A Theory of Learning with Autoregressive Chain of Thought

Nirmit Joshi   NIRMIT@TTIC.EDU
Toyota Technological Institute at Chicago

Gal Vardi   GAL.VARDI@WEIZMANN.AC.IL
Weizmann Institute of Science

Adam Block   ADAM.BLOCK@COLUMBIA.EDU
Microsoft Research, NYC and Columbia University

Surbhi Goel   SURBHIG@CIS.UPENN.EDU
University of Pennsylvania

Zhiyuan Li   ZHIYANLI@TTIC.EDU
Toyota Technological Institute at Chicago

Theodor Misiakiewicz   THEODOR.MISIAKIEWICZ@YALE.EDU
Yale University

Nathan Srebro   NSREBRO@TTIC.EDU
Toyota Technological Institute at Chicago

Editors: Nika Haghtalab and Ankur Moitra

Abstract

For a given base class of sequence-to-next-token generators, we consider learning prompt-to-answer mappings obtained by iterating a fixed, time-invariant generator for multiple steps, thus generating a chain-of-thought, and then taking the final token as the answer. We formalize the learning problems both when the chain-of-thought is observed and when training only on prompt-answer pairs, with the chain-of-thought latent. We analyze the sample and computational complexity both in terms of general properties of the base class (e.g. its VC dimension) and for specific base classes such as linear thresholds. We present a simple base class that allows for universal representability and computationally tractable chain-of-thought learning. Central to our development is that time invariance allows for sample complexity that is independent of the length of the chain-of-thought. Attention arises naturally in our construction.

Keywords: Autoregressive Models, Chain of Thought, Universal Learning

Please see arXiv 2503.07932 for the complete version.

1. Introduction

Autoregressive generation and learning, particularly with attention-based models such as transformers, is driving remarkable advances in Artificial Intelligence and increasingly becoming synonymous with AI itself. To solve complex tasks, especially those requiring multi-step or compositional reasoning and computation, autoregressive generation produces a Chain-of-Thought, consisting of multiple intermediate tokens, that ultimately leads to the desired answer. In this paper, we propose a formal framework for studying this emerging paradigm: learning complex functions through autoregressive Chain-of-Thought generation using a simple next-token generator. We analyze the statistical and computational benefits and pitfalls of this approach, and see how attention naturally arises as a key ingredient for "universal" learning with autoregressive Chain-of-Thought generation.

In our view, a central component of autoregressive generation is that it is **time-invariant**, i.e., the same next-token-generator, with the identical parameters (e.g. a transformer with the same

weights) is used at each step of generation, to generate each token as a function of the prefix thus far. Throughout the paper we emphasize how such time-invariance is crucial for allowing learning with sample complexity independent (perhaps up to a log-factor) of the generation length, and later on, of the compute time (number of steps) of a learned process. In this crucial regard, we deviate significantly from another recent (and inspiring) attempt to formalize autoregressive learning by Malach (2023), who considered time-*dependent* autoregressive learning. We directly contrast with Malach's approach as we progress through our presentation.

We begin by presenting a formal setting for time-invariant autoregressive Chain-of-Thought generation and learning using an abstract base class of next-token-generators (Section 2), studying the basics of learning in this framework for general and abstract base classes (Section 3), and, as an example, for what is perhaps the simplest possible base class, namely linear thresholds (Section 4). We consider training both with and without explicit Chain-of-Thought supervision. Already here we can start seeing the computational benefits of Chain-of-Thought learning and the statistical benefits of time-invariant autoregressive generation. We then turn to the more ambitious goal of "universal" learning of computable functions and see how far the benefits of time-invariance and autoregressive Chain-of-Thought take us, and how attention arises naturally (Sections 5 and 6).

2. Time-Invariant Autoregressive Chain-of-Thought Learning

For a finite set of tokens Σ, a *next token generator* is a mapping $f : \Sigma^* \to \Sigma$. On input $\boldsymbol{x} \in \Sigma^*$, autoregressive Chain-of-Thought (CoT) generation will start with a string containing only the input and will repeatedly apply f to the string and append the newly generated token to it (see Figure 1). Formally, for a next-token generator f, we define the apply-and-append mapping $\overline{f}(\boldsymbol{x}) : \boldsymbol{x} \mapsto$ append$(\boldsymbol{x}, f(\boldsymbol{x})) \in (\Sigma^{|\boldsymbol{x}|+1})$,[1] and apply $\overline{f}$ iteratively for T steps to obtain a mapping from $\boldsymbol{x}$ to its T-step Chain-of-Thought:

$$f^{\mathsf{CoT}\text{-}T}(\boldsymbol{x}) = \underbrace{\overline{f} \circ \overline{f} \circ \ldots \circ \overline{f}}_{T \text{ times}}(\boldsymbol{x}) \in \Sigma^{|\boldsymbol{x}|+T}. \tag{1}$$

We think of all but the last of these T tokens as intermediate "thinking", and only the final token after T steps as the "answer" and so consider the following end-to-end mapping between an input and the final token in $f^{\mathsf{CoT}\text{-}T}(\boldsymbol{x})$:

$$f^{\mathsf{e2e}\text{-}T}(\boldsymbol{x}) = f^{\mathsf{CoT}\text{-}T}(\boldsymbol{x})[-1]. \tag{2}$$

For any base class $\mathcal{F} = \{f : \Sigma^* \to \Sigma\} \subseteq \Sigma^{\Sigma^*}$ of generators, and a generation length T, we define the corresponding end-to-end and CoT classes:

$$\mathcal{F}^{\mathsf{e2e}\text{-}T} = \{f^{\mathsf{e2e}\text{-}T} : \Sigma^* \to \Sigma \mid f \in \mathcal{F}\} \quad \text{and} \quad \mathcal{F}^{\mathsf{CoT}\text{-}T} = \{f^{\mathsf{CoT}\text{-}T} : \Sigma^* \to \Sigma^* \mid f \in \mathcal{F}\}. \tag{3}$$

For simplicity, throughout the paper we consider a fixed thought generation length (i.e. number of iterations) T. This number of iterations T will be a crucial parameter for us and we will study how the learning complexity and expressivity depend on T.[2]

---

1. For any $\boldsymbol{x} \in \Sigma^*$, we use $|\boldsymbol{x}|$ to denote its length. We use negative indices to refer to elements of a vector or string from the end, i.e. $\boldsymbol{v}[-1]$ is the last element. We use inclusive slice indexing, so that $\boldsymbol{v}[i : j]$ are the elements from position i to j inclusive. Dropping i or j means that the slice extends from the beginning or to the end, respectively.

2. One could also define a variant with variable thought lengths and, e.g., a special token indicating when to stop. But if we would anyway be interested in the maximal thought length, we could just as well extend each thought with a padding token to make them all of the same length.
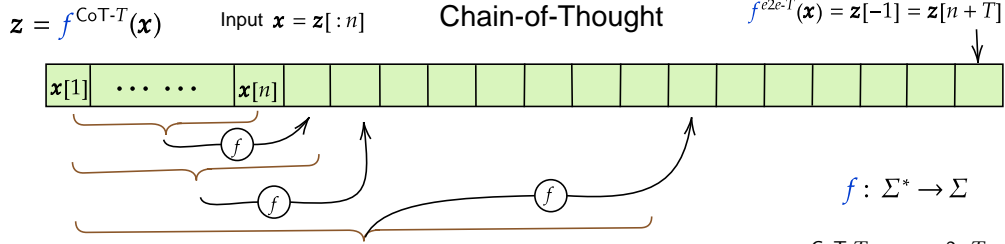
Figure 1: Illustration of Chain-of-Thought generation and the functions $f^{\text{CoT-}T}$ and $f^{\text{e2e-}T}$.

Learning. Our goal is to learn the hypothesis class $\mathcal{F}^{\text{e2e-}T}$. Throughout we consider a realizable setting, i.e. we assume there exists some ground truth $f_* \in \mathcal{F}$, and for an input distribution $\mathcal{D}$ over $\mathcal{X} \subseteq \Sigma^*$ (if not explicitly specified, we always take the domain as $\mathcal{X} = \Sigma^*$), we would like to find a predictor $h : \mathcal{X} \to \Sigma$ that minimizes the population error:

$$L^{\text{0-1}}_{\mathcal{D},f_*}(h) := \mathbb{P}_{\boldsymbol{x} \sim \mathcal{D}}\left(h(\boldsymbol{x}) \neq f_*^{\text{e2e-}T}(\boldsymbol{x}) \right).$$

We will consider two learning settings that differ by what data are available during training. First, we will consider learning based only on observing the final answer $f_*^{\text{e2e-}T}(\boldsymbol{x})$:

Definition 1 (Realizable End-to-End Learnability) *For a base class $\mathcal{F} \subseteq \Sigma^{\Sigma^*}$, and generation length $T \in \mathbb{N}_+$, we say that $\mathcal{F}^{\text{e2e-}T}$ is e2e-learnable over a domain $\mathcal{X} \subseteq \Sigma^*$, with sample complexity $m(\varepsilon,\delta)$ if there exists a learning rule $A : (\mathcal{X} \times \Sigma)^* \to \Sigma^{\mathcal{X}}$ such that for every distribution $\mathcal{D}$ over $\mathcal{X}$ and $f_* \in \mathcal{F}$, for every $0 < \varepsilon, \delta < 1$, with probability at least $1 - \delta$ over $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m \sim_{iid} \mathcal{D}$, with $S_{\text{e2e}} = ((\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m))$, $y_i = f_*^{\text{e2e-}T}(\boldsymbol{x}_i)$ and $m = m(\varepsilon,\delta)$, we have $L^{\text{0-1}}_{\mathcal{D},f_*}(A(S_{\text{e2e}})) \leq \varepsilon$.*

Definition 1 amounts precisely to the standard definition of PAC-learning the hypothesis class $\mathcal{F}^{\text{e2e-}T}$, but we give it explicitly in order to contrast it with learning in situations where the entire Chain-of-Thought is available during training. In contemporary LLM training, such CoT supervision can be found in data sources such scrapped over internet consisting of step-by-step answers to questions in discussion forums and in textbooks (Lewkowycz et al., 2022; Li et al., 2023; Gao et al., 2020; Achiam et al., 2023), as well as in training data specifically curated (Ott et al., 2023; Chung et al., 2024; Hendrycks et al., 2021).

Definition 2 (Realizable Chain-of-Thought Learnability) *For a base class $\mathcal{F} \subseteq \Sigma^{\Sigma^*}$ and generation length $T \in \mathbb{N}_+$, we say that[3] $\mathcal{F}^{\text{e2e-}T}$ is CoT-learnable over a domain $\mathcal{X} \subseteq \Sigma^*$, with sample complexity $m(\varepsilon,\delta)$, if there exists a learning rule $A : (\mathcal{X} \times \Sigma^T)^* \to \Sigma^{\mathcal{X}}$ such that for every distribution $\mathcal{D}$ over $\mathcal{X}$ and $f_* \in \mathcal{F}$, for every $0 < \varepsilon, \delta < 1$, with probability at least $1 - \delta$ over $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m \sim_{iid} \mathcal{D}$, with $S_{\text{CoT}} = (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_m)$ where $\boldsymbol{z}_i = f_*^{\text{CoT-}T}(\boldsymbol{x}_i)$, and $m = m(\varepsilon,\delta)$, we have*

$$L^{\text{0-1}}_{\mathcal{D},f_*}(A(S_{\text{CoT}})) = \mathbb{P}_{\boldsymbol{x} \sim \mathcal{D}}\left(A(S_{\text{CoT}})(\boldsymbol{x}) \neq f_*^{\text{e2e-}T}(\boldsymbol{x}) \right) \leq \varepsilon.$$

Even though the entire Chain-of-Thought is available during training, the learned predictor is still evaluated only on predicting the final answer. We do not explicitly require learning to be *proper*, but all our positive results provide a proper learning rule outputting $A(S) \in \mathcal{F}^{\text{e2e-}T}$, and all our

---

3. We emphasize that CoT-learnability is a property of the base class $\mathcal{F}$ and length T, not only of the set $\mathcal{F}^{\text{e2e-}T}$.

negative results preclude also improper learning, i.e. outputting some $A(S) : \mathcal{X} \to \Sigma$, even if $A(S) \notin \mathcal{F}^{\text{e2e-}T}$. We do not observe any gaps between proper and improper learning.

Time-Invariance vs. Time-Dependence. In our setup we emphasize *time invariance*, i.e. that the same base function $f \in \mathcal{F}$ is used in every step of autoregressive generation (e.g., a transformer with the same parameters, in a practical setting of interest). We then need to learn just a single function $f \in \mathcal{F}$, and our goal is for the learning complexity to be independent of (or at least nearly independent of) T. Time-invariance here can also be viewed as "parameter sharing" (Rajaraman et al., 2021) across steps of generation: for a parametric class $\mathcal{F} = \{f_{\boldsymbol{\theta}} : \boldsymbol{\theta} \in \boldsymbol{\Theta}\}$, the parameter $\boldsymbol{\theta}$ is shared among all time steps and we only need to learn a single $\boldsymbol{\theta}$. This invariance is in contradistinction to a *Time-Dependent* notion of autoregressive learning suggested by Malach (2023), where a different base predictor $f_t \in \mathcal{F}$ is used in every step t, yielding the Time-Dependent Chain-of-Thought class

$$\mathcal{F}^{\text{TD-CoT-}T} = \left\{ x \mapsto \overline{f}_T \circ \overline{f}_{T-1} \circ \cdots \circ \overline{f}_2 \circ \overline{f}_1(\boldsymbol{x}) \,\middle|\, f_1, f_2, \ldots, f_T \in \mathcal{F} \right\} . \tag{4}$$

Malach considered learning the class $\mathcal{F}^{\text{TD-CoT-}T}$, i.e. with the CoT available during training (as in our Definition 2)[4]. One can also consider end-to-end learning (as in our Definition 1) in such a time-dependent setting, which amounts to PAC-Learning the hypothesis class:

$$\mathcal{F}^{\text{TD-e2e-}T} = \left\{ \boldsymbol{x} \mapsto g(\boldsymbol{x})[-1] \,\middle|\, g \in \mathcal{F}^{\text{TD-CoT-}T} \right\} . \tag{5}$$

Sample complexities obtained by Malach all scale linearly with T, and this is unavoidable without time-invariance—see also discussion in Section 3. Our goal is to avoid such dependence.

3. Statistical and Computational Complexity for a General $\mathcal{F}$

In this section, we discuss the learnability of $\mathcal{F}^{\text{e2e-}T}$ for a hypothesis class $\mathcal{F}$ in terms of general properties of $\mathcal{F}$; see Table 1. Complete proofs can be found in Appendix C.

3.1. Learnability and Sample Complexity

Our first observation is simple, yet as we will see, also powerful: if the cardinality $|\mathcal{F}|$ of the base class is bounded, then so is the cardinality of the e2e and CoT classes: $\left|\mathcal{F}^{\text{e2e-}T}\right|, \left|\mathcal{F}^{\text{CoT-}T}\right| \le |\mathcal{F}|$. This is already sufficient for obtaining learning guarantees with sample complexity $\log |\mathcal{F}|$, in particular, using a learning rule requiring "consistency" with the final answer:

Given $S_{\text{e2e}} = ((\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m))$,

  Return $\hat{f}^{\text{e2e-}T}$, for some $\hat{f} \in \mathcal{F}$ such that $\hat{f}^{\text{e2e-}T}(\boldsymbol{x}_i) = y_i, \forall (\boldsymbol{x}_i, y_i) \in S_{\text{e2e}}$. $(\text{CONS}_{\text{e2e}})$

Theorem 3 *For any Σ, base class $\mathcal{F}$, and generation length T, we have that $\mathcal{F}^{\text{e2e-}T}$ is e2e-learnable, and so also CoT-learnable, with CONS_{e2e} and sample complexity*

$$m^{\text{e2e-}T}, m^{\text{CoT-}T} \le \frac{2}{\varepsilon} \left(\log |\mathcal{F}| + \log (1/\delta) \right) .$$

---

4. In Malach (2023), the learning goal is to be able to generate the entire CoT, not just the final answer. This is different from our Definition 2, but this is not a substantial difference.

		e2e (Latent CoT)	CoT (Available CoT)
	Learning Rule	$\textsc{Cons}_{\text{e2e}}$	$\textsc{Cons}_{\text{CoT}}$
Sample Complexity	$\|\mathcal{F}\|$ bounded	$\log\|\mathcal{F}\|$ (Theorem 3)	$\log\|\mathcal{F}\|$
	$\text{VCdim}(\mathcal{F})$ bounded	$\boldsymbol{T}\cdot\text{VCdim}(\mathcal{F})$ (Theorems 4 and 39)	$\text{VCdim}(\mathcal{F})\log T$ (Theorem 5)
	$\text{Ldim}(\mathcal{F})$ bounded	$\text{Ldim}(\mathcal{F})\log T$ (Theorem 6)	$\text{Ldim}(\mathcal{F})\log T$ (dominated by $\text{VCdim}(\mathcal{F})\log T$)
Computational complexity when $\textsc{Cons}$ for $(\mathcal{F}_d)$ is tractable		Not necessarily $\mathcal{P}oly(n,T,d)$ (Section 4)	$\mathcal{P}oly(n,T,d)$ (Corollary 9)

Table 1: A summary of the results in Section 3 for time-invariant autoregressive learning for a general base class $\mathcal{F}$. Each sample complexity row indicates the best possible guarantee, up to $O(\log(1/(\varepsilon\delta))/\varepsilon)$, on the sample complexity of e2e or CoT learning under the corresponding assumption on $\mathcal{F}$. All indicated sample complexities are tight except for $\log T$ factors, i.e. for each cell there exists classes $\mathcal{F}$ matching the indicated upper bound except for the $\log T$ factor (see Appendix D).

While this is a promising start, it is natural to wonder if analogous results hold for more general function classes satisfying less stringent complexity bounds, such as requiring only bounded VC dimension. Indeed, for classes over $\Sigma = \{0,1\}$, we can bound $\text{VCdim}(\mathcal{F}^{\text{e2e-}T}) = O(T\cdot\text{VCdim}(\mathcal{F}))$, which leads to the following sample complexity guarantee for e2e-learning [5] .

Theorem 4 *For any base class $\mathcal{F} \subseteq \{0,1\}^{\{0,1\}^*}$ and generation length $T \in \mathbb{N}_+$, the class $\mathcal{F}^{\text{e2e-}T}$ is* e2e-*learnable using* $\textsc{Cons}_{\text{e2e}}$ *with* $m^{\text{e2e-}T} = O\left(\varepsilon^{-1}\left(T\cdot\text{VCdim}(\mathcal{F})\log\left(\varepsilon^{-1}\right) + \log\left(\delta^{-1}\right)\right)\right).$

The linear dependence on T is extremely disappointing, as our goal with time-invariance is to avoid such generation-length dependence. In fact, we can learn the *time-dependent* $\mathcal{F}^{\text{TD-e2e-}T}$, with $O(T\cdot\text{VCdim}(\mathcal{F})\log T)$ samples, i.e. as in Theorem 4 up to a $\log T$ factor (Theorem 41 in Section D.2). Unfortunately, even with time-invariance, the linear dependence on T is unavoidable in the worst case—see Theorem 39. Now, we see the first benefit of CoT training. With the Chain-of-Thought available, we can define a stronger learning rule:

> Given $S_{\text{CoT}} = (\boldsymbol{z}_1,\ldots,\boldsymbol{z}_m)$, and letting $\boldsymbol{x}_i = \boldsymbol{z}_i[: -(T+1)]$,
>
> Return $\hat{f}^{\text{e2e-}T}$, for some $\hat{f} \in \mathcal{F}$ such that $\boldsymbol{z}_i = \hat{f}^{\text{CoT-}T}(\boldsymbol{x}_i), \forall\boldsymbol{z}_i \in S_{\text{CoT}}.$ ($\textsc{Cons}_{\text{CoT}}$)

As $\textsc{Cons}_{\text{CoT}}$ is a special case of[6] $\textsc{Cons}_{\text{e2e}}$, we have that $\textsc{Cons}_{\text{CoT}}$ enjoys the same guarantees as $\textsc{Cons}_{\text{e2e}}$, including the cardinality based Theorem 3. But for VC base classes, $\textsc{Cons}_{\text{CoT}}$ enjoys a much stronger (nearly independent of T) guarantee than in Theorem 4:

Theorem 5 *For any base class $\mathcal{F} \subseteq \{0,1\}^{\{0,1\}^*}$ and generation length $T \in \mathbb{N}_+$, the class $\mathcal{F}^{\text{e2e-}T}$ is* CoT-*learnable using* $\textsc{Cons}_{\text{CoT}}$ *with* $m^{\text{CoT-}T} = O\left(\varepsilon^{-1}\left(\text{VCdim}(\mathcal{F})\log T\log\left(\varepsilon^{-1}\right) + \log\left(\delta^{-1}\right)\right)\right).$

---

5. For simplicity, we are limiting the discussion to $\Sigma = \{0,1\}$. See Remark 7 for non-binary alphabet sets.

6. As with many standard learning rules, the rule is not a specific mapping, but a specification that can be implemented by many different mappings. Since any CoT consistent f is also e2e consistent, any valid response of $\textsc{Cons}_{\text{CoT}}$ is also valid for $\textsc{Cons}_{\text{e2e}}$, but not vice versa.

Returning to e2e learnability, we ask whether an assumption stronger than bounded VC dimension, but not as strong as finite cardinality, can allow us to obtain T-independent (or nearly independent) end-to-end sample complexity. The answer is yes—we can do so in terms of the familiar Littlestone dimension (Littlestone, 1988), which characterizes online learning (Ben-David et al., 2009), and has also found applications in other domains (e.g. Alon et al., 2022; Bun et al., 2020).

Theorem 6 *For any base class $\mathcal{F} \subseteq \{0,1\}^{\{0,1\}^*}$ with Littlestone dimension $\mathrm{Ldim}(\mathcal{F})$, and any generation length $T \in \mathbb{N}_+$, the class $\mathcal{F}^{\mathsf{e2e}\text{-}T}$ is e2e-learnable using $\mathrm{CONS}_{\mathsf{e2e}}$ with*

$$m^{\mathsf{e2e}\text{-}T} = O\left(\varepsilon^{-1}\left(\mathrm{Ldim}(\mathcal{F})\log T \log\left(\varepsilon^{-1}\right) + \log\left(\delta^{-1}\right)\right)\right).$$

Our Theorems 4 to 6 follow from simple but elegant covering number arguments (sketched out in Section A.1), followed by the standard uniform convergence of the empirical and population losses.

Remark 7 (General finite Σ) *In Appendix C, Corollaries 34, 36 and 38, we provide extensions of our Theorems 4 to 6 for any general finite alphabet Σ. The bounds are similar, with the VC dimension replaced with the Natarajan dimension and the Littlestone dimension replaced with the sequential shattering dimension, but there is also an additional factor of $\log|\Sigma|$ in the sample complexities. We also show in Theorem 45 that with the infinite alphabet $\Sigma = \mathbb{R}$, even if the base class $\mathcal{F}$ has bounded VC-subgraph dimension (a.k.a. Pollard, or pseudo dimension), the end-to-end class $\mathcal{F}^{\mathsf{e2e}\text{-}T}$ could have infinite VC-subgraph dimension.*

3.2. Computational Complexity

We now investigate the computational complexity of implementing the rules $\mathrm{CONS}_{\mathsf{e2e}}$ and $\mathrm{CONS}_{\mathsf{CoT}}$. Although all our learning guarantees are for learning over the entire $\mathcal{X} = \Sigma^*$, and do not have any dependence on the length of the input, when discussing computational complexity, the length of the input will be important (see Appendix B for the standard convention). We consider a hierarchy of base classes $\mathcal{F}_d$ with some size parameter d. If the base class $\mathcal{F}_d$ itself is computationally hard to learn, we cannot generally hope for the iterated classes to be easier. But what can we say if $\mathcal{F}_d$ itself is computationally easy to learn? Can we then implement $\mathrm{CONS}_{\mathsf{e2e}}$ and $\mathrm{CONS}_{\mathsf{CoT}}$ efficiently?

If we do have the entire Chain-of-Thought data S_{CoT}, then implementing $\mathrm{CONS}_{\mathsf{CoT}}$ amounts to finding $\hat{f} \in \mathcal{F}$ that is consistent with the generation of each of the last T tokens in each z_i:

$$\text{Return } \hat{f}^{\mathsf{e2e}\text{-}T} \text{ for some } \hat{f} \in \mathcal{F} \text{ such that } \forall_{z_i \in S_{\mathsf{CoT}}} \forall_{t=1,\dots,T}, \; \hat{f}(z_i[:-(t+1)]) = z_i[-t] \quad (6)$$

But (6) essentially amounts to solving a consistency problem $\mathrm{CONS}_{\mathcal{F}}$.

> Given $(u_i, v_i)_{i=1,\dots,\tilde{m}}$, return some $\hat{f} \in \mathcal{F}$ such that $\hat{f}(u_i) = v_i, \forall i \in [\tilde{m}]$. $\quad (\mathrm{CONS}_{\mathcal{F}})$

Theorem 8 *Consider any hypothesis class $\mathcal{F}$ over an alphabet set Σ. The rule $\mathrm{CONS}_{\mathsf{CoT}}$ on m examples of input length at most n can be implemented by a single call to $\mathrm{CONS}_{\mathcal{F}}$, with $\tilde{m} \leq m \cdot T$ samples of input length at most $n + T$, and $O(\tilde{m})$ additional runtime.*

As a corollary, we have the following tractability of CoT-learnability, if the base class has a tractable consistent oracle.

Corollary 9 *For a family $\mathcal{F}_d$, if $\text{CONS}_{\mathcal{F}_d}$ can be implemented in time polynomial in its input and the size parameter d, and $\text{VCdim}(\mathcal{F}_d) \leq \mathcal{P}oly(d)$, then $\mathcal{F}_d^{\text{e2e-}T}$ is CoT learnable in time $\mathcal{P}oly(n, d, T, \varepsilon^{-1}, \log \delta^{-1})$.*

In the next section, we will see that the situation is much grimmer for e2e-learnability. Even for tractably learnable base classes, where $\text{CONS}_{\mathcal{F}}$ is computationally easy, implementing CONS_{e2e}, or in fact e2e-learning using any other possible rule, could be computationally hard. This computational gap is perhaps the biggest advantage of Chain-of-Thought training.

4. Autoregressive Linear Thresholds

In this section, we study the base class, over the binary alphabet $\Sigma = \{0, 1\}$, where generators are d-dimensional linear thresholds applied to the last d bits in their input:

$$\mathcal{F}_{d,\text{lin}} := \left\{ f_{\boldsymbol{w}, b} =: \mathbb{1}\left[\sum_{i=1}^{d \wedge |\boldsymbol{x}|} \boldsymbol{w}[-i]\boldsymbol{x}[-i] + b \geq 0 \right] \,\middle|\, \boldsymbol{w} \in \mathbb{R}^d,\ b \in \mathbb{R} \right\} .$$

We provide the formal results for the sample complexity of end-to-end and CoT-learnability in Section A.2. The main advantage, however, of CoT learning over e2e learning here is computational, rather than statistical. Since the consistency problem $\text{CONS}_{\mathcal{F}_{d,\text{lin}}}$ amounts to Linear Programming feasibility and is thus easily solvable in polynomial time (Dantzig, 2016), Corollary 9 ensures that we can CoT learn $\mathcal{F}_{d,\text{lin}}^{\text{e2e-}T}$ with $O(d \log T \wedge d^2)$ samples and implement the rule on inputs of length n in time $\mathcal{P}oly(n, d, T, \varepsilon^{-1}, \log(\delta^{-1}))$:

Corollary 10 *The class $\mathcal{F}_{d,\text{lin}}^{\text{e2e-}T}$ is CoT-learnable in time $\mathcal{P}oly(d, T, n, 1/\varepsilon, \log(1/\delta))$.*

On the other hand, we show that $\mathcal{F}_{d,\text{lin}}^{\text{e2e-}T}$ is not tractably e2e-learnable, i.e. when Chain-of-Thought is not provided. Not only is CONS_{e2e} hard to implement, but we prove that no learning rule can e2e-learn $\mathcal{F}_{d,\text{lin}}^{\text{e2e-}T}$ in polynomial time (even improperly). To show this, we rely on the hardness of learning constant depth threshold circuits—a well-established hardness assumption in learning theory. See Section E.1 for a formal definition of the class of linear threshold circuits.

