Conditions for Length Generalization in Learning Reasoning Skills

Changnan Xiao[1] and Bing Liu[2]

[1]ChangnXX.github.io
[2] Department of Computer Science, University of Illinois Chicago
changnanxiao@gmail.com, liub@uic.edu

Abstract

Reasoning is a fundamental capability of AI agents. Recently, large language models (LLMs) have shown remarkable abilities to perform reasoning tasks. However, numerous evaluations of the reasoning capabilities of LLMs have also showed some limitations. An outstanding limitation is *length generalization*, meaning that when trained on reasoning problems of smaller lengths or sizes, the resulting models struggle with problems of larger sizes or lengths. This potentially indicates some theoretical limitations of generalization in learning reasoning skills. These evaluations and their observations motivated us to perform a theoretical study of the length generalization problem. This work focused on reasoning tasks that can be formulated as Markov dynamic processes (MDPs) and/or directed acyclic graphs (DAGs). It identifies and proves conditions that decide whether the length generalization problem can be solved or not for a reasoning task in a particular representation. Experiments are also conducted to verify the theoretical results.

1 Introduction

Reasoning is an important part of Artificial Intelligence (AI) as it is a necessary capability of an intelligent agent. In recent years, learning to reason has been actively studied in the machine learning and natural language processing communities [YZW23, SGC19, YDM+23, HCS21]. Recent *large language models* (LLMs) like ChatGPT [Ope22] and GPT4 [Ope23] have been shown to perform sophisticated reasoning tasks in very impressive manners [BMR+20, SSS+22, SH22, LNT+23, XLH+23].

This success also invited many researchers to conduct systematic evaluations of the reasoning capabilities of LLMs. These evaluations showed impressive results on many reasoning problems, but at the same time also revealed some limitations [LNT+23, XLH+23]. For example, several researchers showed that LLMs often encounter difficulties on simple multiplication, division, and addition of large numbers [NJL21, QWL+22]. [Ark23] listed a diverse set of reasoning tasks that GPT4 struggles with, which led the author to argue that most of the reasoning tasks successfully performed by GPT4 might already be in the pre-training dataset or very similar to some problems in the pre-training dataset. Evaluations conducted on several latest LLMs also showed that they struggled with many reasoning tasks [GBWD23, TZL+23].

The popular solution for improving the reasoning capabilities of LLMs is to use the approach called *Chain of Thought* (**CoT**) [WWS+22b] or *Scratchpad* [NAGA+21]. The CoT idea is natural. It simply adds the *intermediate steps* for each reasoning problem in the training data. For example, the training sample for calculating $2 + 3 \times 4$ may be presented as $2 + 3 \times 4 = 2 + 12 = 14$. Several researchers have used CoT to improve the accuracy of arithmetic operations with good success [WWS+22b, LL23, LSL+23]. However, [DLS+23] demonstrated experimentally that even with the detailed reasoning steps given in the training data, the learned models still fail to learn the operations in a generalizable manner for several reasoning problems. Specifically, the authors showed that when trained with smaller problems for a reasoning task (e.g., multiplication of two numbers), the model cannot generalize to larger problems. Several other authors have also noticed the problem, which is called the ***length generalization*** problem [AWA+22, ZBB+22].

The paper investigates the length generalization problem. It identifies and proves conditions that decide whether the length generalization problem can be solved or not for a reasoning task in a particular representation. Empirical evidences are also given to verify the theoretical results (Sec. 5).

2 Overview of Our Results

This work focuses on studying the length generalization problem for reasoning tasks that can be formulated as *Markov dynamic processes* (MDP) and/or *directed acyclic graphs* (DAGs) structures, where MDP is a special case of DAG. We progressively show: (1) given the MDP/DAG structure of a reasoning task, the condition for learning the *causal function* (which performs a single reasoning step), (2) given the MDP/DAG structure and a well-learned causal function, a *recursive formula* being able to solve the length generalization problem, and (3) given only unstructured data, the condition for learning the structure (i.e., predicting the elements involved in the next reasoning step) and solving the length generalization problem.

To illustrate our notations, we take the problem of calculating $2 + 3 \times 4$ as an example. A causal function infers one step in the reasoning/calculation process (e.g., as specified in CoT). In this example, $3 \times 4 = 12$ and $2 + 12 = 14$ are both one casual/reasoning step. In the arithmetic calculation on natural numbers $\mathbb{N}$, the causal function is $f : \mathbf{X} \to \mathbb{N}$, where $\mathbf{X} = \mathbb{N} \times \{+, -, \times, /\} \times \mathbb{N}$ is the input space of the causal function.

We define a dynamic process as a process that maps a sequence of elements to a sequence of elements from left to right representing a reasoning process. For instance, $3 \times 4 + 2$ defines a dynamic process with $x_0 = 3, x_1 = (\times, 4), x_2 = (+, 2), y_1 = f(x_0, x_1) = 3 \times 4 = 12, y_2 = f(y_1, x_2) = 12 + 2 = 14$, where f is the dynamic/causal function. Since f only depends on the latest x and y, this dynamic process is called a Markov dynamic process (MDP) (see Figure 1), which reasons only sequentially and it is very limiting. DAG is a generalization of MDP
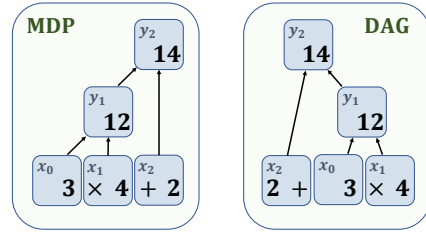

Figure 1: **An example of MDP and DAG.**

and is much more powerful.[1] For instance, the reasoning process of $2 + 3 \times 4$ can be represented as the DAG in Figure 1. The topological ordering of the DAG gives $x_0 = 3, x_1 = (\times, 4), x_2 = (2, +), y_1 = f(x_0, x_1) = 3 \times 4 = 12, y_2 = f(x_2, y_1) = 2 + 12 = 14$, where f is the causal function. For simplicity, we will always use *causal function* and it also means *dynamic function* for dynamic process.

We define a *recursive formula* as a function that recursively applies the causal function step-by-step to solve a reasoning problem according to its structure, e.g., $2 + 3 \times 4 = 2 + 12 = 14$ (see the DAG image in Figure 1). In Secs 3.1 and 3.2, the structure (MDP/DAG with calculation steps) is given. In Sec. 3.3, the structure is unknown and needs to be learned, which is the practical scenario.

In Sec. 3.1 and Sec. 3.2, we start from a simple or ideal scenario where the structure of the reasoning problem is given as an MDP or a DAG. We prove the following:

- the causal function can be perfectly learned (called *well-learned*) only when the input space of the causal function is finite, $|\mathbf{X}| < \infty$.
- when evaluating after training, given the MDP/DAG of the reasoning problem, solving the problem by recursively applying a well-learned causal function can generalize from smaller training problem/sample sizes to testing of larger problem sizes.

In Sec. 3.3, we study the realistic usage scenario where the structure of the problem is unknown and only unstructured sequence data with CoT is given in training, e.g., a sequence of elements like $2 + 3 \times 4 = 2 + 12 = 14$. We will not study the case when the CoT (or the intermediate reasoning) steps are not given but only the input and the output like $2 + 3 \times 4 = 14$ as several researchers [WLS23, FGZ+23, Mal23] have shown that this case is not learnable. We focus on the following,

- learning to predict which elements should be the input to the well-learned causal function to calculate or reason next, i.e., learning the ordering of calculations. We propose an important notion R, the *maximal input element distance for a causal/reasoning step*. For instance, for the arithmetic expression $2 + 3 \times 4$

---

[1] Some prior empirical works have used DAGs to represent reasoning problems [SHH22, CHLL21, HZL+22]. Their goal was to generate DAGs from natural language descriptions of the reasoning problems. We also note that some reasoning problems cannot be modeled as DAGs, e.g., temporal and spatial reasoning problems.

without using '(' and ')', we have $R = 2$ because the elements that should be calculated next are in a window of length 3, e.g., 3×4, where the maximal input elements distance is 2 (the distance between 3 and 4). If the arithmetic calculation contains '(' and ')', e.g., $2 + (3 \times 4)$, then $R = 4$ because the elements that should be calculated next is in a window of at most of length 5, e.g., (3×4).

- proving that the condition for (length) generalization from smaller/shorter training problems to larger/longer test problems is $R < \infty$.

As discussed earlier, several papers have reported experiments showing that deep learning models struggle with length generalization for some reasoning problems even with the help of CoT or Scratchpad [AWA+22, DLS+23, ZBB+22]. Based on our theory, the CoT formulations of these problems suffer from either $|\mathbf{X}| = \infty$ or $R = \infty$, which causes the length generalization issue. In Sec. 5, we will see that the same reasoning problem may be solved if a different CoT representation is used, which satisfies $R < \infty$.

Learnability studies of reasoning problems using CoT have been reported in [WLS23, FGZ+23, Mal23] for neural networks. However, these studies all under the given problem length/size N. Their statements are like "for $\forall N > 0$, given a dataset with training problems no longer than N, for any problem with length $N' \leq N$, it can be solved under a PAC upper bound." The key limitation is that training problem length and testing problem length are the same, or the upper bound depends on the testing problem length. Our theory doesn't have this limitation and works on a more general scenario with training on smaller length problems and testing on larger length problems. Our statement is that "for $\forall N > 0$, given a training dataset with problems no longer than N, problems of any length N' can be solved if $|\mathbf{X}| < \infty$ and $R < \infty$." Our work can be seen as complementary to theirs in the context of neural networks (as our results are independent of specific learning paradigm or algorithm). Their learnability results still apply. We add conditions under which the learned function can extrapolate to larger lengths than N to achieve length generalization.

Length generalization is related to *out-of-distribution* (OOD) generalization in [ABLR23]. However, there is a major difference. In length generalization, since the training data in practice is always finite, the maximal size or length of the training problems is also finite, but a larger size/length problem can always appear in testing, which can be seen as OOD, but such an OOD is unavoidable regardless how much data is used in training as long as it is finite. In [ABLR23], OOD is defined as some subspace (i.e., some values or value combinations) never appeared in training but appeared in testing. This type of OOD generalization can be solved with more and/or diverse training data. We do not study this problem. Further, the study in [ABLR23] was under some specific model architectures and activation functions. Our work is algorithm and architecture independent and we focus on length generalization.

