PROVABLE COMPOSITIONAL REASONING VIA CURRICULUM LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Compositional generalization—the ability to solve complex problems by combining solutions to simpler sub-problems—is a fundamental capability of both natural and artificial intelligence, and a key mechanism underlying chain-of-thought reasoning. However, the theoretical underpinnings of compositional generalization remain poorly understood: when and why does decomposing a problem into parts yield more efficient learning than solving it directly? We study this question through the canonical problem of learning to simulate semiautomata (predicting the outcome of T steps of sequential computation), a model that captures state tracking, regular language recognition, and modular arithmetic. We show that a *curriculum* based approach—recursively decomposing longer sequences into shorter sub-problems, learning to solve them, and composing the solutions—achieves dramatically better statistical complexity than direct methods. (i) For a setting inspired by supervised fine-tuning (SFT) where the learner receives interactive feedback on intermediate states of the computation, curriculum facilitates learning from only $2^{\widetilde{\mathcal{O}}(\sqrt{\log T})}$ tokens of supervision; i.e., *subpolynomial* in the sequence length T, overcoming the $\Omega(T)$ token barrier required by direct simulation. (ii) For a setting inspired by reinforcement learning with verifiable rewards (RLVR), where the learner improves a pre-trained reference model using an outcome verifier, we show that curriculum reduces the requirement on the reference model from coverage at the full sequence length T to coverage at a shorter block length $B \ll T$, an exponentially weaker condition.

1 INTRODUCTION

Chain-of-thought reasoning, where models expend additional computation by producing intermediate reasoning tokens prior to a final answer, has driven significant advances in the capabilities of language models. A key mechanism underlying these advances is *compositional generalization*—the ability to combine solutions to simpler sub-problems in order to solve harder ones—with a growing body of evidence that it is central to how reasoning models scale to difficult tasks (Lake & Baroni, 2018; Saxton et al., 2019; Press et al., 2023; Dziri et al., 2023; Zhao et al., 2024).

Many works emphasize the importance of *curriculum-based training* (Bengio et al., 2009) for enabling compositional generalization in reasoning models (Zeng et al., 2025; Motwani et al., 2025; Kimi Team et al., 2025). Here, problems of increasing difficulty teach the learner increasingly complex compositions of low-level skills. Recent work has further demonstrated that curricula can be induced by the learner itself, avoiding the need for hand-engineered notions of problem difficulty (Chen et al., 2025b; Shi et al., 2025; Yu et al., 2025; Zhao et al., 2025).

While these *autocurriculum* approaches are intuitively appealing, it remains poorly understood as to *when*, *why*, and *to what extent* autocurricula can reduce the cost of learning, and what principles should guide their design. Prior work showed that a form of autocurriculum inspired by boosting—specifically, using the model's own performance to *select which problems to focus on* from a large corpus—provably improves learning efficiency for supervised fine-tuning and reinforcement learning (Rajaraman et al., 2026). In this work, we ask a more ambitious question:

How should a learner *design its own problems* to facilitate solving tasks far beyond its base capabilities, and how can compositional structure guide this process?

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

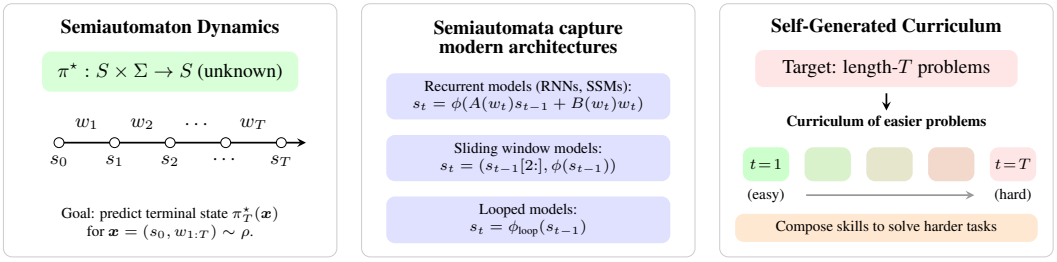


Figure 1: Model overview. *Left:* semiautomata dynamics under an unknown transition function $\pi^\star$. *Middle:* semiautomata capture several modern sequential reasoning architectures. *Right:* self-generated curriculum: a length-T instance is decomposed into shorter sub-instances of length $\tau \ll T$, short-range models are trained, and composed to solve the original problem.

1.1 Contributions

We study these questions through the task of *semiautomaton simulation*—learning to predict the outcome of T steps of sequential computation—a canonical model that captures state tracking, regular language recognition, and modular arithmetic. A semiautomaton is defined by an unknown transition function $\pi^\star : S \times \Sigma \to S$ that updates a state via $s_t \leftarrow \pi^\star(s_{t-1}, w_t)$, where $\boldsymbol{w} = \boldsymbol{w}_{1:T}$ is an input word. The intermediate states $s_{1:T}$ constitute a chain-of-thought, and the goal is to predict the final state s_T after T steps. In the language modeling context, one can think of states as tokens, but they can be more general (say, a block of k tokens, corresponding to a fixed-size attention window). The Markovian structure of semiautomata means that models solving shorter instances can be *composed* to solve longer ones, making this a natural testbed for understanding how curriculum can enable compositional generalization.[1] Our main results—building on Rajaraman et al. (2026)—show that a self-generated compositional curriculum, where the learner adaptively decomposes long problems into shorter sub-problems, yields superpolynomial reductions in supervision and computational cost relative to non-curriculum baselines, in both supervised fine-tuning (SFT) and reinforcement learning with verifiable rewards (RLVR).

Section 3: Fine-tuning with interactive CoT supervision (iSFT). Our first set of results considers a setting in which the learner has interactive access to the semiautomaton $\pi^\star$ of interest and aims to select a set of state transitions on which to fine-tune a model with next-token prediction (concretely, given any word $\boldsymbol{w}$ and step t, the learner can ask for the state s_t, at one unit of cost). The standard approach would be to generate full reasoning traces and train on every intermediate state, but can we do better by deliberately choosing *which* states to supervise? We show that the answer is yes. We give an autocurriculum-based algorithm, `AutoLearn` (Algorithm 2), which interactively selects which intermediate states to label by constructing a curriculum of shorter sub-problems and composing them. The resulting query complexity—the number of state labels requested—is $2^{\widetilde{\mathcal{O}}(\sqrt{\log T})}$, which is *subpolynomial* in the sequence length T. This overcomes the $\Omega(T)$ barrier inherent to direct simulation, where every intermediate state must be labeled.

Section 4: Fine-tuning a reference model (RLVR). Our second set of results considers a setting inspired by reinforcement learning with verifiable rewards (RLVR) (Lambert et al., 2024). Here, the learner has access to a pre-trained *reference model* π^{ref} that can correctly solve short instances of length $B \ll T$ with constant probability, as measured by a coverage coefficient (Zhu et al., 2023; Song et al., 2024), along with an *outcome verifier* that checks correctness of guesses for the terminal state. The reference model's coverage at the full sequence length T may be exponentially worse than at length B, so directly applying prior RL methods (Rajaraman et al., 2026) at the length-T scale is prohibitively expensive. We give an algorithm, `AutoLearn.RL`, which uses autocurriculum to compose short-range predictions into full-length solutions, reducing the requirement on π^{ref} from coverage over full sequences of length T to coverage at the shorter block scale B—an exponentially weaker condition. This is enabled by a *coverage expansion* phenomenon: curriculum enables

[1] This model can be viewed as a Markovian restriction to the chain-of-thought learning framework of Joshi et al. (2025); Rajaraman et al. (2026), which allows for arbitrary dependence on previous states.

the reference model to incrementally build coverage over harder problems, starting from the short length-B instances it can already solve.

1.2 Organization

Section 2 formally introduces semiautomata and the prediction oracle. Section 3 then discusses our main results for iSFT, and we overview the key techniques in Section 3.2. In Section 4, we discuss our main results in the reinforcement learning setting. A more formal presentation of our main results, proofs and supporting results are deferred to the appendix.

2 Problem Setting

This paper formalizes the two learning settings we study—supervised fine-tuning with interactive feedback (iSFT) and reinforcement learning with verifiable rewards (RLVR)—and presents informal statements of our main results. We begin with preliminaries on semiautomata (Section 2.1), then introduce the iSFT setting and results (Sections 3 and 3.1), and finally introduce the RLVR setting and results (Sections 4 and 4.1).

2.1 Preliminaries: Semiautomata and Chain-of-Thought Learning

We focus on learning to generate high-quality chains of thought, through direct supervision or reinforcement learning with verifier feedback. To understand how curriculum and composition can reduce the cost of learning, we adopt *semiautomata* as a stylized yet rich model of chain-of-thought reasoning.

Definition 2.1 (Semiautomaton simulation)**.** *A semiautomaton is defined by a transition function $\pi : S \times \Sigma \to S$ over a state space S and input alphabet Σ. An instance $\boldsymbol{x} = (s_0, \boldsymbol{w}_{1:t})$ is composed of an initial state $s_0 \in S$ and a word $\boldsymbol{w}_{1:t} \in \Sigma^t$ for some $t \in \mathbb{N}$, and t is the length of the instance*[2]*. Words correspond to the input strings that we want the model to process, while the initial state s_0 represents the starting point of the computation. The overall instance $\boldsymbol{x}$ can be interpreted as a prompt in the language modeling context. The semiautomaton simulator $\pi_t(\boldsymbol{x})$ returns the terminal state reached after applying π iteratively for t steps on $\boldsymbol{x}$.*[3] *Namely $\pi_t(\boldsymbol{x}) = s_t$, where $\forall 1 \leqslant t' \leqslant t,\ s_{t'} \leftarrow \pi(s_{t'-1}, w_{t'})$. We will use $\pi_{1:t}(\boldsymbol{x})$ to denote the full sequence of states $(s_1, \cdots, s_t) \in S^t$ generated by this process, which is referred to as the chain-of-thought (CoT).*[4]

We use the term "simulation" to reflect that the objective is to predict the final state (outcome) of the semiautomaton, not specifically to learn the transition function or full CoT (except as a means to predict the outcome). In the language modeling context, the state itself can be a single token, but our setting is more general and allows for more complex state representations (e.g., blocks of tokens).[5]

The key structural property of semiautomata is that the transition function $\pi^\star$ is Markovian and stationary. That is, it is applied identically at every step, and the state s_t is a sufficient summary of the computation history up to time t. This means that any contiguous chunk of a length-T computation can be interpreted as a self-contained computation of shorter length $t \leqslant T$, and models trained to solve shorter instances can be *composed* to solve longer ones. This structure makes semiautomata a natural testbed studying how curriculum, composition, and self-generated sub-problems interact. This refines the setup from Rajaraman et al. (2026), which considered general, potentially non-Markovian transition functions. Note that the Markovian structure is precisely why we allow a state to encode more than a single token: at the granularity of individual tokens, the Markovian assumption would be severe (language models attend to their entire history) whereas letting s_t summarize a block of tokens (such as a fixed attention window) keeps the Markovian condition milder and more realistic.

[2] For a vector $\boldsymbol{w}$, $\boldsymbol{w}_{1:t}$ is the vector obtained by slicing to length t; w_t (non-bold) indicates the t^{th} coordinate.

[3] We consider the *parameter-sharing regime* where transition functions are time invariant.

[4] The notations $\pi_{t'}(\boldsymbol{x})$ and $\pi_{1:t'}(\boldsymbol{x})$ extend to $t' < t$ by truncating $\boldsymbol{x} = (s_0, \boldsymbol{w}_{1:t})$ to $\boldsymbol{x}' = (s_0, \boldsymbol{w}_{1:t'})$ and returning $\pi_{t'}(\boldsymbol{x}')$ or $\pi_{1:t'}(\boldsymbol{x}')$.

[5] Going forward, we will often use the term "state" and "token" interchangeably when clear from context.

Beyond serving as a theoretical testbed, semiautomata capture a broad range of sequential computation tasks, including settings where the state space S and input alphabet Σ can be very large, necessitating function approximation. We discuss a few such examples in Appendix A.1, motivating the general setting we describe in the next section.

2.1.1 Modeling and function approximation

To allow for generalization when the state and input alphabet are large, we assume access to a *model class* $\Pi \subseteq \{S \times \Sigma \to S\}$ of candidate transition functions (equivalently, "next-state/token predictors") containing the unknown transition function $\pi^\star$. This class can represent a parameterized model family (e.g., fixed-window transformers or state-space models). As a computational primitive, our iSFT algorithms make use of *next-state/token prediction* (ERM on the next-state/token prediction loss), but otherwise access the model class only implicitly.

Definition 2.2 (Next-token prediction oracle for Π)**.** *Given any realizable dataset $D = \{(s_i, w_i) \mapsto \pi^\star(s_i, w_i)\}_{i=1}^n$ of input-next-state pairs, a next-state/token prediction oracle for Π, $\mathtt{NTP}(D)$, returns $\widehat{\pi} \in \Pi$ that minimizes the next-state/token prediction loss: $\frac{1}{n}\sum_{i=1}^n \mathbb{I}(\widehat{\pi}(s_i, w_i) \neq \pi^\star(s_i, w_i)) = 0$.*[6]

Many of our results make use of the Natarajan dimension of Π, a complexity measure that characterizes the distribution-free sample complexity of learning Π from i.i.d. examples (up to log-factors).

Definition 2.3 (Natarajan dimension of Π (Natarajan, 1989))**.** *The Natarajan dimension of a class of models Π is the largest $d \in \mathbb{N}$ such that there exist instances $\{(s_i, w_i)\}_{i=1}^d \subseteq S \times \Sigma$ and pairs of labels $\{y_i, y_i'\}_{i=1}^d \subseteq S \times S$ satisfying $y_i \neq y_i'$ for all $i \in [d]$, with the following property: for every subset $I \subseteq [d]$, there exists a model $\pi_I \in \Pi$ such that $\forall i \in I,\ \pi_I(s_i, w_i) = y_i$ and $\forall i \notin I,\ \pi_I(s_i, w_i) = y_i'$.*

Remark 2.4 (Simplifications in the semiautomaton framework)**.** *While our results are motivated by language model reasoning, our problem setting makes two fairly strong simplifications: (1) We restrict the learner to Markovian models, which rules out classes like Transformers that can attend to the full history. (2) We consider a deterministic setting, which rules out stochastic models. Regarding the first point, we note that while this may seem like a strong simplification when we interpret states s_i as tokens, our guarantees allow for general, potentially large state spaces, which makes the framework quite rich nonetheless. For example, s_i may represent a block of tokens of length k, in which case our model allows for fixed-size attention windows of size k. Relaxing both limitations is an important direction for future work, but we believe our framework is still a powerful setting to explore the role of composition.*

3 Learning from Interactive CoT Supervision (iSFT)

Recall from Section 2.1 that the learner's goal is to predict the terminal state $\pi_T^\star(\boldsymbol{x})$ reached by a ground-truth semiautomaton $\pi^\star$ on length-T instances $\boldsymbol{x} \sim \rho$. Standard supervised approaches fix in advance *what* supervision to collect (Joshi et al., 2025): either label the full chain-of-thought $\pi_{1:T}^\star(\boldsymbol{x})$ on each instance and fit a model to its next-state pairs via next-token prediction, or learn *end-to-end* from the terminal state $\pi_T^\star(\boldsymbol{x})$ alone. Our first setting considers a more flexible model—*learning from interactive chain-of-thought supervision*, or *iSFT*—in which the learner instead chooses *which* states of the chain-of-thought to supervise, querying the state reached at any step t at one unit of cost. We formalize this interactive access through the oracle below.

Definition 3.1 (Interactive chain-of-thought oracle)**.** *Let $\pi^\star$ denote the ground-truth semiautomaton. An interactive chain-of-thought oracle, $\mathtt{iCoT}(\cdot)$, is a function which takes as input an instance $\boldsymbol{x} = (s_0, \boldsymbol{w}_{1:t})$ of any length $t \in \mathbb{N}$ and returns the chain-of-thought state $\pi_t^\star(\boldsymbol{x})$ at step t.*

Two interpretations of this oracle will be useful. First, when applied to a length-t instance, $\mathtt{iCoT}(\boldsymbol{x})$ returns that instance's *terminal state*: the outcome of running $\pi^\star$ for t steps. Second, when applied to a length-t *prefix* of a longer length-T instance, the same query returns the *intermediate state* $\pi_t^\star(\boldsymbol{x})$ that the computation passes through at step t. Thus, by querying prefixes, the learner can recover any state along the chain-of-thought of a long instance.

[6] Since semiautomata are deterministic, we work with the 0-1 loss. For general (stochastic) language modeling settings, standard next-token prediction corresponds to ERM under the logarithmic loss.

Problem 3.2 (Learning from Interactive Chain-of-Thought Supervision: iSFT)**.** *The learner is given a class $\Pi \subseteq \{S \times \Sigma \to S\}$ of next-state/token predictors containing the true semiautomaton $\pi^\star$, along with sampling access to a target distribution $\rho \in \Delta_{S\times\Sigma^T}$ of length-T instances. The objective is to learn a model $\widehat{\pi} : S \times \Sigma^T \to S$ such that with probability at least $1-\delta$, the terminal state prediction error satisfies $\Pr_{\boldsymbol{x}\sim\rho}\left[\widehat{\pi}(\boldsymbol{x}) \neq \pi^\star_T(\boldsymbol{x})\right] \leqslant \varepsilon$. To do so, the learner may query the interactive chain-of-thought oracle $\texttt{iCoT}(\cdot)$ of Definition 3.1 on instances of any length. The learner is evaluated according to the following desiderata:*

1. *Query complexity: The number of calls $n_{\texttt{query}}$ made to the interactive CoT oracle $\texttt{iCoT}(\cdot)$.*
2. *Sample complexity: The number of instances $n_{\texttt{sample}}$ drawn from ρ.*

Note that in the iSFT setting, the learner can recover both standard forms of supervision (since $\texttt{iCoT}(\cdot)$ accepts instances of any length): T prefix queries suffice to reconstruct an instance's full chain-of-thought (for next-token prediction (Foster et al., 2024; Joshi et al., 2025)), and a single query suffices to obtain the end-to-end outcome. Full-CoT learners incur $\Omega(T)$ query complexity in the worst case (Proposition C.5); end-to-end lower bounds for the broader autoregressive model are discussed in Proposition C.7. The real leverage of the iSFT framework comes from the Markovian structure of $\pi^\star$ (Section 2.1), which lets the learner *construct its own sub-problems*: splitting a length-T instance at chosen boundaries and labeling the boundary states with $\texttt{iCoT}(\cdot)$ decomposes it into shorter instances, the terminal state of each becoming the start state of the next. In what follows, we will show that this form of self-generated supervision can be used to break the $\Omega(T)$ query complexity barrier for full-CoT supervision.

Motivation for the iSFT interaction model. For general semiautomata, labeling an intermediate state $\pi^\star_t(\boldsymbol{x})$ may be no easier than labeling the entire intermediate CoT $\pi^\star_{1:t}(\boldsymbol{x})$. Yet our framework counts the former as a single unit of cost and counts the latter as t units of cost. Nevertheless, there are two reasons why we bookkeep supervision costs in this manner:

- For many natural semiautomata, labeling a single state *is* cheaper than producing an entire CoT. We observe that for semiautomata including modular arithmetic and linear recurrences (Appendix A.1), the state at step t can be computed in parallel in $\mathcal{O}(\log t)$ time by divide-and-conquer over matrix or group products, whereas simulating $\pi^\star$ step by step to obtain the full length-t chain-of-thought takes $\Omega(t)$ time. Thus, a single state query is a natural unit of cost here.
- Second, the iSFT model underpins our reinforcement learning results in the sequel (Appendix D). There, we show that one can use a weak reference model and outcome verifier to "simulate" $\texttt{iCoT}$, allowing the curriculum and composition machinery we develop here to carry over directly.