Hardness Assumption 1 (Hardness of Learning TC^0) *There exists a constant $L > 0$ and a polynomial $p(n)$ such that threshold circuits of depth L and size $p(n)$ over n binary inputs are not weakly PAC-learnable in time $\mathcal{P}oly(n)$, i.e. to constant accuracy and confidence parameters $\varepsilon < 0.5, \delta < 1$.*

The assumption is implied by core cryptographic assumptions such as hardness of the problems of inverting the RSA encryption function, recognizing quadratic residues, and factoring Blum integers (see Kearns and Valiant, 1994, Theorem 6). The Decisional Diffie-Hellman (DDH) assumption implies hardness already of learning depth-4 circuits, and so also implies Assumption 1 (Naor and Reingold, 1997; Krause and Lucks, 2001). An intersection of halfspaces is simply a depth-two linear threshold circuit, and hence hardness of learning the intersection of polynomially many halfspaces also implies Assumption 1. This is in turn implied by hardness of the unique shortest vector problem (Klivans and Sherstov, 2009), hardness of refuting random K-SAT formulas (Daniely and Shalev-Shwartz, 2016), and the existence of local pseudo-random generators (Daniely and Vardi, 2021).

Theorem 11 *Under Assumption 1, there is no algorithm that e2e-learns $\mathcal{F}_{d,\text{lin}}^{\text{e2e-}T}$ over $\{0, 1\}^n$ in time $\mathcal{P}oly(d, T, n)$ (even to within constants $\varepsilon < 0.5, \delta < 1$).*

Theorem 11 follows from a reduction from circuits to time-invariant iterated linear thresholds: we show that any linear threshold circuit with depth L and size s can be emulated using a time-invariant autoregressive linear threshold of dimension $d = O(s^L)$, up to a fixed poly-time feature map:

Lemma 12 *For any input size n, circuit size s and depth L, there exists a poly-time computable mapping $\phi : \{0,1\}^n \to \{0,1\}^{n'}$, with $n' \leq (s+2)^L(n+1)$, such that for any threshold circuit of size s and depth L computing $C : \{0,1\}^n \to \{0,1\}$, there exists $\boldsymbol{w} \in \mathbb{R}^d$, with $d \leq 2(s+2)^L(n+1)$, and $T \leq (s+2)^L(n+1)$, such that $f_{\boldsymbol{w}}^{\mathsf{e2e\text{-}}T}(\phi(\boldsymbol{x})) = C(\boldsymbol{x})$ for all $\boldsymbol{x} \in \{0,1\}^n$.*

In Lemma 12, the insistence on time-invariance, i.e. using the same linear threshold function in every step of generation, makes the construction much trickier. If instead we allowed a different linear predictor at each step of generation, as in $\mathcal{F}_{d,\mathrm{lin}}^{\mathsf{TD\text{-}e2e\text{-}}T}$ defined in Eqs. (4)–(5), we could directly encode each node in the circuit as a step of generation, with $T = s$ generation steps and dimensionality $d = s + n$. This was the approach taken by Malach (2023) when studying time-dependent autoregressive generation. We leave it as an open question whether it is possible to improve the time-invariant construction of Lemma 12, and reduce the required dimensionality, and especially the dependence on depth. In any case, Lemma 12 is sufficient for obtaining the hardness result of Theorem 11.[7]

5. Expressivity and Universality

One might possibly view Lemma 12 as a positive result about the expressive power of autoregressive time-invariant linear thresholds, suggesting they are "universal" and can express any computable function. In this sense Lemma 12 is disappointing, yielding a sample complexity of size$^{\Omega(\mathrm{depth})}$ for learning circuits. Not only is this exponential in depth, but even for very shallow circuits, e.g. $L = 2$, the sample complexity is much larger than the size of the circuit and of the sample complexity of learning the circuit directly.

Should this motivate us to attempt and improve the construction in Lemma 12? If our goal is to express circuits, the time-dependent class $\mathcal{F}_{d,\mathrm{lin}}^{\mathsf{TD\text{-}e2e\text{-}}T}$ is more appropriate, and we will not be able to improve over the sample complexity it provides for learning circuits (Malach, 2023; Bartlett and Maass, 2003). Circuits are inherently not "time-invariant", and their size and number of parameters are always (at least) linear in the input length, and more importantly, in the time of the computation involved.

What form of "universal" expressive power do we want then? What would we want the sample complexity to depend on? One type of desirable universality is to learn all computable functions with sample complexity proportional to their runtime. More precisely, letting $\mathsf{TIME}(\mathtt{T})$ denote the class of functions computable in time $\mathtt{T}$, we wish to learn $\mathsf{TIME}(\mathtt{T})$ with $\mathcal{P}oly(\mathtt{T})$ samples. All such functions can be expressed as circuits of size $\tilde{O}(\mathtt{T})$ (Arora and Barak, 2009), and hence feedforward neural nets with $E = \mathcal{P}oly(\mathtt{T})$ edges. Since, from a sample complexity perspective, neural nets (e.g. with threshold activations) can be learned from $\tilde{O}(E)$ samples (Anthony and Bartlett, 2009), neural nets already provide for "universal" learning in this sense: minimizing the loss on a neural net of size $\mathcal{P}oly(\mathtt{T})$ would allow us to learn $\mathsf{TIME}(\mathtt{T})$ using $\mathcal{P}oly(\mathtt{T})$ samples. The important caveat here, of course, is that learning a neural network is not computationally tractable, and we cannot actually minimize the loss on a neural net in polynomial time.

---

7. Improving the construction will only enable basing hardness on slightly weaker assumptions, such as hardness of learning log-depth circuits, or even existence of one-way functions.

Time-*Dependent* Autoregressive Chain-of-Thought learning with linear thresholds, i.e. using $\mathcal{F}_{d,\text{lin}}^{\text{TD-e2e-}T}$ is also "universal" in the same way as feed-forward networks: it allows expressing the class $\text{TIME}(\text{T})$ with dimensionality and generation length $d, T = O(\text{T})$, and thus learning with $\mathcal{P}oly(\text{T})$ samples. The advantage here, as pointed out by Malach (2023), is that we can consider Chain-of-Thought training as essentially providing us a peek into the inner working of the computation, and thus allowing *computationally tractable* learning.

While the above results on time-dependent autoregressive Chain-of-Thought learning are promising, we can hope for much stronger universality than just learning with number of samples proportional to computation runtime. Consider the much broader class $\text{PROG}(\text{S})$ of programs of length at most S. Many very-long-runtime functions have a short program but require a very long time to run; runtime usually depends also on the input length while program length does not. This is also a "time-invariant" class, since the program is independent of the input and computation length. An optimistic perspective from statistical learning theory would suggest that, due to the fact that statistical complexity scales only logarithmically in the cardinality of a hypothesis class, one could, in principle, learn $\text{PROG}(\text{S})$ with only $\log|\text{PROG}(\text{S})| = O(\text{S})$ samples; as in the case of neural networks above, however, the caveat in the previous observation is the computational intractability. We really have no hope of learning $\text{PROG}(\text{S})$ tractably–not only is finding a short program with small error uncomputable, but also functions in $\text{PROG}(\text{S})$ might require unbounded amount of time to even evaluate, let alone learn. Such functions are anyway useless as predictors, and it would be sensible to exclude them from consideration in the first place.

This leads us to consider the class $\text{PROG}(\text{S}, \text{T})$ of programs of length at most S that run in time at most T (on inputs we are considering). Typically the program length S will be much lower than the runtime T. Can we learn this class with sample complexity $O(\text{S})$ (or perhaps $\mathcal{P}oly(\text{S})$, or with a mild dependence on T such as $\mathcal{P}oly(\text{S}\log\text{T})$), and training time $\mathcal{P}oly(\text{T})$? Can we do this using time-invariant autoregressive learning? Can we have a (natural and simple?) base class $\mathcal{F}$ for every S and T such that:

1. We have $\text{PROG}(\text{S}, \text{T}) \subseteq \mathcal{F}^{\text{e2e-}\mathcal{P}oly(\text{T})}$;

2. $\mathcal{F}^{\text{e2e-}\mathcal{P}oly(\text{T})}$ is learnable with $\mathcal{P}oly(\text{S})$ or perhaps $\mathcal{P}oly(\text{S}\log\text{T})$ samples;

3. And $\mathcal{F}^{\text{e2e-}\mathcal{P}oly(\text{T})}$ is CoT-learnable in time $\mathcal{P}oly(\text{T})$?

High Context and Low Complexity Classes. One reason time-invariant iterated linear thresholds, $\mathcal{F}_{d,\text{lin}}^{\text{e2e-}T}$ are not sufficient for universal expressibility (as defined above) is that they attend to only a limited *context window*. That is, the output $f_{\boldsymbol{w}}(\boldsymbol{x})$ of a linear threshold generator $f_{\boldsymbol{w}} \in \mathcal{F}_{d,\text{lin}}$ depends only on the last d tokens $\boldsymbol{x}[-d :]$ of its input $\boldsymbol{x}$—we refer to this as having a "context length" of d. When applying $f_{\boldsymbol{w}}$ autoregressively for $T \gg d$ steps to obtain $\boldsymbol{z} = f_{\boldsymbol{w}}^{\text{CoT-}T}$, we have that $\boldsymbol{z}[t :]$ only depends on $\boldsymbol{z}[t - d : t - 1]$, i.e. on a finite state of d bits, and we cannot expect to capture computations requiring more than d bits of memory. Since the sample complexity of learning is also linear in d, we see that with iterated linear thresholds the sample complexity of learning a target h must be at least linear in the space complexity of computing h, and cannot depend only (or primarily) on the program length. To achieve our desiderata on universal expressibility (as defined above) with sample complexity (nearly) independent of program runtime (and space), we

therefore need generator base classes with a long context window (i.e. where $f(\boldsymbol{x})$ does not depend only on a short suffix of $\boldsymbol{x}$) yet small complexity.[8]

6. A Universal Autoregressive Base Class and Natural Emergence of Attention

In this section, we answer the optimistic question asked in the previous section, providing an explicit generator class satisfying all three desiderata. To formalize runtime of a prgram, we will work with a Turing machine as our model of computation, which we formalize in Section 6.1. In Section 6.2, we present our universal base class $\mathcal{F}_{\mathsf{TM}}$ and prove that it satisfies our desiderata from the end of the previous section (with simple input and output transformations). Finally, in Section 6.3, we discuss how attention arises naturally for simulating this class.

6.1. Turing Machine Computation

Definition 13 (Runtime-bounded Turing Machines) *A runtime-bounded Turing machine operates on an infinitely long "Tape" on both of its ends with cells indexed by integers $\mathbb{Z}$. It uses a tape alphabet set $\mathcal{A} = \{0, 1, \square\}$, of which $\{0, 1\}$ are input and working alphabets and $\square$ is a blank symbol. The Turing machine is then specified by a tuple $\mathcal{M} = \langle \mathsf{S}, \mathsf{T}, \tau \rangle$, where $\mathsf{S} \in \mathbb{N}_+$ denotes the number of internal states of the Turing machine, and $\mathsf{T} \in \mathbb{N}_+$ is the number of steps the Turing machine runs for. The transition rule is a mapping of the form*

$$\tau : [\mathsf{S}] \times \mathcal{A} \to [\mathsf{S}] \times \{0, 1\} \times \{-1, 0, +1\} \tag{7}$$
$$\tau : \text{state}, \text{read} \mapsto \text{nextstate}, \text{write}, \text{move} .$$

Computation of Turing Machines. The tape is initialized with an input $\boldsymbol{\omega} \in \{0, 1\}^*$ on the cells indexed from 1 through $|\boldsymbol{\omega}|$ and the other cells contain $\square$. The head of the machine is at the position $p_0 = |\boldsymbol{\omega}| + 1$ and its state is initialized to $s_0 = 1 \in [\mathsf{S}]$. During each time step $1 \le t \le \mathsf{T}$, the machine reads the symbol $r_t = \text{Tape}[p_{t-1}]$ at the current head position p_{t-1}, and according to the transition rule $(s_t, a_t, b_t) = \tau(s_{t-1}, r_t)$ updates its internal state to s_t, writes $\text{Tape}[p_{t-1}] \leftarrow a_t$ on the tape, and updates the head location to $p_t = p_{t-1} + b_t$. Finally, after the completion of T many time steps, the machine stops[9] and outputs the symbol a_T (the symbol it wrote in the final step before moving the head). We denote the function computed by the machine as $g_{\langle \mathsf{S}, \mathsf{T}, \tau \rangle}(\boldsymbol{\omega}) = a_\mathsf{T}$.

Definition 14 (Runtime-Bounded Turning Computable Functions) *For any positive integers $\mathsf{S} \le \mathsf{T}$, the class $\mathsf{TM}(\mathsf{S}, \mathsf{T}) = \big\{ g_{\langle \mathsf{S}, \mathsf{T}, \tau \rangle} : \{0, 1\}^* \to \{0, 1\} \big| \tau : [\mathsf{S}] \times \mathcal{A} \to [\mathsf{S}] \times \{0, 1\} \times \{-1, 0, +1\} \big\}$ is the set of functions computable by some runtime-bounded Turing Machine $\langle \mathsf{S}, \mathsf{T}, \tau \rangle$.*

The number of states of the Turing machine corresponds to program length—roughly speaking, we can think of each state corresponding to a statement in the program flow chart. More formally, the description length of a Turing machine $\langle \mathsf{S}, \mathsf{T}, \tau \rangle$ is $O(\mathsf{S} \log \mathsf{S})$, and by the Complexity-Theoretic Church-Turing (aka Invariance) Thesis, any reasonable model of computation can be simulated by a Turing machine up to polynomial blowup in runtime and additive overhead in description

---

8. One simple candidate to consider is the class of sparse linear thresholds (studied in Section F.1), which has a large context window of d but its sample complexity may be only logarithmic in the context size. We see that this base class indeed satisfies two of the three desiderata, however, its expressive power remains unclear.

9. In a standard textbook definition of a turning machine (e.g. Sipser, 1996), there is a halting state, which we do not require. This is because we have a bounded runtime T. W.l.o.g. one can always consider τ in a way that the tape does not change once the machine reaches either of the halting states, and the computation has effectively stopped.

length. In particular, $\mathsf{PROG}(\mathsf{S},\mathsf{T})$ in the RAM model of computation satisfies $\mathsf{PROG}(\mathsf{S},\mathsf{T}) \subseteq \mathsf{TM}(\tilde{O}(\mathsf{S}), \mathcal{P}oly(\mathsf{T}))$. Thus it suffices to address the universality question for the class $\mathsf{TM}(\mathsf{S},\mathsf{T})$.

6.2. Universal Autoregressive Base Class

In this subsection, we present a base class of generators $\mathcal{F}_{\mathsf{TM}}$ capable of simulating runtime-bounded Turing machines (with minimal pre- and post-processing maps, Pre and Post defined later). To simulate S-state computation, we will use autoregressive generation over the alphabet

$$\Sigma := [\mathsf{S}] \times \mathcal{A} \times \{-1, 0, +1\} \tag{8}$$

where $|\Sigma| = 3 \cdot \mathsf{S} \cdot |\mathcal{A}| = 9\mathsf{S}$. Here, both the token set Σ and our base class $\mathcal{F}_{\mathsf{TM}} \subseteq \Sigma^{\Sigma^*}$ depend on the allowed state space size S (corresponding to program length), but importantly not the runtime T. The generator class dependence on S is unavoidable, since its complexity must grow with S.

Each token $x = (s, a, b) \in \Sigma$ can be used to encode a step of Turing machine computation: a new state $s \in [\mathsf{S}]$, and a symbol $a \in \mathcal{A}$ written on the tape, and a move $b \in \{-1, 0, +1\}$. Indeed, for $x = (s, a, b)$ we will denote $x.\mathsf{state} = s$, $x.\mathsf{symb} = a$, and $x.\mathsf{move} = b$. Our universal base class $\mathcal{F}_{\mathsf{TM}}$ will have next-token generators f_τ which correspond to each Turing machine in $\mathsf{TM}(\mathsf{S},\mathsf{T})$, i.e. each possible transition rule τ as in (7):

$$\mathcal{F}_{\mathsf{TM}} = \{f_\tau : \Sigma^* \to \Sigma \,|\, \tau : [\mathsf{S}] \times \mathcal{A} \to [\mathsf{S}] \times \{0, 1\} \times \{-1, 0, +1\}\}$$

where each f_τ maps a token sequence $\boldsymbol{z} = (\boldsymbol{z}[i].\mathsf{state}, \boldsymbol{z}[i].\mathsf{symb}, \boldsymbol{z}[i].\mathsf{move})_{i \in |\boldsymbol{z}|}$ encoding the input and the history of computation up to time $t = |\boldsymbol{z}| - (\text{input length}) - 1$, to the next step $z = (s_t, a_t, b_t) = f_\tau(\boldsymbol{z})$ the Turing machine specified by τ would make. This is specified explicitly in Algorithm 1. The input to the Turing machine is specified in $\boldsymbol{z}$ through initial tokens that mimic writing the input on the Turing machine tape, and the output can be easily extracted from the final token corresponding to the final step of computation taken by the machine. More specifically, we describe the following input and output transformations:

Pre and Post Processing. Given any input $\boldsymbol{\omega} \in \{0, 1\}^*$ with $|\boldsymbol{\omega}| = n$, the *pre-processing map* Pre is defined as $\mathsf{Pre}(\boldsymbol{\omega}) \in \Sigma^{n+1}$, where

$$\mathsf{Pre}(\boldsymbol{\omega})[i] = \begin{cases} (1, \square, +1) & \text{for } i = 1 \\ (1, \boldsymbol{\omega}[i-1], +1) & \text{for } 2 \le i \le n+1 \,. \end{cases}$$

We need a special token $(1, \square, +1)$ to indicate the beginning of the sequence, similar to the [BOS] token (Beginning Of Sentence) in standard practice. The *post-processing* just amounts to returning the symbol part of the final token, so $\mathsf{Post}(x) = x.\mathsf{symb}$.

Universality of $\mathcal{F}_{\mathsf{TM}}$. We now argue that this universal autoregressive base class satisfies all three desiderata from Section 5. The definition of f_τ as simulating computation by a Turing machine with transition table τ directly ensures that, for any transition rule τ and input $\boldsymbol{\omega}$

$$\mathsf{Post}(f_\tau^{\mathsf{e2e}\text{-}T}(\mathsf{Pre}(\boldsymbol{\omega}))) = g_{\langle \mathsf{S}, \mathsf{T}, \tau \rangle}(\boldsymbol{\omega}), \tag{9}$$

and thus $\mathsf{TM}(\mathsf{S},\mathsf{T}) \subseteq \mathsf{Post} \circ \mathcal{F}_{\mathsf{TM}}^{\mathsf{e2e}\text{-}T} \circ \mathsf{Pre}$. Also, as there are only $(6\mathsf{S})^{3\mathsf{S}}$ possible transition tables, and so $\log|\mathcal{F}_{\mathsf{TM}}| = O(\mathsf{S}\log\mathsf{S})$. Applying the cardinality-based bound from Theorem 3, we have

$$m^{\mathsf{CoT}\text{-}T} = O\left(\varepsilon^{-1}(\mathsf{S}\log\mathsf{S} + \log(1/\delta))\right).$$

11

Algorithm 1 The Predictor $f_\tau : \Sigma^* \to \Sigma$ for a transition table τ

Input: An input string $\boldsymbol{z} \in \Sigma^*$
Output: The next token $z \in \Sigma$
   1 **return** $\tau(\mathsf{state}, \mathsf{read})$ where $(\mathsf{state}, \mathsf{read}) = \mathsf{read\text{-}tape}(\boldsymbol{z})$ given below.

Subroutine: read-tape $\triangleright$ (independent of τ)

Input: An input string $\boldsymbol{z} \in \Sigma^*$ encoding the history of a Turing machine computation.
Output: The current state (state) and the symbol at the current head-position (read).
   1 For each $i \in [N]$, where $N = |\boldsymbol{z}|$:
   2 $\mathsf{pos}[\,i\,] = \sum_{j<i} \boldsymbol{z}[\,j\,].\mathsf{move}$ $\triangleright$ Head position before a move, where $\boldsymbol{z}[\,i\,].\mathsf{symb}$ got written.

   3 $\mathsf{npos}[\,N\,] = \mathsf{pos}[N] + \boldsymbol{z}[N].\mathsf{move}$ $\triangleright$ Final head position.
   4 $\mathsf{read}[N] = \begin{cases} \boldsymbol{z}[j^*].\mathsf{symb} & \text{if there exists } j^* = \max_{j \leq N} j \text{ s.t. } \mathsf{pos}[\,j\,] = \mathsf{npos}[N]); \\ \square & \text{if } j^* \text{ does not exist.} \end{cases}$
   5 **return** $(\boldsymbol{z}[N].\mathsf{state}, \mathsf{read}[N])$

Moreover, the consistency problem $\mathrm{CONS}_{\mathcal{F}_{\mathsf{TM}}}$ (as in $\mathrm{CONS}_{\mathcal{F}}$) on $\tilde{m}$ examples of length $\tilde{n}$ can easily be solved in time $O(\tilde{m}\tilde{n})$:

> Given examples $(\boldsymbol{u}_i, v_i)_{i=1,\ldots,\tilde{m}}$ where $\boldsymbol{u}_i \in \Sigma^*$ and $v_i \in \Sigma$:
>
>   Return $f_{\hat{\tau}}$, for some $\hat{\tau}$ with S states such that $f_{\hat{\tau}}(\boldsymbol{u}_i) = v_i, \ \forall i \in [\tilde{m}]$. $(\mathrm{CONS}_{\mathcal{F}_{\mathsf{TM}}})$
>
> **Implementation:**
>   **1** For each $i \in [\tilde{m}]$: fill $\hat{\tau}(\mathsf{state}_i, \mathsf{read}_i) = v_i$ where $(\mathsf{state}_i, \mathsf{read}_i) = \mathsf{read\text{-}tape}(\boldsymbol{u}_i)$.
>   **2** If there are conflicts, then there is no consistent $f_{\hat{\tau}}$.
>   **3** **return** $f_{\hat{\tau}}$ after completing the other entries of $\hat{\tau}(\cdot, \cdot)$ arbitrarily.

Theorem 15 (Universality) *For every* $\mathsf{S} \in \mathbb{N}_+$ *and* $\mathsf{T} \geq \mathsf{S}$, *we have that* $\mathsf{TM}(\mathsf{S}, \mathsf{T})$ *is contained in* $\mathcal{F}_{\mathsf{TM}}{}^{\mathsf{e2e\text{-}T}}$ *(up to pre and post processing steps), and* $\mathcal{F}_{\mathsf{TM}}{}^{\mathsf{e2e\text{-}T}}$ *is* CoT-*learnable by* $\mathrm{CONS}_{\mathsf{CoT}}$ *with runtime* $\mathcal{P}oly(n, \mathsf{T}, \varepsilon^{-1}, \log \delta^{-1})$ *and sample complexity* $m^{\mathsf{CoT\text{-}T}} = O\left(\varepsilon^{-1}\left(\mathsf{S} \log \mathsf{S} + \log \delta^{-1}\right)\right)$.

This is a corollary of the observations in the preceding paragraphs and Theorem 8.

6.3. Emergence of Attention

We now take a closer look at the generator class $\mathcal{F}_{\mathsf{TM}}$, and its generators $f_\tau \in \mathcal{F}_{\mathsf{TM}}$ specified in Algorithm 1, and how these generators can be implemented using "generic" function classes. We intentionally wrote f_τ as operating on arrays of length N. The two operations involving array elements across multiple locations are the position calculation in Line 2 of read-tape, which involves summing moves over multiple array locations, and the position lookup in Line 4 on read-tape. The lookup operation is explicitly an attention-type operation: we look for a location where the write position (the "key") matches the read position (the "query") and copy its tape symbol (the

"value"). Meanwhile, the summation operation in Line 2 can be implemented as averaging many array locations, as in uniform attention.

From Attention to Transformers. In the preceding paragraphs, we discussed how to implement the "non-local" operations on Lines 2 and 4 using attention. All other operations in the implementation of f_τ are "local" in the sense that they operate independently at each location of the arrays[10]. This directly suggests a decoder-only-transformer architecture (Vaswani, 2017) where attention operations are interleaved with multi-layer perceptrons (MLPs) operating independently and in parallel at each location, as a generic way of implementing the local operations. Most notably, the τ-lookup operation is directly implementable by an MLP with τ encoded in its weights. Using Hard Average Attention as in Definition 18 results in a discontinuous and non-differentiable model, but a natural alternative is to relax the hard max in the score definition (13) to a softmax, resulting in the familiar softmax attention widely used in practice. Implementing f_τ with a practical transformer architecture further requires care to numerical precision, and to the specifics of implementing or approximating the local operations with MLPs and possible layer-normalizations, especially the division and multiplication operations involved in calculating the step-counters and positions. In fact, the Turing Machine expressivity results of Pérez et al. (2021); Wei et al. (2022); Merrill and Sabharwal (2023) essentially do this, following a construction similar to ours, and showing how to implement it with specific transformer architectures.

The difference here is mostly that of perspective: while recent work on Turing universality of transformers started with the transformer as suggested by Vaswani (2017) and showed how to coerce it to simulate a Turing machine, we take the reverse perspective. Our starting point is the desiderata of Section 5, which directly motivates the generator class $\mathcal{F}_{\mathsf{TM}}$, and we then argue how implementing $f_\tau \in \mathcal{F}_{\mathsf{TM}}$ naturally involves attention and yields a transformer-type architecture.

7. Discussion and Future Directions

We investigate training based solely on information about the final answer, as well as training with supervision about the intermediate reasoning steps used to reach that answer. Training based on explicit Chain-of-Thought examples is studied by Nye et al. (2021); Zelikman et al. (2022); Chung et al. (2024); Lewkowycz et al. (2022), and is widely used in contemporary training of frontier LLMs, particularly in the fine-tuning step (OpenAI, 2023; DeepMind, 2024; Touvron et al., 2023).

More recently, there have been attempts to train models using only feedback on the final answer, attempting to learn the reasoning process using reinforcement learning techniques (Guo et al., 2025; Ouyang et al., 2022). Our results help highlight and formalize the intractability of such approaches in general, without additional assumptions. So, why have approaches like DeepSeek and others seen empirical success? One possiexplanation is that these Reinforce-type methods assume access to a sufficiently good (possibly randomized) next-token predictor $f \in \mathcal{F}$ that already produces correct (or at least useful) CoTs with non-negligible probability with some level of coverage over valid reasoning paths. Such a predictor is typically learned during pretraining on large-scale text data, which, while not always matching the target task's CoTs exactly, are often similar enough to enable transfer. This type of training is not exactly CoT supervision, but is arguably closer to it than pure end-to-end supervision. Understanding how transfer from such related-but-different next-

---

10. This includes the operations inside read-tape which operate explicitly at array locations, and the τ-lookup at the end, which can be thought of as operating at location N, or in fact at each location independently and in parallel outputting the computation-step-token

token prediction tasks enables tractable learning—using only final-answer supervision, and little or no direct CoT data for the task of interest—is an extremely interesting direction for future work.