In summary, to learn to solve reasoning problems and to overcome the length generalization issue, we propose *three sufficient conditions*, **(i)** the input space $\mathbf{X}$ of a causal/reasoning step is finite, **(ii)** the problem is solved recursively based on CoT, and **(iii)** the *maximal input element distance R* of the unstructured representation for a causal/reasoning step is finite.

3 Our Results

3.1 Markov Dynamic Process (MDP)

This section considers sequential reasoning problems. In such a problem, the inference steps are made in a single sequence. We take the arithmetic problem in a prime field $F_p = \mathbf{Z}/p\mathbf{Z}$ as an example, which can be formulated into a sequential reasoning problem. For instance, we solve $((((a_0 + a_1) \times a_2) - a_3)/a_4)$ sequentially by $b_1 = f(a_0, +, a_1)$, $b_2 = f(b_1, \times, a_2)$, $b_3 = f(b_2, -, a_3)$, and $b_4 = f(b_3, /, a_4)$, where f is the definition of $(+, -, \times, /)$ in F_p. In this section and the next, we do not regard '(' and ')' as basic elements in the arithmetic problem. When they show up, they only indicate the order of the calculation steps. In Sec. 3.3, we'll revisit them and they are significant when finding the elements to be calculated next.

Denote X to be the domain. X contains all the input elements of all reasoning steps. Denote Y to be the range. Y contains all the output elements of all reasoning steps. X and Y can have multiple dimensions (see the 3-line addition example in Section 5). For the arithmetic problem in F_p, we define $Y = F_p$ and

$X = \{+, -, \times, /\} \times F_p$, since the arithmetic symbols or operators are also the input elements for each calculation step. For simplicity, we denote a sequence of elements in X as

$$x_{[i:j]} = (x_i, x_{i+1}, \ldots, x_j). \tag{1}$$

Let $f : Y \times X^\infty \to Y$ be a **casual function** (we can also call it a *dynamic function*[2]), which performs one reasoning step. We denote the dynamic process τ_f defined by f as

$$\tau_f(x_{[0:\infty]}) = (y_1, y_2, \ldots), \quad \text{where} \quad \begin{cases} y_1 = f(y_0, x_0), \\ \ldots \\ y_{n+1} = f(y_n, x_{[0:n]}). \end{cases} \tag{2}$$

Without loss of generality, we always assume $y_0 = 0$.

For instance, in the arithmetic expression $((((a_0 + a_1) \times a_2) - a_3)/a_4)$, $x_{[1:4]} = ((+, a_1), \ldots, (/, a_4))$, where '(' and ')' only indicate the correct calculation order, the causal function $f : F_p \times \{+, -, \times, /\} \times F_p \to F_p$ defines the arithmetic calculations in F_p. And $\tau_f(x_{[0:4]}) = (b_1, b_2, b_3, b_4)$, where $b_1 = f(a_0, +, a_1)$, $b_2 = f(b_1, \times, a_2)$, $b_3 = f(b_2, -, a_3)$, and $b_4 = f(b_3, /, a_4)$. The first calculation step $b_1 = f(a_0, +, a_1)$ corresponds to $y_1 = f(y_0, x_1)$, where $y_1 = b_1$, $y_0 = a_0$ and $x_1 = (+, a_1)$.

We say τ_f is a Markov dynamic process (MDP) if $y_{n+1} = f(y_n, x_{[0,n]})$ depends only on (y_n, x_n) for $\forall n > 0$. We denote $y_{n+1} = f(y_n, x_n)$, where $f : Y \times X \to Y$ is the causal function for MDP. The **input space** of the causal function is $\mathbf{X} = Y \times X$. We emphasize that the definition of the MDP only requires that f depends on the current x_n rather than the past x_i's.

To understand the learning of MDP, we first study the causal function. Denote the training dataset of f as D, which is a subset of the input space,

$$D \subseteq \{(f(y, x), (y, x)) | (y, x) \in Y \times X\}. \tag{3}$$

For simplicity, we denote $D \subseteq \mathbf{X} = Y \times X$. For instance, in the arithmetic problem in F_p, we have $D \subseteq \{(f(a_1, *, a_2), a_1, *, a_2) | a_1, a_2 \in F_p, * \in \{+, -, \times, /\}\}$. We assume $|D| < \infty$.

Theorem 3.1. *For $|X|, |Y| < \infty$, i.e. $|\mathbf{X}| < \infty$, if $D = \mathbf{X}$, then there exists an approximation function $\hat{f} : Y \times X \to Y$ s.t. $\hat{f}(y, x) = f(y, x), \forall (y, x) \in \mathbf{X}$.*

The proof is given in *Appendix* A. This theorem says that when $|X|, |Y| < \infty$, which implies $|\mathbf{X}| < \infty$, there exists an approximation function that perfectly approximates the causal function f defined on $Y \times X$. For instance, when $D = F_p \times \{+, -, \times, /\} \times F_p$, there must exist a function $\hat{f}$ that learns the same arithmetic rules as f's in F_p. When $D \neq \mathbf{X}$, we have the following Corollary 3.1.1 that shows the **impossibility** of perfectly approximating the causal function f.

Corollary 3.1.1. *For $|X|, |Y| < \infty$, i.e. $|\mathbf{X}| < \infty$, if $|Y| > 1$ and $D \neq \mathbf{X}$, then there exists an approximation function $\hat{f} : Y \times X \to Y$ s.t. $\hat{f}(y, x) = f(y, x), \forall (y, x) \in D$ and $\hat{f}(y, x) \neq f(y, x), \forall (y, x) \in \mathbf{X} \backslash D$.*

The proof is given in *Appendix* A. Corollary 3.1.1 shows that when $D \neq \mathbf{X}$, there always exists a function that predicts correctly on D but wrongly on the complement ($\mathbf{X} \backslash D$). [ABLR23] showed that the learned random feature model tends to learn an low-degree approximation function theoretically, which means that the generalization on the unseen is disappointing unless there exist some subtle underlying connections between the unseen and the hypothesis space. Thus, it supports the ***impossibility*** of learning the causal function when unseen (out-of-distribution) exist in testing. Note that Corollary 3.1.1 is independent of specific learning algorithms or hypothesis spaces. The conclusion can naturally be extended to $|X| = \infty$.

Corollary 3.1.2. *For $|X| = \infty$, i.e. $|\mathbf{X}| = \infty$, if $|Y| > 1$, for $\forall m > 0$, there exists an approximation function $\hat{f} : Y \times X \to Y$ s.t. $\hat{f}(y, x) = f(y, x), \forall (y, x) \in D$ and $|\{(y, x) | \hat{f}(y, x) \neq f(y, x)\}| > m$.*

---

[2] Note that *dynamic function* here is not related to the term *dynamic function* used in programming languages.

The proof is deferred to *Appendix* A. Now we know the causal function f can be well-learned only when $|\mathbf{X}| < \infty$, regardless of the learning algorithm and the hypothesis space. When we look back at the MDP, it can simply be *recursively solved* with the well-learned causal function.

Theorem 3.2. *For $|X|, |Y| < \infty$, i.e. $|\mathbf{X}| < \infty$, if $D = \mathbf{X}$, then there exists an approximation function $\hat{f}$: $Y \times X \to Y$, the MDP τ_f can be recursively solved, i.e. $\forall n > 0$, $\tau_{\hat{f}}(x_{[0:n]}) = \tau_f(x_{[0:n]})$.*

The proof is given in *Appendix* B. The theorem also guarantees that a MDP of *any length* can be recursively solved.

3.2 Directed Acyclic Graph (DAG)

Now we know the MDP of an arbitrary length can be solved recursively by Theorem 3.2. To generalize the result, we introduce the Direct Acyclic Graph (DAG), a directed graph with no directed cycles. In this section, we consider reasoning problems that can be structured into DAGs. A topological ordering can order the vertices of a DAG, where every edge leaves a front vertex and enters a latter vertex. The existence of the topological ordering is an equivalent definition of DAG, i.e. every DAG has at least one topological ordering [BJG08]. Therefore, every DAG can be transformed into a dynamic process. However, the dynamic process from a DAG may not be Markov. For instance, the causal function $f(y_n, x_{[0:n]}) = y_n + x_n + \cdots + x_0$ defines a DAG as well as a dynamic process that always depends on all the previous x_i's, which is not Markov. Even though a DAG is not a MDP, it's still possible to solve the DAG recursively.

Denote a DAG as $G = (V, E)$, where V is the set of all vertices and E is the set of all edges. For easy understanding, we use $|G| \, (= |V|)$ to represent the number of vertices in G. Denote $u \xrightarrow{e} v$ as when v is reachable directly from u by the edge e. Denote $s(v) = \{u \in V \,|\, \exists e \in E, u \xrightarrow{e} v\}$ to be all vertices (immediate in-neighbors) that can reach v directly. Denote the *in-degree* of each vertex v as $|s(v)|$.

For a reasoning problem structured as a DAG, when $u \xrightarrow{e} v$, we say that u is the **input vertex** and v is the **causal vertex**. Since a causal vertex can also be the input vertex of some other vertices, we do not distinguish the domain and the range as in Sec. 3.1. With a slight abuse of notation, we also use v to represent the value of vertex v. We simply denote X to be both the domain and the range of all vertices. Let $f : X^{\sup|s(v)|} \to X$ to be the **causal function**, which is $v = f(s(v))$. The **input space** of $f : X^{\sup|s(v)|} \to X$ is $\mathbf{X} = X^{\sup|s(v)|}$. Note that G defines the structure of the reasoning process and f defines the values. For any $G = (V, E)$, let $V = \{v_1, \ldots, v_{|G|}\}$ be a topological ordering of the vertices in G. Note that the vertices with $|s(v)| = 0$ are pure input vertices of the graph. We say

$$
G_f(\{v_i \,|\, i \le |G|, |s(v_i)| = 0\}) = (v_1, \ldots, v_{|G|}), \text{where}
\begin{cases}
v_1 = f(s(v_1)), \\
\ldots \\
v_n = f(s(v_n)),
\end{cases}
\tag{4}
$$

is the induced graph defined by f. The edge information is included in the topological ordering.

The arithmetic problems in F_p are DAGs. For instance, to calculate $(a_0 + a_1) \times (a_2 - (a_3/a_4))$, we can use the following steps $b_1 = f(a_0, +, a_1)$, $b_2 = f(a_3, /, a_4)$, $b_3 = f(a_2, -, b_2)$, and $b_4 = f(b_1, +, b_3)$, which defines a DAG. The DAG defines the calculation steps and the causal function f defines the arithmetic rules in F_p. When f changes, the induced values on G change. That's why we say G_f is the induced graph of G defined by f.