3.1 Overview of Results for iSFT Setting

Our main result shows that, by querying the interactive chain-of-thought oracle to assemble a self-generated curriculum of shorter instances, semiautomata can be learned in the iSFT setting with sample and query complexity scaling *subpolynomially* in the sequence length T. Our algorithm $\texttt{AutoLearn}$ (Algorithm 2) thereby breaks the $\Omega(T)$ worst-case barrier for full-CoT learners.

Theorem 3.3 (Main result for iSFT setting (informal; see Theorem C.2))**.** $\texttt{AutoLearn}$ *(Algorithm 2) learns a model with $\Pr_{\boldsymbol{x}\sim\rho}\left[\widehat{\pi}(\boldsymbol{x}) \neq \pi^\star_T(\boldsymbol{x})\right] \leqslant \varepsilon$ with probability at least $1-\delta$ in the iSFT setting, and is computationally efficient in terms of a next-token prediction oracle for Π (Definition 2.2). The sample and query complexity used by the algorithm are upper bounded by:*

$$n_{\texttt{sample}}, n_{\texttt{query}} \leqslant 2^{\widetilde{\mathcal{O}}\left(\sqrt{\log(T)}\right)} \cdot \left[\frac{d\log(|S|)\log^2\left(\frac{1}{\varepsilon}\right)\log\left(\frac{1}{\delta}\right)}{\varepsilon}\right].$$

Here, $d \leqslant \log_2(|\Pi|)$ is the Natarajan dimension of Π (Definition 2.3), and $\widetilde{\mathcal{O}}(\cdot)$ ignores logarithmic factors in its argument.

$\texttt{AutoLearn}$ (Algorithm 2) builds this curriculum over the course of training by exploiting *composition*: since $\pi^\star$ is Markovian, a model that simulates $\pi^\star$ accurately for τ steps (under a suitable mixture over intermediate states) can be composed with itself to simulate $\pi^\star$ for $T \gg \tau$ steps. The catch is that errors accumulate under composition, so the short-range accuracy must be driven high

enough to tolerate them. AutoLearn resolves this with a *multiscale* variant of boosting (Schapire & Freund, 2013) that recursively interleaves two operations: (a) boosting, which aggregates several short-range models to drive their error down, and (b) composition, which stitches the boosted models into longer-range ones with degraded guarantees. The central difficulty, treated at length in Appendices C and E, is to turn samples from the target distribution ρ into samples from the distribution over short sub-problems required by boosting, without inflating the sample complexity. The only learning primitive AutoLearn invokes is next-token prediction (Definition 2.2) on i.i.d. next-state data; in fact, this guarantee is a *reduction*—any weak learner with low next-token-prediction error, not only ERM, can be substituted (Appendix C).

Comparison with non-curriculum baselines. The subpolynomial rate in Theorem 3.3 stands in sharp contrast with guarantees achievable by natural non-curriculum approaches.

- *Learning from full chain-of-thought:* A full chain-of-thought learner labels every intermediate state of $\pi^\star$ on each training instance and fits a model to the resulting next-state pairs by next-token prediction. This is computationally efficient—it reduces to next-token prediction over Π—but labels all T states per instance, so its query complexity grows linearly in T. We show this is unavoidable: any such learner needs $\Omega(dT)$ queries to reach constant error, regardless of which instances are queried (Proposition C.5).
- *Learning from end-to-end feedback:* An end-to-end learner queries only the terminal state $\pi^\star_T(\boldsymbol{x})$ on length-T instances. Adapting the construction of Joshi et al. (2025) to the semiautomaton setting, in Proposition C.7 we establish that the statistical complexity scales as $n_{\texttt{sample}} = n_{\texttt{query}} = \Omega(dT)$, and thus the generalization error can be $\Omega(T)$ times larger than that of the base class Π. Under standard cryptographic assumptions (Kearns & Valiant, 1994), this approach is also computationally intractable as semiautomata can encode branching programs (Barrington, 1986).

We give the formal lower bounds and a detailed comparison in Appendix C.3.

3.2 Overview of Techniques: Composition and Boosting

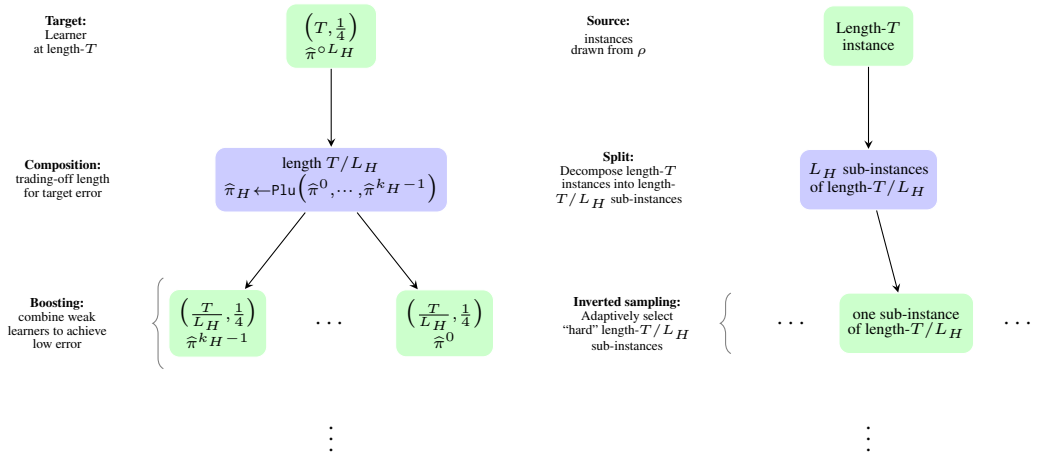
Our main algorithm, AutoLearn (Algorithm 2), exploits the compositional structure of semiautomata to achieve sublinear-in-T query complexity for learning models with high accuracy on length-T instances. The core idea is that a model that predicts the state at time τ with high accuracy for $\tau = \frac{T}{L}$ can be composed L times to obtain a model for the terminal state on length-T instances with constant accuracy. We formalize this idea through the notion of a *composition operator*.

Definition 3.4 (Composition operator)**.** *Fix some $t \in \mathbb{N}$, and consider any model $\widehat{\pi} : S \times \Sigma^t \to S$ for the terminal state on instances of length t. Then for some $L \in \mathbb{N}$, the composed model $\widehat{\pi}^{\circ L} : S \times \Sigma^{tL} \to S$ operates on instances of length tL. $\widehat{\pi}^{\circ L}$ chunks the instance into L segments of length t, applying $\widehat{\pi}$ to predict the terminal state of each chunk, using this as the start state for the next chunk, and repeating this process until termination.*

Composition allows us to extend models solving shorter instances to longer ones, and is equivalent to wrapping the short-length model within a recurrent loop. While this suggests a natural algorithm design approach (learn a model for short instances, then compose it), one expects that inaccuracies in the learned model $\widehat{\pi}$ will compound the more times we compose, degrading performance. This relationship between composition and accuracy is captured in the following observation.

Observation 3.5 (Compositional generalization)**.** *Recall that ρ denotes the distribution over length-T instances given by nature. Let $\overline{\rho}$ denote the uniform mixture distribution over instances of length $\tau = \frac{T}{L}$ obtained by chunking longer length-T instances (defined formally in Eq. (6)) and let $\varepsilon_0 = \frac{1}{4L}$. Let $\widehat{\pi}$ be any model satisfying a high-accuracy correctness guarantee on length-τ test instances drawn from $\overline{\rho}$; namely, $\Pr_{\boldsymbol{x}\sim\overline{\rho}}\big(\widehat{\pi}(\boldsymbol{x}) \neq \pi^\star_\tau(\boldsymbol{x})\big) \leqslant \varepsilon_0$. Then, the L-fold composition model $\widehat{\pi}^{\circ L}$ (defined formally in Definition 3.4) satisfies $\Pr_{\boldsymbol{x}\sim\rho}(\widehat{\pi}^{\circ L}(\boldsymbol{x}) \neq \pi^\star_T(\boldsymbol{x})) \leqslant \varepsilon_0 L = 1/4$.*

Observation 3.5 follows by a simple union bound, and provides a mechanism by which length (i.e., hardness) can be traded off for accuracy. However, this length-versus-accuracy tradeoff is not sufficient on its own to achieve sublinear-in-T query complexity, as the increased accuracy requirement (i.e., ε_0 in Observation 3.5) wipes out potential gains from solving instances of much shorter length.



(a) Tree structure of models. In green boxes, (t, ε) indicates instance length and target error under the corresponding learning distribution. The aggregate block predictor is analyzed through the conditioned potential argument. The k_H weak length-T/L_H models $\widehat{\pi}_H^0, \ldots, \widehat{\pi}_H^{k_H-1}$ are combined by plurality vote into $\widehat{\pi}_H$ and composed L_H-fold. The recursion continues for H levels; models are trained at the bottom-most level to solve 1-step instances, say via next-token prediction.

(b) Data flow on the same tree. Length-T instances $\boldsymbol{x} \sim \rho$ are decomposed by `Split` into L_H contiguous length-T/L_H sub-instances. `InvSampling` selects at most one among these to pass on a given one of the k_H boosting learners in (a). For the j^{th} boosting learner, `InvSampling` selects sub-instances based on the previous $j-1$ models' performance, to adaptively focus on "hard" sub-instances.

Figure 2: Two views of the recursion within $\texttt{AutoLearn}_h$ (Algorithm 1): the *(a)* tree-like structure of induced models and *(b)* flow of data on the same tree.

For instance, if we naively train a model $\widehat{\pi}$ using next-token prediction to very high accuracy on 1-step instances, the target error on 1-step problems would need to be $\varepsilon_0 = \frac{1}{4T}$, for which the standard sample upper bound scales as $\widetilde{\mathcal{O}}(d/\varepsilon_0) = \widetilde{\mathcal{O}}(dT)$.

We improve this length-versus-accuracy tradeoff by combining composition with *boosting* (Schapire & Freund, 2013), specifically Freund's boosting-by-filtering algorithm (Freund, 1995). Boosting lifts weak (i.e., constant-accuracy) models into high-accuracy models by training them on reweighted versions of the target distribution over instances and aggregating them. Combined with composition, the algorithm takes the form:

1. Train models solving short length-τ instances to high accuracy via boosting.
2. Compose the resulting models to solve length-T instances.

With some care, this approach can be shown to achieve query complexity $\widetilde{\mathcal{O}}(d\tau \log(1/\varepsilon_0) + d/\varepsilon_0)$ where $\varepsilon_0 = \tau/T$ when the target error under ρ is a constant. Choosing τ optimally gives us a guarantee scaling as $\widetilde{\mathcal{O}}(d\sqrt{T})$, which is sublinear in T, and improves over standard baselines.

From sublinear to subpolynomial query complexity. To achieve the stronger *subpolynomial* $2^{\widetilde{\mathcal{O}}(\sqrt{\log T})}$ in our main theorem (Theorem C.2), our main approach, `AutoLearn` (Algorithm 2) applies the boosting + composition template above recursively. To train models solving length-τ instances, we decompose instances into even shorter length-$\tau' \ll \tau$ ones, decompose these instances into even shorter length-$\tau'' \ll \tau'$ instances, and so on, until all target instances have length-1. At this "bottom level" of the recursion, all training examples consist of next-state transitions of the form $(s, w) \mapsto \pi^\star(s, w)$, and weak learners can be trained via next-token prediction. Bottom-level models trained this way within `AutoLearn` are then recursively *aggregated* through boosting and *composed* to solve longer length instances. The final length-T model trained through this process recursively aggregates models across multiple scales; we refer to this approach as *multiscale boosting*.

Multiscale boosting realizes a *length-based curriculum*: as the algorithm proceeds to deeper levels, longer (i.e., harder) instances are progressively decomposed into shorter ones which are easier to train to solve. By composition, models which solve shorter instances are combined to realize ones which can solve harder problems. Moreover, the curriculum itself is *self-generated*: at each scale, `AutoLearn` adaptively selects which sub-instances to train on based on the failures of models trained at finer scales (via `InvSampling`, Algorithm 3), rather than following a fixed schedule.

4 Improving a Reference Model from Verifier Feedback (RLVR)

Our second setting is inspired by reinforcement learning with verifiable rewards (RLVR), and models the post-training of a pretrained model. The learner is given a *reference model* π^{ref} (a weak generator of candidate solutions) and an *outcome verifier* that checks whether a guessed terminal state is correct, and aims to improve the accuracy of π^{ref} for the task of simulating the final semiautomaton state output $\pi^\star_T$. When π^{ref} already has nontrivial *coverage* of correct length-T solutions, this is well understood, and standard RLVR fine-tuning succeeds (Foster et al., 2025; Rajaraman et al., 2026). We instead study a harder regime in which π^{ref} is competent only at a much shorter block length $B \ll T$—that is, it solves length-B instances with constant probability, but is exponentially unlikely to solve length-T instances directly—and ask how to *compose* this short-horizon competence into full-length solutions.

We begin by formalizing the verifier and learning problem, before stating our assumption on π^{ref}.

Definition 4.1 (Outcome verifier)**.** *An outcome verifier $\mathcal{V}$ takes in an instance $\boldsymbol{x} \in S \times \Sigma^t$ of length $t \leqslant T$, along with a guess $s \in S$ for the terminal state $\pi^\star_t(\boldsymbol{x})$, and returns a reward $\mathbb{I}\big(s = \pi^\star_t(\boldsymbol{x})\big)$.*

The verifier is much weaker than the interactive chain-of-thought oracle of Definition 3.1. Instead of returning the correct state outright, it gives only binary feedback on a guess, and reveals only that the proposed state is incorrect when the guess is wrong (especially when $|S|$ is large). We will use two equivalent interpretations for the verifier: (1) When applied to a length-t instance and a guess s, $\mathcal{V}(\boldsymbol{x}, s)$ tests whether s is that instance's *terminal state*: the outcome of running $\pi^\star$ for t steps; (2) When applied to a length-t *prefix* of a longer length-T instance (interpreted as a new length-t instance), the same query tests whether s is the *intermediate state* $\pi^\star_t(\boldsymbol{x})$ that the computation passes through at step t. With this in hand, we state the learning problem.

Problem 4.2 (RL setting)**.** *The learner is given a class $\Pi \subseteq \{S \times \Sigma \to S\}$ of next-state/token predictors containing the true semiautomaton $\pi^\star$, along with sampling access to a target distribution ρ over length-T instances. The objective is to learn a model $\widehat{\pi} : S \times \Sigma^T \to S$ such that with probability at least $1-\delta$, the terminal state prediction error satisfies $\Pr_{\boldsymbol{x}\sim\rho}\big[\widehat{\pi}(\boldsymbol{x}) \neq \pi^\star_T(\boldsymbol{x})\big] \leqslant \varepsilon$. To do so, the learner has access to a reference model $\pi^{\mathrm{ref}} : S \times \Sigma \to \Delta_S$ and an outcome verifier $\mathcal{V}$ (Definition 4.1). The learner is evaluated according to the following desiderata:*

1. *Sample complexity: The number of instances n_{sample} drawn from ρ.*
2. *Query complexity: The number of calls n_{query} made to the verifier $\mathcal{V}$.*
3. *Computational cost: The number of state transitions n_{comp} generated from π^{ref} or another model in Π over the course of learning.*

Remark 4.3. *The measure of computational cost we consider focuses on generation costs over the course of training, and ignores other sources of cost for the sake of simplicity. This is inspired by RLVR in practice where the dominant cost of training is autoregressive generation; model updates constitute a lower order cost (Hu et al., 2024).*

Reference model and coverage. We formalize the competence of the reference model π^{ref} via its *coverage*—the probability it assigns to a correct short solution—over blocks of size $B \ll T$.

Definition 4.4 (Block-level coverage)**.** *For a block length $B \in \mathbb{N}$ and constants $C^{(B)}_{\mathsf{seq}} \geqslant C^{(B)}_{\mathsf{ans}} \geqslant 1$, the reference model π^{ref} satisfies the following guarantees:*

$$\inf_{\boldsymbol{x}\in S\times\Sigma^B} \pi^{\mathrm{ref}}_{1:B}(\boldsymbol{y}^\star|\boldsymbol{x}) \geqslant \frac{1}{C^{(B)}_{\mathsf{seq}}}, \text{ and } \inf_{\boldsymbol{x}\in S\times\Sigma^B} \pi^{\mathrm{ref}}_{B}(y^\star_B|\boldsymbol{x}) \geqslant \frac{1}{C^{(B)}_{\mathsf{ans}}}. \tag{1}$$

where $\boldsymbol{y}^\star := \pi^\star_{1:B}(\boldsymbol{x})$. We refer to $C^{(B)}_{\mathsf{ans}}$ as the outcome coverage coefficient at length B, and $C^{(B)}_{\mathsf{seq}}$ as the sequence-level coverage coefficient at length B.

The two coefficients capture different notions of block-level success: $1/C^{(B)}_{\mathsf{ans}}$ lower bounds the probability that π^{ref} produces the correct terminal state of a length-B instance (potentially through an incorrect CoT), and $1/C^{(B)}_{\mathsf{seq}}$ lower bounds the probability that it produces the *entire* length-B CoT correctly. We always have $C^{(B)}_{\mathsf{ans}} \leqslant C^{(B)}_{\mathsf{seq}}$, but using both coefficients will allow us to state more refined guarantees.

Coverage assumptions of this kind are standard in the analysis of reinforcement learning and inference-time methods (Xie et al., 2022; Song et al., 2024; Huang et al., 2025b;a; Foster et al., 2025; Chen et al., 2025a), formalizing the idea that pretraining endows π^{ref} with a weak but nonzero probability of producing correct reasoning traces. While typically RLVR algorithms scale with coverage $C^{(T)}_{\mathsf{seq}}$ at the full sequence level T, our definition assumes competence at length $B \ll T$ only. Rolling π^{ref} out across all T/B blocks of a length-T instance can succeed with probability exponentially small in T/B, so naive RLVR at the target length is hopeless (indeed, extrapolating from short to long reasoning is a notoriously difficult form of out-of-distribution generalization (Anil et al., 2022; Zhou et al., 2024)). This raises the question: can we exploit composition to *expand* π^{ref}'s coverage from length B to length T, avoiding exponential blowup in cost?

4.1 Overview of Results for RLVR Setting

We show that self-generated curriculum makes *coverage expansion* possible. Our main algorithm, `AutoLearn.RL` (Algorithm 6), bootstraps π^{ref} from block-level competence to high accuracy at the full length T, with all costs governed by π^{ref}'s coverage at the short block scale B. `AutoLearn.RL` is a reinforcement learning counterpart to `AutoLearn`, following the same composition/boosting template but replacing the chain-of-thought oracle with verifier feedback.

Theorem 4.5 (Main result for RLVR setting (informal; see Theorem D.3))**.** `AutoLearn.RL` *(Algorithm 6) learns a model with* $\Pr_{\boldsymbol{x}\sim\rho}\left[\widehat{\pi}(\boldsymbol{x}) \neq \pi^{\star}_{T}(\boldsymbol{x})\right] \leqslant \varepsilon$ *with probability at least* $1-\delta$ *in the RLVR setting, assuming access to trajectory probabilities and the trajectory-set optimization oracle in* **??**. *The sample complexity to achieve this guarantee is bounded by* $n_{\mathsf{sample}} \leqslant \widetilde{\mathcal{O}}\left(\frac{d}{\varepsilon}\right)$, *and the query complexity (number of calls to the outcome verifier) is bounded by:*

$$n_{\mathsf{query}} \leqslant \widetilde{\mathcal{O}}\left(\frac{d}{\varepsilon}\right) + \widetilde{\mathcal{O}}\left(\frac{T}{B}\cdot dC^{(B)}_{\mathsf{ans}} + dC^{(B)}_{\mathsf{seq}}\right).$$

The computational cost (number of individual state generations from π^{ref} *or any model in* Π *over the course of training) is bounded by:*

$$n_{\mathsf{comp}} \leqslant \widetilde{\mathcal{O}}\left(\frac{dT}{\varepsilon}\right) + \widetilde{\mathcal{O}}\left(dTC^{(B)}_{\mathsf{ans}} + dBC^{(B)}_{\mathsf{seq}}\right).$$

Above, d *denotes the Natarajan dimension for* Π, *and* $\widetilde{\mathcal{O}}(\cdot)$ *hides logarithmic factors in* $|S|, C^{(B)}_{\mathsf{seq}}, \delta^{-1}, \varepsilon^{-1}$, *and* T *(including* B*).*

We interpret Theorem 4.5 as realizing a coverage expansion phenomenon concretely. Each bound splits into:

1. A leading term, $\widetilde{\mathcal{O}}(d/\varepsilon)$ for the sample and query complexity, and $\widetilde{\mathcal{O}}(dT/\varepsilon)$ for the computational cost, capturing the leading cost of learning to accuracy $1-\varepsilon$.
2. *Burn-in* terms that carry the polynomial dependence on the coverage for π^{ref}, yet are nearly-independent of ε. Crucially, the burn-in depends only on coverage at the *block* scale B: over training, the learned predictor reaches constant accuracy at length T, after which improvement proceeds as if π^{ref} had had constant length-T coverage all along.