In summary, we presented a framework of time-invariant autoregressive Chain-of-Thought learning. Using our framework, we discussed how:

- Time-invariance can (nearly) avoid the dependence on the number of steps of generation and computation in sample complexity, and allows for universal learning with sample complexity dependent only on program length (and not runtime!). This is independent of CoT training and holds also for e2e learning.
- CoT training allows for overcoming computational difficulties and for tractable training even of universal classes. This is independent of time invariance and is true also for time-dependent models.
- Attention arises naturally in time-invariant base classes allowing such universal expressivity.

Learning Goals	Nearly CoT-length Independent Samples	Computational Tractability
Time Invariant e2e	✓	✗
Time Dependent CoT (Malach, 2023)	✗	✓
Time Invariant CoT	✓	✓

We were directly inspired by the work of Malach (2023), and address what we view as its major deficiency, namely time dependence.

Further Open Questions. Our work leaves several technical questions open such as: What is the true sample complexity of e2e and CoT learning of iterated linear thresholds, i.e. lower bounds for Corollary 17? Can the circuit embedding construction of Lemma 12 be improved? Can the dependencies on $\log T$ in Table 1 be resolved, and more significantly the dependence on $\log |\Sigma|$ mentioned in Remark 7? Can e2e learnability be ensured using a notion more relaxed than the Littlestone dimension?

Additionally, it is interesting to study other base classes under our framework, including sparse linear thresholds, gated generators, simple attention mechanisms and transformers. What are the sample and computational complexities of e2e and CoT learning? What is the expressiveness of $\mathcal{F}^{\text{e2e-}T}$? Another major challenge is generalizing our framework to the agnostic setting. In particular, what is the correct way to extend CoT learning to an agnostic setting?

Our framework also allows us to rigorously study the computational complexity of settings on the spectrum between e2e and CoT learning. What can be done tractably if we have a small amount of full CoT data, and a large amount of end-to-end input-output pairs? What is the minimum amount of CoT data required for tractable learning if we have access to an unlimited amount of end-to-end data? How helpful is CoT data from a mixture of different generators f_i all computing the same end-to-end function $f_i^{\text{e2e-}T} = f_j^{\text{e2e-}T}$ but $f_i^{\text{CoT-}T} \neq f_j^{\text{CoT-}T}$?

In this paper, we studied only in-distribution generalization, with a fixed generation length. But insisting on time invariance should allow us to study the much more complex question of length generalization and other forms of out-of-distribution generalization. Finally, much of the actual use of large language models stems from their ability to learn not only from solutions to the task of interest (as we do here), but also transfer from modeling unrelated and unlabeled data. One can consider extending our framework to allow studying of such transfer learning, with access to data not solving the desired task but generated by the same generator.

Acknowledgments

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Noga Alon, Mark Bun, Roi Livni, Maryanthe Malliaris, and Shay Moran. Private and online learnability are equivalent. *ACM Journal of the ACM (JACM)*, 69(4):1–34, 2022.

Martin Anthony and Peter L Bartlett. *Neural network learning: Theoretical foundations*. cambridge university press, 2009.

Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

Michael Bain and Claude Sammut. A framework for behavioural cloning. In *Machine intelligence 15*, pages 103–129, 1995.

Pablo Barceló, Alexander Kozachinskiy, Anthony Widjaja Lin, and Vladimir Podolskii. Logical languages accepted by transformer encoders with hard attention. *arXiv preprint arXiv:2310.03817*, 2023.

Peter L Bartlett and Wolfgang Maass. Vapnik-chervonenkis dimension of neural nets. *The handbook of brain theory and neural networks*, pages 1188–1192, 2003.

Shai Ben-David, Dávid Pál, and Shai Shalev-Shwartz. Agnostic online learning. In *COLT*, volume 3, page 1, 2009.

Adam Block, Ali Jadbabaie, Daniel Pfrommer, Max Simchowitz, and Russ Tedrake. Provable guarantees for generative behavior cloning: Bridging low-level stability and high-level behavior. *Advances in Neural Information Processing Systems*, 36:48534–48547, 2023.

Mark Bun, Roi Livni, and Shay Moran. An equivalence between private classification and online prediction. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 389–402. IEEE, 2020.

Jonathan D Chang, Kiante Brantley, Rajkumar Ramamurthy, Dipendra Misra, and Wen Sun. Learning to generate better than your llm. *arXiv preprint arXiv:2306.11816*, 2023.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53, 2024.

Amit Daniely and Shai Shalev-Shwartz. Complexity theoretic limitations on learning dnf's. In *Conference on Learning Theory*, pages 815–830. PMLR, 2016.

Amit Daniely and Gal Vardi. From local pseudorandom generators to hardness of learning. In *Conference on Learning Theory*, pages 1358–1394. PMLR, 2021.

George B Dantzig. Linear programming and extensions. In *Linear programming and extensions*. Princeton university press, 2016.

Google DeepMind. Gemini 1.5 technical report, 2024. https://deepmind.google/technologies/gemini/.

Dylan J Foster, Adam Block, and Dipendra Misra. Is behavior cloning all you need? understanding horizon in imitation learning. *arXiv preprint arXiv:2407.15007*, 2024.

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

Alex Graves. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. arXiv preprint, 2025. Uses sparse final-answer rewards to learn CoT via policy gradient (Group Relative Policy Optimization).

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *NeurIPS*, 2021.

Michael Kearns and Leslie Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *Journal of the ACM (JACM)*, 41(1):67–95, 1994.

Adam R Klivans and Alexander A Sherstov. Cryptographic hardness for learning intersections of halfspaces. *Journal of Computer and System Sciences*, 75(1):2–12, 2009.

Matthias Krause and Stefan Lucks. Pseudorandom functions in and cryptographic limitations to proving lower bounds. *computational complexity*, 10(4):297–313, 2001.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models. *Advances in Neural Information Processing Systems*, 35:3843–3857, 2022.

Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. Textbooks are all you need ii: phi-1.5 technical report. *arXiv preprint arXiv:2309.05463*, 2023.

Nick Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine learning*, 2:285–318, 1988.

Eran Malach. Auto-regressive next-token predictors are universal learners. *arXiv preprint arXiv:2309.06979*, pages 01–15, 2023.

William Merrill and Ashish Sabharwal. The expresssive power of transformers with chain of thought. *arXiv preprint arXiv:2310.07923*, 2023.

William Merrill, Ashish Sabharwal, and Noah A Smith. Saturated transformers are constant-depth threshold circuits. *Transactions of the Association for Computational Linguistics*, 10:843–856, 2022.

M Naor and O Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *Proceedings 38th Annual Symposium on Foundations of Computer Science*, pages 458–458. IEEE Computer Society, 1997.

Balas K Natarajan. On learning sets and functions. *Machine Learning*, 4:67–97, 1989.

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.

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

Simon Ott, Konstantin Hebenstreit, Valentin Liévin, Christoffer Egeberg Hother, Milad Moradi, Maximilian Mayrhauser, Robert Praas, Ole Winther, and Matthias Samwald. Thoughtsource: A central hub for large language model reasoning data, 2023. URL https://arxiv.org/abs/2301.11596.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. arXiv preprint arXiv:2203.02155, 2022. Introduces RLHF (InstructGPT).

Jorge Pérez, Pablo Barceló, and Javier Marinkovic. Attention is turing-complete. *Journal of Machine Learning Research*, 22(75):1–35, 2021.

David Pollard. Asymptotics via empirical processes. *Statistical science*, pages 341–354, 1989.

Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1, 1988.

Nived Rajaraman, Yanjun Han, Lin Yang, Jingbo Liu, Jiantao Jiao, and Kannan Ramchandran. On the value of interaction and function approximation in imitation learning. *Advances in Neural Information Processing Systems*, 34:1325–1336, 2021.

Alexander Rakhlin, Karthik Sridharan, and Ambuj Tewari. Sequential complexities and uniform martingale laws of large numbers. *Probability theory and related fields*, 161:111–153, 2015.

Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 661–668. JMLR Workshop and Conference Proceedings, 2010.

Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

Michael Sipser. Introduction to the theory of computation. *ACM Sigact News*, 27(1):27–29, 1996.

Lena Strobl, William Merrill, Gail Weiss, David Chiang, and Dana Angluin. What formal languages can transformers express? a survey. *Transactions of the Association for Computational Linguistics*, 12:543–561, 2024.

Hugo Touvron, Louis Martin, Kevin Stone, et al. Llama 2: Open foundation and fine-tuned chat models, 2023. https://ai.meta.com/llama/.

A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

Colin Wei, Yining Chen, and Tengyu Ma. Statistically meaningful approximation: a case study on approximating turing machines with transformers. *Advances in Neural Information Processing Systems*, 35:12071–12083, 2022.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.

Appendix Table of Contents

Appendix A. Deferred Details for the Main Paper

A.1. Proof Sketches of Theorems 4, 5 and 6

Our learning guarantees are based on bounding the growth function $\Gamma_{\mathcal{H}}(m)$ (number of possible behaviors of a function class $\mathcal{H}$ on a set of m points) and then applying standard concentration bounds. **CoT Learnability in terms of VC Dimension (Theorem 5).** We define the loss class:

$$\mathcal{L}^{\mathsf{CoT}\text{-}T} := \{\ell_f : \boldsymbol{z} \mapsto \mathbb{1}\{\boldsymbol{z} \neq f^{\mathsf{CoT}\text{-}T}(\boldsymbol{x})\} \mid f \in \mathcal{F}\}, \text{ where } \boldsymbol{x} = \boldsymbol{z}[:-(T+1)]. \qquad (10)$$

The rule $\mathrm{CONS}_{\mathsf{CoT}}$ can be expressed as requiring $\ell_f \in \mathcal{L}^{\mathsf{CoT}\text{-}T}$ whose empirical average is zero. We want to ensure the population mean of this ℓ_f is also small, as this is a direct bound on $L^{0\text{-}1}_{\mathcal{D}, f_*}(f)$. To ensure this via uniform concentration, we bound the growth function of the loss class $\mathcal{L}^{\mathsf{CoT}\text{-}T}$. The important observation is that the behaviors of $\mathcal{L}^{\mathsf{CoT}\text{-}T}$ on a set $S_{\mathsf{CoT}} = (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_m) \in (\mathcal{X} \times \Sigma^T)^m$ of size m, are determined by the behaviors of the base class $\mathcal{F}$ on the set of mT prefixes:

$$\mathrm{pfx}(S_{\mathsf{CoT}}) = \{\, \boldsymbol{z}_i[:-(t+1)], \text{ where } \boldsymbol{z}_i \in S_{\mathsf{CoT}}, t = 1, \ldots, T \,\}. \qquad (11)$$

We can therefore bound:

$$\Gamma_{\mathcal{L}^{\mathsf{CoT}\text{-}T}}(m) \leq \Gamma_{\mathcal{F}}(mT) \leq \left(\frac{e\,m\,T}{\mathrm{VCdim}(\mathcal{F})}\right)^{\mathrm{VCdim}(\mathcal{F})} \Rightarrow \mathrm{VCdim}(\mathcal{L}^{\mathsf{CoT}\text{-}T}) = O(\mathrm{VCdim}(\mathcal{F})\log T)$$

where the second inequality follows from Sauer's Lemma.

e2e Learnability in terms of VC Dimension (Theorem 4). With only the input given, the behavior of $\mathcal{F}^{\mathsf{e2e}\text{-}T}$ on $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m)$ depends on the behavior of $\mathcal{F}$ not only on these inputs, but for each $\boldsymbol{x}_i$ also on the possible string after $t < T$ steps of generation. A crude way of bounding the number of such possible generations is by the total number of possible ways of extending $\boldsymbol{x}_i$ with $t < T$ additional tokens from $\{0, 1\}$. There are $\sum_{t<T} 2^t \leq 2^T$ such possible extensions, so the behaviors of $\mathcal{F}^{\mathsf{e2e}\text{-}T}$ on m points depend on the behaviors of $\mathcal{F}$ on at most $m2^T$ points, and we have:

$$\Gamma_{\mathcal{F}^{\mathsf{e2e}\text{-}T}}(m) \leq \Gamma_{\mathcal{F}}(m \cdot 2^T) \leq \left(\frac{e\,m\,2^T}{\mathrm{VCdim}(\mathcal{F})}\right)^{\mathrm{VCdim}(\mathcal{F})} \Rightarrow \mathrm{VCdim}(\mathcal{F}^{\mathsf{e2e}\text{-}T}) = O(T\,\mathrm{VCdim}(\mathcal{F})).$$

e2e Learnability in terms of the Littlestone Dimension (Theorem 6). Instead of the crude count on the number of behaviors of $\mathcal{F}^{\mathsf{e2e}\text{-}T}$ on $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m)$, we consider the "computation path" of any $f \in \mathcal{F}$ on some $\boldsymbol{x}_i$. This can be thought of as a tree, where the possible out-edges at each node correspond to the possible behaviors of $\mathcal{F}$ on the prefix at the node. We can construct a complete binary tree of depth $m(T+1)$ such that the covering number of $\mathcal{F}$ on this tree (in the sense of Rakhlin et al. (2015)) upper bounds $\Gamma_{\mathcal{F}^{\mathsf{e2e}\text{-}T}}(m)$. We then use the sequential analogue of Sauer's Lemma (Rakhlin et al., 2015, Lemma 5). Thus, we compute for large m:

$$\Gamma_{\mathcal{F}^{\mathsf{e2e}\text{-}T}}(m) \leq (\# \text{ behaviors on a binary tree of depth } m(T+1)) \leq \left(\frac{2em(T+1)}{\mathrm{Ldim}(\mathcal{F})}\right)^{\mathrm{Ldim}(\mathcal{F})},$$

which allows us to obtain that $\mathrm{VCdim}(\mathcal{F}^{\mathsf{e2e}\text{-}T}) = O(\mathrm{Ldim}(\mathcal{F})\log T)$.

A.2. Sample Complexity Results for Autoregressive Linear Thresholds from Section 4

In this section, we provide formal sample complexity results for the base class $\mathcal{F}_{d,\text{lin}}$. From Theorems 4 and 5, we have that $\mathcal{F}_{d,\text{lin}}^{\text{e2e-}T}$ is e2e and CoT learnable with sample complexities $m^{\text{e2e}} \propto O(Td)$ and $m^{\text{CoT}} \propto O(d \log T)$ respectively, with the disappointing $O(T)$ scaling for e2e learning. But in this case, even though $\mathcal{F}_{d,\text{lin}}$ is specified in terms of real-valued parameters, the discreteness of the input domain actually allows us to bound its cardinality:

Lemma 16 *Over the domain $\{0, 1\}^d$, we have*[11] $|\mathcal{F}_{d,\text{lin}}| \leq (2e \cdot 2^d)^{(d+1)} = 2^{O(d^2)}$.

Plugging in Lemma 16 into Theorem 3 we see that e2e learning is also possible with T-independent sample complexity. Combining the VC-dimension and cardinality-based sample complexities:

Corollary 17 $\mathcal{F}_{d,\text{lin}}^{\text{e2e-}T}$ *is* e2e *and* CoT *learnable using* CONS_{e2e} *and* CONS_{CoT} *with samples*

$$m^{\text{e2e}} = O\left(\frac{\left(d^2 \wedge d \cdot T \log(\varepsilon^{-1})\right) + \log(\delta^{-1})}{\varepsilon}\right) \quad and \quad m^{\text{CoT}} = O\left(\frac{\left(d^2 \wedge d \log T \log(\varepsilon^{-1})\right) + \log(\delta^{-1})}{\varepsilon}\right).$$

While we do not know if the d^2 vs. $d \log T$ gap in the sample complexity (when $\log T \ll d$) between e2e and CoT learning is real, it is clear that there remains a statistical advantage over learning the time-dependent $\mathcal{F}_{d,\text{lin}}^{\text{TD-e2e-}T}$ (as in Malach, 2023), which requires $\Omega(d \cdot T)$ samples.

A.3. Implementing Non-local Operations in Algorithm 1

In this, we will show how Lines 2 and 4 can be naturally implemented by Average Hard Attention (e.g. Merrill et al., 2022; Barceló et al., 2023; Strobl et al., 2024):

Definition 18 (Causal Average Hard Attention) *For any positive integers N, p, l and any sequences of vectors* $\mathsf{q}[1], \ldots, \mathsf{q}[N] \in \mathbb{R}^l$ *(queries),* $\mathsf{k}[1], \ldots, \mathsf{k}[N] \in \mathbb{R}^l$ *(keys), and* $\mathsf{v}[1], \ldots, \mathsf{v}[N] \in \mathbb{R}^p$ *(values), and a score function* $\mathsf{score} : \mathbb{R}^i \to \mathbb{R}^i$ *for each* $i \in [N]$, *the* causal attention function $\mathsf{Att} : (\mathbb{R}^l)^N \times (\mathbb{R}^l)^N \times (\mathbb{R}^p)^N \mapsto (\mathbb{R}^p)^N$ *is defined as:*

$$\left[\mathsf{Att}(\{\mathsf{q}[j]\}_{j=1}^N, \{\mathsf{k}[j]\}_{j=1}^N, \{\mathsf{v}[j]\}_{j=1}^N)\right]_i = \sum_{\ell=1}^i \mathsf{score}(\{\langle \mathsf{q}[i], \mathsf{k}[j]\rangle\}_{j=1}^i)_\ell \, \mathsf{v}[\ell] \, . \tag{12}$$

In particular, the average hard attention $\mathsf{AHA}(\mathsf{q}, \mathsf{k}, \mathsf{v})$ *uses the following score function:*

$$\mathsf{score}_{\mathsf{AHA}}(\{s_j\}_{j=1}^i)_\ell = \begin{cases} \frac{1}{|\mathcal{M}|} & \textit{if } \ell \in \mathcal{M} \\ 0 & \textit{otherwise} \end{cases}, \quad \textit{where } \mathcal{M} = \arg\max_{j \leq i} s_j \, . \tag{13}$$

That is, the output $\mathsf{AHA}(\mathsf{q}, \mathsf{k}, \mathsf{v})[i] \in \mathbb{R}^p$ *at position i is an average of the "values" $\mathsf{v}[\ell]$ at positions $\ell \in \arg\max_{j \leq i} \langle \mathsf{q}[i], \mathsf{k}[j]\rangle$ whose "keys" $\mathsf{k}[j]$ have the maximal inner product with the "query" $\mathsf{q}[i]$ at the output position i of interest.*

---

11. The lemma follows by applying Sauer's lemma on the entire domain $\{0, 1\}^d$ of size 2^d: $|\mathcal{F}_{d,\text{lin}}| = \Gamma_{\mathcal{F}_{d,\text{lin}}}(2^d) \leq (2e2^d)^{\text{VCdim}(\mathcal{F}_{d,\text{lin}})} = 2^{O(d^2)}$.

Implementing Line 2: This is a summation over all previous positions, which is the same as an average over all previous positions multiplied by the number of previous position. Such an average over all previous positions can be easily implemented by "uniform" attention, i.e. where all keys and queries are zero and values from all positions are averaged.

There are two minor complications here in using Causal Average Hard Attention as we defined it: as written in Line 2, we need to average over all previous positions, but not the output position, as in *strict* causal attention (a variant of Definition 18 where the arg-max in (13) is over $j < i$ instead of $j \leq i$). We can also implement this using non-strict causal attention (Definition 18) with one additional local step as follows. Define $\mathsf{npos}[i] = \sum_{j=1}^{i} z[i].\mathsf{symb}$ instead; i.e. the summation including the index i. Once we have computed npos, a local operation $\mathsf{pos}[i] := \mathsf{npos}[i] - z[i].\mathsf{move}$ suffices to compute pos. The second minor complication is going between the average and the sum, for which we also need to compute the current index. This can be done thanks to the special "beginning of sequence" token. Specifically, to compute npos using a non-strict causal attention, first define $\mathsf{is\text{-}first}[i] := \mathbf{1}[z[i].\mathsf{symb} = \square]$, which indicates whether the token is the first. We can use attention to get $\mathsf{inv\text{-}pos} := \mathsf{AHA}(\mathbf{0}, \mathbf{0}, \mathsf{is\text{-}first})$ using uniform attention, where all keys and query are the same. It is easy to check that $\mathsf{inv\text{-}pos}[i] = \frac{1}{i}$. Similarly, consider $\mathsf{scaled\text{-}pos} := \mathsf{AHA}(\mathbf{0}, \mathbf{0}, z.\mathsf{move})$. Using the definition of AHA (Definition 18), it is straightforward to verify that $\mathsf{scaled\text{-}pos}[i] = \frac{\sum_{j=1}^{i} z[j].\mathsf{move}}{i} = \frac{\mathsf{npos}[i]}{i}$. Therefore, $\mathsf{npos}[i] = i \times \mathsf{scaled\text{-}pos}[i] = \mathsf{scaled\text{-}pos}[i]/\mathsf{inv\text{-}pos}[i]$.

Implementing Line 4: This lookup is a much more direct application of attention, and indeed is reminiscent of the original motivation for attention as a basic component required for computation (Graves, 2014). Here we directly want to set the query to the current read position and the keys to the write positions and look for an exact match. The complication here is that we want *the most recent* exact match. This can be achieved by augmenting the keys with the step counter and thus seeking the highest step-count computation step among those with the exact match.

Specifically, we will get $\mathsf{read} := \mathsf{AHA}(\mathsf{q}, \mathsf{k}, \mathsf{v})$ with hard-average attention by setting $\mathsf{v}[i] = z[i].\mathsf{symb}$,

$$\mathsf{q}[i] = (-\mathsf{npos}[i]^2, \mathsf{npos}[i], -1, -1), \quad \mathsf{k}[j] = \begin{cases} (0, 0, 0, \mathsf{inv\text{-}pos}[j]) & \text{if } j = 1; \\ (2, 4\mathsf{pos}[j], 2\mathsf{pos}[j]^2, \mathsf{inv\text{-}pos}[j]) & \text{otherwise}. \end{cases}$$

Thus the inner product between queries and products is

$$\langle \mathsf{q}[i], \mathsf{k}[j] \rangle = \begin{cases} -1 & \text{if } j = 1; \\ -2(\mathsf{npos}[i] - \mathsf{pos}[j])^2 - \frac{1}{j} & \text{otherwise}. \end{cases}$$

Therefore, if we look at the last index $\mathsf{read}[N]$, then whenever $j^* = \max_{j \leq N} j$ s.t. $\mathsf{npos}[N] = \mathsf{pos}[j]$ exists, indeed we observe that the inner product $\{\langle \mathsf{q}[N], \mathsf{k}[j] \rangle\}_{j=1}^{N}$ is maximized uniquely at the index j^*, i.e. $j^* = \arg\max_{j \leq i} \langle \mathsf{q}[N], \mathsf{k}[j] \rangle$. As a result, the average-hard attention always focuses j^* and we have that $\mathsf{read}[N] = \mathsf{v}[j^*] = z[j^*].\mathsf{symb}$ as desired. When j^* does not exist, it is easy to observe that the inner product $\{\langle \mathsf{q}[N], \mathsf{k}[j] \rangle\}_{j=1}^{N}$ is maximized uniquely at the index $j = 1$. Therefore, the attention focuses on the first index and returns $\mathsf{v}[1] = z_{\mathsf{in}}[1].\mathsf{symb} = \square$, which is precisely what we need, concluding the implementation.

Avoiding Uniform Attention. While the tape lookup operation in Line 4 very naturally motivates attention, and also seems unavoidable and fundamental to universal computation, here we discuss how the use of uniform attention to calculate the current Turing Machine head position in Line 2

can be avoided. To do so, instead of writing out the relative move b_t of each step of the Turing computation (as in Definition 13), we can write out the absolute tape position p_t. We cannot do so with a single token, since p_t could potentially be as large as the runtime $\mathtt{T}$, necessitating an alphabet of size $\Sigma = \Omega(\mathtt{T})$, something we would like to avoid. But we could easily write down (i.e. generate) p_t over multiple tokens, even over a binary alphabet $\Sigma = \{0, 1\}$, using some prefix unambiguous encoding. The generator f_τ would then need to collect the encoding of p_{t-1} (as well as the previous state and symbol written), do the arithmetic to calculate the next absolute position $p_t = p_{t-1} + b_t$ (as well other operations in Algorithm 1), figure out where in the middle of generating the description of the next compute state we are at, and generate the next bit in the description. Implementing this with a transformer-like architecture involves multiple attention layers, mostly with short-range attention to the last $O(\log \mathtt{T})$ tokens as well as a single long-term sparse attention to implement the lookup on Line 4, and a transformer of size (and depth) that scales with $\log \mathtt{T}$. But it avoids the need for uniform attention. A similar approach can also simulate RAM-computation, where the non-local operations are reads from memory addresses describable with $\log \mathtt{T}$ bits.

A.4. Relationship to Behavior Cloning

Recent work has connected autoregressive generation to *Behavior Cloning* (BC) (Pomerleau, 1988; Bain and Sammut, 1995; Ross and Bagnell, 2010), and indeed BC has been suggested as an approach for understanding modern language models (Chang et al., 2023; Block et al., 2023). Those works view autoregressive generation as a Markov Decision Process (MDP) with a "state" at step t being the entire sequence generated thus far, i.e. $f^{\mathsf{CoT}\text{-}t}(\boldsymbol{x})$, actions corresponding to tokens $a \in \Sigma$, and transitions being deterministic mapping a state $\boldsymbol{z}$ and an action $a \in \Sigma$ to the next state $\mathrm{append}(\boldsymbol{z}, a)$. In that framework, our "generators" f become deterministic "policies" outputting the action to take (i.e. token to append) at each state (i.e. string, or "context"). Importantly, the reward is only on the final state $\boldsymbol{z} = f^{\mathsf{CoT}\text{-}t}(\boldsymbol{x})$ and depends only on the correspondence of the last token in $\boldsymbol{z}$ to the prefix containing the initial state. BC considers learning a policy maximizing some reward based only on observations from an expert, in our case the ground truth f_*, and *without observing the ground-truth rewards*. Most relevant to our paper is the recent work of Foster et al. (2024), which provides general, horizon-free (i.e., independent of T) guarantees on the performance of BC. As in our work, Foster et al. (2024) considers the realizable setting, where the expert f_* is in a given policy class, and distinguishes between the time-invariant (called parameter-sharing in that work) and time-varying cases, noting that the former is essential for the aforementioned horizon-free guarantees. Foster et al. (2024) considers only the CoT framework, where 'actions' at each time step are available to the learner, and indeed, the implication of Theorem 3 for CoT learning is a special case of their result. That work is focused on the statistical task itself and is not interested in either the end-to-end setting or the computational benefits and drawbacks thereof.

Another innovation in Foster et al. (2024) is the consideration of stochasticity in both the environment and the experts. The lack of reward observability (combined with a stochastic expert) necessitates the use of randomized policies—unlike in supervised learning, where deterministic policies are always optimal. In our setting, when mapped to Behavioral Cloning, observing the expert demonstration already reveals the reward, so this consideration is somewhat moot.

Appendix B. Technical Preliminary

Convention for Runtime of Learning. Instead of considering a fixed base class $\mathcal{F}$, we must consider a family $\mathcal{F}_d$ parametrized by (one or more) size parameters d. We will say that $\mathcal{F}_d^{\text{e2e-}T}$ is e2e or CoT learnable in $\text{time}(n, d, T, \varepsilon, \delta)$ using some learning algorithm[12] A, if over the domain $\mathcal{X} = \Sigma^{\leq n}$ (i.e. restricted to prompts of length at most n), $A(S)$ runs in time at most $\text{time}(n, T, d, \varepsilon, \delta)$ almost surely. For an expression $\kappa(\psi_1, \ldots, \psi_k)$ in scalar quantities $(\psi_1, \ldots, \psi_k) \in \mathbb{R}^k$, we say that κ is in $\mathcal{P}oly(\psi_1, \ldots, \psi_k)$ if κ is uniformly bounded by a polynomial for all $(\psi_1, \ldots, \psi_k) \in \mathbb{R}^k$, i.e. there exists a polynomial $p : \mathbb{R}^k \to \mathbb{R}$ such that for every $(\psi_1, \ldots, \psi_k) \in \mathbb{R}^k$, we have $\kappa(\psi_1, \ldots, \psi_k) \leq p(\psi_1, \ldots, \psi_k)$.