We first study the property of f. Denote the training dataset of f as D, which is a subset of the input space as

Figure 2: **An example of notations.**

$$
D \subseteq \{(f(s(v)), s(v)) \,|\, v \in V\}.
\tag{5}
$$

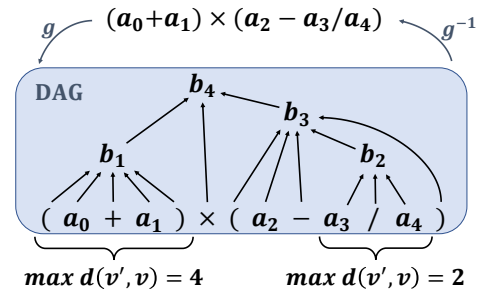D contains vertices from any G. For simplicity, denote $D \subseteq \mathbf{X} = X^{\sup|s(v)|}$ and $f(\mathbf{X}) = \{f(s(v))|s(v) \in \mathbf{X}\}$. We assume $|D| < \infty$, which is the case for any training dataset.

Theorem 3.3. *For $|X| < \infty$ and $\sup|s(v)| < \infty$, i.e. $|\mathbf{X}| < \infty$, if $D = \mathbf{X}$, then there exists an approximation function $\hat{f} : X^{\sup|s(v)|} \to X$, s.t. $\hat{f}(s(v)) = f(s(v)), \forall s(v) \in \mathbf{X}$.*

The proof is given in *Appendix* A. The only difference between Theorem 3.3 and Theorem 3.1 is the assumption $\sup|s(v)| < \infty$, which is similar to the Markov assumption. These assumptions make the domain of f to have finite dimensions, which guarantees the feasibility of learning f with $|D| < \infty$. Similar to the MDP, when $D \neq \mathbf{X}$, we have the following **negative** or **impossibility** corollaries (proofs are given in *Appendix* A).

Corollary 3.3.1. *For $|X| < \infty$ and $\sup|s(v)| < \infty$, i.e. $|\mathbf{X}| < \infty$, if $|f(\mathbf{X})| > 1$ and $D \neq \mathbf{X}$, then there exists an approximation function $\hat{f} : X^{\sup|s(v)|} \to X$ s.t. $\hat{f}(s(v)) = f(s(v)), \forall s(v) \in D$ and $\hat{f}(s(v)) \neq f(s(v)), \forall s(v) \in \mathbf{X} \backslash D$.*

Corollary 3.3.2. *For $\max(|X|, \sup|s(v)|) = \infty$, i.e. $|\mathbf{X}| = \infty$, if $|f(\mathbf{X})| > 1$, for $\forall m > 0$, there exists an approximation function $\hat{f} : X^{\sup|s(v)|} \to X$ s.t. $\hat{f}(s(v)) = f(s(v)), \forall s(v) \in D$ and $|\{s(v)|\hat{f}(s(v)) \neq f(s(v))\}| > m$.*

Similar to that in MDP, the causal function is guaranteed to be well-learned only when $|\mathbf{X}| < \infty$, regardless of the learning algorithm and the hypothesis space. Noticing the similarity between $\sup|s(v)| < \infty$ and the Markov property, it's natural to extend the conclusion of MDP to DAG.

We now show that a DAG can only be solved *recursively*.

Theorem 3.4. *For $|X| < \infty$ and $\sup|s(v)| < \infty$, if $D = X^{\sup|s(v)|}$, then there exists an approximation function $\hat{f} : X^{\sup|s(v)|} \to X$, the DAG can be recursively solved, i.e. $\forall G = (V, E)$, $G_{\hat{f}} = G_f$.*

The proof is given in *Appendix* A. Theorems 3.2 and 3.4 show that when the structure of the problem is given, and the inner causal function f is well-learned, a problem of any length or size can be solved.

So far we have assumed that the MDP or the DAG structure is given for a reasoning task. But in practice, neither of them is given. The next section deals with this more realistic scenario.

3.3 Unstructured Sequence Data

Sec. 3.1 and Sec. 3.2 assume the structure of the reasoning problem is given as a MDP or DAG. This section assumes the structure is unknown. We discuss the possibility to learn to transform an input sequence of unstructured elements into a structured DAG. We achieve this recursively by learning to find elements that should be reasoned next, like recursively doing topological ordering of the underlying DAG (we do not physically construct the DAG). For instance, $(a_0 + a_1) \times (a_2 - (a_3/a_4))$ is a sequence of elements (including '(' and ')') and the ordering of calculation needs to be learned. Note that we will not use MDP subsequently anymore but only DAG as DAG is more general and powerful.

Chain-of-Thought (CoT) [WWS$^+$22b] is a well-known approach to solving reasoning problems for unstructured data by giving the intermediate reasoning steps in the training data. We use $(a_0 + a_1) \times (a_2 - (a_3/a_4))$ in F_p as an example to see how CoT works and how it connects to DAG and recursive reasoning on DAG. The process of CoT is

$$(a_0 + a_1) \times (a_2 - (a_3/a_4)) = b_1 \times (a_2 - b_2) = b_1 \times b_3 = b_4. \tag{6}$$

CoT actually does three things or has three sub-steps in each reasoning/causal step. Sub-step 1 decides a part of the topological order of the underlying DAG, specifically, to find the vertices that will potentially be involved in the next reasoning step, i.e., vertices in $\{s(v)|v \in W^0\}$ (W^0 are vertices to be valued in the next reasoning step, which is defined later), before each reasoning step. Sub-step 2 applies the causal function (previously learned), e.g., $b_1 = f(a_0, +, a_1)$, $b_2 = f(a_3, /, a_4)$, etc. Sub-step 3 puts the result back into the

unstructured sequence. Since the arithmetic example decreases the number of elements at each step, sub-step 3 looks natural. However, it's possible to have a reasoning problem whose underlying DAG induces multiple vertices with multiple elements after one reasoning step. The representation by the unstructured data of the result after one reasoning step shouldn't be ambiguous; otherwise the reasoning problem itself is not well-defined.

To define a well-defined reasoning problem, we need more notations. For a DAG $G = (V, E)$, we have defined $s(v) = \{u \in V \mid \exists e \in E, u \xrightarrow{e} v\}$ to be all vertices that can reach v directly (v's in-neighbors). Now we define $t(v) = \{w \in V \mid \exists e \in E, v \xrightarrow{e} w\}$ to be all vertices that can be reached from v directly (v's out-neighbors). We define $(s \circ t)(v) = \{v' \in V \mid \exists w \in V, \exists e_1, e_2 \in E, v \xrightarrow{e_1} w, v' \xrightarrow{e_2} w\}$ to be vertices that can reach any causal vertex in $t(v)$.

Now we define what a well-defined reasoning problem is in an unstructured sequence. Denote the initial unstructured data to be a sequence of elements $s^0 = s_1^0 s_2^0 \ldots s_{i_0}^0$. Let g be the function that transforms the unstructured data into a structured DAG. We write $G^0 = (V^0, E^0) = g(s^0)$. We say a vertex is *valued* if a value is assigned to the vertex. In G^0, all vertices with $|s(v)| = 0$ have been valued as they are pure input vertices, but all causal vertices with $|s(v)| > 0$ haven't been valued. After one reasoning step on G^0, the vertices $W^0 = \{v \in G^0 \mid s(v) \subseteq \{v \in G^0 \mid |s(v)| = 0\}\}$ are valued. Then the vertices $U^0 = \{v \in \bigcup_{v \in W^0} s(v) \mid t(v) \subseteq W^0\}$, whose causal vertices are valued, become useless in the following reasoning steps. We denote $G^1 = G^0 \setminus U^0$ to be the subgraph of G^0 after removing all vertices in U^0 and all edges from U^0. To represent G^1, let g^{-1} be the inverse function that maps a DAG back to the unstructured data (which may not be unique). We write $s^1 = s_1^1 s_2^1 \ldots s_{i_1}^1 = g^{-1}(G^1)$.

We say the reasoning problem is *well-defined* with the unstructured data if $g(g^{-1}(G)) = G, \forall G$. Note that $g(g^{-1}(G)) = G$ represents that the DAG is isomorphic after the composing transformation $(g \circ g^{-1})$.

Now we describe the three sub-steps of CoT again, with a well-defined reasoning problem. Sub-step 1 is to find the input vertices $\{s(v) \mid v \in W^0\}$ and the next valued vertices $W^0 = \{v \in G^0 \mid s(v) \subseteq \{v \in G^0 \mid |s(v)| = 0\}\}$. Note that CoT only needs a one-step-forward subgraph instead of the entire graph. It's unnecessary to apply the original g. Instead, we define

$$\tilde{g}(s^0) \stackrel{def}{=} \{s(v) \mid v \in g(s^0), s(v) \subseteq \{v \in g(s^0) \mid |s(v)| = 0\}\}, \tag{7}$$

which finds the combinations of input vertices that infer causal vertices for one causal step. Different from g which constructs the entire graph, $\tilde{g}$ only needs to construct the graph for one causal/reasoning step.

Sub-step 2 applies the causal function f to infer

$$f(\tilde{g}(s^0)) \stackrel{def}{=} \{f(s(v)) \mid s(v) \in \tilde{g}(s^0)\}. \tag{8}$$

Sub-step 3 puts the resulting one-step-forward subgraph into the unstructured data. Note that $s_1^0, \ldots, s_{i_0}^0$ are vertices of $g(s^0)$, where $v \in s^0$ means v is a vertex of $g(s^0)$ that is valued by some $s_j^0 \in s^0$. Since the vertices s^0 and the one-step-forward vertices $f(\tilde{g}(s^0))$ construct a subgraph of $g(s^0)$, and g^{-1} is well-defined, we simply map the subgraph of vertices $(s^0, f(\tilde{g}(s^0)))$ into the unstructured data by

$$s^1 = g^{-1}(f(\tilde{g}(s^0)), s^0). \tag{9}$$

Sub-step 3 can be complex. For instance, there may exist input vertices that infer multiply causal vertices by f, i.e. $|f(s(v))| > 1$, then the question is how to represent $f(s(v))$ by the unstructured data. It's also a question where to put $f(s(v))$ back in s^0 when $s(v)$ is not a successive sub-sequence of s^0. The other question is how to deal with the useless vertices $U^0 = \{v \in \bigcup_{v \in W^0} s(v) \mid t(v) \subseteq W^0\}$.

In this work, we define $s^1 = g^{-1}(f(\tilde{g}(s^0)), s^0)$ by specific rules of the reasoning problem. But it's still a question whether sub-step 3 can be learned in general. We leave this for the future study.