In contrast, guarantees for RLVR fine-tuning applied directly at length T depend on π^{ref}'s coverage at that length, $C^{(T)}_{\mathsf{seq}}$.

`AutoLearn.RL` closely follows the `AutoLearn` template, decomposing each length-T instance into a curriculum of length-B sub-problems, which can then be solved through the RLVR primitive of Proposition D.2; as in our iSFT results, this leverages the fact that blocks of long instances can be interpreted as short instances in the semiautomaton setting. The same obstacles arise as in iSFT: short-range models must be accurate enough to survive composition, and the right distribution over sub-problems must be sampled.

AI Disclosure Statement

We did not use generative AI tools to prove the main results of this paper: the problem formulation and key technical ideas were developed by the authors. We, however, used generative AI tools in a final reviewing pass of the proofs of the paper, helping improve presentation, and in the creation of Tikz figures. We have reviewed all AI-assisted work involved in the creation of this manuscript and take full responsibility for the final content of this work, including all proofs contained therein.

Ethics Statement

The nature of our work is theoretical and does not involve human participants, personal data, or animal experiments. As a consequence, we do not identify any significant or direct ethical concerns specific to the research presented.

References

Dana Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 1987.

Dana Angluin and Dongqu Chen. Learning a random dfa from uniform strings and state information. In *International Conference on Algorithmic Learning Theory*, pp. 119–133. Springer, 2015.

Dana Angluin, Leonor Becerra-Bonache, Adrian Horia Dediu, and Lev Reyzin. Learning finite automata using label queries. In *International Conference on Algorithmic Learning Theory*, pp. 171–185. Springer, 2009.

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

David A Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pp. 1–5, 1986.

Leonard E Baum and Ted Petrie. Statistical inference for probabilistic functions of finite state markov chains. *The annals of mathematical statistics*, 37(6):1554–1563, 1966.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th International Conference on Machine Learning (ICML)*, 2009.

Adam Block and Yury Polyanskiy. The sample complexity of approximate rejection sampling with applications to smoothed online learning. In *The Thirty Sixth Annual Conference on Learning Theory*, pp. 228–273. PMLR, 2023.

Fan Chen, Audrey Huang, Noah Golowich, Sadhika Malladi, Adam Block, Jordan T Ash, Akshay Krishnamurthy, and Dylan J Foster. The coverage principle: How pre-training enables post-training. *arXiv preprint arXiv:2510.15020*, 2025a.

Xiaoyin Chen, Jiarui Lu, Minsu Kim, Dinghuai Zhang, Jian Tang, Alexandre Piché, Nicolas Gontier, Yoshua Bengio, and Ehsan Kamalloo. Self-evolving curriculum for llm reasoning. *arXiv preprint arXiv:2505.14970*, 2025b.

Amit Daniely, Sivan Sabato, Shai Ben-David, and Shai Shalev-Shwartz. Multiclass learnability and the erm principle. *Journal of Machine Learning Research*, 2015.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Sean Welleck, Peter West, Chandra Bhagavatula, Ronan Le Bras, et al. Faith and fate: Limits of transformers on compositionality. *Advances in neural information processing systems*, 36:70293–70332, 2023.

Funda Ergün, S Ravi Kumar, and Ronitt Rubinfeld. On learning bounded-width branching programs. In *Proceedings of the eighth annual conference on Computational learning theory*, pp. 361–368, 1995.

Dylan J Foster, Adam Block, and Dipendra Misra. Is behavior cloning all you need? understanding horizon in imitation learning. *Advances in Neural Information Processing Systems*, 2024.

Dylan J Foster, Zakaria Mhammedi, and Dhruv Rohatgi. Is a good foundation necessary for efficient reinforcement learning? the computational role of the base model in exploration. *arXiv preprint arXiv:2503.07453*, 2025.

Yoav Freund. Boosting a weak learning algorithm by majority. *Information and computation*, 121 (2):256–285, 1995.

Yoav Freund, Michael Kearns, Dana Ron, Ronitt Rubinfeld, Robert E Schapire, and Linda Sellie. Efficient learning of typical finite automata from random walks. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pp. 315–324, 1993.

Jason Gaitonde, Ankur Moitra, and Elchanan Mossel. Bypassing the noisy parity barrier: Learning higher-order markov random fields from dynamics. In *Proceedings of the 57th Annual ACM Symposium on Theory of Computing*, pp. 348–359, 2025.

George Giapitzakis, Kimon Fountoulakis, Eshaan Nichani, and Jason D Lee. On the statistical query complexity of learning semiautomata: a random walk approach. *arXiv preprint arXiv:2510.04115*, 2025.

Victor Mikhaylovich Glushkov. The abstract theory of automata. *Russian Mathematical Surveys*, 16(5):1–53, 1961.

Hermann Gruber, Jonathan Lee, and Jeffrey Shallit. Enumerating regular expressions and their languages. *arXiv preprint arXiv:1204.4982*, 2012.

Yanjun Han, Soham Jana, and Yihong Wu. Optimal prediction of markov chains with and without spectral gap. *IEEE Transactions on Information Theory*, 69(6):3920–3959, 2023.

John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to automata theory, languages, and computation*. Addison-Wesley, 2 edition, 2001.

Jian Hu, Xibin Wu, Zilin Zhu, Weixun Wang, Dehao Zhang, Yu Cao, et al. Openrlhf: An easy-to-use, scalable and high-performance rlhf framework. *arXiv preprint arXiv:2405.11143*, 2024.

Audrey Huang, Adam Block, Dylan Foster, Dhruv Rohatgi, Cyril Zhang, Max Simchowitz, Jordan Ash, and Akshay Krishnamurthy. Self-improvement in language models: The sharpening mechanism. In *International Conference on Learning Representations*, 2025a.

Audrey Huang, Adam Block, Qinghua Liu, Nan Jiang, Akshay Krishnamurthy, and Dylan J Foster. Is best-of-n the best of them? coverage, scaling, and optimality in inference-time alignment. *International Conference on Machine Learning (ICML)*, 2025b.

Nirmit Joshi, Gal Vardi, Adam Block, Surbhi Goel, Zhiyuan Li, Theodor Misiakiewicz, and Nathan Srebro. A theory of learning with autoregressive chain of thought. *arXiv:2503.07932*, 2025.

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

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1.5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*, 2025.

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

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

Nick Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine learning*, 2(4):285–318, 1988.

Gaurav Mahajan, Sham Kakade, Akshay Krishnamurthy, and Cyril Zhang. Learning hidden markov models using conditional samples. In *The Thirty Sixth Annual Conference on Learning Theory*, pp. 2014–2066. PMLR, 2023.

Sumeet Ramesh Motwani, Alesia Ivanova, Ziyang Cai, Philip Torr, Riashat Islam, Shital Shah, Christian Schroeder de Witt, and Charles London. h1: Bootstrapping llms to reason over longer horizons via reinforcement learning. *arXiv preprint arXiv:2510.07312*, 2025.

Balas K Natarajan. On learning sets and functions. *Machine Learning*, 4(1):67–97, 1989.

Chirag Pabbaraju. The optimal sample complexity of multiclass and list learning. *arXiv preprint arXiv:2604.24749*, 2026.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 5687–5711, 2023.

Nived Rajaraman, Audrey Huang, Miro Dudik, Robert Schapire, Dylan J Foster, and Akshay Krishnamurthy. Learning to reason with curriculum i: Provable benefits of autocurriculum. *arXiv preprint arXiv:2603.18325*, 2026.

Ronald L Rivest and Robert E Schapire. Inference of finite automata using homing sequences. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pp. 411–420, 1989.

David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. Analysing mathematical reasoning abilities of neural models. *arXiv preprint arXiv:1904.01557*, 2019.

Robert E Schapire and Yoav Freund. *Boosting: Foundations and algorithms*. MIT Press, 2013.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv:2402.03300*, 2024.

Taiwei Shi, Yiyang Wu, Linxin Song, Tianyi Zhou, and Jieyu Zhao. Efficient reinforcement finetuning via adaptive curriculum learning. *arXiv preprint arXiv:2504.05520*, 2025.

Yuda Song, Gokul Swamy, Aarti Singh, J Bagnell, and Wen Sun. The importance of online data: Understanding preference fine-tuning via coverage. *Advances in Neural Information Processing Systems*, 37:12243–12270, 2024.

John von Neumann. Various techniques used in connection with random digits. In *John von Neumann: Collected Works*, volume 5, pp. 768–770. Pergamon Press, 1963.

Tengyang Xie, Dylan J Foster, Yu Bai, Nan Jiang, and Sham M Kakade. The role of coverage in online reinforcement learning. *arXiv:2210.04157*, 2022.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Zhiyuan Zeng, Hamish Ivison, Yiping Wang, Lifan Yuan, Shuyue Stella Li, Zhuorui Ye, Siting Li, Jacqueline He, Runlong Zhou, Tong Chen, et al. Rlve: Scaling up reinforcement learning for language models with adaptive verifiable environments. *arXiv preprint arXiv:2511.07317*, 2025.

Andrew Zhao, Yiran Wu, Yang Yue, Tong Wu, Quentin Xu, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. Absolute zero: Reinforced self-play reasoning with zero data. *arXiv preprint arXiv:2505.03335*, 2025.

Haoyu Zhao, Simran Kaur, Dingli Yu, Anirudh Goyal, and Sanjeev Arora. Can models learn skill composition from examples? *Advances in Neural Information Processing Systems*, 37:102393–102427, 2024.

Yongchao Zhou, Uri Alon, Xinyun Chen, Xuezhi Wang, Rishabh Agarwal, and Denny Zhou. Transformers can achieve length generalization but not robustly. *arXiv preprint arXiv:2402.09371*, 2024.

Banghua Zhu, Michael Jordan, and Jiantao Jiao. Principled reinforcement learning with human feedback from pairwise or k-wise comparisons. In *International Conference on Machine Learning*, pp. 43037–43067. PMLR, 2023.

APPENDIX CONTENTS

A Additional Related Work **15**
A.1 Examples of semiautomata . . . 15
A.2 An Example of a Semiautomaton Induced by a Regular Expression . . . 16

B Discussion **17**
B.1 Simplifications in the Problem Formulation . . . 17
B.2 Optimality of T Dependency in Theorem 3.3 . . . 17

C iSFT: Learning from Interactive Chain-of-Thought Supervision **18**
C.1 Overview of `AutoLearn` . . . 18
C.2 Main Result: Compositional Curriculum Achieves Subpolynomial Query Complexity 19
C.3 Comparison with Non-Curriculum Baselines . . . 21
C.4 Applications of Theorem C.2 . . . 23

D RLVR: Fine-Tuning a Weak Reference Model **24**
D.1 Motivation: Naive RLVR Methods and the Need for Coverage . . . 24
D.2 Overview of `AutoLearn.RL` . . . 25
D.3 Main Result: Coverage Expansion via Compositional Curriculum . . . 27

E Overview of Analysis Techniques: Recursion and Inverted Sampling **29**
E.1 A Single Level of Recursion: Achieving $n_{\texttt{sample}} = \widetilde{\mathcal{O}}(d\sqrt{T})$ and $n_{\texttt{query}} = \widetilde{\mathcal{O}}(d\sqrt{T})$ 29
E.2 Improving Rejection Sampling: A Path to Sub-Polynomial Query Complexity . . . 30
E.3 Detour: Rejection Sampling from Reweighted Mixtures of Marginals . . . 31
E.4 Inverted Sampling: More Efficient Sampling from Reweighted Mixtures . . . 33
E.5 Inverted Sampling Achieves $n_{\texttt{sample}} = \widetilde{\mathcal{O}}(d)$ and $n_{\texttt{query}} = \widetilde{\mathcal{O}}(d\sqrt{T})$ . . . 34
E.6 $2^{\widetilde{\mathcal{O}}(\sqrt{\log(T)})}$ Query and Sample Complexity via Deeper Recursion . . . 35

F Analyzing a Simpler Variant of Algorithm 1: `AutoLearn.warmup` **36**
F.1 Proof Sketch of Theorem F.1 . . . 36
F.2 Proving Theorem F.1: relevant lemmata . . . 41
F.3 Proof of Theorem F.1 . . . 44

G Analysis of `AutoLearn` (Algorithm 2): Proof of Theorem C.2 **45**
G.1 Analysis of half-integer steps: Proof of Lemma G.3 . . . 48
G.2 Proof of Lemma G.4 . . . 51
G.3 Final Recursive Guarantee: Proof of Theorem G.1 . . . 55

H Analysis of `AutoLearn.RL` (Algorithm 6): Proof of Theorem D.3 **55**
H.1 Guess-and-Check: Simulating $\texttt{iCoT}(\cdot)$ via π^{ref} and $\mathcal{V}$ . . . 56
H.2 Analysis of the $h = \frac{1}{2}$ Level: Boosting with Composition . . . 56
H.3 Analysis of the $h = 1$ Level: Boosting to Error ε . . . 58
H.4 Proof of Theorem D.3 . . . 60

I Proofs of Lemmas **61**
I.1 Prediction Error Decomposition via Potentials: Proof of Lemma F.2 . . . 61
I.2 Sampling Approximately from λ_j: Proof of Lemma F.3 . . . 62
I.3 $\mathcal{K}_j$ is a High Probability Event: Proof of Lemma F.4 . . . 64
I.4 Truthful Aborts: Proof of Lemma F.5 . . . 65
I.5 Training Weak Models: Proof of Lemma F.6 . . . 65
I.6 Aborted Iterations Have Low Regression: Proof of Lemma F.7 . . . 66

J Proofs of Auxiliary Results **67**
J.1 Lower Bounds Against Full-CoT Learners: Proof of Proposition C.5 . . . 67

J.2 Bounding Prediction Error by Potential Φ_k: Proof of Lemma I.1 67
J.3 Recursion for Φ_j: Proof of Lemma I.2 68
J.4 Density Ratio Bound between $\nu_{\mathcal{K}}$ and $\widehat{\nu}$: Proof of Lemma I.5 69
J.5 Density Ratio Bound between $\nu_{\mathcal{K}_j}$ and $\widehat{\nu}_j$: Proof of Lemma F.3 70
J.6 Sampling Lower Bound: Proof of Lemma I.6 70
J.7 Improved T Dependency when Online Learning is Possible: Proof of Theorem B.4 71

Appendix Organization

Appendix A presents additional related work which studies the problem of learning automata; in Appendix A.2 we discuss a concrete example illustrating the semiautomaton induced by a regular expression. Appendix B discusses extensions to stochastic models and limitations of the current guarantees, including an online-learning approach with improved dependence on T. Appendix C presents the iSFT algorithm, its formal guarantees, and comparisons with non-curriculum baselines. Appendix D presents the RLVR algorithm, coverage assumptions, and formal guarantees. Appendix E overviews the analysis techniques, from the single-level warmup to inverted sampling and deeper recursion. Appendix F gives a self-contained treatment of `AutoLearn.warmup`, which is a simplified, single-composition version of `AutoLearn`. Appendix G lifts this analysis to the full recursive variant of `AutoLearn` and proves Theorem C.2, while Appendix H analyzes `AutoLearn.RL` and proves Theorem D.3. The remaining appendices collect the supporting technical material: Appendix I proves the main lemmas used in the analysis of `AutoLearn.warmup`, and Appendix J gives auxiliary results, including the non-curriculum lower bounds, the potential and density-ratio calculations, the sampling lower bound, and the discussion-section results on online learning and stochastic model classes.

A Additional Related Work

Semiautomata are closely related to *automata*, which have a rich history in theoretical computer science (Hopcroft et al., 2001). In learning theory specifically, automata have been the subject of a rich line of work, originating with Angluin's seminal $L^\star$ algorithm (Angluin, 1987; Rivest & Schapire, 1989; Freund et al., 1993; Kearns & Valiant, 1994; Ergün et al., 1995). The two models differ only in that automata define languages over words $\boldsymbol{w} \in \Sigma^*$ based on state evolution and an acceptance function, whereas semiautomata only define state evolution. This distinction is salient when considering the task of PAC learning: for automata, the learner only accesses the acceptance decisions of the underlying system, while we consider learning semiautomata from feedback that reveals the state evolution more directly.

Most of the literature on learning automata has focused on settings where the state is completely unobserved, and the learner must infer about it through accept/reject decisions. Most closely related to the setting considered in this paper are the works of Angluin et al. (2009) and Angluin & Chen (2015) who consider learning settings where terminal states are either fully or partially observable. The recent work of Giapitzakis et al. (2025) studies the statistical query complexity of PAC learning semiautomata from random examples, and establish strong SQ hardness for this task.

A.1 Examples of semiautomata

Beyond serving as a theoretical testbed, semiautomata capture a broad range of sequential computation tasks, including settings where the state space S and input alphabet Σ can be very large, necessitating function approximation. We discuss a few examples below.

1. *Modular arithmetic and group products.* Let $(G, \cdot)$ be a finite group with $|G| = k$. The corresponding semiautomaton has state space $S = G$, input alphabet $\Sigma = G$, and transition function $\pi(s, w) = s \cdot w$. For an input word $\boldsymbol{w}$, simulating the semiautomaton for T steps computes the group product $s_0 \cdot w_1 \cdot w_2 \cdots w_T$. For the case where $G = \mathbb{Z}/k\mathbb{Z}$ under addition, this reduces to computing modular sums. As a more interesting example, when $G = S_5$ (the symmetric group on 5 symbols), an input-dependent polynomial-length width-five branching-program encoding makes iterated group multiplication simulate any NC^1 circuit by Barrington's

theorem (Barrington, 1986). The class Π of all valid group multiplication tables on $[k]$ has Natarajan dimension at most $\mathrm{Ndim}(\Pi) \leqslant \mathcal{O}(k^2 \log k)$.

2. *Linear recurrences over finite fields.* Consider the semiautomaton with state space $S = \mathbb{F}_q^d$, input alphabet $\Sigma = \mathbb{F}_q^d$, and transition function $\pi(s, w) = As + Bw$ for unknown matrices $A \in \mathbb{F}_q^{d\times d}$ and $B \in \mathbb{F}_q^{d\times d}$. Simulating for T steps computes the linear recurrence $s_T = A^T s_0 + \sum_{t=1}^{T} A^{T-t} B w_t$. The class Π of all such linear transitions is parameterized by the pair (A, B), giving $|\Pi| = q^{2d^2}$, and Natarajan dimension $\mathrm{Ndim}(\Pi) \leqslant \mathcal{O}(d^2 \log q)$ which is exponentially smaller than the number of states, $|S| = q^d$.
3. *Regular languages.* A regular language over an alphabet Σ is the set of strings accepted by some Deterministic Finite Automaton (DFA), where input words are arbitrary strings in Σ^*. These correspond to strings that match a regular expression built from alphabet symbols, concatenation, union ($|$), and Kleene star (*). The *alphabetic length* k—specifically, a regular expression has length k if it is built from k alphabet symbols, counted with multiplicity[7]—is a natural complexity measure for regular languages. In particular, by Glushkov's construction (Glushkov, 1961) and the powerset constructions, a regular language with alphabetic length k can be decided by a DFA with at most 2^{k+1} states. In Appendix A.2, we give a concrete example of a regular expression and describe the induced semiautomaton.