We now provide some preliminary background: (i) the definitions of the complexity measures which we use throughout the paper, (ii) generalization bounds in terms of appropriate complexity measures.

Supervised Learning. Consider the supervised learning setup under the 0-1 loss. Consider an input domain $\mathcal{X}$, a finite label set $\mathcal{Y}$, and a hypothesis class $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$. For any $h : \mathcal{X} \to \mathcal{Y}$, define

$$\mathcal{L}_{\mathcal{D}}(h) = \mathbb{E}_{(\mathbf{x}, \mathbf{y}) \sim \mathcal{D}} \left[\mathbb{1}\{h(\mathbf{x}) \neq \mathbf{y}\} \right],$$

where $\mathcal{D}$ is an unknown distribution over $\mathcal{X} \times \mathcal{Y}$. We will say that $\mathcal{D}$ is realizable by a hypothesis class $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$ if $\mathcal{D}_{\mathbf{y}|\mathbf{x}} = h_*(\mathbf{x})$ for some $h_* \in \mathcal{H}$. The distribution $\mathcal{D}$ is unknown and our goal is to learn from samples $S = ((\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_m, \mathbf{y}_m)) \sim_{iid} \mathcal{D}^m$. In all our learnability results (in the positive direction) will be via analyzing the performance of Empirical Risk Minimization (ERM) rule; it takes $S \in (\mathcal{X} \times \mathcal{Y})^*$ as the input and outputs a predictor from $\mathcal{H}$ that has the lowest 0-1 error on the training set S.

$$\mathcal{L}_S(h) = \frac{1}{|S|} \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in S} \mathbb{1}\{h(\mathbf{x}_i) \neq \mathbf{y}_i\} \quad \text{and} \quad \text{ERM}_{\mathcal{H}}(S) = \hat{h} = \arg \min_{h \in \mathcal{H}} \mathcal{L}_S(h). \quad \text{(ERM)}$$

In the realizable case, this reduces to the following rule. Given $S = ((\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_m, \mathbf{y}_m))$

$$\text{CONS}_{\mathcal{H}}(S) : \text{Return some } \hat{h} \in \mathcal{H} \text{ such that } \hat{h}(\mathbf{x}_i) = \mathbf{y}_i, \forall (\mathbf{x}_i, \mathbf{y}_i) \in S. \quad \text{(14)}$$

The sample complexity of the learning rule is characterized by some combinatorial dimensions, which we discuss now.

B.1. Growth Function, Complexity Measures

We start by defining an important definition of the growth function.

Definition 19 (Growth Function) *For any $h : \mathcal{X} \to \mathcal{Y}$, and $S = (\mathbf{x}_1, \ldots, \mathbf{x}_m) \in \mathcal{X}^m$, we define $h(S) := (h(\mathbf{x}_1), \ldots, h(\mathbf{x}_m))$ and $\mathcal{H}(S) := \{h(S) : h \in \mathcal{H}\}$. The growth function $\Gamma_{\mathcal{H}} : \mathbb{N}_+ \to \mathbb{N}_+$*

$$\Gamma_{\mathcal{H}}(m) = \max_{(\mathbf{x}_1, \ldots, \mathbf{x}_m) \in \mathcal{X}^m} |\{(h(\mathbf{x}_1), \ldots, h(\mathbf{x}_m)) : h \in \mathcal{H}\}| = \max_{S \in \mathcal{X}^m} |\mathcal{H}(S)|.$$

We will also abuse the notation and may denote $\Gamma_{\mathcal{H}}(S) = |\mathcal{H}(S)|$ as the number of behaviors possible on S using hypothesis class $\mathcal{H}$.

---

12. Formally, to allow uniform algorithms, we can think of A being implicitly passed T and d.

When $\mathcal{Y} = \{0, 1\}$ is binary, we can define the VC dimension of the class, and its relationship with the growth function given by Sauer's lemma.

Definition 20 (VC Dimension) *When $\mathcal{Y} \in \{0, 1\}$, for any $\mathcal{H} \subseteq \{0, 1\}^{\mathcal{X}}$, we say that $\mathcal{H}$ shatters a set of points $S \in \mathcal{X}^m$ iff $|\mathcal{H}(S)| = 2^{|S|}$. The Vapnik-Chervonenkis dimension of the class, denoted by $\mathrm{VCdim}(\mathcal{H})$, is the largest integer $D \in \mathbb{N}_+$ such that $\Gamma_{\mathcal{H}}(D) = 2^D$. If no such D exists, then we say that $\mathrm{VCdim}(\mathcal{H}) = \infty$.*

Lemma 21 (Sauer's Lemma) *For a hypothesis class $\mathcal{H} \subseteq \{0, 1\}^{\mathcal{X}}$, for every $m \in \mathbb{N}_+$, we have $\Gamma_{\mathcal{H}}(m) \leq (em)^{\mathrm{VCdim}(\mathcal{H})}$. Additionally, for $m \geq \mathrm{VCdim}(\mathcal{H}) \geq 1$:*

$$\Gamma_{\mathcal{H}}(m) \leq \left(\frac{em}{\mathrm{VCdim}(\mathcal{H})} \right)^{\mathrm{VCdim}(\mathcal{H})} .$$

A classic generalization of the VC dimension to non-binary finite outputs is the Natarajan dimension.

Definition 22 (Natarajan Dimension) *Consider any finite $\mathcal{Y}$. We say that a set $S \in \mathcal{X}^m$ is shattered by $\mathcal{H}$ if there exist two functions $h_0, h_1 : \mathcal{X} \to \mathcal{Y}$ such that*

- *For every $\mathbf{x} \in S$, we have $h_0(\mathbf{x}) \neq h_1(\mathbf{x})$*

- *For every $U \subseteq S$, there exists $h \in \mathcal{H}$ such that*

$$\forall \mathbf{x} \in U, \ h(\mathbf{x}) = h_0(\mathbf{x}) \quad and \quad \forall \mathbf{x} \in S \setminus U, \ h(\mathbf{x}) = h_1(\mathbf{x}) .$$

The Natarajan dimension of $\mathcal{H}$, denoted by $\mathrm{Ndim}(\mathcal{H})$, is the cardinality of the largest S that is shattered by $\mathcal{H}$. If there is no largest size, then $\mathrm{Ndim}(\mathcal{H}) = \infty$.

There is a classic generalization of the Sauer's lemma (Lemma 21) also for multiclass labels due to Natarajan Natarajan (1989). Below is a variant (Shalev-Shwartz and Ben-David, 2014, Lemma 29.4).

Lemma 23 (Natarajan's Lemma) *Recall Definition 19 of the growth function. For any $m \in \mathbb{N}_+$*

$$\Gamma_{\mathcal{H}}(m) \leq (|\mathcal{Y}|^2 \cdot m)^{\mathrm{Ndim}(\mathcal{H})} .$$

We will bound the VC dimension (or Natarajan dimension) of the end-to-end class in terms of the Littlestone dimension (or sequential fat shattering dimension) of the base class to achieve an improved dependence in terms of T (Theorem 6). We define these measures now.

Definition 24 ($\mathcal{X}$-labeled Tree) *A $\mathcal{X}$-labeled tree $\boldsymbol{x}$ of depth d is a rooted complete binary tree with nodes labeled by elements of $\mathcal{X}$. The tree $\boldsymbol{x}$ is identified by the sequence $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_d)$ of labeling functions $\boldsymbol{x}_i : \{0, 1\}^{i-1} \to \mathcal{X}$ that provide the labels for each node. Here, $\boldsymbol{x}_1 \in \mathcal{X}$ is the label for the root of the tree, and $\boldsymbol{x}_i$ for $i > 1$ is the label of the node obtained by following the path of length $i - 1$ from the root, with 1 indicating 'right' and 0 indicating 'left'.*

A path of length d is denoted by the sequence $\epsilon = (\epsilon_1, \ldots, \epsilon_d) \in \{0, 1\}^d$. For brevity, we write $\boldsymbol{x}_t(\epsilon)$, but it is understood that $\boldsymbol{x}_t$ only depends on the prefix $(\epsilon_1, \ldots, \epsilon_{t-1})$ of ϵ.

Definition 25 (Littlestone Dimension) *A $\mathcal{X}$-labeled tree $\boldsymbol{x}$ of depth d is said to be shattered by a binary function class $\mathcal{H} \subseteq \{0,1\}^{\mathcal{X}}$ if for all $\epsilon \in \{0,1\}^d$, there exists $h \in \mathcal{H}$ such that for all $i \in [d]$, $h(\boldsymbol{x}_i(\epsilon)) = \epsilon_i$. The Littlestone dimension $\mathrm{Ldim}(\mathcal{H})$ is the largest d such that $\mathcal{H}$ shatters an $\mathcal{X}$-labeled tree of depth d.*

Definition 26 (Sequential Fat-Shattering Dimension) *Given a function class $\mathcal{F} \subseteq \mathbb{R}^{\mathcal{X}}$, we say that $\mathcal{F}$ sequentially shatters at scale α a binary tree $\boldsymbol{x}$ of depth m if there exists a real-valued complete binary tree $\boldsymbol{s}$ of depth m such that for all $\epsilon \in \{\pm 1\}^m$, there is some $f_{\epsilon} \in \mathcal{F}$ such that $\epsilon_i(f_{\epsilon}(\boldsymbol{x}_i(\epsilon)) - \boldsymbol{s}_i(\epsilon)) \geq \alpha/2$. We let $\mathrm{sfat}_{\alpha}(\mathcal{F})$ be the maximal m such that $\mathcal{F}$ sequentially shatters at scale α a binary tree of depth m.*

Lemma 27 (Theorem 7 from Rakhlin et al. (2015)) *Consider a binary-valued function class $\mathcal{H} \subseteq \{0,1\}^{\mathcal{X}}$ with $\mathrm{Ldim}(\mathcal{H}) < \infty$. Then for all $\mathcal{X}$-labeled trees $\boldsymbol{x}$ of depth $d \geq \mathrm{Ldim}(\mathcal{H}) \geq 1$, there exists a set of trees $V(\boldsymbol{x})$ of size*

$$|V(\boldsymbol{x})| \leq \left(\frac{2ed}{\mathrm{Ldim}(\mathcal{H})} \right)^{\mathrm{Ldim}(\mathcal{H})}, \tag{15}$$

such that for all $\epsilon \in \{0,1\}^d$, $h \in \mathcal{H}$, there exists $v \in V$ such that $v(\epsilon) = h(\boldsymbol{x}(\epsilon))$.
More generally, for $\mathcal{H} \subseteq \{0,\ldots,K\}^{\mathcal{X}}$ with finite $\mathrm{sfat}_1(\mathcal{H})$, it holds for $d \geq \mathrm{sfat}_1(\mathcal{H})$ that

$$|V(\boldsymbol{x})| \leq \left(\frac{eKd}{\mathrm{sfat}_1(\mathcal{H})} \right)^{\mathrm{sfat}_1(\mathcal{H})}. \tag{16}$$

Finally, we end by noting an algebraic inequality that we use to get a bound on our desired dimension from a bound on the growth function. For any $a, b > 0$, we have

$$\ln a \leq ab - \ln b - 1, \tag{17}$$

with equality only if $ab = 1$. See (Anthony and Bartlett, 2009, Inequality (1.2), Appendix 1). We have the following corollary, which we will use throughout.

Lemma 28 *For any $N \in \mathbb{N}_+, M \in \mathbb{R}_+$ with $NM \geq 1$, there exists $m \in \mathbb{N}_+$ such that*

$$N \leq m \leq 3N \log_2 \left(\frac{2NM}{\ln 2} \right) \text{ and } m > N \log_2(emM).$$

Proof Using Eq. (17), with $a = emM$ and $b = \frac{\ln 2}{2eNM}$, we have for every $m \in \mathbb{N}_+$,

$$\ln(emM) \leq (emM) \left(\frac{\ln 2}{2eNM} \right) - \ln \left(\frac{\ln 2}{2eNM} \right) - 1.$$

$$A \log_2(emM) \leq \frac{m}{2} + N \log_2 \left(\frac{2MN}{\ln 2} \right).$$

Therefore, in order to ensure $m > N \log_2(emM)$, it suffices to have $\frac{m}{2} > N \log_2 \left(\frac{2NM}{\ln 2} \right)$, which is equivalent to $m > 2A \log_2 \left(\frac{2NM}{\ln 2} \right)$. Finally, noting that $N \log_2 \left(\frac{2NM}{\ln 2} \right) > 1$, so there always exists $m \in \mathbb{N}_+$ such that $N < 2N \log_2 \left(\frac{2NM}{\ln 2} \right) < m \leq 3N \log_2 \left(\frac{2NM}{\ln 2} \right)$, concludes the proof. ∎

B.2. Generalization Bounds

First of all, for finite hypothesis classes, we have the following classical guarantee.

Proposition 29 (Corollary 2.3 from (Shalev-Shwartz and Ben-David, 2014)) *Consider any domain $\mathcal{X}$, a label space $\mathcal{Y}$, and a finite hypothesis class $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$. For any realizable distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$ (i.e. $\mathcal{D}_{\mathbf{y}|\mathbf{x}} = h_*(\mathbf{x})$ for some $h_* \in \mathcal{H}$) over the draw of $S \sim \mathcal{D}^m$ with $m = m(\varepsilon, \delta)$, we have that with probability at least $1 - \delta$,*

$$\mathcal{L}_{\mathcal{D}}(\text{Cons}_{\mathcal{H}}(S)) \leq \varepsilon \quad \text{where} \quad m(\varepsilon, \delta) \leq 2 \left(\frac{\log |\mathcal{H}| + \log(1/\delta)}{\varepsilon} \right).$$

To go beyond cardinality based bounds, we need to consider dimension based guarantees. When $\mathcal{Y} = \{0, 1\}$, VCdim completely characterizes the learnability via the fundamental theorem of statistical learning. For example, see (Shalev-Shwartz and Ben-David, 2014, Theorems 6.7, 6.8).

Proposition 30 (The Fundamental Theorem of Statistical Learning) *There is a universal constant $c > 0$ such that for any domain $\mathcal{X}$, a hypothesis class $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$ with $\mathcal{Y} = \{0, 1\}$ and $\text{VCdim}(\mathcal{H}) < \infty$, and any distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, the following holds. With probability at least $1 - \delta$ over $S \sim \mathcal{D}^m$ with $m = m(\varepsilon, \delta)$,*

$$\mathcal{L}_{\mathcal{D}}(\text{ERM}_{\mathcal{H}}(S)) \leq \inf_{h \in \mathcal{H}} \mathcal{L}_{\mathcal{D}}(h) + \varepsilon \quad \text{and} \quad m(\varepsilon, \delta) \leq c \left(\frac{\text{VCdim}(\mathcal{H}) + \log(1/\delta)}{\varepsilon^2} \right).$$

Moreover, if $\mathcal{D}$ is realizable by $\mathcal{H}$, we have

$$\mathcal{L}_{\mathcal{D}}(\text{Cons}_{\mathcal{H}}(S)) \leq \varepsilon \quad \text{with} \quad m(\varepsilon, \delta) \leq c \left(\frac{\text{VCdim}(\mathcal{H}) \log(1/\varepsilon) + \log(1/\delta)}{\varepsilon} \right).$$

Moreover, for any learning rule $A : (\mathcal{X} \times \mathcal{Y})^* \to \mathcal{Y}^{\mathcal{X}}$, if $m < \frac{\text{VCdim}(\mathcal{H})}{2}$, then there exists a realizable distribution $\mathcal{D}$ such that over the draw of $S \sim \mathcal{D}^m$, we have

$$\mathbb{P}(\mathcal{L}_{\mathcal{D}}(A(S)) \geq 1/4) \geq 0.8.$$

Therefore, the VC dimension completely characterizes the learnability when the labels are $\{0, 1\}$. We will also be interested in the upper bounds in more generality, i.e. general finite alphabet set for the tokens. The performance of (ERM) rule and its sample complexity in terms of the Natarajan dimension is given below. See (Shalev-Shwartz and Ben-David, 2014, Theorem 29.3).

Proposition 31 (The Fundamental Theorem for Multiclass Labels) *There is a universal constant $c > 0$ such that for any domain $\mathcal{X}$, a finite $\mathcal{Y}$, a hypothesis class $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$ with $\text{Ndim}(\mathcal{H}) < \infty$, and any distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, the following holds. Over the draw of $S \sim \mathcal{D}^m$ with $m = m(\varepsilon, \delta)$ we have*

$$\mathcal{L}_{\mathcal{D}}(\text{ERM}_{\mathcal{H}}(S)) \leq \inf_{h \in \mathcal{H}} \mathcal{L}_{\mathcal{D}}(h) + \varepsilon \quad \text{and} \quad m(\varepsilon, \delta) \leq c \left(\frac{\text{Ndim}(\mathcal{H}) \log |\mathcal{Y}| + \log(1/\delta)}{\varepsilon^2} \right).$$

Moreover, if $\mathcal{D}$ is realizable by $\mathcal{H}$, i.e. $\mathcal{D}_{\mathbf{y}|\mathbf{x}} = h_*(\mathbf{x})$ for some $h_* \in \mathcal{H}$, we have

$$\mathcal{L}_{\mathcal{D}}(\text{Cons}_{\mathcal{H}}(S)) \leq \varepsilon \quad \text{with} \quad m(\varepsilon, \delta) \leq \frac{c}{\varepsilon} \left(\text{Ndim}(\mathcal{H}) \log \left(\frac{|\mathcal{Y}| \cdot \text{Ndim}(\mathcal{H})}{\varepsilon} \right) + \log(1/\delta) \right).$$

For our sample complexity results on learning with CoT, we have a more general label space $\mathcal{Y}$, i.e. the entire CoT output. For this, we will rely on the same uniform convergence argument but for the loss class instead.

Generalization Bound in terms of the Loss Class. We let $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ and for any function class $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$, consider the associated loss class $\mathcal{L}^{\text{0-1}}(\mathcal{H}) \subseteq \{0,1\}^{\mathcal{Z}}$ defined by

$$\mathcal{L}^{\text{0-1}}(\mathcal{H}) = \{\ell_h : (\mathbf{x}, \mathbf{y}) \mapsto \mathbb{1}\{h(\mathbf{x}) \neq \mathbf{y}\} \mid h \in \mathcal{H}\}.$$

One can now consider $\text{VCdim}(\mathcal{L}^{\text{0-1}}(\mathcal{H}))$ (of the loss class) over the domain $\mathcal{Z}$. Using the uniform convergence argument for Proposition 30 but now for the loss class $\mathcal{L}^{\text{0-1}}(\mathcal{H})$, we have the following guarantee in the realizable case.

Proposition 32 (General Guarantee) *There is a universal constant $c > 0$ such that for any domain $\mathcal{X}$, a label space $\mathcal{Y}$, a hypothesis class $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$ with $\text{VCdim}(\mathcal{L}^{\text{0-1}}(\mathcal{H})) < \infty$, and any distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$ which is realizable by $\mathcal{H}$. With probability at least $1 - \delta$ over $S \sim \mathcal{D}^m$ with $m = m(\varepsilon, \delta)$,*

$$\mathcal{L}_{\mathcal{D}}(\text{CONS}_{\mathcal{H}}(S)) \leq \varepsilon \quad \text{where} \quad m(\varepsilon, \delta) \leq c \left(\frac{\text{VCdim}(\mathcal{L}^{\text{0-1}}(\mathcal{H})) \log(1/\varepsilon) + \log(1/\delta)}{\varepsilon} \right).$$

Appendix C. Proofs from Section 3

In this section, we will prove all our results from Section 3 except Theorem 39 (which will be shown in Appendix D). Along with that, in Appendix D, we will also show other complementary lower bounds and results mentioned during the discussion in Section 3).

Proof [Proof of Theorem 3] This is a corollary of Proposition 29 after noting that $\left| \mathcal{F}^{\text{e2e-}T} \right| \leq |\mathcal{F}|$ and $m^{\text{CoT-}T} \leq m^{\text{e2e-}T}$. ∎

The proof of sample complexity upper bounds, namely, Theorems 4 to 6 are in Sections C.1 to C.3. The computational complexity result from Section 3.2 is proven in Section C.4.

C.1. Proof of Theorem 4 and its extension to non-binary alphabets

In order to prove Theorem 4 and its analog for general finite Σ, we will need to bound the VC dimension or Natarajan dimension of the end-to-end class.

Theorem 33 (VC of $\mathcal{F}^{\text{e2e-}T}$ in terms VC of $\mathcal{F}$) *Consider any finite Σ, a base function class $\mathcal{F} \subseteq \Sigma^{\Sigma^*}$ and generation length $T \in \mathbb{N}_+$.*

- *For binary $\Sigma = \{0,1\}$:*

$$\text{VCdim}(\mathcal{F}^{\text{e2e-}T}) \leq 6\,T \cdot \text{VCdim}(\mathcal{F}).$$

- *For non-binary finite Σ:*

$$\text{Ndim}(\mathcal{F}^{\text{e2e-}T}) \leq 9\,T \cdot \text{Ndim}(\mathcal{F}) \log_2 \left(\frac{2\,\text{Ndim}(\mathcal{F})|\Sigma|}{e \ln 2} \right).$$

Proof [Proof of Theorem 33] Consider any $\mathcal{F} \subseteq \Sigma^{\Sigma^*}$. Our goal is to bound the growth function $\Gamma_T := \Gamma_{\mathcal{F}^{\text{e2e-}T}}$ of the class $\mathcal{F}^{\text{e2e-}T}$ for $T \in \mathbb{N}_+$. To this end, consider any fixed set $S \in (\Sigma^*)^m$ of size $|S| = m$. Let's consider the set S' of all possible "partial" extensions of the examples in S.

$$S' = \{(\boldsymbol{x}, \boldsymbol{u}) : \boldsymbol{x} \in S, \boldsymbol{u} \in \Sigma^t, 0 \le t \le (T-1)\}.$$

Clearly, $|S'| \le m \cdot |\Sigma|^T$. For any $f_1, f_2 \in \mathcal{F}$ such that $f_1^{\text{e2e-}T}(S) \ne f_2^{\text{e2e-}T}(S)$, we have that $f_1(S') \ne f_2(S')$. Therefore, $\Gamma_T(S) \le \Gamma_{\mathcal{F}}(S') \le \Gamma_{\mathcal{F}}(m\,|\Sigma|^T).$[13] The argument holds for any S with $|S| = m$, and thus,

$$\Gamma_T(m) = \max_{|S|=m} \Gamma_T(S) \le \Gamma_{\mathcal{F}}(m\,|\Sigma|^T).$$

- $\Sigma = \{0, 1\}$ (binary alphabets): In this case, using Sauer's lemma (Lemma 21), for any $m \ge \text{VCdim}(\mathcal{F})$,

$$\Gamma_T(m) \le \left(\frac{em2^T}{\text{VCdim}(\mathcal{F})}\right)^{\text{VCdim}(\mathcal{F})}.$$

Therefore, $\text{VCdim}(\mathcal{F}^{\text{e2e-}T})$ can be bounded by any $m \in \mathbb{N}_+$ satisfying $2^m > \Gamma_T(m)$ and $m \ge \text{VCdim}(\mathcal{F})$. Using the derived upper bound on $\Gamma_T(m)$, it suffices to choose m such that

$$m > \text{VCdim}(\mathcal{F}) \log_2 \left(\frac{em\,2^T}{\text{VCdim}(\mathcal{F})}\right) \quad \text{and} \quad m \ge \text{VCdim}(\mathcal{F}).$$

Using Lemma 28 with $N = \text{VCdim}(\mathcal{F})$ and $M = 2^T/\text{VCdim}(\mathcal{F})$, we directly argue the existence of $m \in \mathbb{N}_+$ such that

$$\text{VCdim}(\mathcal{F}^{\text{e2e-}T}) \le m \le 3\,\text{VCdim}(\mathcal{F}) \log_2 \left(\frac{2 \cdot 2^T}{\ln 2}\right) \le 6\,T\,\text{VCdim}(\mathcal{F}).$$

- For a finite Σ : Using Natarajan's lemma (Lemma 23) for any $m \in \mathbb{N}_+$,

$$\Gamma_{\mathcal{F}^{\text{e2e-}T}}(m) \le \Gamma_{\mathcal{F}}(m \cdot |\Sigma|^T) \le \left(|\Sigma|^2 \cdot m \cdot |\Sigma|^T\right)^{\text{Ndim}(\mathcal{F})}.$$

Again, $\text{Ndim}(\mathcal{F}^{\text{e2e-}T})$ can be bounded by any $m \in \mathbb{N}_+$ for which $m > \text{Ndim}(\mathcal{F}) \log_2 \left(|\Sigma|^{T+2} \cdot m\right).$ Applying Lemma 28 with $N = \text{Ndim}(\mathcal{F})$ and $M = |\Sigma|^{T+2}/e$, we obtain that there exists such $m \in \mathbb{N}_+$ such that

$$\text{Ndim}(\mathcal{F}^{\text{e2e-}T}) \le m \le 3\text{Ndim}(\mathcal{F}) \log_2 \left(\frac{2\text{Ndim}(\mathcal{F})|\Sigma|^{T+2}}{e \ln 2}\right)$$

$$\le 9\,T\,\text{Ndim}(\mathcal{F}) \log_2 \left(\frac{2\text{Ndim}(\mathcal{F})|\Sigma|}{e \ln 2}\right).$$

$\blacksquare$

Once we have the bound on the VC dimension of $\mathcal{F}^{\text{e2e-}T}$ in terms of $\text{VCdim}(\mathcal{F})$, using Proposition 30 we have the following proof.

---

13. Note that this step in the proof critically requires time-invariance.

Proof [Proof of Theorem 4] The proof directly follows by combining Theorem 5 with Proposition 29, after recalling that e2e-learnability of $\mathcal{F}^{\text{e2e-}T}$ (Definition 1) is just standard supervised learning setup from Appendix B for $\mathcal{H} = \mathcal{F}^{\text{e2e-lin}}$. ∎

We also have the following corollary for non-binary but finite Σ.

Corollary 34 *There exists a universal constant $c > 0$ such that for any base class $\mathcal{F}$ over a finite Σ and generation length $T \in \mathbb{N}_+$, the class $\mathcal{F}^{\text{e2e-}T}$ is e2e-learnable using CONS_{e2e} with*

$$m^{\text{e2e-}T} \leq \frac{c}{\varepsilon}\left(T \cdot \text{Ndim}(\mathcal{F}) \log\left(\text{Ndim}(\mathcal{F})|\Sigma|\right) \log\left(\frac{1}{\varepsilon}\right) + \log\left(\frac{1}{\delta}\right)\right) .$$

Proof [Proof of Corollary 34] This directly follows from substituting the bound on $\text{Ndim}(\mathcal{F}^{\text{e2e-}T})$ from Theorem 33 in Proposition 31. ∎

C.2. Proof of Theorem 6 and its extension for non-binary alphabets

We now prove the guarantee based on the Littlestone dimension to bound $\text{VCdim}(\mathcal{F}^{\text{e2e-}T})$.