In Sec. 3.2, we discussed the causal function, which is used in sub-step 2 of CoT. Now we discuss the possibility of learning sub-step 1. Denote

$$D \subseteq \{(\tilde{g}(s), s) \mid s = g^{-1}(G)\} \tag{10}$$

to be the dataset of $\tilde{g}$. G represents the underlying structure of the reasoning problem, and $s = g^{-1}(G)$ is the corresponding unstructured or sequence data of G. When $v_i = s_i^0$ and $v_j = s_j^0$, denote

$$d(v_i, v_j) \overset{def}{=} |i - j|, \tag{11}$$

which defines the distance between two vertices that were originally in s^0 to be their index distance in s^0.

We now introduce the important notion of R, the *maximal input element distance for a reasoning step*, which decides whether **length generation** can be achieved. Define

$$R \overset{def}{=} \sup_{} \max_{v' \in (s \circ t)(v)} d(v', v). \tag{12}$$

The sup is for any element v in the unstructured sequence of a problem, the $\max_{v' \in (s \circ t)(v)}$ is the maximal distance between any two elements that should involved in the next calculation/reasoning. R is the maximal distance between the elements that should be calculated next in the same calculation step of the problem.

Theorem 3.5. *For $R < \infty$, if $D = X^{4R+1}$, then there exists an approximation function $\hat{g} : X^{4R+1} \to 2^{4R+1}$ s.t. $\hat{g}(s) = \tilde{g}(s), \forall s \in X^{4R+1}$.*

The proof is given in *Appendix* C. It says that for any problem with G as the underlying DAG structure of the problem (G is not accessible in this section), $\forall v \in g^{-1}(G), \forall v' \in (s \circ t)(v), d(v', v)$ is uniformly bounded by a constant R. $R = \sup \max_{v' \in (s \circ t)(v)} d(v', v)$ is a primitive property of the reasoning DAG G and function g^{-1} transforms G into the unstructured data. For instance, if $s^0 = (a_0 + a_1) \times (a_2 - (a_3/a_4))$, we have $\tilde{g}(s^0) = \{(a_0 + a_1), (a_3/a_4)\}$. It's easy to see $R = 4$ for arithmetic problems with '(' and ')'.

Theorem 3.5 requires $D = X^{4R+1}$ because the combinations within a $(4R+1)$-length window cover all cases that enable a correct order to be learned. For instance, in '$\ldots e \times a + b + c \times d \ldots$', it's obvious that $R = 2$ because each operation or reasoning step involves 3 elements, e.g., $e \times a$, $a + b$, $b + c$, $c \times d$. To decide whether b should be calculated first, we look at neighbors of b in radius R, which is a window of length $2R + 1$. In this window $a + b + c$, we know $a + b$ should be calculated first. However, this is not enough because we also need to consider neighbors of a in radius R to validate whether $a + b$ should be calculated first. In the window $e \times a + b$, we know $e \times a$ should be calculated first. Therefore, in this $(4R+1)$-length window $e \times a + b + c \times d$, we know b shouldn't be calculated first. A $(4R+1)$-length window guarantees correctness when an element belongs to multiple combinations of input vertices. $D = X^M$ ($M < 4R+1$) cannot guarantee the correctness and an example is given in Appendix C.

When $R < \infty$, Theorem 3.5 holds, where a learned function $\hat{g}$ predicts which elements in a $(4R+1)$-length window should be calculated/reasoned next. When $|\mathbf{X}| < \infty$, Theorem 3.3 holds, where a learned causal function $\hat{f}$ takes the predicted elements of $\hat{g}$ as input to predict the value of the next/causal vertex. By Theorem 3.4, the recursive process solves the problem of arbitrary length, i.e., achieving **length generalization**.

When $R = \infty$, it means for any $N > 0$ in training, there exists an input element v s.t. $\max_{v' \in (s \circ t)(v)} d(v', v) > N$ in testing, i.e. the distance between elements to be reasoned next is arbitrarily large. This challenges learning as it's hard to learn an approximation function that adapts well to arbitrary input length. Our experiments haven't achieved length generalization for problems with $R = \infty$ (see Table 2 and Figure 3).

4   Related Work

This section reviews the related theoretical work. The empirical work is reviewed in *Appendix* E, which mianly consists of evaluations of and improvements made to the reasoning capabilities of LLMs and Chain-of-Thought (CoT) and its variants.

[ABLR23] studied out-of-distribution (OOD) generalisation of reasoning, in particular, an extreme case where part of the domain is entirely unseen during training, e.g., some value combinations are missing in training. It studies learning biases of different network architectures and activation functions. All resulting

models may predict wrong on the OOD data. They also analyzed length generalization based on their proposed bias and used curriculum learning to improve the performance of the parity problem. We work on a different problem as we identify and prove conditions under which the length generalization can be solved and our results are independent of learning algorithms, activate functions, and architectures.

[WLS23] proved that when sufficient intermediate steps (or CoT) are available, a neural network can efficiently learn any function in the **P** time complexity class. In addition, there exist functions in the **P** time complexity class that cannot be learned by any polynomial time learning algorithm without intermediate supervision (or CoT). Similarly, [FGZ$^+$23] showed why CoT works for mathematical and decision-making problems, especially those composite problems that can be decomposed into sub-problems, which the authors call dynamic programming problems (which is smilar to DAG in our paper). They first proved that it is impossible to predict the correct result directly without CoT, and then proved that CoT is capable of solving these problems. [LSG$^+$23] also attempted to demystify the mechanics underlying CoT. They showed that CoT can enable the model to identify each step (which they call *filtering*) and then to work on the step (which they call *in-context learning*) before moving to the next step in the CoT chain. [Mal23] proved that that not only sophisticated auto-regressive next-token predictors like Transformer can be an universal learner, but also simple models such as linear next-token predictors, trained on CoT data. The paper also introduces the *length complexity* to measure how many intermediate tokens are required to learn a function.

However, the theorems and proofs in these papers are based on the traditional i.i.d setting. Specifically, as discussed in Sec. 2, these studies all under the given length/size N. Their statements are something like "for $\forall N > 0$, given a dataset with training problems no longer than N, for any problem with length $N' \leq N$, it can be solved under some PAC upper bound." The primary limitation of their works is that either training problem length and testing problem length are the same, or the upper bound depends on the testing problem length. Our theory doesn't have this limitation and works on a more general scenario where training on smaller length problems and testing on larger length problems. Our statement is that "for $\forall N > 0$, given a dataset with problems no longer than N, for any problem with arbitrary length N', it can be solved if $|\mathbf{X}| < \infty$ and $R < \infty$."

5 Experiments

In previous sections, we have shown three key aspects for solving multi-step reasoning problems.

1) Given the structure of the problem in an MDP or a DAG, the causal function is well-learned only if the input space is finite i.e. $|\mathbf{X}| < \infty$, where the positive result is shown in Theorem 3.1 and Theorem 3.3 and the negative result is shown in Corollary 3.1.2 and Corollary 3.3.2.
2) Given the structure of the problem and a well-learned causal function, the reasoning problem of arbitrary sizes can be recursively solved, which is shown in Theorem 3.2 and Theorem 3.4.
3) Given unstructured sequence data (the realistic scenario), if $R < \infty$, we can find the elements to be calculated next to solve the problem recursively via CoT and to deal with length generalization. The sufficient condition is $R = \sup \max_{v' \in (s \circ t)(v)} d(v', v) < \infty$, which is an important primitive property that depends on the underlying DAG of the problem and the unstructured representation of the problem. For the same problem, one representation may have $R = \infty$, but another representation may have $R < \infty$.

Table 1: **Experimental settings**

	Train	Test 1	Test 2	Test 3	Test 4	Test 5
arctan	$r \in (1/2, 2)$	$r \in (1/3, 3)$	$r \in (1/4, 4)$	$r \in (1/5, 5)$	$r \in (1/6, 6)$	$r \in (1/10, 10)$
arithmetic in F_7	$L \in [3, 20)$	$L \in [3, 30)$	$L \in [3, 40)$	$L \in [3, 50)$	$L \in [3, 60)$	$L \in [3, 100)$
1-line addition	$a, b \in [0, 10^6)$	$a, b \in [0, 10^7)$	$a, b \in [0, 10^8)$	$a, b \in [0, 10^9)$	$a, b \in [0, 10^{10})$	$a, b \in [0, 10^{21})$
3-line addition	$a, b \in [0, 10^6)$	$a, b \in [0, 10^7)$	$a, b \in [0, 10^8)$	$a, b \in [0, 10^9)$	$a, b \in [0, 10^{10})$	$a, b \in [0, 10^{21})$
1-line multiplication	$a, b \in [0, 5)$	$a, b \in [0, 6)$	$a, b \in [0, 7)$	$a, b \in [0, 8)$	$a, b \in [0, 9)$	$a, b \in [0, 10)$

Our experiments fall into three categories. The *arctan* problem and the *arithmetic in F_7* (the finite prime field with seven elements) problem verify the 1st aspect. The *arithmetic* problem and the *3-line addition* problem verify the 2nd aspect. The *1-line addition* problem, *3-line addition* problem and *1-line multiplication* problem verify the 3rd aspect. We list the detailed parameters of training and testing datasets of each problem in Table 1 (explained below). The properties of each problem are listed in Table 2, for ease of connecting our theory and experimental results. The accuracy results are reported in Figure 3. Some examples in each experiment are given in Table 3 in *Appendix* D. The implementation details are also given in *Appendix* D.

The *arctan* problem verifies a simple fact that the causal function may make mistakes when *the input space is not finite*. The training data is sampled from an annulus of radius $r \in (1/2, 2)$. We test the performance on different annuluses, listed in Table 1. The performance decays as the annulus becomes larger, which satisfies Corollary 3.3.2.