A.2 An Example of a Semiautomaton Induced by a Regular Expression

In this section we expand upon the regular language example in Appendix A.1, by giving a detailed example of a regular expression with alphabetic length k and induced semiautomaton. Consider the regular expression $\mathtt{RegEx} \equiv (a|b)^*ab$ over $\Sigma = \{a, b\}$. This corresponds to the set of all strings which terminate in ab. The alphabetic length of $\mathtt{RegEx}$ is 4, with symbol occurrences a_1, b_2, a_3, b_4, where a_1 and b_2 correspond to occurrences in the subexpression $(a|b)^*$, while a_3 and b_4 correspond to occurrences in the final ab term. Each regular expression can be compiled into a non-deterministic finite automaton, as we discuss next.

Definition A.1 (Non-deterministic finite automaton)**.** *Fix an input vocabulary Σ and a finite state space Q. A nondeterministic finite automaton (NFA) is defined by a transition function $f : Q \times \Sigma \to 2^Q$, which defines a set of states reachable from the current state for each input symbol, and a subset of states $Q_{\mathsf{acc}} \subseteq Q$ known as the accepting states. Given an initial state $q_0 \in Q$ and an input sequence $\boldsymbol{w}_{1:T} \in \Sigma^T$, the NFA realizes the sequences, $(\boldsymbol{q}_t)_{t=0}^T$, where $\boldsymbol{q}_0 = \{q_0\}$ and $\boldsymbol{q}_t = \cup_{q\in\boldsymbol{q}_{t-1}} f(q, w_t)$ captures all states reachable from q_0 under the input sequence $\boldsymbol{w}_{1:t}$. The NFA accepts the input string $\boldsymbol{w}_{1:T}$ if $Q_{\mathsf{acc}} \cap \boldsymbol{q}_T \neq \emptyset$.*

Glushkov (1961) shows how to write down an automaton corresponding to a regular language by associating one state for each symbol occurrence, as well as an initial state 0. Namely, the state space of the Glushkov NFA is $Q = \{0, \cdots, 4\}$. The NFA is initialized at the state 0; when the first letter of an input word is processed, the subsequent state can be either $a_1 \equiv 1$, $b_2 \equiv 2$ or $a_3 \equiv 3$. The (set-valued) transition function of the corresponding NFA, π^{NFA}, is described by the following map:

$$\begin{aligned} \pi^{\mathsf{NFA}}(0, a) = \pi^{\mathsf{NFA}}(1, a) = \pi^{\mathsf{NFA}}(2, a) &= \{1, 3\} \\ \pi^{\mathsf{NFA}}(0, b) = \pi^{\mathsf{NFA}}(1, b) = \pi^{\mathsf{NFA}}(2, b) &= \{2\} \\ \pi^{\mathsf{NFA}}(3, b) &= \{4\} \\ \text{For all other } (s, w) \in Q \times \Sigma,\ \pi^{\mathsf{NFA}}(s, w) &= \emptyset \end{aligned}$$

An input word is accepted by π^{NFA} if there exists a computational path which terminates in the state 4. π^{NFA} can be translated into a DFA π^{DFA} using powerset determinization, which tracks the set of all possible states reachable by some computational path in the NFA. Thus, the states in π^{DFA} correspond to subsets $S \subseteq Q$. For example, on the input word $\boldsymbol{w} = (b, a, a, b)$, the subset-state trajectory is:

$$\{0\} \xrightarrow{b} \{2\} \xrightarrow{a} \{1, 3\} \xrightarrow{a} \{1, 3\} \xrightarrow{b} \{2, 4\}.$$

[7] Other related notions of length, such as the ordinary length and the reverse polish length, are upper bounded by a constant factor times the alphabetic length after removing redundant stars and empty subexpressions (Gruber et al., 2012).

π^{DFA} accepts the subset state as long as the final returned set contains any of the accepting state in π^{NFA}, i.e., 4. Thus (b, a, a, b) is accepted because its terminal subset is $\{2, 4\}$, which contains the accepting position 4. This is correct as RegEx recognizes the set of strings which end in ab.

B Discussion

This work asks how *curriculum* and *composition* let a learner solve problems far beyond the reach of its base capabilities—a mechanism widely credited for the success of modern reasoning models, yet one with little theoretical foundation. Adopting semiautomaton simulation as a testbed, our results show that composition combined with curriculum can achieve superpolynomial reductions in supervision and computational cost relative to non-curriculum baselines, in both supervised fine-tuning (SFT) and reinforcement learning with verifiable rewards (RLVR). We close by discussing the limitations of our results, followed by several concrete technical questions they leave open.

B.1 Simplifications in the Problem Formulation

While our work is motivated by language model reasoning, our problem setting makes several simplifications; relaxing these is an important direction for future work.

Non-Markovian models. Our results use the Markovian structure of semiautomata simulation task to decompose the problem into shorter sub-problems. However, our results also require that the *model we train* is Markovian, which rules out classes like Transformers that can attend to the full history. It is an interesting question to understand how to extend our results to settings in which the underlying transition function is still Markovian, but the model we train may not be.

Stochastic transition functions. Our main results apply for semiautomata transition functions, which are inherently deterministic. However, in many natural settings, including language model reasoning, it is natural to consider stochastic transition functions $\pi : S \times \Sigma \to \Delta_S$. This encompasses Markov chains, and other (controlled) stochastic processes, for which there is a rich history of work studying learnability (Baum & Petrie, 1966; Han et al., 2023; Gaitonde et al., 2025). A natural objective is to return a terminal-state distribution $\widehat{\pi} : (S \times \Sigma^T) \to \Delta_S$ which minimizes some divergence to the ground-truth terminal state distribution $\pi^\star_T$.[8] When the learning objective is the (forward) KL divergence, $\mathbb{E}_{\boldsymbol{x}\sim\rho}\big[D_{\mathrm{KL}}\big(\pi^\star_T(\cdot|\boldsymbol{x}) \,\big\|\, \widehat{\pi}(\cdot|\boldsymbol{x})\big)\big]$, the idea of composition is still functional, as suggested by the chain rule for KL divergences.

Proposition B.1. *Let $\widehat{\pi} : S \times \Sigma^\tau \to \Delta_S$ be a block kernel. For $\tau \mid T$ and $L = T/\tau$, let $\overline{\rho}$ be the uniform mixture of block instances whose initial states are drawn along the stochastic target trajectory. Then,*

$$\mathbb{E}_{\boldsymbol{x}\sim\rho}\big[D_{\mathrm{KL}}\big(\pi^\star_T(\cdot|\boldsymbol{x}) \,\big\|\, \widehat{\pi}^{\circ L}(\cdot|\boldsymbol{x})\big)\big] \leqslant L \cdot \mathbb{E}_{\boldsymbol{x}\sim\overline{\rho}}\big[D_{\mathrm{KL}}\big(\pi^\star_\tau(\cdot|\boldsymbol{x}) \,\big\|\, \widehat{\pi}(\cdot|\boldsymbol{x})\big)\big]$$

A coupling argument also gives the analogous bound with total variation in place of KL: run the two chains together until the first block discrepancy, and sum the one-block TV errors along the target chain. Extending the learning algorithm requires specifying how distributional prediction errors are estimated from feedback. State samples may suffice under additional assumptions and with additional sample complexity; exact target-probability access is one possible stronger observation model (Mahajan et al., 2023).

Our main results leave open several interesting technical questions, including tightening our sample complexity and computational guarantees, as well as further relaxing the coverage assumptions in our RLVR results. Below, we highlight a particularly interesting question: Improving the T dependence in our iSFT results.

B.2 Optimality of T Dependency in Theorem 3.3

Our main result in the SFT setting establishes query and sample complexity scaling $2^{\widetilde{\mathcal{O}}(\sqrt{\log T})}$ for learning semiautomata. Understanding whether there are statistical or computational barriers to improving this further to achieve *polylogarithmic* dependency on T appears to be a fascinating and

[8] we will use $\pi^\star_T(\cdot|\boldsymbol{x})$ to denote the marginal distribution of the terminal state generated by $\pi^\star$ on the instance $\boldsymbol{x}$.

deep question. $\log(T)$ query complexity is a natural endpoint for divide-and-conquer approaches like AutoLearn, but our current results fall short due to additional complications around error accumulation. Below we show a special case where $\mathrm{polylog}(T)$ is indeed achievable under stronger assumptions on the class Π than i.i.d. learnability: namely that it admits a finite mistake bound. We define the relevant notation below.

Definition B.2 (Mistake-bounded online learning for Π (Littlestone, 1988))**.** *Consider the following online learning game between an algorithm* $\mathtt{Alg}_{\mathrm{on}}$ *and an adversary. The adversary fixes a transition function* $\pi^\star \in \Pi$*, and at each step* $i = 1, \dots, N$*: (1) the adversary adaptively selects a state-letter pair* $(s^i, w^i) \in S \times \Sigma$ *(possibly depending on all prior predictions of* $\mathtt{Alg}_{\mathrm{on}}$*), (2) the algorithm predicts* $\widehat{s}^i_+ \in S$ *based on all prior observations, and (3) the true next-state* $\pi^\star(s^i, w^i)$ *is revealed. The worst-case number of mistakes of* $\mathtt{Alg}_{\mathrm{on}}$ *is defined as,*

$$M_N(\mathtt{Alg}_{\mathrm{on}}) = \sup_{\pi^\star \in \Pi} \sup \sum_{i=1}^{N} \mathbb{I}\big(\widehat{s}^i_+ \neq \pi^\star(s^i, w^i)\big)$$

where the inner supremum is over all adaptive adversary strategies. Π *admits a finite mistake bound if there exists an online learning algorithm* $\mathtt{Alg}_{\mathrm{on}}$ *such that* $\sup_{N \geqslant 1} M_N(\mathtt{Alg}_{\mathrm{on}}) \triangleq M^\star(\mathtt{Alg}_{\mathrm{on}}) < \infty$.

Remark B.3. *A finite mistake bound for* Π *is a stronger assumption than i.i.d. learnability. When* $S = \{0, 1\}$ *and* Σ *is arbitrary, i.i.d. learnability is equivalent to the finiteness of the VC dimension of* Π*. A finite mistake bound is equivalent to the finiteness of the Littlestone dimension,* $\mathrm{Ldim}(\Pi)$ *(Littlestone, 1988), which is no smaller than the VC dimension. For simple classes such as thresholds,* $\mathrm{VC}(\Pi) = 1$*, while* $\mathrm{Ldim}(\Pi)$ *can scale with the size of the domain of the functions.*

Assuming that Π admits a finite mistake bound, we can improve the sample and query complexity of learning semiautomata to achieve logarithmic dependence on T.

Theorem B.4. *Let* $\mathtt{Alg}_{\mathrm{on}}$ *denote an online learning algorithm with mistake bound* $M^\star(\mathtt{Alg}_{\mathrm{on}}) < \infty$*. Consider any* $\varepsilon \in (0, 1)$*. There exists an algorithm (Algorithm 9) which uses* $\mathtt{Alg}_{\mathrm{on}}$ *as a subroutine; the randomized model* $\widehat{\pi}$ *returned by this algorithm satisfies,*

$$\mathbb{E}_D \mathbb{E}_{\boldsymbol{x} \sim \rho, y_T \sim \widehat{\pi}(\cdot|\boldsymbol{x})} \big[\mathbb{I}(y_T \neq \pi^\star_T(\boldsymbol{x}))\big] \leqslant \varepsilon,$$

as long as $n_{\mathtt{sample}} \geqslant C\big(M^\star(\mathtt{Alg}_{\mathrm{on}}) \cdot \varepsilon^{-1}\big)$*. The query complexity is* $n_{\mathtt{query}} \leqslant O\big(n_{\mathtt{sample}} \log(T+1)\big)$.

The outer expectation includes the training sample and training randomness; the output draws one iterate uniformly and uses it for the entire test trajectory. For finite S, Σ*, a version-space majority algorithm makes at most* $\log_2 |\Pi| = O(d \log(e|S||\Sigma|))$ *mistakes by the counting argument in Lemma J.2, where* $d = \mathrm{Ndim}(\Pi)$*. This gives* $O(d \log(e|S||\Sigma|)/\varepsilon)$ *samples and at most an additional* $O(\log(T+1))$ *factor in queries.*

The proof of this result is given in Appendix J.7.

C iSFT: Learning from Interactive Chain-of-Thought Supervision

This section presents our algorithms and main guarantees for learning semiautomata in the iSFT setting (Problem 3.2). Earlier, in Section 3.2, we motivated our main algorithm, AutoLearn, by highlighting the compositional structure of our semiautomaton learning setup. We formally describe AutoLearn in Appendix C.1 and present our main guarantee in Appendix C.2. Finally, in Appendix C.3 we compare with guarantees achieved by natural non-curriculum baselines.

C.1 Overview of AutoLearn

We now formally introduce AutoLearn (Algorithm 2), which instantiates the multiscale boosting template above with $2H + 1$ levels where L is a decomposition factor and $H = \log_L(T)$. The algorithm is visualized in Fig. 2a. We present the recursion in terms of half-integer levels $h = H + \frac{1}{2}, H, H - \frac{1}{2}, \dots, 0$, where the base level $h = 0$ corresponds to length-1 instances, and larger h corresponds to longer instances. AutoLearn is parameterized by two schedules:

- *Decomposition schedule* (L_h): determines how many sub-instances to break instances (of a certain length) into at each level of recursion; the corresponding model trained in each level is composed L_h-fold to induce a model solving longer instances.
- *Branching schedule* (k_h): determines how many weak models to aggregate at each level of recursion.

The half-step levels delineate two types of recursion steps. At integer levels h, $\texttt{AutoLearn}_h$ implements both composition and boosting, setting $L_h = 2^{\Theta(\sqrt{\log(T)})}$ and $k_h = \widetilde{\mathcal{O}}(1)$. At half-integer depths h, $\texttt{AutoLearn}_h$ implements only boosting, not composition, setting $L_h = 1$ and $k_h = \widetilde{\mathcal{O}}(1)$.

Splitting and inverted sampling. At integer depths $h \in \mathbb{N}$ where composition occurs, length-τ_h instances are broken down into length-τ_{h-1} sub-instances via subroutines `Split` and `InvSampling` (Algorithm 4 and Algorithm 3). A key structural property is that each length-τ_h instance contributes at most one length-τ_{h-1} sub-instance, preserving independence; this is visualized in Fig. 2b, and ensures that length-1 instances (the base case in the recursion), can be solved using off-the-shelf methods for classification/next-token prediction. The subroutine `InvSampling` addresses a central technical issue we have not yet discussed—*distribution shift*—which complicates the process of generating examples from the appropriate distribution for boosting; we defer a dedicated discussion of the motivation and design of this component to Appendix E.

Base learner and next-token prediction. Crucially, all of the model training in `AutoLearn` takes place at the bottom-level of the recursion, where $h = 0$. By this level, instances are reduced to a collection of next-state examples $(s, w) \mapsto \pi^\star(s, w)$. To fit a model $\widehat{\pi}$ in Π, `AutoLearn` makes use of a generic "base learner" `Base`, representing next-token prediction or another off-the-shelf algorithm. We can interpret $\texttt{AutoLearn}_h$ for $h \geqslant \frac{1}{2}$ as adaptively constructing a curriculum of shorter instances on which to fit models using `Base`, concentrating on instances where the current model fails and decomposing them into sub-instances where necessary.

Our main guarantee for `AutoLearn` is stated in terms of the sample complexity required to fit a next-token predictor model to constant accuracy using `Base`, under an arbitrary but fixed distribution over instances.

Definition C.1 (Weak learning Π from i.i.d. next-state examples)**.** *Let* $D = \{(s_i, w_i) \mapsto \pi^\star(s_i, w_i)\}_{i=1}^n$ *where* $(s_i, w_i) \overset{\text{i.i.d.}}{\sim} \sigma \in \Delta_{S\times\Sigma}$ *denotes a dataset of i.i.d. next-state examples. A weak learner for* Π *is any algorithm* $\texttt{Base}(D \,\|\, \delta')$ *parameterized by a failure probability* $\delta' \in (0, 1)$*, which learns a model for* $\pi^\star$ *to constant accuracy from* D *with* sample complexity $n_{\texttt{weak}}(\delta')$*. Namely, for any* σ *and* $\pi^\star \in \Pi$*, as long as* $n \geqslant n_{\texttt{weak}}(\delta')$ *the model* $\widehat{\pi} : S \times \Sigma \to S$ *returned by* `Base` *satisfies w.p. at least* $1 - \delta'$*,*

$$\Pr_{(s,w)\sim\sigma}\left[\widehat{\pi}(s, w) \neq \pi^\star(s, w)\right] \leqslant \frac{1}{4}.$$

Furthermore, we say that `Base` *admits a generic weak-learning guarantee if* $n_{\texttt{weak}}(\delta') \leqslant \texttt{comp}(\Pi)\log(e/\delta')$*, where* $\texttt{comp}(\Pi) \geqslant 1$ *is some notion of complexity of the class* Π*.*

Definition C.1 posits that the class Π is learnable from *i.i.d.* next-state examples to constant accuracy. In the definition above, we demarcate the "generic" weak-learning guarantee to capture typical scaling behavior of the sample complexity with respect to the failure probability δ'. For instance, when `Base` is instantiated as empirical risk minimization (ERM) under the 0-1 loss, we obtain a generic weak-learning guarantee with $\texttt{comp}(\Pi) = d\log(|S|)$ where d is the Natarajan dimension of Π (Definition 2.3) (Natarajan, 1989).

C.2 Main Result: Compositional Curriculum Achieves Subpolynomial Query Complexity

Our main guarantee for `AutoLearn` shows that it achieves subpolynomial-in-T sample and query complexity when invoked with off-the-shelf base learners such as next-token prediction.

Parameter conventions. All datasets and model ensembles are indexed multisets. Integer budgets are rounded up (and enlarged to permit equal epoch sizes); logarithms in complexity bounds

Algorithm 1 $\texttt{AutoLearn}_h\big(D_{\text{input}} \,\|\, \tau_h, \delta\big)$

Learning semiautomata via multiscale boosting

Input: Class of semiautomaton transitions Π over state space S and alphabet Σ,
Current depth $h \in \{0, \frac{1}{2}, 1, \frac{3}{2}, \cdots\}$,
Current length of instances $\tau_h \in \mathbb{N}$,
Target failure probability δ,
Dataset of instances $D_{\text{input}} = \{\boldsymbol{x}^i = (s_0^i, \boldsymbol{w}^i)\}_{i=1}^n$.
Learner alg. $\texttt{Base}(\cdot \,\|\, \delta')$ for Π from i.i.d. next-state data (Definition C.1).
▶ $\texttt{Base}(\cdot \,\|\, \delta')$ has sample complexity $n_{\texttt{weak}}(\delta')$ parameterized by failure probability.

Hyperparameters: Decomposition schedule, $(L_h : h \in \{\frac{1}{2}, 1, \frac{3}{2}, \cdots\})$ where $\tau_h \pmod{L_h} \equiv 0$,
Branching schedule, $(k_h : h \in \{\frac{1}{2}, 1, \frac{3}{2}, \cdots\})$.

Instantiate: $\tau_{h-\frac{1}{2}} = \tau_h / L_h$ **if** $h \geqslant \frac{1}{2}$.

if $|D_{\text{input}}| < n^\star_{\texttt{sample}}(h, \delta)$, return a fixed predictor; otherwise truncate D_{input} to the first $n^\star_{\texttt{sample}}(h, \delta)$ instances. ▶ $n^\star_{\texttt{sample}}$ is defined in Eq. (34).

if $h = 0$**:** label the one-step instances using $\texttt{iCoT}$, and return $\texttt{Base}(D_{\text{labeled}} \,\|\, \delta)$.