Theorem 35 (VCdim of $\mathcal{F}^{\text{e2e-}T}$ in terms of Ldim of $\mathcal{F}$) *Consider any finite Σ, a base hypothesis class $\mathcal{F} \subseteq \Sigma^{\Sigma^*}$ and generation length $T \in \mathbb{N}_+$. When $\Sigma = \{0,1\}$, we have*

$$\text{VCdim}(\mathcal{F}^{\text{e2e-}T}) \leq 10 \, \text{Ldim}(\mathcal{F}) \cdot \log(T).$$

More generally, for any finite Σ, it holds that

$$\text{Ndim}(\mathcal{F}^{\text{e2e-}T}) \leq 20 \, \text{sfat}_1(\mathcal{F}) \cdot \log\left(T|\Sigma|\right).$$

Proof [Proof of Theorem 35] *The case when $\Sigma = \{0,1\}$*: Let $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m \in \Sigma^*$ be m points. Consider the tree $\boldsymbol{x}$ of depth $M = \overline{m(T+1)}$ where for $t \in [M]$,

$$\boldsymbol{x}_t(\epsilon) = \begin{cases} \boldsymbol{x}_i & \text{if } t = (i-1)(T+1) \\ [\boldsymbol{x}_i, \epsilon_{t-(i-1)(T+1)}, \ldots, \epsilon_t] & \text{otherwise.} \end{cases}$$

For every $f \in \mathcal{F}$ define the path $\epsilon^f \in \{0,1\}^M$ such that for all $0 \leq i < m$ and $1 \leq s \leq T$,

$$f(\boldsymbol{x}_{(T+1)i+s}(\epsilon^f)) = f^{\text{e2e-}s}(\boldsymbol{x}_i).$$

Let $P_{\mathcal{F},\boldsymbol{x}} = \left\{\epsilon^f : f \in \mathcal{F}\right\}$ be the set of root-to-leaf paths realized by $\mathcal{F}$ on $\boldsymbol{x}$. Now by definition,

$$\Gamma_T(m) = \Gamma_{\mathcal{F}^{\text{e2e-}T}}(m) \leq \max_{\boldsymbol{x}} |(f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_m)) : f \in \mathcal{F}| \leq \max_{\boldsymbol{x}} |P_{\mathcal{F},\boldsymbol{x}}|.$$

By Lemma 27, we know that there exists a set of trees $V(\boldsymbol{x})$ such that

$$|V(\boldsymbol{x})| \leq \left(\frac{2eM}{\text{Ldim}(\mathcal{F})}\right)^{\text{Ldim}(\mathcal{F})}, \text{ and } \forall f \in \mathcal{F}, \epsilon \in \{0,1\}^M; \exists v \in V(\boldsymbol{x}), v(\epsilon) = f(\boldsymbol{x}(\epsilon)).$$

We will show that $|P_{\mathcal{F},\boldsymbol{x}}| \leq |V(\boldsymbol{x})|$ by proving that no two distinct paths in $P_{\mathcal{F},\boldsymbol{x}}$ can belong to the same tree in $V(\boldsymbol{x})$. Let us prove by contradiction. Consider two distinct paths $\epsilon^f, \epsilon^g \in P_{\mathcal{F},\boldsymbol{x}}$ and let

t be the first index where they differ and consider the tree v that covers both. Up to node $t-1$, both share the same path in v however at node t they assign opposite labels. Since the tree can only assign one value to the node t, it cannot cover both paths. Hence, $|P_{\mathcal{F},\boldsymbol{x}}| \leq |V(\boldsymbol{x})| \leq \left(\frac{2eM}{\mathrm{Ldim}(\mathcal{F})} \right)^{\mathrm{Ldim}(\mathcal{F})}$.

To conclude, note that $\Gamma_T(m) < 2^m$ if and only if $m \geq \mathrm{VCdim}(\mathcal{F}^{\text{e2e-}T})$. Thus, it holds that if $m > \mathrm{Ldim}(\mathcal{F})$ and

$$(2emT/\mathrm{Ldim}(\mathcal{F}))^{\mathrm{Ldim}(\mathcal{F})} < 2^m, \tag{18}$$

then m upper bounds $\mathrm{VCdim}(\mathcal{F}^{\text{e2e-}T})$. Setting $m = 10\,\mathrm{Ldim}(\mathcal{F}) \cdot \log(T)$ and observing that such an m satisfies the desired bound concludes the proof.

General finite Σ: WLOG we assume that $0, 1 \notin \Sigma$. Let $k = \lceil \log_2(|\Sigma|) \rceil$ and let $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m \in \Sigma^*$. Consider the binary tree $\boldsymbol{x}$ of depth $M = mk(T+1)$ where for $t \in [M]$,

$$\boldsymbol{x}_t(\epsilon) = \begin{cases} \boldsymbol{x}_i & \text{if } t = (i-1)k(T+1) \\ [\boldsymbol{x}_i, \epsilon_{t-(i-1)k(T+1)}, \ldots, \epsilon_t] & \text{otherwise.} \end{cases} \tag{19}$$

Choose some injection $\rho : \Sigma \to \{0,1\}^k$ and for any $f \in \mathcal{F}$, let ϵ^f be such that for all $0 \leq i < m$ and all $1 \leq s \leq T$,

$$\rho \circ f(\boldsymbol{z}_{(T+1)ik+sk}(\epsilon^f)) = \epsilon_{(T+1)ik+sk+1:(T+1)ik+(s+1)k}. \tag{20}$$

Let $P_{\mathcal{F},\boldsymbol{x}} = \left\{ \epsilon^f : f \in \mathcal{F} \right\}$ be the set of root-to-leaf paths realized by $\mathcal{F}$ on $\boldsymbol{x}$. By virtue of this mapping, we see that $|\left\{ (f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_m)) | f \in \mathcal{F} \right\}| \leq |P_{\mathcal{F},\boldsymbol{x}}|$. Since the nodes in the constructed $\boldsymbol{x}$ are not in Σ^* necessarily, to apply Lemma 27, we first extend $\mathcal{F}$ to $\mathcal{F}'$ to handle these inputs. For each $f \in \mathcal{F}$, we construct $f' : \Sigma'^* \to \Sigma'$ where $\Sigma' = \Sigma \cup \{0,1\}$ such that f' matches f on all inputs of the form Σ^* and is 0 everywhere else. Since this is a trivial extension of $\mathcal{F}$, it does not change the complexity of this class. Now applying 27 on the tree and $\mathcal{F}'$, we know that there exists a set of trees $V(\boldsymbol{x})$ such that

$$|V(\boldsymbol{x})| \leq \left(\frac{e(|\Sigma|+2)M}{\mathrm{sfat}_1(\mathcal{H})} \right)^{\mathrm{sfat}_1(\mathcal{H})}, \text{ and } \forall f' \in \mathcal{F}', \epsilon \in \{0,1\}^M; \exists v \in V(\boldsymbol{z}), v(\epsilon) = f'(\boldsymbol{z}(\epsilon)).$$

We can use the same argument as before about two paths not sharing the same tree to show that $|P_{\mathcal{F},\boldsymbol{x}}| \leq |V(\boldsymbol{x})|$. Then we have that for any $m > \mathrm{sfat}_1(\mathcal{F})$, it holds that

$$|\mathcal{F}^{\text{e2e-}T}(\boldsymbol{z})| \leq (e(|\Sigma|+2)(T+1)km/\mathrm{sfat}_1(\mathcal{F}))^{\mathrm{sfat}_1(\mathcal{F})} \leq (4e|\Sigma|Tkm/\mathrm{sfat}_1(\mathcal{F}))^{\mathrm{sfat}_1(\mathcal{F})}. \tag{21}$$

As above, if $2^m > \Gamma_T(m)$ then it must hold that $m > \mathrm{Ndim}(\mathcal{F}^{\text{e2e-}T})$. Thus if $m > \mathrm{sfat}_1(\mathcal{F})$ and

$$\mathrm{sfat}_1(\mathcal{F}) \log \left(\frac{4e|\Sigma| \log(|\Sigma|)Tm}{\mathrm{sfat}_1(\mathcal{F})} \right) < m, \tag{22}$$

then $\mathrm{Ndim}(\mathcal{F}^{\text{e2e-}T}) \leq m$. The result follows. $\blacksquare$

Our sample complexity bound in Theorem 6 follows directly from this bound.
Proof [Proof of Theorem 6] Substitute the bound of $\mathrm{VCdim}(\mathcal{F}^{\text{e2e-}T})$ from Theorem 35 in Proposition 30. $\blacksquare$

We also have the following corollary of Theorem 35 for non-binary alphabets.

Corollary 36 *There exists a universal constant $c > 0$ such that for any base class $\mathcal{F}$ over a finite Σ and generation length $T \in \mathbb{N}_+$, the class $\mathcal{F}^{\text{e2e-}T}$ is e2e-learnable using* CONS_{e2e} *with*

$$m^{\text{e2e-}T} \leq \frac{c}{\varepsilon} \left(\text{sfat}_1(\mathcal{F}) \log \left(T \left| \Sigma \right| \right) \log \left(\frac{1}{\varepsilon} \right) + \log \left(\frac{1}{\delta} \right) \right) .$$

Proof [Proof of Corollary 36] This follows from substituting the bound on $\text{Ndim}(\mathcal{F}^{\text{e2e-}T})$ from Theorem 35 in Proposition 31. ∎

C.3. Proof of Theorem 5 and its extension for non-binary alphabets

We now prove the sample complexity of CoT-learnability given in Theorem 5. For every $\boldsymbol{z} \in \Sigma^*$, which corresponds to a chain-of-thought of length T on some input $\boldsymbol{x}$, we can decompose $\boldsymbol{z} = (\boldsymbol{x}, \boldsymbol{u})$ where $\boldsymbol{u} = \boldsymbol{z}[-T :]$ are the last T tokens. Then CoT-learnability of $\mathcal{F}^{\text{e2e-}T}$ is equivalent to a supervised learning problem (Appendix B) with the domain $\mathcal{X} = \Sigma^*$, the label space $\mathcal{Y} = \Sigma^T$ and $\mathcal{H} = \mathcal{F}^{\text{CoT-}T}$. The learning rule $\text{CONS}_{\mathcal{H}}$ in (14) is equivalent to the rule CONS_{CoT} in this setup.

Our goal is to apply Proposition 32 on the corresponding loss class. The loss class $\mathcal{L}^{0\text{-}1}(\mathcal{H})$ discussed in Section B.2 exactly corresponds to the loss class from (10) in Section A.1, also written below.

$$\mathcal{L}(\mathcal{F}^{\text{CoT-}T}) := \{\ell_f : \boldsymbol{z} \mapsto \mathbb{1}\{\boldsymbol{z} \neq f^{\text{CoT-}T}(\boldsymbol{x})\} \mid f \in \mathcal{F}\}, \text{ where } \boldsymbol{x} = \boldsymbol{z}[: -(T+1)]. \tag{23}$$

Therefore, our goal is to bound the VC dimension of the loss class.

Theorem 37 *Consider any finite Σ, a base class $\mathcal{F} \subseteq \Sigma^{\Sigma^*}$, and the associated $\mathcal{L}(\mathcal{F}^{\text{CoT-}T})$.*

- *For $\Sigma = \{0, 1\}$:*

$$\text{VCdim}(\mathcal{L}(\mathcal{F}^{\text{CoT-}T})) \leq 3 \, \text{VCdim}(\mathcal{F}) \log_2 \left(\frac{2\,T}{\ln 2} \right) .$$

- *For any non-binary finite Σ:*

$$\text{VCdim}(\mathcal{L}(\mathcal{F}^{\text{CoT-}T})) \leq 3 \, \text{Ndim}(\mathcal{F}) \log_2 \left(\frac{2\,\text{Ndim}(\mathcal{F})|\Sigma|^2 T}{e \ln 2} \right) .$$

Proof [Proof of Theorem 37] The main idea is to bound the growth function of $\Gamma_{\mathcal{L}(\mathcal{F}^{\text{CoT-}T})}(m)$ in terms of the growth function of $\Gamma_{\mathcal{F}}(m)$, and then use Sauer's Lemma (or Natarajan Lemma for non-binary Σ). Towards this, consider any $S = ((\boldsymbol{x}_1, \boldsymbol{u}_1), \ldots, (\boldsymbol{x}_m, \boldsymbol{u}_m)) \in (\Sigma^* \times \Sigma^T)^m$. From this, we can consider the set $\text{pfx}(S) = \big(\boldsymbol{p}_{(i,t)} : 1 \leq i \leq m, 0 \leq t \leq T - 1\big)$, of all the prefixes where

$$\boldsymbol{p}_{(i,t)} = [\boldsymbol{x}_i, \boldsymbol{u}_1[\,1\,], \ldots, \boldsymbol{u}_1[\,t\,]] \in \Sigma^* \times \Sigma^t . \tag{24}$$

We then have

$$\begin{aligned}
\Gamma_{\mathcal{L}(\mathcal{F}^{\text{CoT-}T})}(S) = |\mathcal{L}(\mathcal{F}^{\text{CoT-}T})(S)| &= \{(\ell_f(\boldsymbol{x}_1, \boldsymbol{u}_1), \ldots, \ell_f(\boldsymbol{x}_m, \boldsymbol{u}_m)) : f \in \mathcal{F}\}| \quad \text{(By definition)} \\
&\leq |\{\big(f(\boldsymbol{p}_{(1,0)}), \ldots, f(\boldsymbol{p}_{(m,T-1)})\big) : f \in \mathcal{F}\}| \\
&= |\mathcal{F}(\text{pfx}(S))| \leq \Gamma_{\mathcal{F}}(mT) . \quad \text{(As } |\text{pfx}(S)| = mT)
\end{aligned}$$

Here the only inequality followed from this critical observation: for any two $f, g \in \mathcal{F}$ such that

$$(\ell_f(\boldsymbol{x}_1, \boldsymbol{u}_1), \ldots, \ell_f(\boldsymbol{x}_m, \boldsymbol{u}_m)) \neq (\ell_g(\boldsymbol{x}_1, \boldsymbol{u}_1), \ldots, \ell_g(\boldsymbol{x}_m, \boldsymbol{u}_m)),$$

there exists $1 \leq i^* \leq m$, $0 \leq t^* \leq T - 1$ such that $f(\boldsymbol{p}_{(i^*, t^*)}) \neq g(\boldsymbol{p}_{(i^*, t^*)})$. Therefore, the number of behaviors of the loss class on S, can be bounded by the number of behaviors of the function class $\mathcal{F}$ on the set of all prefixes, i.e. $|\mathcal{L}(\mathcal{F}^{\mathsf{CoT}\text{-}T})| \leq |\mathcal{F}(\mathrm{pfx}(S))|$. As the argument holds for any $|S| = m$, we have

$$\Gamma_{\mathcal{L}(\mathcal{F}^{\mathsf{CoT}\text{-}T})}(m) \leq \Gamma_{\mathcal{F}}(m \cdot T).$$

We now break into each case and bound $\mathrm{VCdim}(\mathcal{L}(\mathcal{F}^{\mathsf{CoT}\text{-}T}))$.

- Binary $\Sigma = \{0, 1\}$: In this case, using Sauer's lemma (Lemma 21), for any $m \geq \mathrm{VCdim}(\mathcal{F})$,

$$\Gamma_{\mathcal{L}(\mathcal{F}^{\mathsf{CoT}\text{-}T})}(m) \leq \Gamma_{\mathcal{F}}(m \cdot T) \leq \left(\frac{emT}{\mathrm{VCdim}(\mathcal{F})} \right)^{\mathrm{VCdim}(\mathcal{F})}.$$

  Therefore, $\mathrm{VCdim}(\mathcal{L}(\mathcal{F}^{\mathsf{CoT}\text{-}T}))$ can be bounded by m such that

$$m > \mathrm{VCdim}(\mathcal{F}) \log_2 \left(\frac{emT}{\mathrm{VCdim}(\mathcal{F})} \right) \quad \text{and} \quad m \geq \mathrm{VCdim}(\mathcal{F}).$$

  Using Lemma 28 with $N = \mathrm{VCdim}(\mathcal{F})$ and $M = T / \mathrm{VCdim}(\mathcal{F})$, we directly obtain that there exists such $m \in \mathbb{N}_+$ such that

$$\mathrm{VCdim}(\mathcal{L}(\mathcal{F}^{\mathsf{CoT}\text{-}T})) \leq m \leq 3 \cdot \mathrm{VCdim}(\mathcal{F}) \log_2 \left(\frac{2T}{\ln 2} \right).$$

- For any non-binary finite Σ : Using Natarajan's lemma (Lemma 23) for any $m \in \mathbb{N}_+$,

$$\Gamma_{\mathcal{L}(\mathcal{F}^{\mathsf{CoT}\text{-}T})}(m) \leq \Gamma_{\mathcal{F}}(m \cdot T) \leq \left(|\Sigma|^2 \cdot m \cdot T \right)^{\mathrm{Ndim}(\mathcal{F})}.$$

  Again, $\mathrm{VCdim}(\mathcal{L}(\mathcal{F}^{\mathsf{CoT}\text{-}T})) \leq m$ for any m s.t. $m > \mathrm{Ndim}(\mathcal{F}) \log_2 \left(|\Sigma|^2 \cdot m \cdot T \right)$. Applying Lemma 28 with $N = \mathrm{Ndim}(\mathcal{F})$ and $M = \frac{|\Sigma|^2 T}{e}$, we obtain that there exists such $m \in \mathbb{N}_+$ such that

$$\mathrm{VCdim}(\mathcal{L}(\mathcal{F}^{\mathsf{CoT}\text{-}T})) \leq m \leq 3\mathrm{Ndim}(\mathcal{F}) \log_2 \left(\frac{2\,\mathrm{Ndim}(\mathcal{F}) |\Sigma|^2 T}{e \ln 2} \right).$$

$\blacksquare$

This bound on $\mathrm{VCdim}(\mathcal{L}(\mathcal{F}^{\mathsf{CoT}\text{-}T}))$ implies the desired sample complexity result.

Proof [Proof of Theorem 5] Recall the discussion of the equivalence CoT-learnability problem with a supervised learning problem (Section B.2) with $\mathcal{H} = \mathcal{F}^{\mathsf{CoT}\text{-}T}$. By plugging the bound from Theorem 37 on $\mathrm{VCdim}(\mathcal{L}(\mathcal{F}^{\mathsf{e2e}\text{-}T}))$ in Proposition 32 as $\mathcal{L}(\mathcal{F}^{\mathsf{CoT}\text{-}T})$ corresponds to $\mathcal{L}^{0\text{-}1}(\mathcal{H})$, the theorem follows. $\blacksquare$

We also have the following non-binary but finite Σ corollary for CoT-learnability. The proof again follows from combining Theorem 37 and Proposition 32.

Corollary 38 *There exists a universal constant $c > 0$ such that for any base class $\mathcal{F} \subseteq \Sigma^{\Sigma^*}$ over a finite Σ and generation length $T \in \mathbb{N}_+$, the class $\mathcal{F}^{\mathsf{e2e}\text{-}T}$ is CoT-learnable using $\mathrm{CONS}_{\mathsf{CoT}}$ with*

$$m^{\mathsf{CoT}\text{-}T} \leq \frac{c}{\varepsilon} \left(\mathrm{Ndim}(\mathcal{F}) \log \left(T \, |\Sigma| \, \mathrm{Ndim}(\mathcal{F}) \right) \log \left(\frac{1}{\varepsilon} \right) + \log \left(\frac{1}{\delta} \right) \right).$$

C.4. Computational Complexity

We finally show that a tractable $\text{CONS}_{\mathcal{F}}$ oracle implies a computationally tractable CoT learnability of $\mathcal{F}^{\text{e2e-}T}$ via a chain-of-thought generated by $\mathcal{F}$. We first start with the proof of Theorem 8 which reduces CoT-learnability to a consistency problem on the base class $\mathcal{F}$.

Proof [Proof of Theorem 8] For any CoT dataset $S_{\text{CoT}} = (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_m) \in (\Sigma^* \times \Sigma^T)^m$, one can first create the dataset of a prefix and the associated next-token pairs for all the examples. Formally, we decompose $\boldsymbol{z}_i = (\boldsymbol{x}_i, \boldsymbol{u}_i)$ again and consider a dataset $\tilde{S} = ((\boldsymbol{p}_{(i,t)}, y_{i,t}) : i \in [m], 0 \leq t \leq (T-1))$ of (prefix,next-token) pairs where

$$\boldsymbol{p}_{(i,t)} = [\boldsymbol{x}_i, \boldsymbol{u}_i[1], \ldots, \boldsymbol{u}_i[t]] \in \Sigma^* \times \Sigma^t, \quad y_{(i,t)} = \boldsymbol{u}_i[t+1] \in \Sigma, \quad \text{for } i \in [m], 0 \leq i \leq (T-1).$$

It is easy to observe that if we find a consistent $\hat{f} \in \mathcal{F}$ on $\tilde{S}$ (in the sense of $\text{CONS}_{\mathcal{F}}$), then we also have that this predictor is consistent with the entire chain-of-thought, i.e. $\hat{f}^{\text{CoT-}T}(\boldsymbol{x}_i) = \boldsymbol{z}_i$ for all $i \in [m]$. If the original inputs $\boldsymbol{x}_i$'s are of length at most n, then the prefixes $p_{(i,t)}$ are of length at most $n + T$. Moreover, we have $|\tilde{S}| = \tilde{m} \leq m \cdot T$ and thus, one such call of $\text{CONS}_{\mathcal{F}}$ on $\tilde{S}$ suffices. The initial input processing of creating $\tilde{S}$ from S, and also finally returning $\hat{f}^{\text{e2e-}T}$ can be done in additional time $O(\tilde{m})$, concluding the implementation of CONS_{CoT}. ∎

Proof [Proof of Corollary 9] This is a direct corollary of Theorems 5 and 8. If $\text{VCdim}(\mathcal{F}_d) = \mathcal{P}oly(d)$, then implementing CONS_{CoT} on a CoT training set S_{CoT} of size

$$|S_{\text{CoT}}| = O(\varepsilon^{-1}(\text{VCdim}(\mathcal{F}_d) \log T \log \varepsilon^{-1} + \log \delta^{-1})) = \mathcal{P}oly(d, T, \varepsilon^{-1}, \delta^{-1})$$

suffices for CoT-learnability of $\mathcal{F}_d^{\text{e2e-}T}$ by Theorem 5. Moreover, by Theorem 8, we can implement this by calling $\text{CONS}_{\mathcal{F}_d}$ on a sample set $\tilde{S}$ of size at most $|\tilde{S}| = O(T \cdot |S_{\text{CoT}}|) = \mathcal{P}oly(d, T, \varepsilon^{-1}, \log \delta^{-1})$. Finally, for input distributions that are supported on sequences of length at most n, the description length of S_{CoT} is $\mathcal{P}oly(n + T, d, |S_{\text{CoT}}|)$. Here we are subsuming the $\log |\Sigma|$ dependence to be already captured in the size parameter as it is inherent to the base class $\mathcal{F}_d$. Thus even the description length of the set $\tilde{S}$ is $\mathcal{P}oly(n + T, d, |\tilde{S}|) = \mathcal{P}oly(n, d, T, \varepsilon^{-1}, \log \delta^{-1})$. As such, as long as $\text{CONS}_{\mathcal{F}_d}$ is implementable in time polynomial in its input and the size parameter d, we have that $\mathcal{F}^{\text{e2e-}T}$ is CoT-learnable in time $\mathcal{P}oly(n, d, T, \varepsilon^{-1}, \log \delta^{-1})$. ∎

Appendix D. Complementary Results from Section 3

In this section, we provide all our complementary results mentioned during the discussion in Section 3.

D.1. T Dependence is Unavoidable for VC-based End-to-End Learning Sample Complexity

We start with the following theorem that establishes that a linear dependence on the generation length T in the sample complexity is necessary for a guarantee based on VCdim of the base class in the worst-case.

Theorem 39 *For every $D, T \in \mathbb{N}_+$, there exists a base class $\mathcal{F} \subseteq \{0,1\}^{\{0,1\}^*}$ with $\text{VCdim}(\mathcal{F}) = D$, and a distribution $\mathcal{D}$ over $\{0,1\}^n$ for $n = \lceil \log(DT) \rceil + 1$ such that for any learning rule A there exists $f_* \in \mathcal{F}$, s.t. with probability at least 0.8 over S_{e2e} of size $m < \frac{DT}{2}$ (sampled as in Definition 1), we have $L_{\mathcal{D}, f_*}^{0\text{-}1}(A(S_{\text{e2e}})) \geq \frac{1}{4}$.*

The proof will follow by using Proposition 30 once we have a construction of a family of base classes with $\mathrm{VCdim}(\mathcal{F}^{\text{e2e-}T}) = \Omega(T \cdot \mathrm{VCdim}(\mathcal{F}))$. Formally, we show the following.

Theorem 40 *For $\Sigma = \{0,1\}$, for any $D, T \in \mathbb{N}_+$, there exists $\mathcal{F} \subseteq \Sigma^{\Sigma^*}$ such that*

$$\mathrm{VCdim}(\mathcal{F}) = D \quad but \quad \mathrm{VCdim}(\mathcal{F}^{\text{e2e-}T}) = T{\cdot}D, \ over\ domain\ \mathcal{X} = \Sigma^n\ with\ n = \lceil \log_2(DT) \rceil + 1.$$

We will return to the proof of this theorem—for now it is immediate to see the following proof.
Proof [Proof of Theorem 39] Use the negative direction of the Fundamental Theorem of Statistical Learning (Proposition 30), and recall that Definition 1 is simply a supervised learning for $\mathcal{F}^{\text{e2e-}T}$ class. ∎

Let us now show the construction of such base classes.
Proof [Proof of Theorem 40] The claim trivially holds for $T = 1$. Thus, below we consider $T \geq 2$.
Description of the Class. For any target $D, T \in \mathbb{N}_+$, we aim to construct a hypothesis class $\mathcal{F}$ from $\{0,1\}^*$ to $\{0,1\}$. Consider the following set of $M := D \cdot T$ strings of length $n = \lceil \log_2 M \rceil + 1$ points.

$$S = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M), \quad \text{where } \boldsymbol{x}_i = 1 \underbrace{(\text{bit representation of } i)}_{\lceil \log_2 M \rceil \text{ bits}}.$$

Our base class $\mathcal{F}$ has functions $f_{\boldsymbol{b}}$ that will correspond to sign-patterns on M points:

$$\mathcal{F} = \{f_{\boldsymbol{b}} : \boldsymbol{b} \in \{0,1\}^M\}.$$

We will define each hypothesis in such a way that $\mathrm{VCdim}(\mathcal{F}) = D$, but when we apply each $f_{\boldsymbol{b}}$ auto-regressively T times on the points in S, we realize the sign-pattern $\boldsymbol{b}$ at the end, i.e. $f_{\boldsymbol{b}}^{\text{e2e-}T}(S) = \boldsymbol{b}$.

$$f_{\boldsymbol{b}}(\boldsymbol{x}) = \begin{cases} b_{jD+k}, & \text{if } \boldsymbol{x} = 0^* \underbrace{\boldsymbol{x}_i}_{n \text{ bits}} \underbrace{b_k b_{D+k} \ldots b_{(j-1)D+k}}_{j \text{ bits}}, \text{ for } ((i-1) \bmod D \equiv k-1), \text{ and } 0 \leq j \leq (T-2), \\ b_i, & \text{if } \boldsymbol{x} = 0^* \underbrace{\boldsymbol{x}_i}_{n \text{ bits}} \underbrace{b_k\, b_{D+k} \ldots b_{(T-2)D+k}}_{(T-1) \text{ bits}}, \text{ for some } ((i-1) \bmod D \equiv k-1), \\ 0, & \text{otherwise.} \end{cases}$$
(25)

In simple words, the hypothesis $f_{\boldsymbol{b}} : \Sigma^* \to \Sigma$ is defined in such a way that on the point $\boldsymbol{x}_i$ such that $(i-1) \bmod D = (k-1)$ for $k \in [D]$ outputs $b_k, b_{D+k}, b_{2D+k}, \ldots, b_{(T-2)D+k}$ in the next $(T-1)$ steps of auto-regression before outputting b_i at the Tth step.