Table 2: **Experimental primitive properties**

	$|X|$ in Theorem 3.3	R in Theorem 3.5
arctan	$|X| = \infty$	$R = 1$
arithmetic in F_7	$|X| = 13^5$	$R = 4$
1-line addition	$|X| = 14^3$	$R = \infty$
3-line addition	$|X| = 14^9$	$R = 1$
1-line multiplication	unknown	$R = \infty$

The *arithmetic in F_7* problem verifies three things. First, the input space of the causal function is finite. For each calculation step, we need at most 5 elements as the input, e.g., $(1+2)$. Therefore, since $|\mathbf{X}| = |\{(,), +, -, \times, /, 0, 1, 2, 3, 4, 5, 6\}|^5 < \infty$, by Theorem 3.3, the causal function can be well-learned. Second, it's obvious that when the DAG is known, we can simply apply the well-learned causal function to solve the problem of any length, which satisfies Theorem 3.4. Third, it's important to ask how we acquire the underlying DAG. By Theorem 3.5, if $R < \infty$, it is possible to perfectly predict the elements that should be calculated next, which is like progressively following the underlying DAG, but there is no need to physically construct the DAG. In the *arithmetic in F_7* problem, we have $R = 4$, because any combinations of elements that should be calculated next in a step is within a window of at most 5 elements, e.g., $(2+3)$. Finally, the problem satisfies all the positive theorems. The training and testing settings are listed in Table 1. The training data has at most 20 elements, i.e. $L \in [3, 20)$, and the testing data has at most 30, 40, 50, 60, 100 elements. The results in Figure 3 show this problem is perfectly solved.

The *1-line addition* problem and the *3-line addition* problem consider the same DAG with different unstructured data for representation. They are trained on 5-digits additions, i.e. $a, b \in [0, 10^6)$, and tested on 5, 6, 7, 8, 9, 20-digits additions respectively. The *1-line addition* problem fails to generalize beyond 5-digits, shown in Table 3. The *3-line addition* problem performs with 100% accuracy on 5,

Figure 3: **Experimental results in accuracy.**

6, 7, 8, 9, 20-digits. The two addition problems share the same DAG, but have different R's due to their different representations. For *1-line addition*, let us consider $x = $ '285 + 9805 =?'. The elements (or digits) to be calculated first are $x[2] = $ '5', $x[7] = $ '5' and $x[9] = $ '?'. The maximal distance between them is 7. This distance increases as number of digits increases. By definition, $R = \sup \max_{v' \in (s \circ t)(v)} d(v', v) = \infty$. But for 3-line addition, x becomes

$$x = \begin{pmatrix} 285 \\ 9805 \\ ? \end{pmatrix} = (\text{'}\ 285\text{'}, \text{'}9805\text{'}, \text{'}\quad ?\text{'})^T. \tag{13}$$

For each step, $x[i]$ only needs to consider its right neighbor $x[i+1]$. $x[3] = (\text{‘5’}, \text{‘5’}, \text{‘?’})^T$ is enough for calculation. $x[2] = (\text{‘8’}, \text{‘0’}, \text{‘ ’})^T$ only needs to consider $x[3]$. The maximal distance of elements to be calculated next is always 1, which doesn't depend on the number of digits. Therefore, $R = 1$.

For the *1-line multiplication* of two integers, we decompose multiplication into two stages. In the first stage, we transform multiplication into a summation of multiple integers. In the second stage, we solve the summation recursively. An example is shown in Table 3(d) in *Appendix* D. For the second stage, it's solvable by *3-line addition* in principle. However, we have $R = \infty$ when dealing with the first stage. For instance, let input[k] = '$a \times b = \underbrace{a + \cdots + a}_{k} + ?$'. When $k < b - 1$, and output[k] = '$a \times b = \underbrace{a + \cdots + a}_{k+1} + ?$'. when $k = b - 1$, output[k] = '$a \times b = \underbrace{a + \cdots + a}_{k+1}$'. In this example, Whether to add '+?' or to go to the second stage depends on b and the number of existing a's. Therefore, b and all existing a's are input elements in a reasoning step, where the maximal distance between them is arbitrary large as $b \to \infty$.

We have tried several other kinds of unstructured data for representing multiplication, including the traditional multi-line representation, but have not been able to design a good representation that solves the multiplication problem through CoT that can overcome the length generalization problem.

6 Discussions

In the discussion of the causal function, i.e. Theorem 3.1, Corollary, 3.1.1, Corollary 3.1.2, Theorem 3.3, Corollary 3.3.1 and Corollary 3.3.2, the theories show only the existence of a positive/negative function. Thus, our theories are qualitative rather than quantitative and are not connected to specific learning algorithms or model structures. Quantitatively, in the context of neural networks, existed works [WLS23, FGZ+23] have shown the learnability of the CoT-based learning for reasoning under the PAC-learning framework. Our work focuses only on length generalization, which is learning paradigm and algorithm independent and clearly is applicable to neural networks. [WLS23, FGZ+23] did not study the conditions for length generalization.

Our work considers reasoning problems that can be structured as a DAG. We don't know whether a reasoning problem (e.g., temporal and spatial reasoning) that cannot be represented as a DAG could be solved or not by CoT or the conditions under which it may be solvable to deal with length generalization.

We found the maximal input elements distance R to be an important quantity that decides whether a reasoning problem represented as a unstructured sequence could be learned to solve the length generalization problem. As demonstrated in the *1-line addition* and *3-lines addition* experiments, it is important to note the fact that different representations of the same reasoning problem may have different R's, which may decide whether the problem is solvable or not solvable. Intuitively, the *3-line addition* is more similar to how humans calculate a summation on a piece of 2-dimensional paper. We believe that it may not be rational to represent an inherently 2-dimensional task in one dimension like *1-line addition*. Thus, using a suitable dimension to represent a reasoning problem may be important. An interesting question is whether all the reasoning problems could be represented by a high dimensional representation with $R < \infty$. In other words, it is unclear whether there exist any reasoning problem that has no CoT based representations with $R < \infty$. We leave these questions to our future work.

References

[ABLR23] Emmanuel Abbe, Samy Bengio, Aryo Lotfi, and Kevin Rizk. Generalization on the unseen, logic reasoning and degree curriculum. *arXiv preprint arXiv:2301.13105*, 2023.

[AMA⁺23] Risako Ando, Takanobu Morishita, Hirohiko Abe, Koji Mineshima, and Mitsuhiro Okada. Evaluating large language models with neubaroco: Syllogistic reasoning ability and human-like biases. *arXiv preprint arXiv:2306.12567*, 2023.

[Ark23] Konstantine Arkoudas. Gpt-4 can't reason. *arXiv preprint arXiv:2308.03762*, 2023.

[AWA⁺22] 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.

[BJG08] Jørgen Bang-Jensen and Gregory Z Gutin. *Digraphs: theory, algorithms and applications*. Springer Science & Business Media, 2008.

[BMR⁺20] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[BZJ⁺23] Zhen Bi, Ningyu Zhang, Yinuo Jiang, Shumin Deng, Guozhou Zheng, and Huajun Chen. When do program-of-thoughts work for reasoning? *arXiv preprint arXiv:2308.15452*, 2023.

[CHL⁺22] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*, 2022.

[CHLL21] Yixuan Cao, Feng Hong, Hongwei Li, and Ping Luo. A bottom-up dag structure extraction model for math word problems. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 39–46, 2021.

[CMWC22] Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*, 2022.

[CSC⁺23] Yangyi Chen, Karan Sikka, Michael Cogswell, Heng Ji, and Ajay Divakaran. Measuring and improving chain-of-thought reasoning in vision-language models. *arXiv preprint arXiv:2309.04461*, 2023.

[CXS⁺23] Zhoujun Cheng, Tianbao Xie, Peng Shi, Chengzu Li, Rahul Nadkarni, Yushi Hu, Caiming Xiong, Dragomir Radev, Mari Ostendorf, Luke Zettlemoyer, et al. Binding language models in symbolic languages. *ICLR-2023*, 2023.

[DCLT18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[DLS⁺23] 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.

[FGZ⁺23] 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.

[FOC⁺23] Yao Fu, Litu Ou, Mingyu Chen, Yuhao Wan, Hao Peng, and Tushar Khot. Chain-of-thought hub: A continuous effort to measure large language models' reasoning performance. *arXiv preprint arXiv:2305.17306*, 2023.

[GBWD23] Gaël Gendron, Qiming Bao, Michael Witbrock, and Gillian Dobbie. Large language models are not abstract reasoners. *arXiv preprint arXiv:2305.19555*, 2023.

[GMZ+23] Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. In *International Conference on Machine Learning*, pages 10764–10799. PMLR, 2023.

[Has95] Mohamad H Hassoun. *Fundamentals of artificial neural networks*. MIT press, 1995.

[Hay98] Simon Haykin. *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1998.

[HCS21] Chadi Helwe, Chloé Clavel, and Fabian Suchanek. Reasoning with transformer-based models: Deep learning, but shallow reasoning. In *International Conference on Automated Knowledge Base Construction (AKBC)*, 2021.

[HLY+23] Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alexander Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. *Findings of the Association for Computational Linguistics (ACL2023)*, 2023.

[HQL+23] Pengbo Hu, Ji Qi, Xingyu Li, Hong Li, Xinqi Wang, Bing Quan, Ruiyu Wang, and Yi Zhou. Tree-of-mixed-thought: Combining fast and slow thinking for multi-hop visual reasoning. *arXiv preprint arXiv:2308.09658*, 2023.

[HYLZ23] Yi Hu, Haotong Yang, Zhouchen Lin, and Muhan Zhang. Code prompting: a neural symbolic method for complex reasoning in large language models. *arXiv preprint arXiv:2305.18507*, 2023.

[HZL+22] Fei Huang, Hao Zhou, Yang Liu, Hang Li, and Minlie Huang. Directed acyclic transformer for non-autoregressive machine translation. In *International Conference on Machine Learning*, pages 9410–9428, 2022.

[LFL+23] Zhan Ling, Yunhao Fang, Xuanlin Li, Zhiao Huang, Mingu Lee, Roland Memisevic, and Hao Su. Deductive verification of chain-of-thought reasoning. *arXiv preprint arXiv:2306.03872*, 2023.

[LHS+23] Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. Faithful chain-of-thought reasoning. *arXiv preprint arXiv:2301.13379*, 2023.

[LK23] Soochan Lee and Gunhee Kim. Recursion of thought: A divide-and-conquer approach to multi-context reasoning with language models. *arXiv preprint arXiv:2306.06891*, 2023.

[LL23] Tiedong Liu and Bryan Kian Hsiang Low. Goat: Fine-tuned llama outperforms gpt-4 on arithmetic tasks. *arXiv preprint arXiv:2305.14201*, 2023.

[LNT+23] Hanmeng Liu, Ruoxi Ning, Zhiyang Teng, Jian Liu, Qiji Zhou, and Yue Zhang. Evaluating the logical reasoning ability of chatgpt and gpt-4. *arXiv preprint arXiv:2304.03439*, 2023.

[Lon23] Jieyi Long. Large language model guided tree-of-thought. *arXiv preprint arXiv:2305.08291*, 2023.