Split D_{input} into k_h equal parts, $\{D^j_{\text{input}} : j \in \{0, \cdots, k_h - 1\}\}$.

for $j = 0, 1, \cdots, k_h - 1$ **do**

  Let $\Pi_j \leftarrow \{\widehat{\pi}^0, \cdots, \widehat{\pi}^{j-1}\}$ and $D^j_{\text{out}} \leftarrow \emptyset$. ▶ By convention, $\Pi_0 = \emptyset$.

  for instances $\boldsymbol{x} = (s_0, \boldsymbol{w}) \in D^j_{\text{input}}$ **do**

    $D_{\boldsymbol{x}} \leftarrow \texttt{Split}(\boldsymbol{x} \| L_h)$. ▶ $\boldsymbol{x}$ is split into L_h shorter instances which are labeled by their terminal states.

    $O \leftarrow \texttt{InvSampling}\big(D_{\boldsymbol{x}} \| \Pi_j, k_h\big)$.

    if $O \neq \perp$, **then** update the dataset $D^j_{\text{out}} \leftarrow D^j_{\text{out}} \cup \{O\}$.

  Train model $\widehat{\pi}^j \leftarrow \texttt{AutoLearn}_{h-\frac{1}{2}}\big(D^j_{\text{out}} \,\|\, \tau_{h-\frac{1}{2}}, \delta/2k_h\big)$.

Return: $\widehat{\pi}^{\circ L_h}$, where $\widehat{\pi} = \texttt{Plu}\big(\{\widehat{\pi}^j : j \in \{0, \cdots, k_h - 1\}\}\big)$.

and parameter schedules are clipped below at one. Sample complexity functions are taken nonincreasing in the allowed failure probability. A call with too few examples for its prescribed budget returns a fixed predictor; zero acceptance probability means that all examples are rejected. A tilted law with zero normalizing mass is assigned arbitrarily, since its weighted error contribution is zero. These conventions also apply to the base case. For the final boosting step, let

$$k_\varepsilon = \min\{k \in \mathbb{N} : k \geqslant k_{\min},\ 3\sqrt{k}e^{-ck} \leqslant \varepsilon\},$$

where c is from Lemma F.9 and $k_{\min} \geqslant \max\{1, 1/c\}$ is fixed. Thus $k_\varepsilon = O(\log(e/\varepsilon))$ and $k_\varepsilon e^{ck_\varepsilon} = O(\varepsilon^{-1}\log^{3/2}(e/\varepsilon))$.

Theorem C.2 (Main guarantee for $\texttt{AutoLearn}$)**.** *For any $\delta' \in (0,1)$, let $\texttt{Base}(\cdot \,\|\, \delta')$ be any weak learner for Π from i.i.d. next-state data, as defined in Definition C.1. Suppose $\texttt{Base}$ admits a generic weak-learning guarantee satisfying $n_{\texttt{weak}}(\delta') \leqslant \texttt{comp}(\Pi)\log(e/\delta')$.*

Let $\varepsilon, \delta \in (0,1)$ and suppose $H = \sqrt{\log_2(T)} \in \mathbb{N}$. Suppose $\texttt{AutoLearn}(\cdot \,\|\, T, \varepsilon, \delta)$ (Algorithm 2) is invoked with a base learner $\texttt{Base}(\cdot \,\|\, \delta')$ with weak-learning sample complexity $n_{\texttt{weak}}(\delta') \leqslant \texttt{comp}(\Pi)\log(e/\delta')$. Then $\texttt{AutoLearn}$ learns a model $\widehat{\pi} : S \times \Sigma^T \to S$ with $\Pr_{\boldsymbol{x}\sim\rho}\big[\widehat{\pi}(\boldsymbol{x}) \neq \pi^\star_T(\boldsymbol{x})\big] \leqslant \varepsilon$ with probability at least $1 - \delta$, using sample and query complexity upper bounded by:

$$n_{\texttt{sample}}, n_{\texttt{query}} \leqslant 2^{\tilde{\mathcal{O}}\left(\sqrt{\log(T)}\right)} \cdot \left[\frac{\texttt{comp}(\Pi)\log^2(1/\varepsilon)\log(1/\delta)}{\varepsilon}\right].$$

The main proof is provided in Appendix G.3. Relevant lemmas are sketched in Appendix G. Theorem 3.3 follows from this result by instantiating $\texttt{Base}$ as empirical risk minimization (ERM) on the next-token prediction objective (Definition 2.2), and noting that $\texttt{comp}(\Pi)$ can be chosen as $d\log(|S|)$, where d is the Natarajan dimension. With more sophisticated choices for $\texttt{Base}$, such as the recently proposed algorithm of Pabbaraju (2026), we can achieve improved sample complexity guarantees such as $\texttt{comp}(\Pi) = d_{\texttt{DS}}$, where $d_{\texttt{DS}}$ is the DS dimension (Daniely et al., 2015).

Algorithm 2 `AutoLearn`$\big(D_{\text{input}} \parallel T, \varepsilon, \delta\big)$

High-accuracy extension of Algorithm 1

Input: Class of semiautomaton transitions Π over state space S and alphabet Σ,
Target length of instances T,
Target error ε and failure probability δ.
Dataset of instances $D_{\text{input}} = \{\boldsymbol{x}^i = (s_0^i, \boldsymbol{w}^i)\}_{i=1}^n \overset{\text{i.i.d.}}{\sim} \rho$.
Learner alg. `Base`$(\cdot \parallel \delta')$ for Π from i.i.d. next-state data (Definition C.1).

Instantiate: $H \leftarrow \sqrt{\log_2(T)}$

Return: $\widehat{\pi} \leftarrow \texttt{AutoLearn}_{H+\frac{1}{2}}\big(D_{\text{input}} \parallel T, \delta\big)$, ▶ $\texttt{AutoLearn}_h$ defined in Algorithm 1.

decomposition schedule: $L_h = L \triangleq 2^{\sqrt{\log_2(T)}}$ for $h \in \mathbb{Z}$, else $L_h = 1$, and,

branching schedule: $k_h = C\log(L)$ for $h \in \mathbb{Z}$,
$k_{h+\frac{1}{2}} = C\log\log(T)$ for $h \in \{0, 1, \cdots, H-1\}$.
and $k_{H+\frac{1}{2}} = k_\varepsilon$ for large $C > 0$.

Our most general guarantees for `AutoLearn` (given in Theorem G.1) are directly stated in terms of $n_{\texttt{weak}}(\cdot)$ (and do not assume a generic weak learning guarantee for `Base`).

In Appendix F we analyze a simpler variant of `AutoLearn` which considers a single round of boosting + aggregation to understand the core mechanism.

Remark C.3 (On the technical condition on length T)**.** *The schedule in Algorithm 2 requires the condition $\sqrt{\log_2(T)} \in \mathbb{N}$. The guarantees can be extended to any value of $T \in \mathbb{N}$ by replacing T by T', where T' is the smallest value larger than T such that there exists $H \in \mathbb{N}$ satisfying $T' = (L')^H$, where $L' = 2^{\sqrt{\log_2(T')}}$,*[9] *and changing the class Π to Π' as follows: the state space S is changed to $S' = S \times \{0, \ldots, T\}$ enabling the "timestep" to be tracked, and, each $\pi \in \Pi$ is changed to $\pi' \in \Pi'$ as,*

$$\text{For } s' = (s,t) \in S \times \{0,\ldots,T\}, \quad \pi'(s', w) = \begin{cases} (\pi(s,w), t+1) & \text{if } t < T, \\ (s,t) & \text{if } t = T, \end{cases}$$

Initialize the clock at 0 and append arbitrary alphabet symbols to length T. An oracle query of length ℓ from clock t is simulated using at most one original query of length $\min\{\ell, T-t\}$, followed by the known frozen transitions. Changing Π to Π' results in the state space growing by a factor of $T+1$ in size. However, this transformation itself does not affect the learnability of Π, since an algorithm for learning an unknown $\pi^\star \in \Pi$ from i.i.d. next-state data also induces a model for learning the corresponding $(\pi')^\star \in \Pi'$ from i.i.d. next-state data achieving the same error guarantee, and vice versa.

C.3 Comparison with Non-Curriculum Baselines

To interpret Theorem C.2, we contrast the dependence on the sequence length-T achieved by `AutoLearn` with that of two natural non-curriculum baselines: SFT on full sequences and end-to-end learning from final-state supervision (Joshi et al., 2025).[10]

C.3.1 Vanilla SFT: Learning from Full Chains-of-Thought

Arguably the most natural approach to learning semiautomata is to use supervised fine-tuning (SFT) on full chains-of-thought. Concretely, in our framework, this corresponds to querying the $\texttt{iCoT}(\cdot)$ oracle T times on each instance in the training dataset to label the full state sequence $\pi^\star_{1:T}(\boldsymbol{x}^i)$. Such a labeled dataset is composed of many next-state examples of the form $(s, w) \mapsto \pi^\star(s, w)$, and a model $\widehat{\pi}$ can be fit to this dataset via empirical risk minimization on the next-token prediction

[9] It suffices to choose $T' = 2^{m^2}$ for some $m \in \mathbb{N}$. Such a T' always exists in $[T, 2T^2]$.

[10] We remark that these baselines are more general than `AutoLearn`, in the sense that they can be applied to general autoregressive models, not just semiautomata.

Algorithm 3 $\texttt{InvSampling}(D_{\boldsymbol{x}} \| \Pi_j, k)$

Subsample an instance from D_x based on how many models in Π_j correctly label the instance.

Input: Dataset $D_{\boldsymbol{x}} = \{(\boldsymbol{x}^i, s^i) \text{ for } \boldsymbol{x}^i = (s_0^i, \boldsymbol{w}^i)\}_{i=0}^{L'-1}$ where $L' \in \{1, L\}$,
Indexed ensemble of j block predictors Π_j.
Instantiate: $\texttt{err}_\star \leftarrow \frac{1}{4}, \mathcal{I} \leftarrow \emptyset$.
for $i = 0, 1, \cdots, L' - 1$ **do**
  Draw $\eta_i \sim \mathrm{Unif}([0, 1])$.
  if $\eta_i \leqslant w_j(\boldsymbol{x}^i)/\alpha_{\max}^{j,k}$ **then** ▶ Defined below across Eqs. (2) and (3). Computing $w_j(\boldsymbol{x}^i)$ uses knowledge of $s^i = \pi_\tau^\star(\boldsymbol{x}^i)$ where $\tau = |\boldsymbol{x}^i|$.
    $\mathcal{I} \leftarrow \mathcal{I} \cup \{i\}$
Return: $\boldsymbol{x}^I$ for $I \sim \mathrm{Unif}(\mathcal{I})$ if $\mathcal{I} \neq \emptyset$. Else return $\perp$.

For $0 \leqslant r \leqslant j < k$, the weight w_j is defined as $w_j(\boldsymbol{x}) = \alpha^{j,k}_{\mathrm{rank}_j(\boldsymbol{x})}$ and $\alpha^{j,k}_{\max} := \max_{0 \leqslant r \leqslant j} \alpha_r^{j,k}$, where,

$$\alpha_r^{j,k} = \beta_r^{j+1,k} - \beta_{r+1}^{j+1,k}, \text{ where, } \beta_r^{j,k} = \begin{cases} \mathbb{I}(r \leqslant k/2) & \text{if } j = k \\ \texttt{err}_\star \cdot \beta_r^{j+1,k} + (1 - \texttt{err}_\star) \cdot \beta_{r+1}^{j+1,k} & \text{if } j < k \end{cases} \tag{2}$$

$\mathrm{rank}_j(\boldsymbol{x})$ counts the number of models in Π_j which guess the terminal-state label on $\boldsymbol{x}$ correctly: for $j \geqslant 0$,

$$\mathrm{rank}_j(\boldsymbol{x}) \leftarrow \sum_{\ell=0}^{j-1} \mathbb{I}\big(\widehat{\pi}^\ell(\boldsymbol{x}) = \pi_\tau^\star(\boldsymbol{x})\big) \in [0, j] \tag{3}$$

where $\tau = |\boldsymbol{x}|$.

Algorithm 4 $\texttt{Split}(\boldsymbol{x} \| L')$ where $\boldsymbol{x} = (s_0, \boldsymbol{w}_{1:t})$

Split a length-t instance into L' instances of length t/L' by querying $\texttt{iCoT}(\cdot)$ at boundary states.

Input: Instance $\boldsymbol{x} = (s_0, \boldsymbol{w}_{1:t})$ with $t \pmod{L'} \equiv 0$, supervision oracle $\texttt{iCoT}(\cdot)$.
Let s_0 be the input state and $\tau \leftarrow t/L'$.
for $i = 0, \cdots, L' - 1$ **do**
  $s_{\tau(i+1)} \leftarrow \texttt{iCoT}\big((s_{\tau i}, \boldsymbol{w}_{\tau i+1:\tau(i+1)})\big)$. ▶ L' calls to the $\texttt{iCoT}(\cdot)$ oracle in total
Return: $D_{\boldsymbol{x}} = \big\{(\boldsymbol{x}^i, s_{\tau(i+1)}) \text{ where } \boldsymbol{x}^i = (s_{\tau i}, \boldsymbol{w}_{\tau i+1:\tau(i+1)}) : i = 0, \cdots, L' - 1\big\}$

objective:

$$\widehat{\pi} \in \arg\min_{\pi \in \Pi} \sum_{i=1}^{n} \sum_{t=1}^{T} \mathbb{I}\big(\pi(s_{t-1}^i, w_t^i) \neq \pi^\star(s_{t-1}^i, w_t^i)\big).$$

where $\boldsymbol{x}^i = (s_0^i, \boldsymbol{w}_{1:T}^i)$ and $s_t^i = \pi_t^\star(\boldsymbol{x}^i)$ for all $i \in [n]$ and $t \in [T]$, given an input dataset of n instances $\{\boldsymbol{x}^i\}_{i=1}^n$. This is an instance of a more general class of *full-CoT* learners we define as follows.

Definition C.4 (Full CoT learner)**.** *Given a dataset of instances $D = \{\boldsymbol{x}^i\}_{i=1}^n \overset{\text{i.i.d.}}{\sim} \rho$, a full CoT learning algorithm $\texttt{Alg}(D)$ chooses a subset of instances indexed $S \subseteq [n]$ and queries $\texttt{iCoT}(\cdot)$ T times on each instance $\boldsymbol{x}^i$ for $i \in S$ to observe the full state sequence $\pi_{1:T}^\star(\boldsymbol{x}^i)$. The final model returned is trained on the CoT labeled dataset—for example, by using next-token prediction.*

As full CoT learners commit to querying full state sequences on every instance (even when potentially unnecessary), we show that any learner in this class requires linear-in-T query complexity; the proof of this result is given in Appendix J.1.

Proposition C.5. *Consider any full CoT learner $\texttt{Alg}$ (cf. Definition C.4). For every sufficiently large d, there exists a class of next-state/token predictors Π with Natarajan dimension d and a target distribution ρ such that for $\texttt{Alg}$ to return a model $\widehat{\pi}$ such that $\Pr_{\boldsymbol{x} \sim \rho}\big[\widehat{\pi}(\boldsymbol{x}) \neq \pi_T^\star(\boldsymbol{x})\big] \leqslant \frac{1}{4}$ with probability at least $\frac{1}{2}$, its sample complexity must satisfy $n_{\texttt{sample}} \geqslant cd$ for a universal constant*

$c > 0$. *Furthermore, this implies that for any such approach, the query complexity must satisfy* $n_{\texttt{query}} \geqslant cdT$ *to achieve the same guarantee.*

Rajaraman et al. (2026) showed that for the goal of achieving high accuracy $(1-\varepsilon)$, it is possible to improve over vanilla SFT by adaptively choosing which instances to gather full CoTs for. This leads to query complexity $\widetilde{\mathcal{O}}(dT/\varepsilon)$ to $\widetilde{\mathcal{O}}\big(d/\varepsilon + dT\big)$, pushing the dependency on T into a burn-in term. Ultimately though, each CoT requires making $\Omega(T)$ queries to the $\texttt{iCoT}(\cdot)$ oracle, so the query complexity is still linear in T.

C.3.2 End-to-End Feedback

Since the final objective in our setting is to accurately predict the final state $\pi_T^\star(\boldsymbol{x}^i)$, another natural learning approach is to ignore the intermediate CoT altogether, and label each instance in the training dataset only with the final state $\pi_T^\star(\boldsymbol{x}^i)$. One can then train a model which directly classifies the final state based on the instance. We formally define this class of *end-to-end* learners as follows.

Definition C.6 (End-to-end learner)**.** *Given a dataset of instances* $D = \{\boldsymbol{x}^i\}_{i=1}^n \overset{\text{i.i.d.}}{\sim} \rho$*, an end-to-end learning algorithm* $\texttt{Alg}(D)$ *queries* $\texttt{iCoT}(\cdot)$ *once per instance to label the terminal state* $\pi_T^\star(\boldsymbol{x}^i)$*. A model is then trained on this dataset. For example, by finding a model* $\widehat{\pi}$ *consistent with all terminal labels—that is,* $\widehat{\pi}_T(\boldsymbol{x}^i) = \pi_T^\star(\boldsymbol{x}^i)$ *for all* $i \in [n]$*—and predicting the terminal state on new instances using* $\widehat{\pi}_T$*. This example corresponds to learning a model by empirical risk minimizer over the induced "end-to-end" class* Π_T *defined as* $\Pi_T = \{\pi_T : \pi \in \Pi\}$ *(Joshi et al., 2025).*

Proposition C.7 (Corollary of Theorem E.1 of Joshi et al. (2025))**.** *Fix* $d, T \geqslant 1$ *and consider any end-to-end learning algorithm* $\texttt{Alg}$ *(cf. Definition C.6). For any* $\varepsilon, \delta \in (0, 1/8)$*, there exist a class of next-state/token predictors* Π *with Natarajan dimension* d*, a target transition* $\pi^\star \in \Pi$*, and a target distribution* ρ *over length-*T *instances, such that for* $\texttt{Alg}$ *to return a model* $\widehat{\pi}$ *such that* $\Pr_{\boldsymbol{x}\sim\rho}\big[\widehat{\pi}(\boldsymbol{x}) \neq \pi_T^\star(\boldsymbol{x})\big] \leqslant \varepsilon$ *with probability at least* $1-\delta$*,* $\texttt{Alg}$ *must incur sample complexity* $n_{\texttt{sample}} \geqslant \Omega\big(\big(dT + \log(1/\delta)\big) \cdot \varepsilon^{-1}\big)$.

Proof. The binary autoregressive class $\mathcal{G}$ in Joshi et al. (2025, Theorem E.1) has VC dimension d, while its T-step terminal-output class shatters dT strings of length $m = \lceil \log_2(dT) \rceil + 1$. Encode prefixes as states: take $\Sigma = \{a\}$ and $S = \{q_u : u \in \{0,1\}^\ell,\ m \leqslant \ell \leqslant m+T-1\} \cup \{\top_0, \top_1\}$. For each $g \in \mathcal{G}$, define

$$\pi_g(q_u, a) = \begin{cases} q_{u\,g(u)}, & |u| < m+T-1, \\ \top_{g(u)}, & |u| = m+T-1, \end{cases} \qquad \pi_g(\top_b, a) = \top_b.$$

This is a stationary Markovian transition. Starting from q_x with $|x| = m$, exactly T steps return only the final autoregressive bit, encoded by $\top_0$ or $\top_1$; the terminal label does not reveal the preceding prefix. At each prefix state, the two possible successors are a fixed recoding of $g(u)$, so Natarajan shattering by $\{\pi_g : g \in \mathcal{G}\}$ implies VC shattering by $\mathcal{G}$. Conversely, the cited construction shatters d length-m strings with its one-step predictions, giving Natarajan dimension exactly d. The terminal-state class therefore shatters dT instances, yielding the realizable PAC lower bound above (the absorbing state $\top_0$ also supplies a fixed-label instance). This encoding has $|S| = 2^m(2^T-1)+2$, so $\log|S| = \Theta(T + \log(dT))$; the lower bound does not keep the state-space size fixed as T grows. □

C.4 Applications of Theorem C.2

For a finite class, consistent learning gives $\texttt{comp}(\Pi) = O(1 + \log|\Pi|)$; we use this bound directly in both applications below. In this section we instantiate Theorem C.2 for the specific classes of semiautomata previously discussed in Appendix A.1: Regular languages and linear recurrences over finite fields.