It is first important to note that each $f_{\boldsymbol{b}} : \Sigma^* \to \Sigma$ is well-defined. Because, for any input $\boldsymbol{x} \in \Sigma^*$, we can first drop the leading 0s since $\boldsymbol{x}_i$ always starts with 1. Therefore, we can uniquely decode if the input point's prefix is of the form $0^* \boldsymbol{x}_i$. Once that is determined, we can find the value of $k = ((i-1) \bmod D) + 1$ and check whether the remaining tokens to be followed (if there are any) are of the form $b_k, b_{D+k}, \ldots$, to decide the output on $\boldsymbol{x}$. If $\boldsymbol{x}$ does not match the description, then it simply outputs 0.
Showing end-to-end VCdim **is large.** It is straightforward to note that for the set of points $S = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M)$, at the end of T steps, by definition, we will realize all sign-patterns of these M points. In particular, for any $\boldsymbol{x}_i \in [M]$, we have $f_{\boldsymbol{b}}^{\text{e2e}(T)}(\boldsymbol{x}_i) = b_i$. We achieve

$$\left| \{(f_{\boldsymbol{b}}^{\text{e2e-}T}(\boldsymbol{x}_1), \ldots, f_{\boldsymbol{b}}^{\text{e2e-}T}(\boldsymbol{x}_M)) : f_{\boldsymbol{b}} \in \mathcal{F}\} \right| = \left| \{\boldsymbol{b} : \boldsymbol{b} \in \{0,1\}^M\} \right| = 2^M,$$

shattering all points in S, giving us $\mathrm{VCdim}(\mathcal{F}^{\text{e2e-}T}) \geq M$. Also, we know that $\mathrm{VCdim}(\mathcal{F}^{\text{e2e-}T}) \leq \log_2 |\mathcal{F}| \leq M$. Combining, we obtain

$$\mathrm{VCdim}(\mathcal{F}^{\text{e2e-}T}) = M = T \cdot D.$$

Showing that Base Class has small VCdim. We first start by showing that $\mathrm{VCdim}(\mathcal{F}) \geq D$. Fix the set $S' = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_D) \subset S$, of the first D points in S. It is immediate to see that $k = i$ for any $i \in [D]$, and therefore, applying the first case in (25)

$$\text{for any } i \in [D], \text{ we have } f_{\boldsymbol{b}}(\boldsymbol{x}_i) = b_k = b_i.$$

The main challenge is to show that our base class $\mathcal{F}$ has $\mathrm{VCdim}(\mathcal{F}) \leq D$. To this end, suppose for the purpose of contradiction, that $\mathrm{VCdim}(\mathcal{F}) \geq D + 1$. Then there exists a set $\tilde{S} \in (\Sigma^*)^{D+1}$ such that all 2^{D+1} possible labelings can be realized by some $f_{\boldsymbol{b}} \in \mathcal{F}$. Then it must be that any point $\tilde{\boldsymbol{x}} \in \tilde{S}$ is of the form

$$\boldsymbol{x} = 0^* \boldsymbol{x}_i \tilde{b}_1 \ldots \tilde{b}_\ell, \text{ for some } 1 \leq \ell \leq (T-1), \text{ and } \tilde{b}_1, \ldots, \tilde{b}_\ell \in \{0, 1\}.$$

Because otherwise the functions in $\mathcal{F}$ just outputs 0 on that point. Then by pigeon-hole argument, there exists $i_1, i_2 \in M$ such that

$$(i_1 - 1) \bmod D = (i_2 - 1) \bmod D = (k - 1), \text{ for some } k \in [D],$$

and there are two points $\tilde{\boldsymbol{x}}_1, \tilde{\boldsymbol{x}}_2 \in \tilde{S}$ such that

$$\tilde{\boldsymbol{x}}_1 = 0^* \boldsymbol{x}_{i_1} \tilde{b}_k \tilde{b}_{D+k} \ldots \tilde{b}_{(\ell_1-1)D+k}$$
$$\tilde{\boldsymbol{x}}_2 = 0^* \boldsymbol{x}_{i_2} \tilde{b}_k \tilde{b}_{D+k} \ldots \tilde{b}_{(\ell_1-1)D+k} \ldots \tilde{b}_{(\ell_2-1)D+k}, \text{ for some } \tilde{\boldsymbol{b}} \in \{0,1\}^M.$$

where w.l.o.g., we let $\ell_2 \geq \ell_1$. Moreover, we can continue to shatter $\{\tilde{\boldsymbol{x}}_1, \tilde{\boldsymbol{x}}_2\}$ using $\mathcal{F}$. We break into the cases to show that there is a contradiction.

1. $\ell_1 < \ell_2 \leq (T-1)$: The main intuition is that, any predictor, that outputs 1 on the point $\tilde{\boldsymbol{x}}_2$, is also required to output $\tilde{b}_{\ell_1 D+k}$ on $\tilde{\boldsymbol{x}}_1$, due to the first case (25). Formally, consider the target labels $(\tilde{y}_1, \tilde{y}_2) = (\neg \tilde{b}_{\ell_1 \cdot D+k}, 1)$. Then we claim that, for any predictor $f_{\boldsymbol{b}} \in \mathcal{F}$

$$\tilde{y}_1 \neq f_{\boldsymbol{b}}(\tilde{\boldsymbol{x}}_1) \quad \text{or} \quad \tilde{y}_2 \neq f_{\boldsymbol{b}}(\tilde{\boldsymbol{x}}_2).$$

This is because in order for any $f_{\boldsymbol{b}}(\tilde{\boldsymbol{x}}_2) = 1$, we must have that $\boldsymbol{b}$ agrees with $(\tilde{b}_k, \tilde{b}_{D+k}, \ldots, \tilde{b}_{(\ell_2-1)D+k})$, on the respective indices; otherwise $f_{\boldsymbol{b}}(\tilde{\boldsymbol{x}}_2) = 0$, by the third case of (25). So

$$(b_k, b_{D+k}, \ldots, b_{(\ell_1-1)D+k}, b_{\ell_1 D+k}, \ldots, b_{(\ell_2-1)D+k}) = (\tilde{b}_k, \tilde{b}_{D+k}, \ldots, \tilde{b}_{(\ell_1-1)D+k}, \tilde{b}_{\ell_1 D+k}, \ldots, \tilde{b}_{(\ell_2-1)D+k}).$$

But this also subsumes that $(b_k, b_{D+k}, \ldots, b_{(\ell_1-1)D+k}) = (\tilde{b}_k, \tilde{b}_{D+k}, \ldots, \tilde{b}_{(\ell_1-1)D+k})$. According to first case of (25), we must have $f_{\boldsymbol{b}}(\tilde{\boldsymbol{x}}_1) = b_{\ell_1 \cdot D+k} = \tilde{b}_{\ell_1 \cdot D+k}$, giving us that we cannot realize the behavior $(\tilde{y}_1, \tilde{y}_2) = (\neg \tilde{b}_{\ell_1 \cdot D+k}, 1)$, contradicting that we can shatter $\{\tilde{\boldsymbol{x}}_1, \tilde{\boldsymbol{x}}_2\}$.

36

2. $\ell_1 = \ell_2 < (T-1)$: In this case, the main intuition is that, any predictor $f_{\boldsymbol{b}} \in \mathcal{F}$, either agrees with the bit pattern of $\tilde{\boldsymbol{b}}$ on the respective bits and outputs the same bit $b_{\ell_1 D+k} = b_{\ell_2 D+k}$ or otherwise, it outputs 0s on both points. So, effectively, we cannot have different outputs on the two points. Formalizing this, consider the target labels $(\tilde{y}_1, \tilde{y}_2) = (0, 1)$. Then for any predictor $f_{\boldsymbol{b}} \in \mathcal{F}$

$$\tilde{y}_1 \neq f_{\boldsymbol{b}}(\tilde{\boldsymbol{x}}_1) \quad \text{or} \quad \tilde{y}_2 \neq f_{\boldsymbol{b}}(\tilde{\boldsymbol{x}}_2).$$

This is because in order for any $f_{\boldsymbol{b}}(\tilde{\boldsymbol{x}}_2) = 1$, we must have that $\boldsymbol{b}$ is such that

$$(b_k, b_{D+k}, \ldots, b_{(\ell_2-1)D+k}) = (\tilde{b}_k, \tilde{b}_{D+k}, \ldots, \tilde{b}_{(\ell_2-1)D+k}) \text{ and } 1 = b_{\ell_2 D+k} = b_{\ell_1 D+k}.$$

But this immediately also implies that, according to first case of (25), we must have $f_{\boldsymbol{b}}(\tilde{\boldsymbol{x}}_1) = b_{\ell_1 \cdot D+k} = 1$, contradicting that we can shatter $\{\tilde{\boldsymbol{x}}_1, \tilde{\boldsymbol{x}}_2\}$.

3. $\ell_2 = \ell_1 = (T-1)$: In this case, in order to shatter $\{\tilde{\boldsymbol{x}}_1, \tilde{\boldsymbol{x}}_2\}$, we must have $i_1 \neq i_2$ because otherwise any $f_{\boldsymbol{b}} \in \mathcal{F}$ will have the identical outputs on both the points. W.l.o.g. let $i_1 < i_2$. Then we have $i_1 = (j_1 - 1) \cdot D + k$ for some $1 \leq j_1 \leq T-1$. Then we shall show that for the target labels $(\tilde{y}_1, \tilde{y}_2) = (\neg \tilde{b}_{(j_1-1)\cdot D+k}, 1)$, for any predictor $f_{\boldsymbol{b}} \in \mathcal{F}$

$$\tilde{y}_1 \neq f_{\boldsymbol{b}}(\tilde{\boldsymbol{x}}_1) \quad \text{or} \quad \tilde{y}_2 \neq f_{\boldsymbol{b}}(\tilde{\boldsymbol{x}}_2).$$

This holds true because in order for any $f_{\boldsymbol{b}}(\tilde{\boldsymbol{x}}_1) = \neg \tilde{b}_{(j_1-1)\cdot D+k}$, we must have $\boldsymbol{b}$ such that

$$(b_k, b_{D+k}, \ldots, b_{(T-2)D+k}) \neq (\tilde{b}_k, \tilde{b}_{D+k}, \ldots, \tilde{b}_{(T-2)D+k});$$

otherwise, by the second case of (25), the output $f_{\boldsymbol{b}}(\tilde{\boldsymbol{x}}_1) = b_{i_1} = b_{(j_1-1)\cdot D+k} = \tilde{b}_{(j_1-1)\cdot D+k}$. But if this is the case, by the third case of (25), we must also have $f_{\boldsymbol{b}}(\tilde{\boldsymbol{x}}_2) = 0$. Therefore, we cannot realize $(\tilde{y}_1, \tilde{y}_2) = (\neg \tilde{b}_{(j_1-1)\cdot D+k}, 1)$, achieving a contradiction.

$\blacksquare$

D.2. Sample Complexity of Learning Time-Dependent End-to-End Class.

We now discuss that even for the end-to-end classes that have time-dependent compositions have roughly the same complexity, i.e. up to $\log T$ factor, as the worst-case lower bound from Theorem 39. We show that $\text{VCdim}(\mathcal{F}^{\text{TD-e2e-}T}) = O(T \cdot \text{VCdim}(\mathcal{F}) \log T)$, essentially the same as $\text{VCdim}(\mathcal{F}^{\text{e2e-}T}) = O(T \cdot \text{VCdim}(\mathcal{F}))$ up to a $O(\log T)$ factor.

Theorem 41 (Time Dependent Class) *Consider any finite Σ, a base function class $\mathcal{F} \subseteq \Sigma^{\Sigma^*}$ and generation length $T \in \mathbb{N}_+$.*

- *For binary $\Sigma = \{0, 1\}$:*

$$\text{VCdim}(\mathcal{F}^{\text{TD-e2e-}T}) \leq 3\, T \cdot \text{VCdim}(\mathcal{F}) \log\left(\frac{2\,T}{\ln 2}\right).$$

- *For non-binary finite Σ:*

$$\text{Ndim}(\mathcal{F}^{\text{TD-e2e-}T}) \leq 3\, T \cdot \text{Ndim}(\mathcal{F}) \log_2\left(\frac{2\,\text{Ndim}(\mathcal{F})|\Sigma|^2}{e \ln 2}\right).$$

Proof [Proof of Theorem 33] Let $\Gamma_T := \Gamma_{\mathcal{F}^{\text{TD-e2e-}T}}$ be the growth function of the class $\mathcal{F}^{\text{TD-e2e-}T}$ for $T \in \mathbb{N}_+$. Then by definition $\Gamma_1 = \Gamma_{\mathcal{F}}$. We will try to bound Γ_T recursively. Let us first define

$$\mathcal{F}^T := \underbrace{\mathcal{F} \times \cdots \times \mathcal{F}}_{T \text{ times}}.$$

Consider any fixed set $S \in (\Sigma^*)^m$ of size $|S| = m$. For any $T \in \mathbb{N}_+$, divide $\mathcal{F}^T$ into equivalence classes according to the equivalence relation $\equiv_T$ such that

$$f := (f_1, \ldots, f_T) \equiv_T g := (g_1, \ldots, g_T), \text{ iff } f^{\text{CoT-}T}(\boldsymbol{x}) = g^{\text{CoT-}T}(\boldsymbol{x}), \text{ for all } \boldsymbol{x} \in S.$$

In words, a pair of time-dependent functions $f, g \in \mathcal{F}^T$ are equivalent if for the next T steps, for any point in S, they both have identical chain-of-thoughts on that point. Let $\kappa_T(S)$ be the number of equivalence classes. By definition of growth function,

$$\Gamma_T(m) \le \max_{S \in (\Sigma^*)^m} \kappa_T(S). \tag{26}$$

Therefore, it suffices to bound the latter. The main observation is that each equivalence class with relation $\equiv_T$ may split into a partition of at most $\Gamma_{\mathcal{F}}(m)$ many equivalence classes with respect to the relation $\equiv_{T+1}$, because we have at most $\Gamma_{\mathcal{F}}(m)$ many behaviors on any fixed set of m points our hypothesis class. We obtain

$$\kappa_{T+1}(S) \le \Gamma_{\mathcal{F}}(m) \cdot \kappa_T(S) \quad \text{and} \quad \kappa_1(S) \le \Gamma_{\mathcal{F}}(m). \tag{27}$$

Solving this recursion yields

$$\kappa_T(S) \le (\Gamma_{\mathcal{F}}(m))^T \tag{28}$$

This argument holds for any $S \in (\Sigma^*)^m$. Combining the inequalities (26) and (28), we have

$$\Gamma_T(m) \le (\Gamma_{\mathcal{F}}(m))^T .$$

We now break into cases:

- $\Sigma = \{0, 1\}$ (binary alphabets): In this case, using Sauer's lemma (Lemma 21), for any $m \ge \text{VCdim}(\mathcal{F})$,

$$\Gamma_T(m) \le \Gamma_{\mathcal{F}}(m)^T \le \left(\frac{em}{\text{VCdim}(\mathcal{F})}\right)^{T \cdot \text{VCdim}(\mathcal{F})} .$$

  Therefore, $\text{VCdim}(\mathcal{F}^{\text{TD-e2e-}T})$ can be bounded by any $m \in \mathbb{N}_+$ satisfying $2^m > \Gamma_T(m)$ and $m \ge \text{VCdim}(\mathcal{F})$. Using the derived upper bound on $\Gamma_T(m)$, it suffices to choose m such that

$$m > T \cdot \text{VCdim}(\mathcal{F}) \log_2 \left(\frac{em}{\text{VCdim}(\mathcal{F})}\right) \quad \text{and} \quad m \ge \text{VCdim}(\mathcal{F}).$$

  Using Lemma 28 with $N = T \cdot \text{VCdim}(\mathcal{F})$ and $M = 1/\text{VCdim}(\mathcal{F})$, we directly argue the existence of $m \in \mathbb{N}_+$ such that

$$\text{VCdim}(\mathcal{F}^{\text{TD-e2e-}T}) \le m \le 3\,T \cdot \text{VCdim}(\mathcal{F}) \log_2 \left(\frac{2T}{\ln 2}\right) .$$

38

- For a finite Σ : Using Natarajan's lemma (Lemma 23) for any $m \in \mathbb{N}_+$,

$$\Gamma_{\mathcal{F}^{\text{TD-e2e-}T}}(m) \leq \Gamma_{\mathcal{F}}(m)^T \leq \left(|\Sigma|^2 \cdot m\right)^{T \cdot \text{Ndim}(\mathcal{F})} .$$

Again, $\text{Ndim}(\mathcal{F}^{\text{TD-e2e-}T})$ can be bounded by $m \in \mathbb{N}_+$ for which $m > T \cdot \text{Ndim}(\mathcal{F}) \log_2\left(|\Sigma|^2 \cdot m\right)$. Applying Lemma 28 with $N = T \cdot \text{Ndim}(\mathcal{F})$ and $M = |\Sigma|^2/e$, we obtain that there exists such $m \in \mathbb{N}_+$ such that

$$\text{Ndim}(\mathcal{F}^{\text{TD-e2e-}T}) \leq m \leq 3\,T \cdot \text{Ndim}(\mathcal{F}) \log_2\left(\frac{2\,\text{Ndim}(\mathcal{F})|\Sigma|^2}{e \ln 2}\right) .$$

■

D.3. Lower Bound for Theorem 6 in terms of Littlestone Dimension

We now show that $\Omega(\text{Ldim}(\mathcal{F}))$ samples are necessary in the worst case. Our lower bound is non-trivial in the sense that we show this for a family of classes that always have $\text{VCdim}(\mathcal{F}) = 1$. First, note that by the negative direction of the Fundamental Theorem of Statistical Learning (Proposition 30), it suffices to show that the VC dimension of the end-to-end class, $\text{VCdim}(\mathcal{F}^{\text{e2e-}T})$, is lower bounded by $\text{Ldim}(\mathcal{F})$, to establish the lower bound in the sample complexity of end-to-end learning of $\mathcal{F}^{\text{e2e-}T}$. The following theorem establishes this for a non-trivial family of classes.

Theorem 42 (Lower Bound for Littlestone Dimension) *For $\Sigma = \{0, 1\}$, for any $D \in \mathbb{N}_+$, there exists $\mathcal{F} \subseteq \Sigma^{\Sigma^*}$ such that*

$$\text{Ldim}(\mathcal{F}) = D, \text{VCdim}(\mathcal{F}) = 1 \text{ over domain } \Sigma^* ,$$

and

$$\text{VCdim}(\mathcal{F}^{\text{e2e-}T}) = D \text{ for any } T > D \text{ even over domain } \Sigma^n \text{ for } n = \lceil \log D \rceil + 1 .$$

Proof [Theorem 42] **Description of the Class.** For any target $D \in \mathbb{N}_+$, consider a set of D points of length $n = \lceil \log_2 D \rceil + 1$ defined as:

$$S = \{\boldsymbol{x}_i : i \in [D]\}, \quad \text{where } \boldsymbol{x}_i = 1 \underbrace{(\text{bit representation of } i)}_{\lceil \log_2 D \rceil \text{ bits}} .$$

We define the function class $\mathcal{F}$ consisting of functions $f_{\boldsymbol{b}}$ for each $\boldsymbol{b} \in \{0, 1\}^D$, defined as:

$$f_{\boldsymbol{b}}(\boldsymbol{x}) = \begin{cases} b_{j+1}, & \text{if } \boldsymbol{x} = 0^* \underbrace{\boldsymbol{x}_i}_{n \text{ bits}} \underbrace{b_1 b_2 ... b_j}_{j \text{ bits}} \text{ for some } i \in [D], 0 \leq j < D \\ b_i, & \text{if } \boldsymbol{x} = 0^* \underbrace{\boldsymbol{x}_i}_{n \text{ bits}} b_1 b_2 ... b_D b_i^* \text{ for some } i \in [D] \\ 0, & \text{otherwise} \end{cases}$$

In simple words, the function $f_{\boldsymbol{b}}$ is defined in such a way that on the point $\boldsymbol{x}_i$, it outputs $b_1, \ldots, b_D$ in the next D steps of auto-regression and then outputs b_i for the rest of the steps.

Showing Base Class has Littlestone Dimension D. We first show that $\mathrm{Ldim}(\mathcal{F}) \geq D$. Let $\boldsymbol{z}$ be a complete binary tree of depth D. For any binary string $\epsilon \in \{0,1\}^j$ of length $j < D$, let $\boldsymbol{z}(\epsilon)$ denote the example at the node reached by following path ϵ from the root. We label the tree as follows:

$$\boldsymbol{z}(\epsilon) = \boldsymbol{x}_1 \epsilon[1] \epsilon[2] \ldots \epsilon[j]$$

where $\epsilon[i]$ denotes the ith bit of ϵ. Note that we use $\boldsymbol{x}_1$ for simplicity; the same argument works with any $\boldsymbol{x}_i$.

We now show this tree is shattered by $\mathcal{F}$. For any path $\epsilon \in \{0,1\}^D$, consider $f_{\boldsymbol{b}}$ with $\boldsymbol{b} = \epsilon$. By construction of $f_{\boldsymbol{b}}$:

- At root: $f_{\boldsymbol{b}}(\boldsymbol{x}_1) = b_1 = \epsilon[1]$

- For any prefix ϵ of length $0 \leq j < D$:

$$f_{\boldsymbol{b}}(\boldsymbol{z}(\epsilon)) = f_{\boldsymbol{b}}(\boldsymbol{x}_1 \epsilon[1]...\epsilon[j]) = b_{j+1} = \epsilon[j+1]$$

  where the second equality follows from the first case of the function definition

Thus $f_{\boldsymbol{b}}$ realizes the labeling ϵ on this tree. Since this holds for any $\epsilon \in \{0,1\}^D$, we have $\mathrm{Ldim}(\mathcal{F}) \geq D$.

For the upper bound, note that $\mathrm{Ldim}(\mathcal{F}) \leq \log_2 |\mathcal{F}|$ for any function class. Since our construction has $|\mathcal{F}| = 2^D$ functions (one for each $\boldsymbol{b} \in \{0,1\}^D$), we immediately get $\mathrm{Ldim}(\mathcal{F}) \leq D$. Combined with the lower bound, this establishes $\mathrm{Ldim}(\mathcal{F}) = D$.

Showing Base Class has VC Dimension 1. First, we show $\mathrm{VCdim}(\mathcal{F}) \geq 1$ by exhibiting a point that can be shattered. Consider the point $\boldsymbol{x}_1$. We can achieve both labelings:

- For 0: Choose $f_{\boldsymbol{b}}$ with $b_1 = 0$. Then $f_{\boldsymbol{b}}(\boldsymbol{x}_1) = b_1 = 0$

- For 1: Choose $f_{\boldsymbol{b}}$ with $b_1 = 1$. Then $f_{\boldsymbol{b}}(\boldsymbol{x}_1) = b_1 = 1$

To show $\mathrm{VCdim}(\mathcal{F}) \leq 1$, consider any two distinct points $\boldsymbol{y}_1 = \boldsymbol{x}_i p$ and $\boldsymbol{y}_2 = \boldsymbol{x}_j q$ for $i, j \in [D]$ and $p, q \in \{0,1\}^*$. Assume without loss of generality that $|p| \leq |q|$. Suppose for contradiction that we can realize both labelings (1,1) and (1,0)/(0,1). Now consider the following cases:

- $p = q$. If $i = j$ then both points are the same, so they can not be shattered. So we can assume $i \neq j$.

  - If $|p| < D$ then for any $\boldsymbol{b}$ that has p as the prefix, both $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ evaluate to $\boldsymbol{b}[|p|+1]$, and for any $\boldsymbol{b}$ that does not have p as a prefix, they evaluate to 0. So they cannot achieve (0,1) or (1,0) labeling and thus cannot be shattered.

  - If $|p| \geq D$, in order to get output (1,1), we would need some $\boldsymbol{b}$ to match p on its first D bits, have $\boldsymbol{b}[i] = \boldsymbol{b}[j] = 1$ and the rest of the bits of both p and q be 1. Now for any other $\boldsymbol{b}$, on this input, $\boldsymbol{b}$ would not match p on its first D bits, and so $f_{\boldsymbol{b}}(\boldsymbol{y}_1) = 0$ and $f_{\boldsymbol{b}}(\boldsymbol{y}_2) = 0$. So we cannot achieve (1,0) or (0,1) labeling and thus cannot be shattered.

- p is a strict prefix of q.

- If $r \leq D$, then to get labeling $\boldsymbol{y}_1$ as 1 forces $\boldsymbol{b}$ to match p for its first r bits, and have $\boldsymbol{b}[r] = 1$. Now if $q[r]$ is 0, then $f_{\boldsymbol{b}}(\boldsymbol{y}_2) = 0$, so we cannot achieve (1,1). So $q[r] = 1$ must be 1. Now if we want $\boldsymbol{y}_1$ to labeled to 0, then we would need $\boldsymbol{b}$ to not match p. But p is a prefix of q so this qould imply that $f_{\boldsymbol{b}}(\boldsymbol{y}_2) = 0$ implying (0,1) is not achievable.

- If $r > D$, then p and q share the first D bits, So for any $\boldsymbol{b}$, either both $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ are labeled 0, or $\boldsymbol{y}_1$ is labeled $p[i]$ and $\boldsymbol{y}_2$ is labeled $p[j]$. So they cannot be shattered.

• $p \neq q$ and they differ fon some position $r \leq |p|$:

- If $r \leq D$ and $p[r] \neq q[r]$ then to label $\boldsymbol{y}_1$ with 1, the function $f_{\boldsymbol{b}}$ must match p on the first r bits, forcing $\boldsymbol{b}[r] = p[r]$. Because $q[r] \neq p[r]$, we get $f_{\boldsymbol{b}}(\boldsymbol{y}_2) = 0$. This implies that we cannot assign both $\boldsymbol{y}_1, \boldsymbol{y}_2$ the label 1, and these points cannot be shattered.

- If $r > D$, then p and q share all the first D bits but differ after bit D. By definition of $f_{\boldsymbol{b}}$, to get output 1 on $\boldsymbol{y}_1$, we would need $p = \boldsymbol{b}1^*$ and $p[i] = 1$. But since $p[r] \neq q[r]$ beyond bit D, for such $\boldsymbol{b}$, $f_{\boldsymbol{b}}(\boldsymbol{y}_2) = 0$. Hence we cannot assign both $\boldsymbol{y}_1, \boldsymbol{y}_2$ the label 1, and these points cannot be shattered.

Showing End-to-End Class has VC Dimension D. We first show that $\mathrm{VCdim}(\mathcal{F}^{\text{e2e-}T}) \geq D$. Consider the set $S = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_D\}$. For any desired labeling $\epsilon \in \{0,1\}^D$, we construct $f_{\boldsymbol{b}}$ with $\boldsymbol{b} = \epsilon$. After $T > D$ steps, the end-to-end function on input $\boldsymbol{x}_i$ outputs b_i for all $i \in [D]$. Thus we can realize any labeling ϵ on S.

To show $\mathrm{VCdim}(\mathcal{F}^{\text{e2e-}T}) \leq D$, suppose for contradiction that we could shatter $D + 1$ points $\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{D+1}\}$. Consider the behavior of any $f_{\boldsymbol{b}} \in \mathcal{F}$ on these points: for each point $\boldsymbol{y}_j$, after $T > D$ steps: if $\boldsymbol{y}_j$ starts with some $\boldsymbol{x}_i$, follows the path $b_1, \ldots, b_D, b_i^*$ and the output is b_i, otherwise (if it deviates or doesn't start with any $\boldsymbol{x}_i$), the output is 0.