[LSG+23] Yingcong Li, Kartik Sreenivasan, Angeliki Giannou, Dimitris Papailiopoulos, and Samet Oymak. Dissecting chain-of-thought: A study on compositional in-context learning of mlps. *arXiv preprint arXiv:2305.18869*, 2023.

[LSL+23] Nayoung Lee, Kartik Sreenivasan, Jason D Lee, Kangwook Lee, and Dimitris Papailiopoulos. Teaching arithmetic to small transformers. *arXiv preprint arXiv:2307.03381*, 2023.

[LYE23] Jiaxuan Li, Lang Yu, and Allyson Ettinger. Counterfactual reasoning: Testing language models' understanding of hypothetical scenarios. *arXiv preprint arXiv:2305.16572*, 2023.

[Mal23] Eran Malach. Auto-regressive next-token predictors are universal learners. *arXiv preprint arXiv:2309.06979*, 2023.

[MMJ+23] Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. Orca: Progressive learning from complex explanation traces of gpt-4. *arXiv preprint arXiv:2306.02707*, 2023.

[MVTF23] Jordan Meadows, Marco Valentino, Damien Teney, and Andre Freitas. A symbolic framework for systematic evaluation of mathematical reasoning with transformers. *arXiv preprint*

arXiv:2305.12563, 2023.

[MX23] Shentong Mo and Miao Xin. Tree of uncertain thoughts reasoning for large language models, 2023.

[NAGA⁺21] 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.

[NJL21] Rodrigo Nogueira, Zhiying Jiang, and Jimmy Lin. Investigating the limitations of transformers with simple arithmetic tasks. *arXiv preprint arXiv:2102.13019*, 2021.

[Ope22] OpenAI. Chatgpt: Optimizing language models for dialogue, 2022.

[Ope23] OpenAI. Gpt-4 technical report, 2023.

[PGZG23] Gabriel Poesia, Kanishk Gandhi, Eric Zelikman, and Noah D Goodman. Certified reasoning with language models. *arXiv preprint arXiv:2306.04031*, 2023.

[Pin99] Allan Pinkus. Approximation theory of the mlp model in neural networks. *Acta numerica*, 8:143–195, 1999.

[QWL⁺22] 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.

[QXS⁺23] Jingyuan Qi, Zhiyang Xu, Ying Shen, Minqian Liu, Di Jin, Qifan Wang, and Lifu Huang. The art of socratic questioning: Zero-shot multimodal reasoning with recursive thinking and self-questioning. *arXiv preprint arXiv:2305.14999*, 2023.

[SBS23] Alessandro Stolfo, Yonatan Belinkov, and Mrinmaya Sachan. Understanding arithmetic reasoning in language models using causal mediation analysis. *arXiv preprint arXiv:2305.15054*, 2023.

[SGC19] Shane Storks, Qiaozi Gao, and Joyce Y Chai. Commonsense reasoning for natural language understanding: A survey of benchmarks, resources, and approaches. *arXiv preprint arXiv:1904.01172*, pages 1–60, 2019.

[SH22] Abulhair Saparov and He He. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. *arXiv preprint arXiv:2210.01240*, 2022.

[SHH22] Zhihong Shao, Fei Huang, and Minlie Huang. Chaining simultaneous thoughts for numerical reasoning. *arXiv preprint arXiv:2211.16482*, 2022.

[SPBG23] Jingyuan Selena She, Christopher Potts, Samuel R Bowman, and Atticus Geiger. Scone: Benchmarking negation reasoning in language models with fine-tuning and in-context learning. *arXiv preprint arXiv:2305.19426*, 2023.

[SPK⁺23] Rylan Schaeffer, Kateryna Pistunova, Samar Khanna, Sarthak Consul, and Sanmi Koyejo. Invalid logic, equivalent gains: The bizarreness of reasoning in language model prompting. *arXiv preprint arXiv:2307.10573*, 2023.

[SSS⁺22] Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*, 2022.

[TNB23] Qingyu Tan, Hwee Tou Ng, and Lidong Bing. Towards benchmarking and improving the temporal reasoning capability of large language models. *arXiv preprint arXiv:2306.08952*, 2023.

[TZL⁺23] Xiaojuan Tang, Zilong Zheng, Jiaqi Li, Fanxu Meng, Song-Chun Zhu, Yitao Liang, and Muhan Zhang. Large language models are in-context semantic reasoners rather than symbolic reasoners. *arXiv preprint arXiv:2305.14825*, 2023.

[WDZ⁺23] Dingzirui Wang, Longxu Dou, Wenbin Zhang, Junyu Zeng, and Wanxiang Che. Exploring equation as a better intermediate meaning representation for numerical reasoning. *arXiv preprint arXiv:2308.10585*, 2023.

[WL23] Tianduo Wang and Wei Lu. Learning multi-step reasoning by solving arithmetic tasks. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1229–1238, 2023.

[WLC+23] Peiyi Wang, Lei Li, Liang Chen, Feifan Song, Binghuai Lin, Yunbo Cao, Tianyu Liu, and Zhifang Sui. Making large language models better reasoners with alignment. *arXiv preprint arXiv:2309.02144*, 2023.

[WLS23] Noam Wies, Yoav Levine, and Amnon Shashua. Sub-task decomposition enables learning in sequence to sequence tasks. *Procedings of International Conference on Learning Representations (ICLR-2023)*, 2023.

[WMD+22] Boshi Wang, Sewon Min, Xiang Deng, Jiaming Shen, You Wu, Luke Zettlemoyer, and Huan Sun. Towards understanding chain-of-thought prompting: An empirical study of what matters. *arXiv preprint arXiv:2212.10001*, 2022.

[WQR+23] Zhaofeng Wu, Linlu Qiu, Alexis Ross, Ekin Akyürek, Boyuan Chen, Bailin Wang, Najoung Kim, Jacob Andreas, and Yoon Kim. Reasoning or reciting? exploring the capabilities and limitations of language models through counterfactual tasks. *arXiv preprint arXiv:2307.02477*, 2023.

[WSC+23] Jianing Wang, Qiushi Sun, Nuo Chen, Xiang Li, and Ming Gao. Boosting language models reasoning with chain-of-knowledge prompting. *arXiv preprint arXiv:2306.06427*, 2023.

[WWS+22a] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

[WWS+22b] 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.

[XLH+23] Fangzhi Xu, Qika Lin, Jiawei Han, Tianzhe Zhao, Jun Liu, and Erik Cambria. Are large language models really good logical reasoners? a comprehensive evaluation from deductive, inductive and abductive views. *arXiv preprint arXiv:2306.09841*, 2023.

[XPL+23] Binfeng Xu, Zhiyuan Peng, Bowen Lei, Subhabrata Mukherjee, Yuchen Liu, and Dongkuan Xu. Rewoo: Decoupling reasoning from observations for efficient augmented language models. *arXiv preprint arXiv:2305.18323*, 2023.

[YDM+23] Zonglin Yang, Xinya Du, Rui Mao, Jinjie Ni, and Erik Cambria. Logical reasoning over natural language as knowledge representation: A survey. *arXiv preprint arXiv:2303.12023*, 2023.

[YLZ23] Yao Yao, Zuchao Li, and Hai Zhao. Beyond chain-of-thought, effective graph-of-thought reasoning in large language models. *arXiv preprint arXiv:2305.16582*, 2023.

[YSAN22] Mengjiao Sherry Yang, Dale Schuurmans, Pieter Abbeel, and Ofir Nachum. Chain of thought imitation with procedure cloning. *Advances in Neural Information Processing Systems*, 35:36366–36381, 2022.

[YTL+23] Fanglong Yao, Changyuan Tian, Jintao Liu, Zequn Zhang, Qing Liu, Li Jin, Shuchao Li, Xiaoyu Li, and Xian Sun. Thinking like an expert: Multimodal hypergraph-of-thought (hot) reasoning to boost foundation modals. *arXiv preprint arXiv:2308.06207*, 2023.

[YYZ+23] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*, 2023.

[YZW23] Fei Yu, Hongbo Zhang, and Benyou Wang. Nature language reasoning, a survey. *arXiv preprint arXiv:2303.14725*, 2023.

[ZBB+22] 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.

[ZYYY23] Yifan Zhang, Jingqin Yang, Yang Yuan, and Andrew Chi-Chih Yao. Cumulative reasoning with large language models. *arXiv preprint arXiv:2308.04371*, 2023.

Appendix

A Proofs - Causal Function

We firstly provide Lemma A.1, which is fundamental to our subsequent proofs. Lemma A.1 is similar to the *universal approximation theorem* (UAT) [Hay98, Has95, Pin99]. We do not directly use the UAT as it is for neural networks but our results are not bounded to any specific learning paradigm or algorithm. But due to the UAT, our results are applicable to learning in neural networks. The UAT guarantees that $\sup_{x \in K} ||f(x) - g(x)|| < \eta$, where $g(x)$ is the target function and K is compact. Applying the UAT in our setting of Lemma A.1, we have $\sup_{1 \le i \le n} ||f(x_i) - y_i|| < \eta$, where n is the number of training samples. The UAT has a condition that K is compact. In Lemma A.1, since we have a stronger condition that $K = \{(x_i, y_i) | 1 \le i \le n\}$ is finite, we provide a stronger result that $\sup_{1 \le i \le n} ||f(x_i) - y_i|| = 0$.

Lemma A.1 (A Simple Interpolating Function). *Let (X, d_X) be a metric space. Let $(Y, || \cdot ||)$ be a Banach space. For any $D \subseteq \{(x, y) | x \in X, y \in Y\}$, if $x \ne x'$, $\forall (x, y), (x', y') \in D$, then there exists a continuous approximation function $f : X \to Y$, s.t. $f(x) = y, \forall (x, y) \in D$.*

Proof of Lemma A.1: Let

$$\varepsilon = \min_{(x,y),(x',y') \in D} d_X(x, x').$$

Define

$$K_\varepsilon(x, x') = \frac{\varepsilon}{d_X(x, x')}.$$

Denote $D = \{(x_1, y_1), \dots, (x_n, y_n)\}$, let

$$f(x) = \frac{\sum_{i=1}^{n} y_i K_\varepsilon(x, x_i)}{\sum_{i=1}^{n} K_\varepsilon(x, x_i)}, x \in X \backslash \{x_1, \dots, x_n\}. \tag{14}$$

For $\forall 1 \le i_0 \le n$ (i_0 is any integer), since $\lim_{x \to x'} K_\varepsilon(x, x') = +\infty$, $\sup_{d_X(x, x') > \varepsilon} K_\varepsilon(x, x') < 1$ and

$$f(x) = \frac{y_{i_0} K_\varepsilon(x, x_{i_0})}{\sum_{i=1}^{n} K_\varepsilon(x, x_i)} + \frac{\sum_{i \ne i_0} y_i K_\varepsilon(x, x_i)}{\sum_{i=1}^{n} K_\varepsilon(x, x_i)},$$

it's obvious that $\forall \eta > 0$, $\exists \delta > 0$, s.t. $\forall d_X(x, x_{i_0}) < \delta$, $||f(x) - y_{i_0}|| < \eta$, which is $\lim_{x \to x_{i_0}} f(x) = y_{i_0}$. Therefore, we can define f on X and it's obvious that f is a continuous approximation function. ∎

Proof of Theorem 3.1: In the theorem presented in Section 3.1 in the main text, we wrote $D = Y \times X$ for simplicity. In detail, we have $D = \{((y, x), f(y, x)) | (y, x) \in Y \times X\}$.