Regular languages. As discussed earlier, the class of regular languages induced by regular expressions with alphabetic length at most k induces a natural class of semiautomata. Specifically, the induced semiautomaton is always initialized at the state $s_0 = \{0\}$, and input words are strings in Σ^T. The terminal state of the semiautomaton can be used to decide whether the input word is a member of the regular language (i.e., whether it is generated by the corresponding regular expression).

In more detail, let $\mathtt{RE}_k(\Sigma)$ denote the set of regular expressions over alphabet Σ with alphabetic length at most k. Use a normalized expression syntax of total size $O(k)$, removing redundant nested stars and empty expressions. There are at most $(c(1+|\Sigma|))^{O(k)}$ such expressions (Gruber et al., 2012, Table 8). The DFA for each regular language can be obtained by determinizing an NFA on $k+1$ states (Glushkov's construction) via the powerset construction. Thus, $\mathtt{RE}_k(\Sigma)$ corresponds to a class of semiautomata $\Pi_{\mathtt{RE}_k}$ on $|S| = 2^{\Theta(k)}$ states, and the size of the class itself is $|\Pi_{\mathtt{RE}_k}| \leqslant (c(1+|\Sigma|))^{O(k)}$ after canonical compilation onto the common subset state space. See Appendix A.2 for a detailed example.

In this setting, the $\mathtt{iCoT}(\cdot)$ oracle identifies the set of states which are reachable at time t, which is stronger feedback than language membership. By divide-and-conquer, it can be implemented in parallel time $\mathrm{poly}(k, \log(T))$ (and with $T \cdot \mathrm{poly}(k)$ work). Furthermore, when Theorem C.2 is specialized to this setting, the following guarantees on sample complexity and query complexity are established for `AutoLearn` (Algorithm 2),

$$n_{\mathtt{sample}}, n_{\mathtt{query}} \leqslant 2^{\widetilde{\mathcal{O}}(\sqrt{\log(T)})} \cdot \frac{k \log(e|\Sigma|) \log^2(1/\varepsilon) \log(1/\delta)}{\varepsilon},$$

where the target failure probability is δ, error rate is ε. Notably, this guarantee only scales linearly in the bound on the alphabetic length, k, even though the underlying (semi-)automata are on $2^{\Theta(k)}$ states.

Linear recurrences over finite fields. Here d denotes vector dimension, not Natarajan dimension. Linear recurrences over finite fields correspond to semiautomata with state space $S = \mathbb{F}_q^d$, input alphabet $\Sigma = \mathbb{F}_q^d$, and transition function $\pi(s, w) = As + Bw$ for unknown matrices $A \in \mathbb{F}_q^{d \times d}$ and $B \in \mathbb{F}_q^{d \times d}$. Simulating for T steps computes the linear recurrence $s_T = A^T s_0 + \sum_{t=1}^{T} A^{T-t} B w_t$. The class Π of all such linear transitions is parameterized by the pair (A, B), giving $|\Pi| = q^{2d^2}$, and Natarajan dimension $\mathrm{Ndim}(\Pi) \leqslant \mathcal{O}(d^2 \log q)$. Specializing Theorem C.2 to this setting results in the following sample complexity and query complexity upper bounds for `AutoLearn`:

$$n_{\mathtt{sample}}, n_{\mathtt{query}} \leqslant 2^{\widetilde{\mathcal{O}}(\sqrt{\log(T)})} \cdot \frac{d^2 \log(q) \log^2(1/\varepsilon) \log(1/\delta)}{\varepsilon}.$$

Here, the target failure probability is δ, error rate is ε. This guarantee only scales polynomially in the dimension d, even though the underlying semiautomaton has $|S| = q^d$ states. In this setting, the $\mathtt{iCoT}(\cdot)$ oracle again has a natural interpretation as before, computing the state after t steps of recursion, which by divide-and-conquer can be computed in parallel in time $\mathrm{poly}(d, \log(T))$ with $T \cdot \mathrm{poly}(d)$ work, counting field operations.

D RLVR: Fine-Tuning a Weak Reference Model

This section adapts the compositional curriculum from Appendix C to the RLVR setting, where the learner aims to improve a weak reference model π^{ref} using an outcome verifier (Definition 4.1). We show that the compositional curriculum reduces the coverage requirement on π^{ref} from the full sequence length T to the shorter block scale B, an exponentially weaker condition. We begin by motivating the need for a compositional curriculum through a discussion of naive RLVR approaches (Appendix D.1), then describe the algorithm `AutoLearn.RL` (Appendix D.2), and finally present the main guarantee (Appendix D.3).

D.1 Motivation: Naive RLVR Methods and the Need for Coverage

Given a pre-trained model π^{ref} and an outcome verifier $\mathcal{V}$, the basic principle underlying standard RLVR fine-tuning methods (e.g., GRPO (Shao et al., 2024)) is to draw rollouts from π^{ref}, filter by outcome correctness, and train on the survivors. `RLFineTune` (Algorithm 5; also used by Rajaraman et al., 2026) provides a simple, theoretically tractable instantiation of this principle. Given a distribution over problem instances, it repeatedly generates candidate CoTs from π^{ref}, keeps those whose terminal state is verified as correct by $\mathcal{V}$, and trains a model on the filtered dataset. Under a

Algorithm 5 `RLFineTune`$\big(D_{\mathrm{input}} \parallel B, \varepsilon, \delta\big)$

Sharpening a reference model π^{ref} satisfying sequence-level coverage using verifier feedback.

1: **Input:** Class of semiautomaton transitions Π over state space S and alphabet Σ,
Number of steps of semiautomaton simulation B,
Reference model π^{ref} satisfying Eq. (1) with parameters $(B, C_{\mathsf{seq}}^{(B)}, C_{\mathsf{ans}}^{(B)})$,
Outcome verifier $\mathcal{V}$,
Target error rate ε and failure probability δ,
Dataset of instances $D_{\mathrm{input}} = \{\boldsymbol{x}^i = (s_0^i, \boldsymbol{w}^i)\}_{i=1}^n$.

2: **Instantiate:** $m \leftarrow \lceil C_{\mathsf{seq}}^{(B)} \log\Big(4nC_{\mathsf{seq}}^{(B)}/\delta\Big)\rceil$.

3: **for** $\boldsymbol{x}^i \in D_{\mathrm{input}}$ **do**

4: Draw CoTs, $\boldsymbol{y}^{i,j} \sim \pi_{1:B}^{\mathrm{ref}}(\cdot \,|\, \boldsymbol{x}^i)$ for $j = 1, \cdots, m$.

5: Let D_i denote the set of deduplicated CoTs among $\{\boldsymbol{y}^{i,j}\}_{j=1}^m$.

6: Keep only sufficiently probable CoTs with verified correct answers:

$$\widetilde{D}_i \leftarrow \big\{\boldsymbol{y} \in D_i : \mathcal{V}(\boldsymbol{x}^i, y_B) = 1 \text{ and } \pi_{1:B}^{\mathrm{ref}}(\,\boldsymbol{y} \,|\, \boldsymbol{x}^i) \geqslant \big(C_{\mathsf{seq}}^{(B)}\big)^{-1}\big\}.$$

7: **Return:** $\widehat{\pi} \in \operatorname*{argmin}_{\pi \in \Pi} \mathcal{L}(\pi; D_{\mathrm{input}})$, where $\mathcal{L}(\pi; D_{\mathrm{input}}) = \frac{1}{n}\sum_{i=1}^n \mathbb{I}\big(\pi_{1:B}(\boldsymbol{x}^i) \notin \widetilde{D}_i\big)$.

sequence-level coverage assumption Eq. (1) instantiated at block length T, each rollout produces a fully correct CoT with probability at least $(C_{\mathsf{seq}}^{(T)})^{-1}$, and altogether roughly

$$\widetilde{\mathcal{O}}\left(\frac{dC_{\mathsf{seq}}^{(T)}}{\varepsilon}\right) \tag{4}$$

prompts and verifier queries (and T times as many state generations) suffice to find an ε-optimal model. The obvious limitation of this approach is that it requires the reference model to have non-trivial coverage over instances of length T. If the model π^{ref} only satisfies coverage at length $B \ll T$ (i.e., it is not strong enough to solve length-T instances directly), one can at best hope for a bound of the form $C_{\mathsf{seq}}^{(T)} \leqslant (C_{\mathsf{seq}}^{(B)})^{T/B}$, which scales exponentially in the number of blocks.[11]

Remark D.1. `RLFineTune` *(Algorithm 5) assumes access to the trajectory probabilities of* π^{ref} *and an exact oracle for the trajectory-set loss in line 7. This loss is not the same as the NTP objective defined in Definition 2.2, and is a trajectory-level loss.*

D.2 Overview of `AutoLearn.RL`

To overcome the coverage requirement for naive RLVR, we adopt a compositional curriculum that decomposes length-T problem instances into length-B instances, the scale at which π^{ref} is reliable. Our algorithm, `AutoLearn.RL` (Algorithm 6), uses `RLFineTune` as an inner-loop primitive to improve π^{ref} on length-B instances, wrapped in an outer loop that constructs the curriculum using an adapted version of Algorithm 1. As in our previous results, we critically use that the semiautomaton is time-homogeneous: the same transition map governs every step, so any length-B block of a length-T instance—beginning at an intermediate state and treating it as a fresh start state—is itself a valid length-B instance. Consequently, the outcome verifier $\mathcal{V}$ can be applied to check attempts at solving each such block.

In more detail, `AutoLearn.RL` (Algorithm 6) is an invocation of `AutoLearn`$_H$ (Algorithm 1), but uses a simplified decomposition and branching schedule with $H = 1$ (in other words, the algorithm considers instances of length T at the top level and length B at the bottom level, with no intermediate levels):

- At the top level, we set $L_1 = 1$ and $k_1 = k_\varepsilon$, which corresponds to boosting from constant accuracy at length-T to accuracy $1 - \varepsilon$ at length-T.

[11] For unstructured problems, this is the best possible bound for any algorithm, not just `RLFineTune`; see Foster et al. (2025).

Algorithm 6 `AutoLearn.RL`$\big(D_{\text{input}} \,\|\, T, \varepsilon, \delta\big)$

Learning semiautomata from verifier feedback using compositional curriculum.

Input: Class of semiautomaton transitions Π over state space S and alphabet Σ,
Reference model π^{ref} satisfying Eq. (1) with parameters $(B, C_{\mathsf{seq}}^{(B)}, C_{\mathsf{ans}}^{(B)})$,
Outcome verifier $\mathcal{V}$,
Number of steps of semiautomaton simulation, T, with $T \pmod B \equiv 0$,
Target error rate ε and failure probability δ,
Dataset of instances $D_{\text{input}} = \{\boldsymbol{x}^i = (s_0^i, \boldsymbol{w}^i)\}_{i=1}^n$.

Set $L = T/B$, $k_{1/2} = \lceil C \log L \rceil$, $k_1 = k_\varepsilon$, and $\varepsilon_0^{-1} = C k_{1/2}^{7/2} \log L$.

Use the RL budgets $n^\star_{\mathsf{sample}}(0, \delta') = n^{\text{RL}}_{\mathsf{weak}}(\delta')$ (Eq. (50)), and the budgets of Lemmas H.2 and H.5 at levels $1/2, 1$.

Set $M_G = k_1 n^\star_{\mathsf{sample}}(1/2, \delta/(4k_1))$ and $\delta_1 = \delta/(2\max\{1, M_G\})$.

Return: $\texttt{AutoLearn}_1(D_{\text{input}} \| T, \delta/2)$ with $(L_{1/2}, L_1) = (L, 1)$, the branching schedule above, and these overrides:

At $h = 0$, truncate to the base budget and return the block predictor from $\texttt{RLFineTune}(\cdot \| B, \varepsilon_0, \delta')$ on the unlabeled input, using the passed confidence δ'.

At $h = 1$, skip `Split` and compute each rank as $\sum_{\ell<j} \mathcal{V}(\boldsymbol{x}, \widehat{\pi}^\ell(\boldsymbol{x}))$. Apply `InvSampling` to the single original instance, returning it without a label.

At $h = 1/2$, implement `Split` with $\texttt{GuessAndCheck}(\cdot \,\|\, T, B, \delta')$ at failure probability δ_1 per length-T instance; reuse its boundary labels to compute ranks.

- At level $h = \frac{1}{2}$, we set $L_{\frac{1}{2}} = T/B$ and $k_{\frac{1}{2}} = \Theta(\log(T/B))$, which corresponds to aggregating and composing short-range models on length-B blocks with constant accuracy into a model solving length-T instances which also achieves constant accuracy.
- At the bottom level, we set $L_0 = 1$ and $k_0 = 1$, which corresponds to training a length-B model to constant accuracy; for this, we use $\texttt{Base} \equiv \texttt{RLFineTune}$ using feedback from the outcome verifier $\mathcal{V}$.

Similar to its counterpart in the iSFT setting, `AutoLearn.RL` can be viewed as an outer-loop which constructs a curriculum of independent length-B instances, wrapped around an inner loop which improves the reference model π^{ref} on these length-B instances via RL. Our analysis makes use of the following guarantee for `RLFineTune`, which asserts that it can improve π^{ref} to constant accuracy on length-B, but any algorithm with a similar guarantee can be used in its place. In this sense, `AutoLearn.RL` can be viewed as a reduction to RLVR fine-tuning over length-B instances.

Proposition D.2 (Guarantee for `RLFineTune`; adapted from Rajaraman et al., 2026)**.** *Fix $\delta \in (0, 1)$, and suppose the reference model π^{ref} satisfies sequence-level coverage with parameter $C_{\mathsf{seq}}^{(B)}$ at length B (Definition 4.4). There is an absolute constant $C > 0$ such that when $\texttt{RLFineTune}(\cdot \,\|\, B, \varepsilon, \delta)$ is run on a dataset of size*

$$n_{\mathsf{sample}} \geqslant \frac{C\big(d \log\big(eB|S|C_{\mathsf{seq}}^{(B)}\big) \log(e/\varepsilon) + \log(e/\delta)\big)}{\varepsilon}$$

drawn from some target distribution ρ_B over length-B instances, the resulting model $\widehat{\pi} : S \times \Sigma^B \to S$ satisfies $\Pr_{\boldsymbol{x}\sim\rho_B}\big[\widehat{\pi}(\boldsymbol{x}) \neq \pi^\star_B(\boldsymbol{x})\big] \leqslant \varepsilon$ with probability at least $1 - \delta$. Here d is the Natarajan dimension of Π. Furthermore, the number of queries `RLFineTune` makes to the outcome verifier $\mathcal{V}$ is at most, $n_{\mathsf{query}} \leqslant \widetilde{\mathcal{O}}\big(\frac{dC_{\mathsf{seq}}}{\varepsilon}\big)$, and the computational cost of `RLFineTune` is at most, $n_{\mathsf{comp}} \leqslant \widetilde{\mathcal{O}}\big(\frac{dBC_{\mathsf{seq}}^{(B)}}{\varepsilon}\big)$, where $\widetilde{\mathcal{O}}(\cdot)$ hides logarithmic factors in $|S|, B, C_{\mathsf{seq}}^{(B)}, \varepsilon$ and $1/\delta$. Moreover, the verifier is queried only to check predictions for the terminal state of length-B instances. Here the computational cost is the number of states generated from π^{ref} or any other model in Π over the course of training (cf. Remark 4.3).

The displayed algorithm assumes $B < T$. For $B = T$, omit the composition level and boost $\texttt{RLFineTune}(\cdot \| T, 1/4, \delta')$ directly, using the same verifier-based ranks and the rejection-sampling budget of Lemma H.5. No guess-and-check calls are then needed.

Simulating intermediate state labels via guess-and-check. To merge models learned at the length-B scale into a model at the length-T scale, `AutoLearn.RL` uses the `Split`$(\cdot \,\|\, L)$ procedure of Algorithm 4, called by Algorithm 1, which requires intermediate state labels at the length-B scale. Naively such feedback is not available to the learner in the RLVR setting, but the reference model π^{ref} and outcome verifier $\mathcal{V}$ can be used to simulate this feedback using a guess-and-check approach, which we refer to as `GuessAndCheck`$(\cdot \,\|\, T, B, \delta')$.

`GuessAndCheck`$(\cdot \,\|\, T, B, \delta')$. Given a length-$T$ instance $\boldsymbol{x}$, a reference model which satisfies a bound on $C^{(B)}_{\mathsf{ans}}$, outcome coverage at the length-B scale, in conjunction with the outcome verifier can be used to split the instance into T/B blocks of length B using a guess-and-check approach. Fix a prompt $\boldsymbol{x}$. To label the terminal state of the first block, i.e., $\pi^\star_B(\boldsymbol{x})$, we generate $\widetilde{\mathcal{O}}(C^{(B)}_{\mathsf{ans}})$ rollouts from the reference model. With high probability, at least one rollout has a correct state at $t = B$, and we can identify it with the verifier. We then proceed to the next block, interpreting $\pi^\star_B(\boldsymbol{x})$ as a start state, and repeating the process to identify $\pi^\star_{2B}(\boldsymbol{x})$, and so on. We discuss this in more detail in Appendix H.1.

For a given prompt, this procedure recovers the terminal states for all blocks using total query complexity $\widetilde{\mathcal{O}}(T/B \cdot C^{(B)}_{\mathsf{ans}})$. A crucial detail, and the source for most of the technical effort behind the algorithm, is that this process does not identify correct intermediate states within each block, only the final state.

D.3 Main Result: Coverage Expansion via Compositional Curriculum

Our main result shows that `AutoLearn.RL` achieves high accuracy on length-T instances whenever the reference model π^{ref} can solve length-B instances with constant probability, a form of coverage expansion.

Theorem D.3 (Main guarantee for `AutoLearn.RL`). *Fix $\varepsilon, \delta \in (0,1)$, block length $1 \leqslant B \leqslant T$, with $T \pmod B \equiv 0$. Suppose the reference model π^{ref} satisfies Eq. (1) with parameters $(B, C^{(B)}_{\mathsf{seq}}, C^{(B)}_{\mathsf{ans}})$, and suppose the class Π has Natarajan dimension d (Definition 2.3). Then, `AutoLearn.RL` (Algorithm 6) returns a model $\widehat{\pi}$ that satisfies $\Pr_{\boldsymbol{x}\sim\rho}\left[\widehat{\pi}(\boldsymbol{x}) \neq \pi^\star_T(\boldsymbol{x})\right] \leqslant \varepsilon$ with probability at least $1-\delta$. Moreover:*

1. *The sample complexity is bounded by, $n_{\mathsf{sample}} \leqslant n^{\star,\mathsf{RL}}_{\mathsf{sample}} = \widetilde{\mathcal{O}}\left(\frac{d}{\varepsilon}\right)$.*
2. *The total computational cost is bounded by $n_{\mathsf{comp}} \leqslant \widetilde{\mathcal{O}}\left(\frac{dT}{\varepsilon}\right) + \widetilde{\mathcal{O}}\left(dTC^{(B)}_{\mathsf{ans}} + dBC^{(B)}_{\mathsf{seq}}\right)$, as measured by the number of states/tokens generated.*[12]
3. *The query complexity is upper bounded by $n_{\mathsf{query}} \leqslant \widetilde{\mathcal{O}}\left(\frac{d}{\varepsilon}\right) + \widetilde{\mathcal{O}}\left(\frac{T}{B} \cdot dC^{(B)}_{\mathsf{ans}} + dC^{(B)}_{\mathsf{seq}}\right)$.*

Above, $\widetilde{\mathcal{O}}(\cdot)$ hides polylogarithmic factors in $T, |S|, C^{(B)}_{\mathsf{seq}}, \varepsilon^{-1}$ and δ^{-1}.