In order to achieve the labeling with all $D + 1$ points being 1, we would need all $D + 1$ points to start with some $\boldsymbol{x}_i$ and follow the path $b_1, \ldots, b_D, b_i^*$ for some $\boldsymbol{b}$. Since we only have D possible choices for $\boldsymbol{x}_i$, by pigeonhole, two of the points will need to share $\boldsymbol{x}_i$ for some i and have the form $\boldsymbol{x}_1 b_1 \ldots b_D b_i^*$. However, for these two points, all other functions in the class would output 0 since they would not match the prefix $b_1 \ldots b_D$, so we cannot achieve the labeling $(1, 0)$ or $(0, 1)$ on these two points. Therefore we cannot shatter $D + 1$ points, so $\mathrm{VCdim}(\mathcal{F}^{\text{e2e-}T}) \leq D$. ∎

D.4. End-to-End Sample Complexity map Collapse

We now show that learning the end-to-end class sometimes might be simpler than learning the base class (or even trivial). We show a family of base classes with a high VC dimension, but the end-to-end learning complexity collapses after one step of iterative composition.

Theorem 43 *For any natural number $D \in \mathbb{N}_+$, there exists a hypothesis class $\mathcal{F}$ such that $\mathrm{VCdim}(\mathcal{F}) = D$ over domain $\mathcal{X} = \{0,1\}^n$ for $n = \log\lceil D \rceil + 1$ but $\mathrm{VCdim}(\mathcal{F}^{\text{e2e-}T}) = 0$, even with $T = 2$. As a corollary, $\mathcal{F}^{\text{e2e-}T}$ is learnable without any samples. However, for every learning rule A, there exists a distribution $\mathcal{D}$ over $\{0,1\}^n$ realizable by $\mathcal{F}$ such that for any set $S \sim \mathcal{D}^m$ with $m < D/2$, we have*

$$\mathbb{P}\left(\mathcal{L}_{\mathcal{D}}(A(S)) \geq 1/4\right) \geq 1/7 \,.$$

41

Proof [Proof of Theorem 43] Consider the set of D points as follows:

$$S = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_D), \text{ where } \boldsymbol{x}_i = 1 \underbrace{(\text{bit representation of } i)}_{\lceil \log_2 D \rceil \text{ bits}} 1 .$$

Description of the class. For each bit pattern $\boldsymbol{b} \in \{0,1\}^D$, we have one hypothesis $f_{\boldsymbol{b}} \in \mathcal{F}$ which is defined as follows

$$f_{\boldsymbol{b}}(\boldsymbol{x}) = \begin{cases} b_i, & \text{if } \boldsymbol{x} = 0^* \underbrace{\boldsymbol{x}_i}, \\ 0, & \text{otherwise.} \end{cases} \tag{29}$$

Again, note that this class is well-defined: for any input $\boldsymbol{x} \in \Sigma^*$, we drop the leading 0s since $\boldsymbol{x}_i$ always starts with 1 and then uniquely decodes whether $\boldsymbol{x}$ is of the form $0^* \boldsymbol{x}_i$.

VCdim after at step 1 is large. It is immediate to see that $|\mathcal{F}| = 2^D$ and $\mathrm{VCdim}(\mathcal{F}) \leq \log |\mathcal{F}| \leq D$. Also, for the set of points S, by definition for any $\boldsymbol{b} \in \{0,1\}^D$, we have that

$$|\{(f_{\boldsymbol{b}}(\boldsymbol{x}_i), \ldots, f_{\boldsymbol{b}}(\boldsymbol{x}_D)) : f_{\boldsymbol{b}} \in \mathcal{F}\}| = |\{(b_1, \ldots, b_D) : \boldsymbol{b} \in \{0,1\}^D\}| = 2^D.$$

Therefore, $\mathrm{VCdim}(\mathcal{F}) \geq D$. Combining, we obtain $\mathrm{VCdim}(\mathcal{F}) = D$.

VCdim collapses at step 2. We now want to show that $\mathrm{VCdim}(\mathcal{F}^{\text{e2e-}T}) = 0$ for $T = 2$ (fix for the rest of the proof). Suppose for the purpose of contradiction, assume that there is a point $\boldsymbol{x} \in \Sigma^*$ such that $\{f_{\boldsymbol{b}}^{\text{e2e-}T} : \boldsymbol{b} \in \mathcal{F}\} = \{0,1\}$, i.e. we can get both behaviors at two steps.

1. If $\boldsymbol{x} = 0^* \boldsymbol{x}_i$ for some $i \in [D]$: In this case, it is easy to see that for any $\boldsymbol{b} \in \mathcal{F}$

$$f^{\text{e2e-}T}(\boldsymbol{x}) = f^{\text{e2e-}T}(0^* \boldsymbol{x}_i) = f_{\boldsymbol{b}}(0^* \boldsymbol{x}_i b_i) = 0,$$

   contradicting that we can get both possible outputs.

2. If $\boldsymbol{x} \neq 0^* \boldsymbol{x}_i$ for any $i \in [D]$: In this case, we first note that $[\boldsymbol{x}, 0]$ is also not of the form $0^* \boldsymbol{x}_i$, as $\boldsymbol{x}_i$ always has the right most symbol 1. Using this

$$f_{\boldsymbol{b}}^{\text{e2e-}T}(\boldsymbol{x}) = f_{\boldsymbol{b}}(\boldsymbol{x}0) = 0,$$

   again reaching a contradiction.

■

D.5. Real Valued Alphabets

In this section, we formalize the claim mentioned in Remark 7.

Definition 44 (Pseudo Dimension (Pollard, 1989)) *For any domain $\mathcal{X}$ and a real valued function class $\mathcal{H} \subseteq \mathbb{R}^{\mathcal{X}}$. Then pseudo-dimension of $\mathcal{H}$, denoted by $\mathrm{Pdim}(\mathcal{H})$ is the largest $D \in \mathbb{N}$ such that there exists a set of points $\mathbf{x}_1, \ldots, \mathbf{x}_D \in \mathcal{X}$ and real target thresholds $\theta_1, \ldots, \theta_D \in \mathbb{R}$ where*

$$|\{(sign(h(\mathbf{x}_1) - \theta_1), \ldots, sign(h(\mathbf{x}_D) - \theta_D)) : h \in \mathcal{H}\}| = 2^D.$$

Moreover, if there is no such D, then we say that $\mathrm{Pdim}(\mathcal{F}) = \infty$.

Under some assumptions on the loss function, pseudo-dimension guarantees learnability. Below is the construction of $\mathcal{F} \subseteq \mathbb{R}^{\mathbb{R}^*}$, which has bounded pseudo-dimension. However, even one more iterative composition leads to an infinite pseudo-dimension of the end-to-end class.

Theorem 45 *Let $\Sigma = \mathbb{R}$. There exists a class $\mathcal{F} \subseteq \mathbb{R}^{\mathbb{R}^*}$ such that $\mathrm{Pdim}(\mathcal{F}) = 1$, but $\mathrm{Pdim}(\mathcal{F}^{\mathsf{e2e}\text{-}T}) = \infty$, even for $T = 2$.*

Proof [Proof of Theorem 45] We will consider a hypothesis class $\mathcal{F}$ defined as follows. Consider the following set of points

$$S = \{\boldsymbol{x}_n : n \in \mathbb{N}_+\} \text{ where } \boldsymbol{x}_n = \underbrace{1 \ldots 1}_{n \text{ times}} 0 = 1^n 0.$$

Therefore, the cardinality of S is countably infinite. Now the hypotheses class is defined such that there exists $f_{\boldsymbol{b}} \in \mathcal{F}$ for each sequence $\boldsymbol{b} \in \{0, 1\}^\infty$, whose output is defined as follows.

$$f_{\boldsymbol{b}}(\boldsymbol{x}) = \begin{cases} n + 0.\boldsymbol{b} := n + 0.b_1 b_2 \ldots, & \text{if } \boldsymbol{x} = 0^* \boldsymbol{x}_n \\ b_n & \text{if } \boldsymbol{x} = [0^* \boldsymbol{x}_n (n + 0.\boldsymbol{b})] \\ 0, & \text{otherwise.} \end{cases} \tag{30}$$

Throughout the proof, $0.\boldsymbol{b} \in \mathbb{R}$ is a real number representation for any sequence $\boldsymbol{b} \in \{0, 1\}^\infty$.
End-to-End pseudo-dimension is infinite. Fix the set $S = \{\boldsymbol{x}_n : n \in \mathbb{N}_+\}$ as defined previously. We will fix the thresholds to $z_1 = z_2 = \cdots = 1/2$. We know that for any bit pattern $\boldsymbol{b} \in \{0, 1\}^{\mathbb{N}_+}$, we have

$$\mathrm{sign}(f_{\boldsymbol{b}}^{\mathsf{e2e}(2)}(\boldsymbol{x}_n) - z_n) = \mathrm{sign}(b_n - 1/2) = \begin{cases} +1, \text{if } b_n = 1; \\ -1, \text{if } b_n = 0. \end{cases}.$$

This means we can realize all sign patterns on the points in S, given by $\{-1, +1\}^{\mathbb{N}_+}$, using hypothesis from $\mathcal{F}^{\mathsf{e2e}\text{-}2}$ and the thresholds $z_1 = z_2 = \cdots = 1/2$. This establishes that $\mathrm{Pdim}(\mathcal{F}^{\mathsf{e2e}\text{-}2}) = \infty$.
Pseudo-dimension after one step is small. It is easy to see that $\mathrm{Pdim}(\mathcal{F}) \geq 1$. Simply choose $\boldsymbol{x}_1$ and $z_1 = 1.05$. Then for $\boldsymbol{b}^{(1)} = 0^\infty = (000 \ldots)$ and $\boldsymbol{b}^{(2)} = 10^\infty = (1000 \ldots)$,

$$\mathrm{sign}(f_{\boldsymbol{b}^{(1)}}(\boldsymbol{x}_1) - z_1) = \mathrm{sign}(1 + 0.0 - 1.05) = -1 \text{ but } \mathrm{sign}(f_{\boldsymbol{b}^{(2)}}(\boldsymbol{x}_1) - z_1) = \mathrm{sign}(1 + 0.1 - 1.05) = +1.$$

The main challenge is to show that $\mathrm{Pdim}(\mathcal{F}) \leq 1$. Suppose for the purpose of contradiction, $\mathrm{Pdim}(\mathcal{F}) \geq 2$ and there exists points $\{\tilde{\boldsymbol{x}}_1, \tilde{\boldsymbol{x}}_2\}$ and thresholds $z_1, z_2 \in \mathbb{R}$ such that

$$|\{(\mathrm{sign}(f_{\boldsymbol{b}}(\tilde{\boldsymbol{x}}_1) - z_1), \mathrm{sign}(f_{\boldsymbol{b}}(\tilde{\boldsymbol{x}}_2) - z_2)) : f_{\boldsymbol{b}} \in \mathcal{F}\}| = 4.$$

We first note that this enforces that the points $\{\tilde{\boldsymbol{x}}_1, \tilde{\boldsymbol{x}}_2\}$ must be such that they match the description of either of the first two cases in (30). Otherwise on the point that does not match the description, each predictor $f_{\boldsymbol{b}} \in \mathcal{F}$ outputs 0, and irrespective of what (z_1, z_2) is chosen, we can never realize all four sign patterns. We now consider the following two cases and show a contradiction in each.

1. At least one point from $\{\tilde{\boldsymbol{x}}_1, \tilde{\boldsymbol{x}}_2\}$ matches description of the second case: W.l.o.g. assume that $\tilde{\boldsymbol{x}}_1$ is that point, i.e. $\tilde{\boldsymbol{x}}_1 = [0^* \tilde{\boldsymbol{x}}_{n_1} (n_1 + 0.\tilde{\boldsymbol{b}})]$ for some $n_1 \in \mathbb{N}_+$ and $\tilde{\boldsymbol{b}} \in \{0, 1\}^\infty$. This implies that

$$f_{\boldsymbol{b}}(\tilde{\boldsymbol{x}}_1) = \begin{cases} \tilde{b}_{n_1}, & \text{if } \boldsymbol{b} = \tilde{\boldsymbol{b}}, \\ 0, & \text{if } \boldsymbol{b} \neq \tilde{\boldsymbol{b}}. \end{cases}$$

43

So there are only two possible real outputs on $\tilde{x}_1$, therefore, in order to shatter the real threshold $0 < z_1 \leq n_1 + 0.\tilde{\boldsymbol{b}}$. However, there is only one predictor $f_{\tilde{\boldsymbol{b}}} \in \mathcal{F}$ whose output $f_{\tilde{\boldsymbol{b}}}(\tilde{x}_1) \geq z_1$. But then this completely also determines the output on $\tilde{x}_2$, and irrespective of what z_2 is chosen, we cannot realize at least one of the sign patterns from $\{(+1, -1), (+1, +1)\}$, arriving at a contradiction.

2. Both $\{\tilde{x}_1, \tilde{x}_2\}$ matches the description of the first case: In this case, it is important to note that we must have

$$\tilde{x}_1 = 0^* x_{n_1} \quad \tilde{x}_2 = 0^* x_{n_2}, \text{for } n_1 \neq n_2 \in \mathbb{N}_+.$$

Otherwise both points are identical for the purpose of prediction for any $f_{\boldsymbol{b}} \in \mathcal{F}$, and we cannot realize all four sign patterns anyway. Finally, in order to shatter these two points, there exist thresholds $z_1, z_2 \in \mathbb{R}$ such that

$$\{(\text{sign}(f_{\boldsymbol{b}}(x_{n_1}) - z_1), \text{sign}(f_{\boldsymbol{b}}(x_{n_2}) - z_2)) : f_{\boldsymbol{b}} \in \mathcal{F}\} = \{(s_1, s_2) : s_1, s_2 \in \{-1, +1\}\}.$$

Using the output of $f_{\boldsymbol{b}}$ from (30), this is equivalent to

$$\{(\text{sign}(n_1 - z_1 + 0.\boldsymbol{b}), \text{sign}(n_2 - z_2 + 0.\boldsymbol{b})) : \boldsymbol{b} \in \{0, 1\}^\infty\} = \{(s_1, s_2) : s_1, s_2 \in \{-1, +1\}\}.$$

But we know that $(n_1 - z_1), (n_2 - z_2) \in \mathbb{R}$ and w.l.o.g. assume $(n_1 - z_1) \geq (n_2 - z_2)$. This immediately implies there is no real number $0.\boldsymbol{b} \in \mathbb{R}$ such that

$$\text{sign}(n_1 - z_1 + 0.\boldsymbol{b}) = -1 \text{ but sign}(n_2 - z_2 + 0.\boldsymbol{b}) = +1,$$

concluding the proof.

$\blacksquare$

Appendix E. Proofs from Section 4

We first show the sample complexity results for e2e and CoT learnability.

Proof [Proof of Lemma 16] As noted already, this follows by Sauer's lemma. We know that $\text{VCdim}(\mathcal{F}_{d,\text{lin}})$ is at most $(d+1)$. Therefore, the cardinality of $\mathcal{F}_{d,\text{lin}}$ on the hypercube of $\{0, 1\}^d$ of size 2^d is:

$$|\mathcal{F}_{d,\text{lin}}| \leq \Gamma_{\mathcal{F}_{d,\text{lin}}}(2^d) \leq (e2^d)^{\text{VCdim}(\mathcal{F}_{d,\text{lin}})} \leq 2^{O(d^2)}.$$

$\blacksquare$

Proof [Proof of Corollary 17] The bound simply follows from Theorems 3 to 5 after substituting $\text{VCdim}(\mathcal{F}_{d,\text{lin}}) \leq (d+1)$ and $\log |\mathcal{F}_{d,\text{lin}}| = O(d^2)$ by Lemma 16. $\blacksquare$

We now show the computational tractability of the CoT-learnability of $\mathcal{F}_{d,\text{lin}}^{\text{e2e-}T}$.

Proof [Proof of Corollary 10] This directly follows from the facts (i) $\text{CONS}_{\mathcal{F}_{d,\text{lin}}}(S)$ is implementable in time polynomial in d and the length of the input S (ii) noting that $\text{VCdim}(\mathcal{F}_{d,\text{lin}}) \leq (d+1)$. This is a corollary of Corollary 9.

More specifically, we are implementing CONS_{CoT} by forming the following LP feasibility problem. For a given S_{CoT}, consider the set of all prefixes and next token pairs mentioned in Eq. (C.4):

$$\boldsymbol{p}_{(i,t)} = [\boldsymbol{x}_i, \boldsymbol{u}_i[\,1\,], \ldots, \boldsymbol{u}_i[\,t\,]] \in \Sigma^* \times \Sigma^t, \quad y_{(i,t)} = \boldsymbol{u}_i[t+1] \in \Sigma, \quad \text{for } i \in [m], 0 \leq i \leq (T-1).$$

Then we have to solve the following linear program in a variable parameter $\boldsymbol{w} \in \mathbb{R}^d$ with the following constraints. For $1 \leq i \leq m, 0 \leq t \leq (T-1)$

$$\begin{aligned} \langle \boldsymbol{w}, \boldsymbol{p}_{(i,t)} \rangle &\geq 0, \quad \text{if } \mathbf{y}_{(i,t)} = 1; \\ \langle \boldsymbol{w}, \boldsymbol{p}_{(i,t)} \rangle &< 0, \quad \text{if } \mathbf{y}_{(i,t)} = 0. \end{aligned} \quad \text{(LP-Feasibility)}$$

Clearly there are $|S_{\text{CoT}}| \cdot T$ many constraints, and we are implementing CONS_{CoT}, and therefore choosing $|S_{\text{CoT}}| = \mathcal{P}oly(n, d, T, 1/\varepsilon, \log(1/\delta))$ suffices by Corollary 17. The runtime complexity of solving (LP-Feasibility) with d variables and $|S_{\text{CoT}}| \times T$ constraints is bounded by $\mathcal{P}oly(n, d, T, 1/\varepsilon, \log(1/\delta))$; the theorem follows. ∎

E.1. End-to-End Learning is Hard for Iterated Linear Thresholds

We now return to the computational intractability of e2e-learning of $\mathcal{F}_{d,\text{lin}}^{\text{e2e-}T}$; the main technical result of this section. We begin with the description of the class of threshold circuits that we will be using in our reduction. **Bounded Depth and Size Linear Threshold Circuit.** Consider the input $\boldsymbol{x} \in \{0,1\}^n$. A linear threshold circuit $C : \{0,1\}^n \to \{0,1\}$ is a computational model represented by a connected directed acyclic graph (DAG), $G(V, E)$ with

- n input nodes $V_{\text{in}} = \{v_{\text{in},1}, \ldots, v_{\text{in},n}\}$ (the only nodes in V with no incoming edges) that correspond to the input coordinates $(x_1, \ldots, x_n)$,

- One output node v_{out} (the only node with no outgoing edges),

- A weight function $w : E \to \mathbb{R}$.

Each node has a value $val : V \to \{0,1\}$ associated with it computed recursively as follows:

$$val(v_{\text{in},k}) = x_k, \text{ for } k \in [n], \text{ and for any } v \in V \setminus V_{\text{in}}, \quad val(v) = \text{thr}\left(\sum_{(u,v) \in E} w(u,v) val(u) \right) \tag{31}$$

The final output of the circuit is $C(\boldsymbol{x}) = val(v_{\text{out}})$. The *depth* of the circuit is the length of the longest path from an input node in V_{in} to the output node v_{out}. The *size* of the circuit is $|V \setminus V_{\text{in}}|$.

For $n, L, s \in \mathbb{N}_+$, consider the hypothesis class threshold circuits over n input variables, size s, and depth at most L. This class is referred in Assumption 1. We first prove the final hardness results in light of Lemma 12.

Proof [Proof of Theorem 11] Consider the class threshold circuits of depth L and size $p(n)$ from Hardness Assumption 1. By our expressivity result Lemma 12, every such circuit C can be expressed as $f_{\boldsymbol{w}}^{\text{e2e-}T}$ with some $\boldsymbol{w} \in \mathbb{R}^d$ with $d \leq 2(p(n)+2)^L(n+1)$ and $T \leq (p(n)+2)^L(n+1)$, up to some fixed input transformation that runs in time $O(n') = O\left(2(p(n)+2)^L(n+1)\right)$. All d, n', T are bounded by fixed polynomials in n. Therefore, if $\mathcal{F}_{d,\text{lin}}$ is learnable in time $\mathcal{P}oly(n, T, d)$ up to

even constants $\varepsilon, \delta > 0$, we can also learn the class from Assumption 1 in time $\mathcal{P}oly(n)$ through this reduction, contradicting Assumption 1. ∎

We now prove our main expressivity result of this section.

Proof [Proof of Lemma 12] We begin our proof with some notation. In the graph representation on any circuit C (defined in Section E.1), we can partition the internal nodes in $V \setminus V_{\text{in}}$ into different layers $V_1 \cup V_2 \cup \cdots \cup V_L$. The layer number of any internal node v is defined as the length of the longest path from some input node in V_{in} to v. We now number the nodes arbitrarily per layer: the nodes in V_l can be numbered $v_1^{(l)}, \ldots, v_s^{(l)}$, for $1 \leq l \leq L$. Without loss of generality, we may assume that $|V_l| = s$ for $1 \leq l \leq L$ as we can always add nodes to the layer with incoming and outgoing edges having weights zero. And, the final output node $v_{\text{out}} = v_s^{(L)} \in V_L$. Also, w.l.o.g., we consider any node $v_i^{(l)}$ in V_l is connected with all the nodes $V_{\text{in}} \cup V_1 \cup \cdots \cup V_{l-1}$, as we can add the edges with weight 0, without affecting the output of the circuit. As a consequence, we can denote all the weights associated with incoming edges as a vector $\boldsymbol{w}_{li} \in \mathbb{R}^{p_l}$ where $p_l = n + (l-1)s$ and $(l, i) \in [L] \times [s]$. The weight coming from j^{th} node from the ℓth layer in $\boldsymbol{w}_{li}$ is denoted by $\boldsymbol{w}_{li}(\ell, j) \in \mathbb{R}$. Throughout the proof, rather than explicitly defining the coordinate of vectors, we will defined it by listing out its coordinates from left to right.

Without loss of generality, we will assume that the incoming weight associated to the last node of the previous layer is always 0. Formally, we have $\boldsymbol{w}_{1i}(0, n) = 0$ and for $l \geq 2$, we have $\boldsymbol{w}_{li}(l-1, s) = 0$. This can be easily ensured by adding one dummy node per layer in the circuit, and thus, by replacing n with $n + 1$ and s with $(s + 1)$ in the final bounds on T, n', d, which we will do towards the end of the proof. For now, we will proceed assuming this as it simplifies the presentation of our construction.

We now specify our feature map $\phi : \{0, 1\}^n \to \{0, 1\}^{n'}$. For any $\boldsymbol{x} \in \{0, 1\}^n$, the feature $\phi(\boldsymbol{x}) \in \{0, 1\}^{n'}$ written as a string takes the following form

$$\phi(\boldsymbol{x}) = (1 \underbrace{00 \ldots \ldots 00}_{(T-1) \text{ bits}} \underbrace{\boldsymbol{x}}_{n \text{ bits}}) \tag{32}$$

where T is the number of end-to-end steps to be taken which will be specified later in terms (n, s, L). Roughly speaking, we want to construct a linear predictor whose chain-of-thoughts on $\phi(\boldsymbol{x})$ when applied iteratively, produces the output of each internal node sequentially (separated by some number of zeros), i.e.

$$\phi(\boldsymbol{x}), 0, \ldots, 0, y_{11}, 0 \ldots 0, y_{12}, 0, \ldots \ldots, 0, y_{L1}.$$

The idea is to combine all the weights from the circuit into one linear threshold. However, to ensure that the weights associated with the previous output do not contribute when outputting the other node, we need that the 0s are outputted between any two relevant bits in CoT, so that the weights of the previous output align with 0s exactly. This means that the weight vector of the next layer must be now expanded into a higher dimension to adjust for the padded 0s.

Formally, we will embed each $\boldsymbol{w}_{li} \in \mathbb{R}^{p_l}$ into a vector $\tilde{\boldsymbol{w}}_{li} \in \mathbb{R}^{\tilde{p}_l}$ by padding extra zeros between the coordinates of $\boldsymbol{w}_{li}$ (to be specified how). The new dimension $\tilde{p}_l \geq p_l$ defined recursively:

$$\tilde{p}_1 = p_1 = n, \quad \text{and for } l > 1, \text{ define } \tilde{p}_l = (s+1)\tilde{p}_{l-1}. \tag{33}$$

We will create a vector $\boldsymbol{v}_0 = \boldsymbol{0}^{n-1}$ and for $l \geq 1$, we will form a vector $\boldsymbol{v}_l$, which is a concatenation of the vectors $\tilde{\boldsymbol{w}}_{l1}, \ldots, \tilde{\boldsymbol{w}}_{ls}$ in the reverse order:

$$\boldsymbol{v}_l = [\tilde{\boldsymbol{w}}_{ls}, \ldots, \tilde{\boldsymbol{w}}_{l1}], \quad \text{and therefore } |\boldsymbol{v}_l| = s \, \tilde{p}_l. \tag{34}$$

$$
\begin{aligned}
\boldsymbol{v}_0 &= \quad 0^{n-1} & |\boldsymbol{v}_0| &= n-1 \\
\boldsymbol{v}_1 &= [\boldsymbol{w}_{1s}, \boldsymbol{w}_{1(s-1)}, \ldots, \boldsymbol{w}_{12}, \boldsymbol{w}_{11}] & |\boldsymbol{v}_1| &= s\,\tilde{p}_1 = s\,n \\
\boldsymbol{v}_2 &= \left[\widetilde{\boldsymbol{w}}_{2s}, \widetilde{\boldsymbol{w}}_{2(s-1)}, \ldots, \widetilde{\boldsymbol{w}}_{22}, \widetilde{\boldsymbol{w}}_{21}\right] & |\boldsymbol{v}_2| &= s\,\tilde{p}_2 = s\,(s+1)\,n \\
\boldsymbol{v}_3 &= \left[\widetilde{\boldsymbol{w}}_{3s}, \widetilde{\boldsymbol{w}}_{3(s-1)}, \ldots, \widetilde{\boldsymbol{w}}_{32}, \widetilde{\boldsymbol{w}}_{31}\right] & |\boldsymbol{v}_3| &= s\,\tilde{p}_3 = s\,(s+1)^2\,n \\
&\qquad\qquad\vdots & &\quad\vdots \\
\boldsymbol{v}_l &= \left[\widetilde{\boldsymbol{w}}_{ls}, \widetilde{\boldsymbol{w}}_{l(s-1)}, \ldots, \widetilde{\boldsymbol{w}}_{l2}, \widetilde{\boldsymbol{w}}_{l1}\right] & |\boldsymbol{v}_l| &= s\,\tilde{p}_l = s\,(s+1)^{l-1}\,n \\
\boldsymbol{v}_{l+1} &= \left[\widetilde{\boldsymbol{w}}_{(l+1)s}, \ldots\ldots\ldots, \widetilde{\boldsymbol{w}}_{(l+1)1}\right] & |\boldsymbol{v}_{l+1}| &= s\,\tilde{p}_{l+1} = s\,(s+1)^l\,n \\
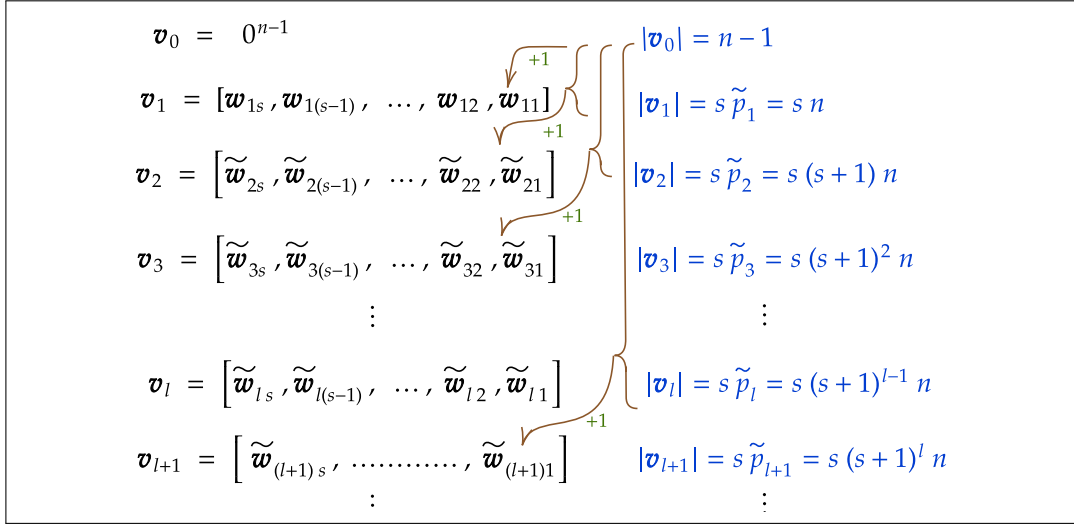&\qquad\qquad\vdots & &\quad\vdots
\end{aligned}
$$

Figure 2: The figure provides an illustrative summary of the construction so far and the sizes of the predictors, formalized in Claim 1.

