mathcal{H}$, $\min_{f\in\mathcal{H}} \mu(f) > 0$.*

Proof. We define a two state deterministic process. Let $\mathcal{S} = \{a, b\}$. When t is even, let

$$T_t(s'|s) = \begin{cases} 1, \text{ if } s' = s, \\ 0, \text{ else.} \end{cases}$$

When t is odd, let

$$T_t(s'|s) = \begin{cases} 1, \text{ if } s' \neq s, \\ 0, \text{ else.} \end{cases}$$

When $p_0 = (x, y)$, we have $p_1 = (y, x), p_2 = (y, x), p_3 = (x, y), p_4 = (x, y), \ldots$. Then for any p_0, q_0, we always have

$$d(p_t, q_t) = \cdots = d(p_0, q_0).$$

Therefore, $N(\mathcal{S}, P) = 0$. But it's obvious that we cannot define $f \in \mathcal{H}$ s.t.

$$(y, x) = p_2 = f(p_1) = f(y, x),$$
$$(x, y) = p_3 = f(p_2) = f(y, x).$$

□

B Definition of (n, r)-Consistency

The following is from Xiao & Liu (2025). Let $\mathscr{S}$ be a reasoning problem (a class of problem, e.g., addition) and $S^0 \in \mathscr{S}$ be an instance of the problem expressed as a string. In a CoT step t, we have the input S^t and output S^{t+1}. Our goal is to learn a function f that predicts S^{t+1} given S^t, i.e. $S^{t+1} = f(S^t)$.[2] The problem of predicting S^{t+1} can be decomposed into sub-problems of predicting each element of S^{t+1}, i.e., $S^{t+1}[i]$ for each position index i. To do so, we further ensure that the lengths S^t and S^{t+1} are the same by adding blank space elements in suitable positions. (n, r) basically means a context with n r-length intervals (or subsequences) in S^t can predict S^{t+1}.

Let us now define (n, r)-*consistency*. For simplicity of notations, we will use S^0 to represent S^t and S^1 to represent S^{t+1}. Similarly, we can also call any S^t a problem instance. We first introduce the concept of ***consistent context***.

Definition B.1 (Consistent Context)**.** Denote a context $h = (a_1 : a_2, \ldots, a_{|h|})$ as a sequence of $|h|$ intervals, where a_1 is called the **anchor interval** and each interval consists of a sequence of r elements, i.e. $a_j = s_{j,1} \ldots s_{j,r}$, $s_{j,l} \in V, 1 \leq j \leq |h|, 1 \leq l \leq r$, where V is a finite vocabulary.[3]

(i) For a problem instance $S^0 = s_1 \ldots s_{|S^0|}$, we say $h \sqsubset S^0$ (contained in) if there exists $1 \leq m_1, \ldots, m_{|h|} \leq |S^0|$,[4] s.t. $s_{m_j} \ldots s_{m_j+r-1} = a_j, 1 \leq j \leq |h|$.

(ii) We say h is a *context* of a problem $\mathscr{S}$ if there exists a problem instance $S^0 \in \mathscr{S}$ s.t. $h \sqsubset S^0$.

(iii) We define $c(h, S^0) = m_1 + \lfloor \frac{r}{2} \rfloor$ as the position index of the central element of the anchor interval $a_1 = s_{m_1} \ldots s_{m_1+r-1}$ in S^0 (due to the flooring, r does not have to be **odd**). For example, given $r = 3$, $S^0 = 1 + 2 \times 1$ and $a_1 = 2 \times 1$, then $m_1 = 3$, $\lfloor \frac{r}{2} \rfloor = 1$, $c(h, S^0) = 4$, and $S^0[c(h, S^0)] = \times$.

(iv) We say h is a **consistent context** if for any $S_1^0, S_2^0 \in \mathscr{S}$ and their respective CoT step outputs S_1^1, S_2^1, when $h \sqsubset S_1^0, h \sqsubset S_2^0$, they always have the same element at the central element position of a_1 of h, i.e., $S_1^1[c(h, S_1^0)] = S_2^1[c(h, S_2^0)]$. To ensure the equal length of S_k^0 and S_k^1, blank space elements are inserted in suitable positions (see below).

(v) For a consistent context h, define $\psi(h) = S^1[c(h, S^0)], \forall\, S^0 \in \mathscr{S}$ s.t. $h \sqsubset S^0$.

When $h = (a_1 : a_2, \ldots, a_{|h|})$, $h' = (a'_1 : a'_2, \ldots, a'_{|h'|})), |h'| > |h|$ and $a_i = a'_i, 1 \leq i \leq |h|$, we say h is the *prefix* of h' and write $h \prec h'$. We write $h \preceq h'$, when $h \prec h'$ or $h = h'$.

Definition B.1 constrains only the anchor interval a_1 of h and it seems to leave the other intervals unconstrained except h is contained in S^0. As indicated in Sec. 4.2.1, the other intervals usually contain the other elements involved in a reasoning step. This leads to the (n, r)-consistency definition.

[2] For simplicity, we omit N in f_N, where N represents the length of the problem instance.

[3] It is not important where the intervals other than the anchor interval are located.

[4] It is not required that $m_1 < m_2 < \cdots < m_{|h|}$ and every a_j in $h = (a_1 : a_2, \ldots, a_{|h|})$ doesn't contain any positional information in S^0.

Definition B.2 ((n, r)-Consistent Problem)**.** A problem $\mathscr{S}$ is said (n, r)-consistent, if for every problem instance S^0 in $\mathscr{S}$, i.e., $\forall\, S^0 \in \mathscr{S}$, denote S^1 as the CoT step output of S^0, for every index or position i of S^1 (i.e. $\forall\, 1 \leq i \leq |S^1|$), there exists a consistent context $h_{[i]}$ s.t. (i) $c(h_{[i]}, S^0) = i$, (ii) $|h_{[i]}| \leq n$, (iii) all intervals in $h_{[i]}$ is r-length.

C PARITY

Position Coupling Scheme for Parity: Figure 4 shows a 4-bit parity example in position coupling, where the input sequence consists of a query (0101) and an associated Scratchpad (0110) Cho et al. (2025). Each bit is regarded as a token. The Scratchpad is designed to record intermediate parity states as the binary query is processed from left to right. Specifically, the k'th token in the Scratchpad (after '=') represents the parity (0 for even and 1 for odd) of the subsequence formed by the first k bits of the query. Consequently, the rightmost token of the Scratchpad corresponds to the final output of the parity task. The coupled position IDs take the form 2 3 4 5 1 2 3 4 5. The starting ID 2 is randomly chosen.

Necessary and Sufficient Conditions: They can be similarly analyzed as for the addition in Sec. 5.1.2.

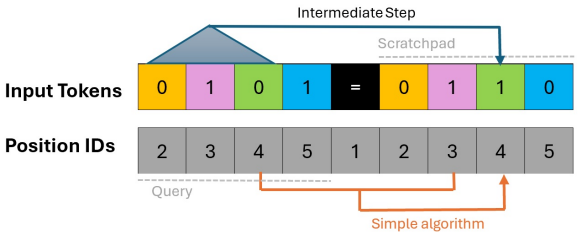


Figure 4: A parity check on the sequence 0101. The 'simple algorithm' just indicates how each party state is obtained.