The proof of Theorem D.3 is deferred to Appendix H.4. The main features of the result are that: (1) All costs only scale with the model's coverage at the length-B scale, representing a form of *coverage expansion* from the length-B scale to the length-T scale; and (2) the dependence on the model's coverage—both for statistical and computational costs—is decoupled from the target accuracy $1-\varepsilon$, apart from logarithmic factors, only appearing in a "burn-in" cost. The latter is a consequence of the boosting effect of the self-generated curriculum, and has a similar flavor to the RL fine-tuning results of Rajaraman et al. (2026). In what follows, we interpret the sample complexity, computational cost, and query complexity in more detail.

[12] A comment on units: the computational cost n_{comp} in Theorem D.3 counts the total number of individual *state generations* produced by π^{ref} or by any model in Π over the course of training, rather than the number of length-T sequences rolled out; the latter is used as the measure of computational cost by Rajaraman et al. (2026). This fact accounts for the T dependency in the bounds on n_{comp}: to even predict the terminal state on a single length-T instance requires generating T states.

Sample complexity. The number of prompts required by `AutoLearn.RL` scales as $n_{\texttt{sample}} \leqslant \widetilde{\mathcal{O}}(d/\varepsilon)$. This matches the usual d/ε scale for worst-case realizable PAC learning, up to logarithmic factors; no matching lower bound for every reference model is asserted.

Computational cost. The computational cost of `AutoLearn.RL`—measured in terms of number of generated states—decomposes into two parts: a "leading order" term, which is polynomial in ε^{-1} but nearly independent of the coverage coefficients, and a "burn-in" term, which is polynomial in the coverage coefficients, but nearly-independent of ε^{-1}:

$$n_{\texttt{comp}} \leqslant \widetilde{\mathcal{O}} \underbrace{\left(\frac{dT}{\varepsilon}\right)}_{\text{leading order}} + \underbrace{\widetilde{\mathcal{O}}\left(dTC^{(B)}_{\texttt{ans}} + dBC^{(B)}_{\texttt{seq}}\right)}_{\text{burn-in cost } n_{\texttt{burn-in}}}. \tag{5}$$

The leading-order term $\widetilde{\mathcal{O}}(dT/\varepsilon)$ reflects the number of state generations from the learned child models needed to predict the terminal state on each of the $\widetilde{\mathcal{O}}(d/\varepsilon)$ instances in the training dataset. We do not prove a matching generation-cost lower bound. The burn-in term depends on the pre-trained model's coverage, but critically, does so only at the length-B scale. The term involving the (smaller) answer-level coverage coefficient $C^{(B)}_{\texttt{ans}}$ is scaled by T, while the term involving the (larger) block-level coverage coefficient $C^{(B)}_{\texttt{seq}}$ is scaled only by the block length B. In more detail:

- *Labeling cost.* The first term, $\widetilde{\mathcal{O}}(dTC^{(B)}_{\texttt{ans}})$, can be interpreted as the computational cost required to label a deliberately chosen set of $\widetilde{\mathcal{O}}(d)$ length-T instances $\boldsymbol{x}$ with ground-truth outcomes for each length-B block using the guess-and-check strategy described in Appendix D.2. In particular, each $\boldsymbol{x}$ is decomposed into T/B segments, and π^{ref} generates $\widetilde{\mathcal{O}}(C^{(B)}_{\texttt{ans}})$ rollouts per segment, with overall cost $\widetilde{\mathcal{O}}(d \times T/B \times C^{(B)}_{\texttt{ans}} \times B)$, where the final factor of B arises because π^{ref} generates B states for each segment rollout.
- *Training cost.* The second term, $\widetilde{\mathcal{O}}(dBC^{(B)}_{\texttt{seq}})$, can be interpreted as the computational cost of using the base learner `RLFineTune` to improve π^{ref} to constant accuracy on length-B instances, as in Proposition D.2. This term can therefore be interpreted as the computational cost emerging from *training* within `AutoLearn.RL`, once again focusing on the cost of generation, and not model updates (cf. Remark 4.3).

Query complexity. The number of queries to the outcome verifier made by `AutoLearn.RL` has a similar decomposition to the computational cost: A leading-order term scaling as $\widetilde{\mathcal{O}}(d/\varepsilon)$, and a burn-in term which inherits the polynomial dependence on the coverage coefficients $C^{(B)}_{\texttt{seq}}$ and $C^{(B)}_{\texttt{ans}}$:

$$n_{\texttt{query}} \leqslant \widetilde{\mathcal{O}}\left(\frac{d}{\varepsilon}\right) + \underbrace{\widetilde{\mathcal{O}}\left(\frac{T}{B} \cdot dC^{(B)}_{\texttt{ans}} + dC^{(B)}_{\texttt{seq}}\right)}_{\text{burn-in cost}},$$

Here: (1) The leading-order term reflects the number of queries needed to verify length-T terminal state correctly on each of the $\widetilde{\mathcal{O}}(d/\varepsilon)$ instances in the training dataset; (2) the burn-in term $\widetilde{\mathcal{O}}\left(\frac{T}{B} \cdot dC^{(B)}_{\texttt{ans}}\right)$ reflects the number of queries needed to label all of the block-level terminal states for a deliberately chosen subset of $\widetilde{\mathcal{O}}(d)$ instances; (3) the burn-in term $\widetilde{\mathcal{O}}(dC^{(B)}_{\texttt{seq}})$ reflects the number of queries needed for `RLFineTune` to improve π^{ref} to constant accuracy on length-B instances.

D.3.1 Comparison to Fixed-Length Curriculum

Rajaraman et al. (2026) showed that in the general autoregressive setting, a simpler form of self-generated curriculum with *no length curriculum* (i.e., purely selecting length-T prompts from the training set adaptively) can be used to improve a reference model π^{ref} to accuracy $1-\varepsilon$ with computational cost $\widetilde{\mathcal{O}}\left(dT/\varepsilon + dTC^{(T)}_{\texttt{seq}}\right)$. Our results above recover this guarantee by setting $B = T$, but—as discussed in the previous section—can be exponentially more efficient if π^{ref} only has constant coverage for blocks of length $B \ll T$.

E Overview of Analysis Techniques: Recursion and Inverted Sampling

Having presented the high-level principles behind our algorithm `AutoLearn` in Appendices C and D, this section gives an in-depth overview of the multiscale boosting technique used within the algorithm. To simplify presentation, we restrict our attention to the iSFT setting (Algorithm 1), and focus on the regime where the target error $\varepsilon = \frac{1}{4}$ is a constant, as this already captures the main technical challenges, and will use $d = \mathsf{comp}(\Pi)$ to refer to the complexity of the model class (in the sense of Definition C.1). The section proceeds in three parts.

- *Warm-up via composition and boosting* (Appendix E.1). As a warmup, we first show that a single layer of boosting and composition is sufficient to achieve query *and* sample complexity $\widetilde{\mathcal{O}}(d\sqrt{T})$. This already improves upon the linear-in-T $\widetilde{\mathcal{O}}(dT)$ full-CoT baseline in Appendix C.3, but is nonetheless still polynomial in T. We then highlight a natural path to improving this guarantee further (Appendix E.2)—improving the sample complexity required by the *rejection sampling* step within boosting.
- *The reweighted-mixture-of-marginals problem and inverted sampling* (Appendix E.3). We isolate an abstract problem, *sampling from a reweighted mixture of marginals*, which captures the key structure of the sampling problem used within our boosting approach (Appendix E.3), and give a new algorithm, *inverted sampling*, which solves it significantly more efficiently than naive rejection sampling (Appendix E.4).
- *Putting everything together* (Appendices E.5 and E.6). We show that using inverted sampling within a single layer of recursion brings the sample complexity down to $\widetilde{\mathcal{O}}(d)$ while maintaining query complexity scaling as $\widetilde{\mathcal{O}}(d\sqrt{T})$ (Appendix E.5), then show that an iterated version of this approach—recursing to depth $H = \mathcal{O}(\sqrt{\log T})$—yields $2^{\widetilde{\mathcal{O}}(\sqrt{\log T})} \cdot d$ sample and query complexity (Appendix E.6).

We begin by introducing notation used throughout the section.

Notation. Recall that ρ denotes the target distribution over length-T instances. For $1 \leqslant t_1 \leqslant t_2 \leqslant T$, we let $\rho_{t_1:t_2}$ denote the law of $(s_{t_1-1}, \boldsymbol{w}_{t_1:t_2})$, where $(s_0, \boldsymbol{w}_{1:T}) = \boldsymbol{x} \sim \rho$ and $s_{t_1-1} = \pi^\star_{t_1-1}(\boldsymbol{x})$. $\rho_{t_1:t_2}$ can be viewed as a distribution over instances of length $t_2 - t_1 + 1$. For some $\tau < T$ define the uniform mixture of length-τ marginals obtained from ρ as follows,

$$\overline{\rho} \;=\; \frac{1}{L}\sum_{i=0}^{L-1} \rho_{\tau i+1:\tau(i+1)}. \tag{6}$$

This is the distribution over instances by breaking down length-T instances drawn from ρ into segments of length τ and selecting one uniformly at random (where the start state of the segment is labeled by $\pi^\star$).

E.1 A Single Level of Recursion: Achieving $n_{\mathsf{sample}} = \widetilde{\mathcal{O}}(d\sqrt{T})$ and $n_{\mathsf{query}} = \widetilde{\mathcal{O}}(d\sqrt{T})$

We first recap the idea of *composition* in `AutoLearn`. For a model $\widehat{\pi} : S \times \Sigma^\tau \to S$ on length-τ instances, we let $\widehat{\pi}^{\circ L}$ denote its L-fold composition (Definition 3.4), which operates on length-T instances by chunking them into $L = T/\tau$ chunks of length τ. The composition framework (Observation 3.5) asserts that whenever $\widehat{\pi}$ has error at most $\varepsilon_0 = \frac{1}{4L}$ under the distribution $\overline{\rho}$, the composed model $\widehat{\pi}^{\circ L}$ has error at most $\frac{1}{4}$ under the original distribution ρ. Thus, compositional generalization gives a mechanism by which length can be traded off for accuracy at the length-τ scale. However, this tradeoff is not sufficient on its own to achieve sublinear-in-T query complexity: a single model $\widehat{\pi}$ learned to error $\varepsilon_0 = \frac{1}{4L}$ under $\overline{\rho}$ by querying full CoTs on length-τ instances has query complexity scaling as $\mathcal{O}\big(d\tau/\varepsilon_0\big) = \mathcal{O}(dT)$, matching the full-CoT baseline of Section C.3.1.

Breaking the $\mathcal{O}(dT)$ barrier requires combining composition with *boosting*, building on Rajaraman et al. (2026). In the general autoregressive setting (without composition), they showed that boosting

can be used to decide which instances to query *full CoTs* on, adaptively focusing on harder instances as determined by outcome correctness of the current ensemble of models. Given target error $\varepsilon > 0$, boosting trains $\mathcal{O}(\log(1/\varepsilon))$ weak models to constant accuracy on an adaptively chosen sequence of distributions, each of which requires only $\widetilde{\mathcal{O}}(d)$ CoTs. The total query complexity required to train a length-T model in our iSFT setting is $\widetilde{\mathcal{O}}(dT + d/\varepsilon)$, where the former term accounts for the query complexity of training the weak learners (recall that one full CoT requires T iSFT queries), and the latter term accounts for the query complexity required to evaluate the weak learners and update the training distribution. Furthermore, the sample complexity is $\widetilde{\mathcal{O}}(d)$ (more general, $\widetilde{\mathcal{O}}(d/\varepsilon)$ for error ε).

Combining boosting with composition gives the following template:

1. For a parameter $\tau \ll T$ and $L = T/\tau$, use boosting to train a model $\widehat{\pi}$ to accuracy $\varepsilon_0 = 1/4L$ on length-τ sequences drawn from $\overline{\rho}$.
2. Return the L-fold composed model $\widehat{\pi}^{\circ L}$, giving error $\varepsilon = \varepsilon_0 L = \frac{1}{4}$ as desired.

Implementing boosting in this fashion requires rejection sampling from reweighted versions of the distribution $\overline{\rho}$—a non-trivial detail, as we will highlight below. After accounting for this, the total query complexity for this idealized template is:

$$n_{\texttt{query}} = \widetilde{\mathcal{O}}(d\tau + d/\varepsilon_0) = \widetilde{\mathcal{O}}(d\tau + dL).$$

Recalling that $L = T/\tau$ and choosing $\tau = \sqrt{T}$ to balance the two terms yields $n_{\texttt{query}} = \widetilde{\mathcal{O}}(d\sqrt{T})$ query complexity, improving over the $\widetilde{\mathcal{O}}(dT)$ query complexity of the full CoT baseline. Furthermore, the sample complexity is $n_{\texttt{sample}} = \widetilde{\mathcal{O}}(d/\varepsilon_0) = \widetilde{\mathcal{O}}(dL) = \widetilde{\mathcal{O}}(d\sqrt{T})$.

E.2 Improving Rejection Sampling: A Path to Sub-Polynomial Query Complexity

The full version of `AutoLearn` (Algorithm 1) uses multiple levels of recursion, solving the length-τ blocks above by breaking them into smaller blocks, and so on. The main bottleneck in making this recursion fruitful is the *sample complexity* in the single-level algorithm in the prequel, which scales as $\widetilde{\mathcal{O}}(d/\varepsilon_0) = \widetilde{\mathcal{O}}(dL)$. To ensure that errors do not compound as we recurse, it is essential to improve this guarantee so that it scales as $\widetilde{\mathcal{O}}(d)$—i.e., nearly independent of the number of blocks L. We will return to the full derivation in Appendix E.6, but for now we set our sights on improving the sample complexity to $\widetilde{\mathcal{O}}(d)$.

Showing that the sample complexity can be improved to be strictly better than $\mathcal{O}(d/\varepsilon_0)$ is essential for the recursion to be fruitful. With multiple rounds of recursion, instances are decomposed into shorter ones, following the schedule $T \to T/L \to \cdots \to T/L^H$. At each depth, a model with error $\frac{1}{4}$ is desired; this is obtained by taking the L-fold composition of a model achieving error $\varepsilon_0 = \frac{1}{4L}$ under the appropriate mixture distribution of instances (akin to $\overline{\rho}$). This model itself is trained by boosting, by training and aggregating an ensemble of models each of which achieves error $\frac{1}{4}$ across appropriately reweighted versions of the mixture distribution, and so on. As the depth increases, the length of instances decreases geometrically and at the deepest level, models are trained to solve instances of length T/L^H. As we discuss in the next paragraph, generating samples from the appropriate reweighted mixture distributions at the deepest level via rejection sampling will require drawing d/ε_0^H instances at the top level (i.e., from ρ). Since the length of instances at the bottom level is $\mathcal{O}(T/L^H) = \mathcal{O}(T(4\varepsilon_0)^H)$, all in all, this gives query complexity,

$$n_{\texttt{query}} = \widetilde{\mathcal{O}}(dT(4\varepsilon_0)^H + d/\varepsilon_0^H)$$

And optimizing over ε_0 and H still results in a guarantee that is no better than $\widetilde{\Theta}(d\sqrt{T})$.

Key bottleneck: Rejection sampling. To highlight why the sample complexity for the naive approach in the prequel scales as $n_{\texttt{sample}} = \widetilde{\mathcal{O}}(d/\varepsilon_0) = \widetilde{\mathcal{O}}(dL)$, let us dig into the details of the boosting stage. Formally, boosting proceeds in $k = \widetilde{\mathcal{O}}(\log \varepsilon_0^{-1}) = \widetilde{\mathcal{O}}(\log L)$ iterations. At iteration

j, given weak models $\widehat{\pi}^0, \ldots, \widehat{\pi}^{j-1}$, we train a new model $\widehat{\pi}^j$ on a distribution $\overline{\rho}_j$, which reweights $\overline{\rho}$ to focus on instances where the current ensemble $\left(\widehat{\pi}^0, \cdots, \widehat{\pi}^{j-1}\right)$ errs; $\widetilde{\mathcal{O}}(d)$ samples from this distribution are required to ensure that $\widehat{\pi}^j$ achieves constant accuracy. After all iterations conclude, we aggregate the models through a majority vote to produce a final length-τ model

$$\widehat{\pi} = \texttt{Plu}\left(\{\widehat{\pi}^0, \cdots, \widehat{\pi}^{k-1}\}\right).$$

As discussed above, this model has error $\varepsilon_0 = 1/4L$ on $\overline{\rho}$, leading to error $\frac{1}{4}$ for the composed model $\widehat{\pi}^{\circ L}$. This is depicted pictorially in Figure 3a.

Training the jth model above requires generating $\widetilde{\mathcal{O}}(d)$ inputs from the reweighted distribution $\overline{\rho}_j$, which has the structure $\overline{\rho}_j(\cdot) \propto \overline{\rho}(\cdot) w_j(\cdot)$ for a non-negative weight function w_j; informally, w_j downweights regions where the ensemble already has high accuracy and upweights regions with low accuracy (refer to Eq. (2) for a formal definition of w_j). Standard boosting—as in Rajaraman et al. (2026)—generates samples from this distribution using *rejection sampling* (von Neumann, 1963; Block & Polyanskiy, 2023): We repeatedly draw $\boldsymbol{x} \sim \overline{\rho}$, then draw $\eta \sim \mathrm{Unif}([0,1])$ and accept if $\eta \leqslant \frac{w_j(\boldsymbol{x})}{\|w_j\|_\infty}$. Accepted samples are guaranteed to follow $\overline{\rho}_j$, so the only question is how many attempts are required. For this, one of two good situations must occur. First, if

$$\Pr_{\boldsymbol{x}\sim\overline{\rho}, \eta \overset{\text{i.i.d}}{\sim} \mathrm{Unif}([0,1])}\left(\eta \leqslant \frac{w_j(\boldsymbol{x})}{\|w_j\|_\infty}\right) \geqslant \varepsilon_0 \tag{7}$$

then only $\mathcal{O}(\varepsilon_0^{-1})$ attempts are required in expectation, leading to $\widetilde{\mathcal{O}}(d\varepsilon_0^{-1})$ attempts in total to generate the full dataset for $\widehat{\pi}^j$. If Eq. (7) fails, the small mean weight instead bounds this round's contribution to the boosting potential. The formal analysis controls the sum of these contributions; small acceptance alone does not certify the current majority's accuracy.

While the $\mathcal{O}(\varepsilon_0^{-1})$ sample complexity for rejection sampling that we sketch above is tight for generic sampling problems (Block & Polyanskiy, 2023), our compositional application has additional structure. In particular, ρ_j has the form

$$\overline{\rho}_j(\cdot) \propto \overline{\rho}(\cdot) w_j(\cdot) = \frac{1}{L}\sum_{i=0}^{L-1} \rho_{\tau i+1:\tau(i+1)}(\cdot) w_j(\cdot),$$

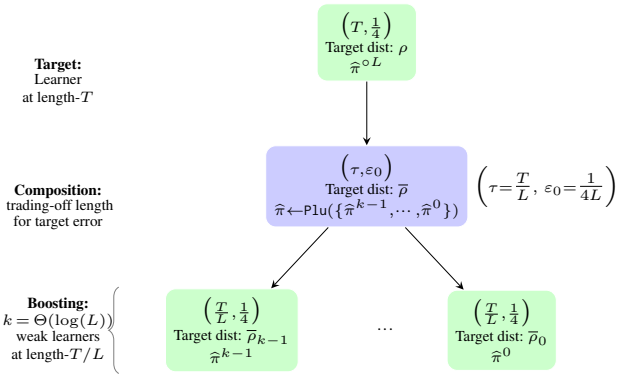
which we recall arises from mixing blocks at different positions. Writing the details explicitly, standard rejection sampling with this distribution takes the following form:

- Generate a length-T instance $\boldsymbol{x} \sim \rho$.
- Sample $i \sim \{0, \ldots, L-1\}$ uniformly at random.
- Query the start state $\pi^\star_{\tau i}(\boldsymbol{x})$, and terminal state $\pi^\star_{\tau(i+1)-1}(\boldsymbol{x})$ for block i and define the length-τ instance $\boldsymbol{x}^i = (\pi^\star_{\tau i}(\boldsymbol{x}), \boldsymbol{w}_{\tau i+1:\tau(i+1)})$, which is distributed according to $\rho_{\tau i+1:\tau(i+1)}$. This requires a single query to the $\texttt{iCoT}(\cdot)$ oracle.
- Draw $\eta \sim \mathrm{Unif}([0,1])$ and accept if $\eta \leqslant w_j(\boldsymbol{x}^i)/\|w_j\|_\infty$.