Here is a simple claim that specifies the size of each of these linear predictors $\boldsymbol{v}_l$.

Claim 1 *For any $l \geq 1$, we have $\tilde{p}_l = (s+1)^{l-1}n$ and $\sum_{\ell=0}^{l-1} |\boldsymbol{v}_\ell| = (s+1)^{l-1} \cdot n - 1 = \tilde{p}_l - 1$.*

So roughly speaking, the size of each predictor $\tilde{\boldsymbol{w}}_{li} \in \mathbb{R}^{\tilde{p}_l}$ in layer l that we embedded our $\boldsymbol{w}_{li} \in \mathbb{R}^{p_l}$ into, is one more than the length of the entire concatenation of vectors $\boldsymbol{v}_{l-1}, \ldots, \boldsymbol{v}_0$. And the vector $\boldsymbol{v}_l$ is a concatenation of s such vectors, giving us $\tilde{p}_l = O(s^l)$. See Figure 2.

It is now time to specify T and the final linear predictor $\boldsymbol{w}$. The number of end-to-end steps T is given by

$$
T := \sum_{\ell=1}^{L} |\boldsymbol{v}_\ell| = \tilde{p}_{L+1} - 1 - |\boldsymbol{v}_0| = (s+1)^L \cdot n - n, \tag{35}
$$

where we used Claim 1 in the second equality. Eq. (32) gives us

$$
n' = |\phi(\boldsymbol{x})| = T + n = n + \sum_{\ell=1}^{L} |\boldsymbol{v}_\ell| = (s+1)^L \cdot n. \tag{36}
$$

Having constructed $\boldsymbol{v}_0, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_L$, the final predictor is then the concatenation of all these predictors along with another $\boldsymbol{v} \in \mathbb{R}^T$ (to be specified later) as follows

$$
\boldsymbol{w} = [\boldsymbol{v}, \boldsymbol{v}_L, \boldsymbol{v}_{L-1}, \ldots, \boldsymbol{v}_2, \boldsymbol{v}_1, \boldsymbol{v}_0]. \tag{37}
$$

Hence, at the start the linear predictor $\boldsymbol{w}$ and $\phi(\boldsymbol{x})$ aligns as follows.

$$
\begin{aligned}
\boldsymbol{w} &= [\underbrace{\text{———} \boldsymbol{v} \text{———}}, \underbrace{\boldsymbol{v}_L, \boldsymbol{v}_{L-1}, \ldots, \boldsymbol{v}_2, \boldsymbol{v}_1}_{T \text{ coordinates}}, 0^{n-1}] \\
\phi(\boldsymbol{x}) &= \qquad [1, \underbrace{0,0,0,\ldots\ldots,0,0,0}_{(T-1)\text{ times}}, \underbrace{-\boldsymbol{x}-}_{n \text{ bits}}]
\end{aligned} \tag{38}
$$

47

As $\boldsymbol{v} \in \mathbb{R}^T$, we have

$$d := |\boldsymbol{w}| = T + \sum_{\ell=0}^{L} |\boldsymbol{v}_\ell| \leq 2 \cdot (s+1)^L \cdot n \,, \tag{39}$$

where we used Claim 1 and Eq. (35). Overall, our constructions so far satisfies the bounds on T, n', d according to Lemma 12. Therefore, it remains to show the final construction of $\tilde{\boldsymbol{w}}_{li} \in \mathbb{R}^{\tilde{p}_l}$ and $\boldsymbol{v} \in \mathbb{R}^T$, and that this construction achieves the desired end-to-end computation of the circuit. To this end, we first specify the set of time-steps at which we output a gate in the circuit. Let t_{li} for $l \in [L]$ and $i \in [s]$ be the time when we are going to be writing y_{li} (i.e. output of the ith gate from the layer l). We define these time-steps recursively as

$$t_{1i} = n \cdot i, \text{ for } i \in [s] \quad \text{and} \quad t_{li} = t_{(l-1)s} + i \cdot \tilde{p}_l \quad \text{for } l > 1, i \in [s] \,. \tag{40}$$

Let $\mathcal{I} := \{t_{li} : l \in [L], i \in [s]\}$ be the collection of these indices. It is straight-forward to verify that $t_{Ls} = \sum_{\ell=1}^{L} s \cdot \tilde{p}_\ell = \sum_{\ell=1}^{L} |\boldsymbol{v}_\ell| = T$ by Eq. (35). Therefore, the final output of the circuit y_{Ls} will be computed at the Tth step (to be shown).

We would want that the iterated predictor on the time steps $t \in [T] \setminus \mathcal{I}$ when applied iteratively outputs zero. By the construction of $\mathcal{I}$, this corresponds to outputting exactly $\tilde{p}_l - 1$ zeros before outputting y_{li} for any $l \in [L], i \in [s]$. So the desired chain-of-thought is of the following form

$$\phi(\boldsymbol{x}), \underbrace{0 \ldots 0}_{(n-1) \text{ bits}}, y_{11}, \ldots \ldots, y_{l(i-1)}, \underbrace{0 \ldots 0}_{\tilde{p}_l - 1 \text{ bits}}, y_{li}, \ldots \ldots \ldots, y_{L(s-1)}, \underbrace{0 \ldots 0}_{\tilde{p}_L - 1 \text{ bits}} y_{Ls} \,. \tag{41}$$

This is exactly where the 1 at the beginning of the feature map ϕ in Eq. (32) plays the role of "positional encoding" together with the vector $\boldsymbol{v} \in \mathbb{R}^T$. The main idea is to create a vector $\boldsymbol{v} \in \{-B, 0\}^T$ where $-B$ is a "very large" negative value to be specified such that for every $t \in [T] \setminus \mathcal{I}$, the 1 in $\phi(\boldsymbol{x})$ aligns with $-B$ in $\boldsymbol{v}$, which on thresholding gives us the output 0. Formally,

$$\boldsymbol{v}[-t] := \begin{cases} -B & , \text{if } t \in [T] \setminus \mathcal{I} \,; \\ 0 & , \text{if } t \in \mathcal{I} \,. \end{cases} \tag{42}$$

Here $B = 1 + \sum_{l=1}^{L} \sum_{i=1}^{s} \|\boldsymbol{w}_{li}\|_1$ is a real number greater than the total ℓ_1 norm of the weights in the circuit. We have the following claim due to this construction.

Claim 2 *Consider any $\boldsymbol{w}$ of the form Eq. (37), where $\tilde{\boldsymbol{w}}_{li}$ are just constructed by padding zeros in $\boldsymbol{w}_{li}$ for $l \in [L], i \in [s]$, i.e., without increasing the total ℓ_1 norm of the weights. Then*

$$\text{for every } t \in [T] \setminus \mathcal{I} \,, \text{ we have } f_{\boldsymbol{w}}^{\text{e2e-}t}(\phi(\boldsymbol{x})) = 0 \,.$$

Finally, it is time to specify the constructions of $\tilde{\boldsymbol{w}}_{li} \in \mathbb{R}^{\tilde{p}_l}$. Note that the goal of $\tilde{\boldsymbol{w}}_{li}$ is to compute y_{li} in our CoT. For this, it needs to compute the threshold of the inner product $\langle \boldsymbol{w}_{li}, [\boldsymbol{x}, y_{11}, y_{12}, \ldots, y_{(l-1)(s-1)} y_{(l-1)s}] \rangle$, where $\boldsymbol{w}_{li} \in \mathbb{R}^{p_l}$ is the weight vector from the circuit

$$\boldsymbol{w}_{li} = [\boldsymbol{w}_{li}(0, 1), \boldsymbol{w}_{li}(0, 2), \ldots, \boldsymbol{w}_{li}(l-1, s-1), \boldsymbol{w}_{li}(l-1, s)],$$

where we denote the scalar weight coming from the jth node in the layer $\ell \leq l - 1$ by $\boldsymbol{w}_{li}(\ell, j)$. However, note that by Claim 2 and Eq. (41), we have to now pad zeros so that the $\boldsymbol{w}_{li}$ aligns with

48

the desired output gates. Formally, construct $\tilde{\boldsymbol{w}}_{li}$ by padding $\tilde{p}_\ell - 1$ zeros before $\boldsymbol{w}_{li}(\ell, j)$ for any $i \in [s]$.

$$\tilde{\boldsymbol{w}}_{li} = [\boldsymbol{w}_{li}[(0,1) \to (0,n)], \underbrace{0 \ldots 0}_{(n-1) \text{ bits}}, \boldsymbol{w}_{li}(1,1), \ldots, \boldsymbol{w}_{li}(\ell, i-1), \underbrace{0 \ldots 0}_{\tilde{p}_\ell - 1 \text{ bits}}, \boldsymbol{w}_{li}(\ell, i), \ldots \ldots, \boldsymbol{w}_{li}(l-1, s)] \tag{43}$$

Clearly, we are only padding zeros ensuring that the total ℓ_1 norm $\sum_{l=0}^L \|\boldsymbol{v}_l\|_1 = \sum_{l=1}^L \sum_{i=1}^s \|\boldsymbol{w}_{li}\|_1$ is the total ℓ_1 norm of the circuit as needed. Formally, the following claim finalizes that this construction outputs y_{li} after t_{li} end-to-end steps for all $t_{li} \in \mathcal{I}$.

Claim 3 *For every $l \in [L], i \in [s]$, we indeed have $|\tilde{\boldsymbol{w}}_{li}| = \tilde{p}_l$, and $f_{\boldsymbol{w}}^{\text{e2e-}t_{li}}(\phi(\boldsymbol{x})) = y_{li}$ for $t_{li} \in \mathcal{I}$.*

Invoking this claim at $t_{Ls} = T$ directly gives us that $f_{\boldsymbol{w}}^{\text{e2e-}T}(\phi(\boldsymbol{x})) = y_{Ls} = C(\boldsymbol{x})$ is the output of the circuit. Finally, noting that bounds on T, d, n' by Eqs. (35),(39), and (36) even after replacing (n, s) with $(n+1, s+1)$ is according to the lemma statement. This removes the posed restriction on the circuit that the weight coming from the last gate of the previous layer is zero, concluding the proof. ∎

We now return to the deferred proofs of the claims in order.
Proof [Proof of Claim 1] Solving the recurrence in Eq.(33) immediately gives us $\tilde{p}_l = (s+1)^{l-1} \cdot n$. The second part will be shown inductively. Observe that $\sum_{\ell=0}^0 |\boldsymbol{v}_\ell| = |\boldsymbol{v}_0| = n - 1 = \tilde{p}_1 - 1$. For any $l > 1$, inductively

$$\sum_{\ell=0}^{l-1} |\boldsymbol{v}_\ell| = |\boldsymbol{v}_{l-1}| + \sum_{\ell=0}^{l-2} |\boldsymbol{v}_\ell| = s\,\tilde{p}_{l-1} + \tilde{p}_{l-1} - 1 = (s+1)\tilde{p}_{l-1} - 1 = \tilde{p}_l - 1\,.$$

∎

Proof [Proof of Claim 2] We first note that during the output of the first T predictions, our linear predictor has only shifted by at most $T - 1$ positions to the right. At the beginning, the right-most coordinate $\boldsymbol{v}[-1]$ is aligned with 1 to the left-end of $\phi(\boldsymbol{x})$ according to the construction Eq. (38). As there are $T - 1$ additional 0's padded before $\boldsymbol{x}$, the weight $\boldsymbol{v}[-1]$ never gets aligned with any coordinates in $\boldsymbol{x}$ (or the new ones to be added to its right) in the first T predictions.

In summary, letting $\boldsymbol{z}^t = f_{\boldsymbol{w}}^{\text{CoT-}t}(\phi(\boldsymbol{x}))$, for any $1 \le t \le T - 1$, we can say

$$\langle \boldsymbol{w}[-n - t :], \boldsymbol{z}^t[-n - t :]\rangle \le \sum_{l=1}^L \sum_{i=1}^s \|\boldsymbol{w}_{ls}\|_1 \tag{44}$$

where we used the fact that only weights from $[\boldsymbol{v}_1, \ldots, \boldsymbol{v}_L]$ align with $[\boldsymbol{x}, \boldsymbol{z}^t[-t :]]$ during the first T iterative steps. And these vectors are only constructed from $\{\boldsymbol{w}_{li} : l \in [L], i \in [s]\}$ by padding zeros, and $\boldsymbol{z}^t$ has binary coordinates. Hence, the inner-product is bounded by the total ℓ_1 norm of the circuit.

Finally, during the tth iterative prediction for any $t \in [T] \setminus \mathcal{I}$, we have that the coordinate of $\boldsymbol{v}$ that aligns with 1 to the left in $\phi(\boldsymbol{x})$ is $\boldsymbol{v}[-t]$; this is because $\boldsymbol{v}[-1]$ is aligned at the start and the predictor moved $t - 1$ steps to the right. Therefore, using Eq. (44), for any $t \in [T] \setminus \mathcal{I}$,

$$\langle \boldsymbol{w}, \boldsymbol{z}^{t-1}[-d :]\rangle \le \boldsymbol{v}[-t] + \sum_{l=1}^L \sum_{i=1}^s \|\boldsymbol{w}_{li}\|_1 \le -1\,,$$

where, in the last inequality, we used $v[-t] = -B$ from Eq. (42) and the value of B. Thresholding immediately gives us $f_w^{\text{e2e-}t}(\phi(x)) = \text{thr}(\langle w, z^{t-1}[-d :]\rangle) = 0$, as desired. ∎

Proof [Proof of Claim 3] We first start by inductively (on l) verifying that $\tilde{w}_{li} = \tilde{p}_l$ for $l \in [L], i \in [s]$. Clearly, by our description before (43), we don't insert any zeros in w_{1i} for $1 \leq i \leq s$, and therefore $|\tilde{w}_{1i}| = p_1 = n = \tilde{p}_1$. Now for any $l > 1$, on the top of weights from the first $(l-2)$ layers, there are also s number of weights associated with $(l-1)^{\text{th}}$ layer in w_{li}. Before all of these weights we have padded $\tilde{p}_{l-1} - 1$ zeros. Therefore, using this and the induction hypothesis

$$|\tilde{w}_{li}| = \tilde{p}_{l-1} + s(\tilde{p}_{l-1} - 1) + s = (s+1) \cdot \tilde{p}_{l-1} = \tilde{p}_l \,.$$

Finally, we return to the most important claim that for any $t_{li} \in \mathcal{I}$, we have $f_w^{\text{e2e-}t_{li}}(\phi(x)) = y_{li}$. First of all, by our construction of the vector v in Eq. (42), for any of these time values, the coordinate of v that aligns with the left-most 1 in $\phi(x)$ has the value simply zero. This allows us to focus on the coordinates starting from x and to its right from newly added CoT. We will show this by induction on the indices from $\mathcal{I}$ in their increasing order.

The base case is $t_{11} = n$. That means $n - 1$ steps have already elapsed. By Claim 2, we only output 0 in them. The linear predictor also moved $n-1$ positions to the right, and $v_0 = 0^{n-1}$ aligns with the last $n-1$ tokens of CoT. The tokens before that are x which align with $v_1[-n :] = \tilde{w}_{11} = w_{11}$. Therefore, the output

$$f_w^{\text{e2e-}t_{11}}(\phi(x)) = \text{thr}(\langle w_{11}, x \rangle) = y_{11} \,.$$

We now consider any time step $t \in \mathcal{I}$ for $t = t_{li}$. Let $z := f_w^{\text{CoT-}(t-1)}(\phi(x))$ be the current CoT. We break into two cases.

1. *Case 1: $t = t_{li}$ with $i = 1$ for some $l > 1$.* In this case, note that $t_{li} = t_{(l-1)s} + \tilde{p}_l$. And thus, the CoT z has $\tilde{p}_l - 1$ zeros at the end. Also, by Claim 1, these zeros exactly align with $w[-(\tilde{p}_l-1) :] = [v_{l-1}, \ldots, v_0]$, i.e. the weights associated with the previous layers. And also note that $(n + t_{(l-1)s}) = \tilde{p}_l$, and thus these many coordinates prior to that are $v_l[-\tilde{p}_l :] = \tilde{w}_{l1}$ using Eqs. (37) and (34). Finally, using the induction hypothesis this $\tilde{w}_{li}$ exactly aligns with the coordinates of z that has the following form

$$x, \; \underbrace{0 \ldots 0}_{(n-1) \text{ bits}}, y_{11}, \; \underbrace{0 \ldots 0}_{(n-1) \text{ bits}}, y_{12}, \ldots \ldots, y_{\ell(i-1)}, \underbrace{0 \ldots 0}_{\tilde{p}_\ell - 1 \text{ bits}}, y_{\ell i}, \ldots \ldots \ldots, y_{(l-1)(s-1)}, \underbrace{0 \ldots 0}_{\tilde{p}_{l-1} - 1 \text{ bits}} y_{(l-1)s} \,.$$

Based on the way, we constructed $\tilde{w}_{li}$, we directly conclude that

$$f_w^{\text{e2e-}t} = \text{thr}(\langle w_{li}, [x, y_{11}, y_{12}, \ldots, y_{(l-1)(s-1)}, y_{(l-1)s}]\rangle) = y_{li} \,.$$

2. *Case 2: $t = t_{li}$ with $i > 1$ for some $l \in [L]$.* The argument is very similar to the previous case. If we only isolate the CoT after the time $t_{(l-1)s}$, then by the induction hypothesis and Claim 2, it has the following form:

$$\underbrace{0 \ldots 0}_{\tilde{p}_l - 1 \text{ bits}}, y_{l1}, \underbrace{0 \ldots 0}_{\tilde{p}_l - 1 \text{ bits}}, y_{l2}, \ldots \ldots, y_{l(i-1)}, \underbrace{0 \ldots 0}_{\tilde{p}_\ell - 1 \text{ bits}} \,.$$

Again $[v_{l-1}, \ldots, v_0]$ align with the last $\tilde{p}_\ell - 1$ zeros and does not contribute. The $(i-1)\tilde{p}_l$ many weights before that in w are $v_l[-(i-1)\tilde{p}_l :] = [\tilde{w}_{l(i-1)}, \ldots, \tilde{w}_{l1}]$ by Eq (34). Note that

the only variables with potentially non-zero values are at the indices which are multiples of $\tilde{p}_l$. These correspond to the last coordinates of each $\tilde{\boldsymbol{w}}_{l(i-1)}, \dots, \tilde{\boldsymbol{w}}_{l1}$ respectively; the value of all these coordinates is zero by our assumption that the last weight coming from the previous layer is always 0. The CoT before this (from $\boldsymbol{x}$ and new $t_{(l-1)s}$ CoT tokens) is exactly of the form we described in Case 1. Moreover, this now aligns with the $\tilde{p}_l$ many coordinates prior to $\tilde{\boldsymbol{w}}_{l(i-1)}$, which by Eq. (34) is exactly $\tilde{\boldsymbol{w}}_{li}$. Again noticing the form $\tilde{\boldsymbol{w}}_{li}$, we conclude

$$f_{\boldsymbol{w}}^{\text{e2e-}t} = \text{thr}(\langle \boldsymbol{w}_{li}, [\boldsymbol{x}, y_{11}, y_{12}, \dots, y_{(l-1)(s-1)}, y_{(l-1)s}]\rangle) = y_{li}\,.$$

∎

Appendix F. Proof of Theorem 15 & Miscellaneous Discussions

F.1. Context and Iterated Sparse Linear Thresholds

Here, we discuss how the class of sparse linear thresholds (at least with bounded norm) already satisfies two of the three desiderata in Section 5 simultaneously. However, it remains open whether it satisfies the first desideratum of the expressive power. Let

$$\mathcal{F}_{d,k,\text{lin}} = \{f_{\boldsymbol{w},b} \in \mathcal{F}_{d,\text{lin}} : \|\boldsymbol{w}\|_0 \le k\}\,. \tag{45}$$

As mentioned, it remains open whether $\text{TM}(\text{S}, \text{T}) \subseteq \mathcal{F}_{d,\text{lin}}^{\text{e2e-}T}$ with $d, T = \mathcal{P}oly(\text{T})$ and $k \lll d$. However, the computational tractability is easy to see. The important thing is that it has the property of "high-context" and "low-sample-complexity". Indeed, the e2e sample complexity is $O(k^2 + k \log d)$.

We now prove that indeed our universal base class satisfies all three desiderata.

F.2. Proof of Theorem 15

We break the entire proof into three parts.
Expressivity. We first have that $\text{TM}(\text{S}, \text{T}) \subseteq post \circ \mathcal{F}_{\text{TM}}^{\text{e2e-}T} \circ pre$. (i.e. $\mathcal{F}_{\text{TM}}^{\text{e2e-}T}$ expresses $\text{TM}(\text{S}, \text{T})$ up to pre and post-processing steps). This is because our autoregressive function essentially is designed to hard code each transition (s_t, a_t, b_t), i.e. for any possible transition table τ and input $\boldsymbol{\omega} \in \{0, 1\}^*$, we have

$$f_{\tau}^{\text{e2e-}t}(\text{Pre}(\boldsymbol{\omega})) = (s_t, a_t, b_t)\,.$$

This can be shown by induction on t. For the base case, consider $t = 1$. The head is initially at the $p_0 = |\boldsymbol{\omega}| + 1$, and $\text{Tape}[p_0] = \square$, and the internal state is initialized to $s_0 = 1$. Therefore, we have $(s_1, a_1, b_1) = \tau(1, \square)$. We must show that $f_{\tau}^{\text{e2e-}(t=1)}(\text{Pre}(\boldsymbol{\omega})) = f_{\tau}(\text{Pre}(\boldsymbol{\omega})) = (s_1, a_1, b_1)$ to verify the base case. Let $\boldsymbol{x} = \text{Pre}(\boldsymbol{\omega})$. Indeed, from the pre-processing step we have $\boldsymbol{x}[i].\text{move} = +1$ for all $i \in |\boldsymbol{x}|$, and thus $\text{pos}[i] = i - 1$ for any $i \in |\boldsymbol{x}|$, and there will be no position i for which $\text{pos}[i] = |\boldsymbol{x}|$. Also, the last index in $\boldsymbol{x}$ has $\boldsymbol{x}[-1].\text{state} = 1$ by construction in the pre-processing step. Therefore, $(1, \square) = \text{read-tape}(\boldsymbol{x})$. Clearly, by Algorithm 1, we output $(s_1, a_1, b_1) = \tau(1, \square)$.

For the inductive case, let's assume that the claim holds for all steps i in $1 \le i \le t$ for some t. Then we must show that

$$f_{\tau}^{\text{e2e-}(t+1)}(\boldsymbol{x}) = f_{\tau} \circ f_{\tau}^{\text{CoT-}t}(\boldsymbol{x}) = (s_{t+1}, a_{t+1}, b_{t+1})\,.$$

Let $z = f_\tau^{\mathsf{CoT}\text{-}t}(x)$. By induction, we know that for any $1 \leq i \leq t$, we have $p_{i-1} = \sum_{j<i} z[j].\mathsf{move} = \mathsf{pos}[i]$ is the head position of the machine just before the time-step i and this is where the symbol $z[i].\mathsf{symb}$ got written. By the way a Turing machine operates, the current head position after the end of t steps is $p_t = p_{t-1} + b_t = \mathsf{pos}[-1] + z[-1].\mathsf{move} = \mathsf{npos}[|z|]$ (by the induction hypothesis). So our goal is to retrieve the symbol $r_{t+1} = \mathsf{Tape}[p_t]$ currently present at this location, i.e. the symbol that was written most recently at this position on the tape. This exactly corresponds to finding the largest j^* such that $\mathsf{pos}[j^*] = \mathsf{npos}[|z|]$ and retrieve $z[j^*].\mathsf{symb}$, and if j^* does not exist then at no point in the past, the head of the machine has been at the position so the tape must contain $\square$. Our read-tape(z) operation is exactly implementing this step (Algorithm 1). So we indeed correctly find the symbol r_{t+1} that was read by the Turing machine in order to make the next $(t+1)^{\text{th}}$ move using our read-tape operation. Also, the read-tape operation outputs the state at the final location $z[-1].\mathsf{state}$ which is s_t by the induction hypothesis. Therefore, read-tape$(z) = (s_t, r_{t+1})$ is satisfied. Then we indeed have

$$f_\tau^{\mathsf{e2e}\text{-}(t+1)}(x) = f_\tau(z) = \tau(\text{read-tape}(z)) = \tau(s_t, r_{t+1}) = (s_{t+1}, a_{t+1}, b_{t+1}),$$

where the second inequality follows from the definition of f_τ (Algorithm 1), the next one follows from the discussion above, and the last one follows directly from the way a Turing machine works. Therefore, at the end-to-end step with $t = \mathsf{T}$, we have $f_\tau^{\mathsf{e2e}\text{-}\mathsf{T}}(\mathsf{Pre}(\omega)) = (s_\mathsf{T}, b_\mathsf{T}, a_\mathsf{T})$. Applying post-processing step returns $a_\mathsf{T} = g_{\langle \mathsf{S},\mathsf{T},\tau\rangle}(\omega)$, which is the desired answer returned to the Turing machine with the transition rule τ. This establishes $\mathsf{TM}(\mathsf{S},\mathsf{T}) \subseteq \mathsf{Post} \circ \mathcal{F}_\mathsf{TM}{}^{\mathsf{e2e}\text{-}\mathsf{T}} \circ \mathsf{Pre}$.

Sample Complexity. By Theorem 3, we have that $\mathcal{F}_\mathsf{TM}{}^{\mathsf{e2e}\text{-}\mathsf{T}}$ is CoT-learnable by the rule CONS$_\mathsf{CoT}$ on a sample set S_CoT of size

$$m^{\mathsf{CoT}\text{-}\mathsf{T}} = O(\varepsilon^{-1}(\mathsf{S}\log\mathsf{S} + \log\delta^{-1})),$$

where we used the facts that $\log|\mathcal{F}_\mathsf{TM}| \leq \log(6\mathsf{S})^{3\mathsf{S}} = O(\mathsf{S}\log\mathsf{S})$ and CONS$_\mathsf{CoT}$ is also CONS$_\mathsf{e2e}$.

Tractable CoT-learnability. Finally, we need to show that CONS$_\mathsf{CoT}$ for the class $\mathcal{F}_\mathsf{TM}$ on S_CoT of size $|S_\mathsf{CoT}| = O(\varepsilon^{-1}(\mathsf{S}\log\mathsf{S} + \log\delta^{-1}))$ containing inputs of length at most n, is implementable in time $\mathcal{P}oly(n, \mathsf{T}, \varepsilon^{-1}, \log\delta^{-1})$. Due to Theorem 8, it suffices to show that the base class consistency CONS$_{\mathcal{F}_\mathsf{TM}}$ is solvable in time polynomial in its input. This is clearly true for the implementation of CONS$_{\mathcal{F}_\mathsf{TM}}$ we provide, concluding the proof of this theorem.