By Lemma A.1, let $Y \times X$ be X and Y be Y, there exists $\hat{f}$ s.t. $\hat{f}(y, x) = f(y, x), \forall (y, x) \in D$. Since $D = Y \times X$, the proof is done. ∎

Proof of Corollary 3.1.1: In the corollary presented in Section 3.1 in the main text, we wrote $D \ne Y \times X$ for simplicity. In detail, we have $D \ne \{((y, x), f(y, x)) | (y, x) \in Y \times X\}$. For simplicity, denote $A = \{((y, x), f(y, x)) | (y, x) \in Y \times X\}$.

Since $D \ne A$, let $((y_0, x_0), f(y_0, x_0)) \in A \backslash D$. Since $|Y| > 1$, let $y^{err} \in Y$ s.t. $y^{err} \ne f(y_0, x_0)$. Let

$$D^{err} = D \cup \{((y_0, x_0), y^{err})\}.$$

By Lemma A.1, let $Y \times X$ be X and Y be Y, there exists $\hat{f}$ s.t. $\hat{f}(y, x) = y', \forall ((y, x), y') \in D^{err}$. Since $(y_0, x_0), y^{err}) \in D^{err}$, $\hat{f}$ makes a wrong prediction as

$$\hat{f}(y_0, x_0) = y^{err} \ne f(y_0, x_0).$$

Since $D \subseteq D^{err}$, $\hat{f}$ is correct on training dataset as

$$\hat{f}(y,x) = f(y,x), \, \forall \, ((y,x), f(y,x)) \in D.$$

∎

Proof of Corollary 3.1.2: Denote $D \subseteq \{((y,x), f(y,x)) \,|\, (y,x) \in Y \times X\}$. For simplicity, denote $A = \{((y,x), f(y,x)) \,|\, (y,x) \in Y \times X\}$.

Since $|X| = \infty$, $|A| = \infty$. Since $|D| < \infty$, we know $D \neq A$ and $|A \backslash D| = \infty$. Let $((y_i, x_i), f(y_i, x_i)) \in A \backslash D$, $i = 0, 1 \ldots, m$. Since $|Y| > 1$, let $y_i^{err} \in Y$ s.t. $y_i^{err} \neq f(y_i, x_i)$. Let

$$D^{err} = D \cup \{((y_i, x_i), y_i^{err}), i = 0, \ldots, m)\}.$$

By Lemma A.1, let $Y \times X$ be X and Y be Y, there exists $\hat{f}$ s.t. $\hat{f}(y,x) = y', \, \forall \, ((y,x), y') \in D^{err}$. Since $((y_i, x_i), y_i^{err}) \in D^{err}$, $\hat{f}$ makes $m+1$ mistakes as

$$\hat{f}(y_i, x_i) = y_i^{err} \neq f(y_i, x_i), i = 0, 1, \ldots, m.$$

Since $D \subseteq D^{err}$, $\hat{f}$ is correct on training dataset as

$$\hat{f}(y,x) = f(y,x), \, \forall \, ((y,x), f(y,x)) \in D.$$

∎

Proof of Theorem 3.3: By Lemma A.1, let $X^{\sup |s(v)|}$ be X and X be Y, there exists $\hat{f}$ s.t. $\hat{f}(s(v)) = v', \, \forall \, (v', s(v)) \in D$. Since $D = \{(f(s(v)), s(v)) \,|\, v \in V\}$, the proof is done. ∎

Proof of Corollary 3.3.1: In the corollary presented in Section 3.2 in the main text, we wrote $D \neq X^{\sup |s(v)|}$ for simplicity. In detail, we have $D \neq \{(f(s(v)), s(v)) \,|\, v \in V\}$. For simplicity, denote $A = \{(f(s(v)), s(v)) \,|\, v \in V\}$.

Since $D \neq A$, let $(f(s(v_0)), s(v_0)) \in A \backslash D$. Since $|\{f(s(v) \,|\, s(v) \in X^{\sup |s(v)|})\}| > 1$, let $v^{err} \in \{f(s(v) \,|\, s(v) \in X^{\sup |s(v)|})\}$ s.t. $v^{err} \neq f(s(v_0))$. Let

$$D^{err} = D \cup \{(v^{err}, s(v_0))\}.$$

By Lemma A.1, let $X^{\sup |s(v)|}$ be X and X be Y, there exists $\hat{f}$ s.t. $\hat{f}(s(v)) = v', \, \forall \, (v', s(v)) \in D^{err}$. Since $(v^{err}, s(v_0)) \in D^{err}$, $\hat{f}$ makes a wrong prediction as

$$\hat{f}(s(v_0)) = v^{err} \neq f(s(v_0)).$$

Since $D \subseteq D^{err}$, $\hat{f}$ is correct on training dataset as

$$\hat{f}(s(v)) = f(s(v)), \, \forall \, (f(s(v)), s(v)) \in D.$$

∎

Proof of Corollary 3.3.2: Denote $D \subseteq \{(f(s(v)), s(v)) \,|\, v \in V\}$. For simplicity, denote $A = \{(f(s(v)), s(v)) \,|\, v \in V\}$.

Since $\max(|X|, \sup |s(v)|) = \infty$, $|X^{\sup |s(v)|}| = \infty$, $|A| = \infty$. Since $|D| < \infty$, we know $D \neq A$ and $|A \backslash D| = \infty$. Let $(f(s(v_i)), v_i)) \in A \backslash D$, $i = 0, 1 \ldots, m$. Since $|\{f(s(v) \,|\, s(v) \in X^{\sup |s(v)|})\}| > 1$, let $v_i^{err} \in \{f(s(v) \,|\, s(v) \in X^{\sup |s(v)|})\}$ s.t. $v_i^{err} \neq f(s(v_i))$. Let

$$D^{err} = D \cup \{(f(s(v_i), v_i)), i = 0, \ldots, m)\}.$$

By Lemma A.1, let $X^{\sup|s(v)|}$ be X and X be Y, there exists $\hat{f}$ s.t. $\hat{f}(s(v)) = v'$, $\forall (v', s(v)) \in D^{err}$. Since $(v_i^{err}, s(v_i)) \in D^{err}$, $\hat{f}$ makes $m+1$ mistakes as

$$\hat{f}(s(v_i)) = v_i^{err} \neq f(s(v_i)), i = 0, 1, \ldots, m.$$

Since $D \subseteq D^{err}$, $\hat{f}$ is correct on training dataset as

$$\hat{f}(s(v)) = f(s(v)), \forall (f(s(v)), s(v)) \in D.$$

■

B Proofs - Recursive Formula

Proof of Theorem 3.2: By Theorem 3.1, there exists $\hat{f}$ s.t. $\hat{f}(y,x) = f(y,x)$, $\forall (y,x) \in Y \times X$. Since τ is a Markov dynamic process, by the definition in Eq. (2), we have

$$\tau_f(x_{[0:\infty]}) = (y_1, y_2, \ldots), \text{ where } \begin{cases} y_1 = f(y_0, x_0), \\ \ldots \\ y_{n+1} = f(y_n, x_n). \end{cases}$$

By induction, we have

$$y_1 = f(y_0, x_0) = \hat{f}(y_0, x_0) = \hat{y}_1,$$
$$\ldots$$
$$y_{n+1} = f(y_n, x_n) = \hat{f}(\hat{y}_n, x_n) = \hat{y}_{n+1}.$$

■

Proof of Theorem 3.4: By Theorem 3.3, there exists $\hat{f}$, s.t. $\hat{f}(s(v)) = f(s(v))$, $\forall (f(s(v)), s(v)) \in \{(f(s(v)), s(v)) | v \in V\}$.

By the definition in Eq. (4) in Section 3.2, for an arbitrary topological ordering of the graph G, we have

$$G_f(\{v_i, i \leq |G| \,|\, d(v_i) = 0\}) = (v_1, \ldots, v_{|G|}), \text{where} \begin{cases} v_1 = f(s(v_1)), \\ \ldots \\ v_n = f(s(v_n)). \end{cases}$$

Given the graph G and the topological ordering, by induction, we have

$$v_1 = f(s(v_1)) = \hat{f}(s(v_1)) = \hat{v}_1,$$
$$\ldots$$
$$v_n = f(s(v_n)) = \hat{f}(s(\hat{v}_n)) = \hat{v}_n.$$

■

C Proof - Maximal Input Element Distance of a Reasoning step

Before presenting the proof for Theorem 3.5, we first provide an example to show that a $4R$-length window is not enough to decide which elements should be used in the next reasoning or calculation step.

Let's consider a simple one-dimensional *ko* problem. Let $s = \ldots 001001111 \ldots$ be a sequence of 0's and 1's. We say s_i is captured if $s_i \neq s_{i-1}$, $s_i \neq s_{i+1}$. We say s_i is not a ko if both s_{i-1} and s_{i+1} are not captured. We say s_i is a ko if s_i is captured and at least one of s_{i-1} and s_{i+1}
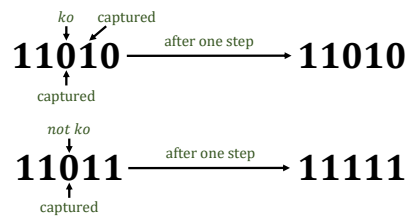

Figure 4: **Two examples of the *ko* problem.**

19

is captured. When some s_i is captured and is not a ko, we set $s_i = 1 - s_i$. We continue the process until s won't change, which is the final settlement of the problem. In this problem, if s_i is captured and is not a ko, it should be acted upon next (changing s_i's value). The neighbors s_{i-1} and s_{i+1} are enough to decide the value of s_i, i.e. $\max_{v' \in (s \circ t)(s_i)} d(v', s_i) = \max(d(s_{i-1}, s_i), d(s_{i+1}, s_i)) = 1$. Therefore, $R = 1$ in this problem.

However, a $4R$-length window is not enough to decide whether s_i should be reasoned or acted upon next. For instance, let $s = 11010$. In this problem, $s_{[2]} = 0$ is captured and is a ko, so it cannot be changed. Therefore, when we only consider 4 elements $s_{[0:4]} = 1101$, we will receive the ground truth label $y_{[0:4]} = 0000$, where 0 indicates the element shouldn't be changed and 1 indicates the element should be changed. However, when we consider a problem $s' = 11011$, $s'_{[2]} = 0$ is captured and is not a ko, so it should be changed. Therefore, when we only consider 4 elements $s'_{[0:4]} = 1101$, the ground truth label is $y'_{[0:4]} = 0010$. In these two problems, they share an identical $4R$-length window 1101, but the labels of this window are different, which is obviously ambiguous.

Proof of Theorem 3.5: Denote the training dataset as

$$D \subseteq \{(\tilde{g}(s), s) | s = s_0 s_1 \ldots s_{4R}, s = g^{-1}(G)\},$$

where $\tilde{g}(s) = 1_0(s_0) 1_1(s_1) \ldots 1_{4R}(s_{4R})$ is a $(4R+1)$-length binary sequence that $1_i(s_i)$ indicates whether s_i should be reasoned next.

By Lemma A.1, let X^{4R+1} to be X and 2^{4R+1} to be Y, there exists $\hat{f}$ s.t. $\hat{f}(s) = I, \forall (I, s(v)) \in D$. Since $D = \{(\tilde{g}(s), s) | s = s_0 s_1 \ldots s_{4R}, s = g^{-1}(G)\}$, the proof is done. ∎

Table 3: **Experimental data examples**

(a) arctan

Input[0]:	a, b
Output[0]:	arctan(a/b)

(b) arithmetic in prime field F_7

Input[0]:	(0+4-(2-3*6))*(4+0)
Output[0]:	(4 -(2- 4))* 4
Input[1]:	(4-(2-4))*4
Output[1]:	(4- 5)*4
Input[2]:	(4-5)*4
Output[2]:	6 *4
Input[3]:	6*4
Output[3]:	3

(c) 1-line addition

Input[0]:	285+9805=?
Output[0]:	285+9805=c0
Input[1]:	285+9805=c0
Output[1]:	285+9805=?90
Input[2]:	285+9805=?90
Output[2]:	285+9805=c090
Input[3]:	285+9805=c090
Output[3]:	285+9805=10090

(d) 1-line multiplication

Input[0]:	1*3=?
Output[0]:	1*3=1+?
Input[1]:	1*3=1+?
Output[1]:	1*3=1+1+?
Input[2]:	1*3=1+1+?
Output[2]:	1*3=1+1+1
Input[3]:	1*3=1+1+1
Output[3]:	1*3=2+1
Input[4]:	1*3=2+1
Output[4]:	1*3=3

(e) 3-line addition

Input[0]:	89283 3360 ?
Output[0]:	?3
Input[1]:	89283 3360 ?3
Output[1]:	c43
Input[2]:	89283 3360 c43
Output[2]:	?643
Input[3]:	89283 3360 ?643
Output[3]:	c2643
Input[4]:	89283 3360 c2643
Output[4]:	92643

D Experimental Data and Implementation Details

The *arithmetic in F_7* problem is defined in the prime field F_7. The *1-line addition* problem and the *3-line addition* problem use '?' to represent 0 being carried from the right and 'c' to represent 1 being carried from the right. In all problems, * is equivalent to $\times$. We use * instead of $\times$ for ease of aligning chars.

All the problems are trained with 50*k* batches. Each batch contains 256 CoT steps. For each problem, we first randomly generate a question and then its detailed CoT steps. Each CoT step is a pair (Input[i], Output[i]), as shown in Table 3. We put the CoT steps into a batch until the batch size reaches 256.

When testing the performance after training, we test 6 different datasets for each problem, which is shown in Table 1. Each testing dataset is independently generated with 1*k* questions. We solve each question by CoT using the trained model. For the *arctan* problem, since it only has one step, we only inference in one step. For the *arithmetic* problem, we stop the CoT output generation in a step when (i) the output of the step has only one token/element, or (ii) the output of the step is identical to the input of the step. For *1-line addition* and *3-line addition*, we stop the CoT output generation if (i) '?' and 'c' aren't shown in the output, or (ii) the number of CoT steps is greater than the number of digits. For the *1-line multiplication* problem, we stop the CoT output generation if (i) '+' is not shown in the output, or (ii) the number of CoT steps is greater than the multipliers.

For the *arctan* problem, the predicted value is considered correct if the absolute error is smaller than 0.01. For the other problems, the final output is considered correct only when it is identical to the ground truth. For the dataset of each problem, the accuracy is the number of correctly answered questions divided by the total number of questions.

The model of the *arctan* problem has 3 fully connected layers. The model of the *arithmetic* problem, the *1-line addition* problem, the *3-lines addition* problem and the *1-line multiplication* problem has 3 Transformer encoders with relative position embedding. The optimizer is Adam and the learning rate is 0.0001. The training data for each task contains 12.8M CoT steps and was trained for 1 epoch.

E   Related Empirical Work

Evaluations and limitations of LLM reasoning. Continuing with the discussion about evaluations of the reasoning capabilities of LLMs in Section 1, we present a more extensive literature survey here. Note that Sections 1 and 2 have discussed empirical works about length generalization [AWA$^+$22, DLS$^+$23, ZBB$^+$22]. In these papers, the authors also tried to mitigate the problem through improved training [AWA$^+$22], improved promoting and fine-tuning of LLMs [ZBB$^+$22], and curriculum learning [ABLR23]. However, none of them studied the length generalization problem theoretically as we do. Below, we focus on surveying other empirical works. Many of them identified limitations of LLMs in solving different reasoning problems, but few have characterized the limitations in a formal manner to facilitate theoretical investigation.

[MVTF23] created a dataset specifically for mathematical reasoning that can be perturbed. They showed that perturbations of the tasks heavily affect the results, reducing F1 score from 97% to 17%, which suggests that inference is likely to be dominated by surface-level patterns unrelated to the deeper understanding of the mathematical operators. However, this evaluation was done using only BERT [DCLT18] based models, but not on more recent LLMs like ChatGPT and GPT4. [WQR$^+$23] used "counterfactual" tasks that deviate from the standard reasoning tasks to evaluate LLMs. It was found that the performance degrades substantially compared to the default conditions, which again suggests that while LLMs can perform reasoning to some extend, they often rely on narrow, non-transferable procedures or surface patterns for task-solving. A counterfactual based evaluation was also done in [LYE23], which reached the same conclusion.

[LNT$^+$23] evaluated ChatGPT and GPT-4 on logical reasoning. The results showed that they do relatively well on well-known public domain datasets, but their performances drop substantially when newly released and out-of-distribution datasets are used. [XLH$^+$23] too evaluated LLMs using logical reasoning (deductive, inductive, abductive and mixed-form reasoning) and gave pros and cons of LLMs. [SPBG23] created a dataset for reasoning involving negations and evaluated LLMs and showed poor results. [AMA$^+$23] created a dataset, originally designed for psychological experiments to assess human logical abilities in syllogistic reasoning. The authors examined three types of biases observed in human syllogistic reasoning: *belief biases*, *conversion errors*, and *atmosphere effects*. The evaluation on LLMs showed that they struggle with problems involving these biases too. [TNB23] created a dataset to evaluate LLMs on temporal reasoning and showed some weaknesses of LLMs. They then proposed an approach to improve the results.

Chain of thoughts (CoT) and variants. Earlier prompting for solving reasoning problems using LLMs only states the question and the answer. They found that these two pieces of information are insufficient for LLMs to learn to perform effective reasoning, which we have proved. Then *chain of thought* (CoT) prompting [WWS+22b] was proposed to improve the situation. CoT basically contains the detailed intermediate reasoning steps between the question and the answer for fine-tuning the LLMs, which significantly enhance LLMs' reasoning capabilities [CHL+22, HLY+23, MMJ+23, FOC+23]. [SH22] created a synthetic dataset generated based on first-order logic. They then parsed the generated CoT into symbolic proofs for formal analysis. It was shown that LLMs are capable of reasoning. The success of CoT has encouraged researchers to refine the technique and also propose variations of the technique.

For example, [CSC+23] proposed a metric to measure the effectiveness of CoT and a technique to improve CoT for vision-language models. [WLC+23] studied using multiple reasoning paths and positive and negative answers to improve CoT reasoning. [ZYYY23] proposed cumulative reasoning, which employs LLMs in a cumulative and iterative manner to emulate the human thought process. [QXS+23] proposed a divide-and-conquer algorithm that simulates the self-questioning and recursive thinking process of humans to improve CoT. [WL23] investigated how to incorporate into relatively small LMs the capabilities of multi-step reasoning and CoT. [WMD+22] found that even logically invalid CoT also helps reasoning. This was confirmed in [SPK+23]. To deal with unsound inferences, [PGZG23] introduced a class of external tools for LLMs called guides that use states and incremental constraints to guide the generation in reasoning. A related work on using external tools was done in [XPL+23]. [WWS+22a] improved CoT using multiple-paths and consistency check. [LFL+23] studied the verification of CoT. [SBS23] identified part of an LLM responsible for reasoning. In a different direction, [YSAN22] argued that the prevailing approach to CoT prompt selection through trial and error is unsatisfactory. They then proposed a principled approach for multi-domain LLM CoT prompt selection.

Several researchers also broadened the CoT method and proposed the neural symbolic *code prompting* [HYLZ23], *program of thoughts* [CMWC22, CXS+23], *tree-of-thoughts* [YYZ+23, Lon23], *tree-of-mixed-thoughts* [HQL+23], *tree of uncertain thoughts* [MX23], *hypergraph-of-thoughts* [YTL+23], *recursion of thoughts* [LK23], *chain of knowledge* [WSC+23], *chain of simultaneous thoughts* [SHH22], *graph-of-thoughts* [YLZ23] and *faithful chain of thoughts* [LHS+23]. Further, [BZJ+23] proposed a complexity measure and chose the optimal complexity to improve the *program of thoughts* [CMWC22]. [WDZ+23] proposed a method to improve the generation of equations from natural language questions as the intermediate step to answer the original question. [GMZ+23] combined CoT and Program-Aided Language Models (PAL) for improved reasoning.