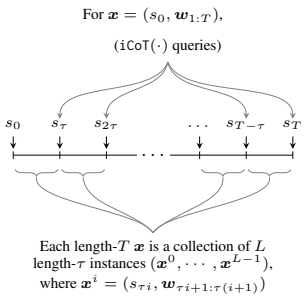
Our first key observation is as follows: Instead of only generating the start state $\boldsymbol{x}^i$ for the uniformly sampled block i, we can generate the start states $\boldsymbol{x}^1, \ldots, \boldsymbol{x}^L$ *for all L blocks simultaneously*, at the cost of increasing the number of $\texttt{iCoT}(\cdot)$ queries to L. Can we use this extra information to more efficiently generate a sample from the target distribution $\overline{\rho}_j$?

E.3 Detour: Rejection Sampling from Reweighted Mixtures of Marginals

To highlight our goal as crisply as possible, we define an abstract setting, *sampling from reweighted mixtures of marginals*, which captures the essential structure of the problem above, but may be of interest in its own right. In this setting, we assume sampling access to a joint distribution μ over potentially correlated random variables $(\boldsymbol{x}^0, \cdots, \boldsymbol{x}^{L-1})$, and aim to sample from a reweighted mixture of their marginals according to a weight function $w(\boldsymbol{x})$. In the boosting context, these random variables correspond to the length-τ segments of a length-T input instance, with μ playing the role of ρ.



(a) Idealized boosting and composition under exact tilted-marginal sampling. The inverted-sampling analysis instead uses conditioned target laws. The tuple (t, ε) in each box indicates the length of the instances and the target error for the model trained on these instances. $\widehat{\pi}^0, \cdots, \widehat{\pi}^{k-1}$, are combined by plurality vote to give $\widehat{\pi}$, and its L-fold composition is a model on length T instances.



(b) Each length-T instance $\boldsymbol{x}$ can be viewed as a collection of L shorter instances, $(\boldsymbol{x}^0, \cdots, \boldsymbol{x}^{L-1})$, by labeling every τ^{th} state (via querying the $\texttt{iCoT}(\cdot)$ oracle).

Figure 3: *Left:* Multiscale boosting. *Right:* Decomposing longer instances into shorter ones.

Definition E.1 (Sampling from reweighted mixtures of marginals)**.** *Let $\boldsymbol{X}$ denote a probability space, and fix $L \in \mathbb{N}$. We are given sampling access to a joint distribution μ over $\boldsymbol{X}^L$, and query access to a weight function $w : \boldsymbol{X} \to [0, 1]$. Define $\nu(\cdot) \propto \overline{\mu}(\cdot)w(\cdot) \in \Delta_{\boldsymbol{X}}$, where $\overline{\mu}(\cdot) = \frac{1}{L}\sum_{i=0}^{L-1} \mu(X_i = \cdot\,)$ is the uniform mixture of marginals of μ. Our objective is to generate samples (approximately) from ν, making as few queries to $w(\cdot)$ and drawing as few joint samples from μ as possible.*

For this setting, standard rejection sampling takes the following form:

(a) Draw $X = (X_0, \cdots, X_{L-1}) \sim \mu$. Sample an index $I \sim \mathrm{Unif}(\{0, \cdots, L-1\})$ and select X_I. Since I was selected uniformly at random, the random variable X_I is sampled according to $\overline{\mu}$.

(b) Use rejection sampling to accept/reject X_I. Namely, draw $\eta \sim \mathrm{Unif}([0, 1])$ and accept X_I if $\eta \leqslant w(X_I)/\|w\|_\infty$ (the RHS is the acceptance probability). Repeat from (a) until a sample is accepted and declare success if a sample is accepted.

As discussed in the prequel, a standard sufficient condition for this process to accept in $\widetilde{\mathcal{O}}(\varepsilon_0^{-1})$ attempts is that

$$\Pr_{X\sim\mu, i\sim\mathrm{Unif}(\{0,\ldots,L-1\}), \eta_i\sim\mathrm{Unif}([0,1])}\left(\eta_i \leqslant \frac{w(X_i)}{\|w\|_\infty}\right) \geqslant \varepsilon_0. \tag{8}$$

To see why we might hope for better, consider the case where $X_0, \ldots, X_{L-1}$ are independent and identically distributed. If each X_i has acceptance probability ε_0 marginally, then Eq. (8) holds with parameter ε_0, and vanilla rejection sampling will indeed use ε_0^{-1} draws from μ. However, this is wasteful—we are effectively throwing away $L-1$ perfectly good samples—and we can increase the success probability to $1-(1-\varepsilon_0)^L$ by simply testing and accepting/rejecting all $X_0, \ldots, X_{L-1}$.

Motivated by this observation, a more optimistic sampling procedure should aim for the following condition:

$$p_{\mu,w} = \Pr_{X\sim\mu, \eta_i \overset{\text{i.i.d.}}{\sim} \mathrm{Unif}([0,1])}\left(\exists i \in \{0, \ldots, L-1\} : \eta_i \leqslant \frac{w(X_i)}{\|w\|_\infty}\right) \geqslant \varepsilon. \tag{9}$$

This condition asserts that *counterfactually, at least one of samples X_i would have been accepted had we chosen the index i to begin with*. Note that we have $\varepsilon = 1-(1-\varepsilon_0)^L \approx L\varepsilon_0$ for independent X_i when $L\varepsilon_0 \ll 1$, suggesting that the quantity in Eq. (9) captures the easiness of this setting.

A barrier to further improvement: Hardness of sampling. Taking advantage of the condition in Eq. (9) is more challenging when $X_0, \ldots, X_{L-1}$ are correlated. For example, the following lemma shows that vanilla rejection sampling incurs $\Omega(L)$ sample complexity even when Eq. (9) holds with $\varepsilon = 1$.

Lemma E.2 (Informal; see Lemma I.6)**.** *Even when $\varepsilon = 1$ in Eq. (9), the expected number of samples from μ required for rejection sampling to succeed scales with L.*

More generally, we show now that *any approach* to generate a sample from a distribution $\widehat{\nu}$ such that $\mathrm{TV}(\widehat{\nu}, \nu) < \frac{1}{2}$ requires drawing at least $\Omega(\mathrm{poly}(L))$ samples from μ in the worst case. The full version of this result (Lemma I.6) holds even when w and μ satisfy Eq. (9) with $\varepsilon = 1$; we sketch a simplified version below.

Example E.3 (An example showcasing the hardness of sampling from ν)**.** *Consider the setting where $\boldsymbol{X} = \{0, 1, 2\}$, and the weight function $w : \boldsymbol{X} \to [0, 1]$ is defined as $w(z) = \mathbb{I}(z \in \{1, 2\})$. Now, consider the distribution $\mu \in \Delta_{\boldsymbol{X}^L}$ which is uniform on a set of $L + 1$ strings, namely, $\{(1, 1, \cdots, 1), (2, 0, \cdots, 0), (0, 2, \cdots, 0), \cdots, (0, 0, \cdots, 2)\}$. The distribution μ with probability $\approx \frac{1}{L}$ generates the all-1 sequence, and with remaining probability sets a uniformly random position as 2 and the remaining positions as 0. A simple calculation by Bayes rule shows that the distribution $\overline{\mu}$ satisfies $\overline{\mu}(1) = \overline{\mu}(2) = 1/(L + 1)$ and $\overline{\mu}(0) = (L - 1)/(L + 1)$. Consequently, $\nu \propto \overline{\mu}(\cdot)w(\cdot)$ is the uniform distribution over $\{1, 2\}$.*

For this example, Eq. (9) holds with $\varepsilon = 1$, since every string contains either 1 or 2 at some position. However, note the structure of the distribution μ: the symbol 1 is only ever seen in an instance drawn from μ with probability $\approx \frac{1}{L}$, even though this symbol is highly represented under ν. Rejection sampling requires $\Theta(L)$ joint draws here. A known uniform target on $\{1, 2\}$ can, however, be sampled without data, and the point mass at 2 has TV exactly $1/2$. A lower bound for arbitrary algorithms must hide the rare symbol and require uniform performance over an unknown family, as in Lemma I.6.

Thus, in the worst case, sampling (even very approximately) from $\overline{\rho}_j$ cannot be carried out with fewer than $\mathrm{poly}(L)$ samples from ρ. This makes it unlikely to improve the sample and query complexity by developing better methods for sampling from the reweighted mixture of marginals (Definition E.1) under standard notions of distribution approximation (say, TV-distance or KL divergence).

E.4 Inverted Sampling: More Efficient Sampling from Reweighted Mixtures

Recall that standard rejection sampling—when applied to the problem of sampling from reweighted mixtures of marginals—samples a shorter instance from each longer instance uniformly at random, and checks whether it passes a filter. The remaining $L - 1$ sub-instances contained within are thrown away. This is inherently wasteful since one of the remaining (unselected) instances may have passed the filter, but the index of the uniformly random instance selected by rejection sampling will only "hit" this index with small probability $1/L$. A natural algorithm which does not suffer from this issue reverses the order of operands: first accept/reject each of the L shorter instances by applying rejection sampling independently, then select one of the accepted instances uniformly at random. We refer to this approach as *inverted sampling*, implemented as follows:

(a) Draw $X = (X_0, \cdots, X_{L-1}) \sim \mu$. Let $\mathcal{I}$ denote the set of indices,

$$\{i \in \{0, \ldots, L-1\} : \eta_i \leqslant w(X_i)/\|w\|_\infty\}$$

Here $\eta_i \sim \mathrm{Unif}([0, 1])$ independently across i. If $\mathcal{I} = \emptyset$, reject and repeat from (a).

(b) Let I be an index drawn uniformly at random from $\mathcal{I}$. Return X_I.

The benefit of inverted sampling is that it is guaranteed to accept as long as at least one of the X_i's passes the filter; this is much less pessimistic than rejection sampling, which may fail to select this X_i to filter in the first place. However, this approach is still subject to the strong sampling lower bound of Lemma I.6, and so the increased acceptance rate must come at a large cost in the TV distance to ν.

In spite of the TV distance lower bound, we show that inverted sampling can be used to sample from a distribution which is close to ν in the sense of a weaker notion of approximation, which is nonetheless sufficient for our application within `AutoLearn`.

Lemma E.4 (Informal; see Lemmas I.3 and I.5)**.** *Fix $L \geqslant 2$, $p_{\mu,w} > 0$, and a constant $c \in (0,1)$. Let $\widehat{\nu}$ denote the distribution over instances accepted by inverted sampling. There exists an event $\mathcal{K} \subseteq \boldsymbol{X}^L \times [0,1]^L$ which is a measurable function of $(X, \boldsymbol{\eta}) \sim \mu \times \mathrm{Unif}([0,1])^{\otimes L}$ where $\boldsymbol{\eta} = (\eta_i)_{i=0}^{L-1}$, such that $\Pr((X, \boldsymbol{\eta}) \in \mathcal{K}) \geqslant 1 - c$, and the following density ratio bound is satisfied,*

$$\left\| \frac{\nu_{\mathcal{K}}(\cdot)}{\widehat{\nu}(\cdot)} \right\|_{\infty} \leqslant \mathcal{O}(c^{-1} \cdot \log(L)) \tag{10}$$

Here, $\nu_{\mathcal{K}}(\cdot) \propto \frac{1}{L} \sum_{i=0}^{L-1} \Pr(X_i = \cdot \,|\, (X, \boldsymbol{\eta}) \in \mathcal{K})\, w(\cdot)$ is the reweighted mixture distribution of marginals when the underlying sample and random coins, $(X, \boldsymbol{\eta})$, are conditioned on the event $\mathcal{K}$. Finally, the marginal probability that inverted sampling accepts a sample $X \sim \mu$ is $p_{\mu,w}$, defined in Eq. (9).

That is, by shifting our target to the distribution $\nu_{\mathcal{K}}$—where $\mathcal{K}$ is an appropriate high-probability event defined formally in Eq. (66)—we can sample from a distribution $\widehat{\nu}$ that approximates the target in the density ratio sense up to only a $\log(L)$ factor (as opposed to polynomial dependence on L), and achieve an acceptance probability that matches the skyline in Eq. (9). This result sidesteps the hardness of sampling (Lemma I.6) because the distribution approximated by inverted sampling, $\nu_{\mathcal{K}}$, can be very different from ν. Nonetheless, because $\mathcal{K}$ has high probability, this guarantee is sufficient for our application within `AutoLearn`.

How does Lemma E.4 sidestep the hard sampling instance of Example E.3? In order to better understand this result, we put it into context of the hard instance in Example E.3. For the construction in the example, if we define $\mathcal{K}$ as $\{X \neq (1, \cdots, 1)\}$, we see that $\Pr(\mathcal{K}) \geqslant 1 - \frac{1}{L}$ is a high probability event. This conditioning on $\mathcal{K}$ has a dramatic effect on the resulting distribution $\nu_{\mathcal{K}}$: since the all-1 sequence is the only sequence which contains the symbol 1, conditioning on $\mathcal{K}$ results in $\nu_{\mathcal{K}}$ becoming the delta distribution on the symbol 2. This is a very different target from $\nu = \mathrm{Unif}(\{1,2\})$, and sampling from $\nu_{\mathcal{K}}$ no longer presents the same pathologies of sampling from ν.

E.5 Inverted Sampling Achieves $n_{\texttt{sample}} = \widetilde{\mathcal{O}}(d)$ and $n_{\texttt{query}} = \widetilde{\mathcal{O}}(d\sqrt{T})$

We now return to the context of `AutoLearn`, where we use inverted sampling to implement boosting at the length-τ scale. Let σ denote the joint distribution over the L length-τ instances obtained by labeling the boundary states on $\boldsymbol{x} \sim \rho$. In each iteration j of the boosting process, we apply inverted sampling guarantees to draw instances from a distribution that *covers* λ_j, which is the distribution proportional to $\overline{\sigma_{\mathcal{K}_j}}(\cdot)\, w_j(\cdot)$ where $\overline{\sigma_{\mathcal{K}_j}} = \frac{1}{L} \sum_{i=0}^{L-1} \Pr(X_i = \cdot \mid (X, \boldsymbol{\eta}) \in \mathcal{K}_j)$ for $X \sim \sigma$, where $\mathcal{K}_j$ denotes the high probability event defined in Lemma E.4 for an appropriate choice of c. The construction keeps high-probability regions of the joint tuple space while controlling the density ratio for each adaptive round. Small acceptance probability controls that round's regression contribution; it need not mean that every earlier model is correct on every coordinate.

The actual events are nested: each construction loses at most $c = 1/(32k)$ of the remaining mass, and excluding the regression tail loses at most $1/(32k^2)$. Summing these losses gives $\Pr(\mathcal{K}_{k-1}^c) \leqslant 1/16$. The potential argument accounts for this lost mass separately and bounds the remaining regression contributions.

The analysis sketched here requires some subtle additional conditioning arguments to make everything go through, and we defer the discussion of these details to Appendix F.1. Overall, by using inverted sampling to appropriately define the target weak learning distributions, we obtain an algorithm with the following query and sample complexity guarantees:

$$n_{\texttt{query}} = \widetilde{\mathcal{O}}\left(d\tau \cdot \mathrm{polylog}(L) + d/\varepsilon_0\right), \quad \text{and,} \quad n_{\texttt{sample}} = \widetilde{\mathcal{O}}\big(d \cdot \mathrm{polylog}(L)\big)$$

Where $\varepsilon_0 = \mathcal{O}(1/L)$. The d/ε_0 term in the query complexity comes from the cost of using inverted sampling to generate $\mathcal{O}(d)$ samples for the full-CoT learners at the length-τ scale, while the

$d\tau \cdot \mathrm{polylog}(L)$ term comes from labeling all τ intermediate states on the instances that are selected by inverted sampling (to train full-CoT learners via next-token prediction). Optimizing over L, we get an algorithm with query complexity scaling as $\widetilde{\mathcal{O}}(d\sqrt{T})$, but with an *improved sample complexity of* $\widetilde{\mathcal{O}}(d)$ compared to the rejection sampling based approach described in Appendix E.1. This improvement shows that it is possible to achieve sublinear-in-T query complexity without paying for a comparable blowup in sample complexity.

For more detail, we refer the reader to Appendix F, which presents full pseudocode for a simplified version of `AutoLearn` *that uses a single level of recursion, along with a complete proof that it achieves* $n_{\texttt{sample}} = \widetilde{\mathcal{O}}(d)$ *and* $n_{\texttt{query}} = \widetilde{\mathcal{O}}(d\sqrt{T})$.

E.6 $2^{\widetilde{\mathcal{O}}(\sqrt{\log(T)})}$ Query and Sample Complexity via Deeper Recursion

The depth-1 algorithm described above uses a full-CoT learner (i.e., vanilla next-token prediction) to weakly learn $\pi^\star_\tau$ for each distribution over length-τ instances, $\widehat{\rho}_j$. Each invocation of this weak learning algorithm requires $\Omega(\tau)$ queries. To achieve the main guarantee for the full version of `AutoLearn` in Theorem 3.3, we use a deeper recursion in which the weak learner at the length-τ scale is itself an instantiation of the depth-1 algorithm in the prequel (for $T = \tau$). We iterate this recursion over multiple levels, decomposing instances into smaller and smaller sub-instances, with the length decreasing by a factor of L at each level. We refer the reader back to Figure 5 for the overall structure of the algorithm, with the conditioning and change-of-measure steps in Appendix G replacing the idealized block-mixture guarantee.

In the approach above, the depth of the recursion H and the decomposition factor L are free parameters, but need to satisfy $T = L^H$ to ensure that the deepest level of recursion (i.e., depth 0) corresponds to solving length-1 problems (which is achieved through fitting a next-token predictor). The analysis of inverted sampling we worked out in Lemma E.4 plays a key role in analyzing the sample complexity and query complexity of the resulting algorithm. In particular, when we extend the same analysis to deeper levels of recursion, at any depth $h \leqslant H$ (corresponding to solving instances of length T/L^{H-h}), generating samples from the appropriate weak learning distribution at this depth requires drawing $(\log(L))^{\mathcal{O}(H-h)}$ examples from ρ. In particular, this implies that the total sample complexity of the algorithm scales as $n_{\texttt{sample}} = d \cdot (\log(L))^{\mathcal{O}(H)}$, noting that we require $\widetilde{\mathcal{O}}(d)$ samples at the deepest level to train a model via next-token prediction to achieve constant accuracy. On the other hand, the overall query complexity of the algorithm scales as $n_{\texttt{query}} \leqslant LHn_{\texttt{sample}}$: at each depth h of the algorithm, inverted sampling subselects a single length-T/L^{H-h+1} instance from a length T/L^{H-h}-length instance. This means that on each input in the dataset, inverted sampling makes LH `iCoT` queries, leading to the following bound on the overall query complexity:

$$n_{\texttt{sample}} = d \cdot (\log(L))^{\mathcal{O}(H)} \text{ and } n_{\texttt{query}} = LH \cdot d \cdot (\log(L))^{\mathcal{O}(H)}$$

While the sample complexity alone is minimized with $H = 1$ and $L = T$, the query complexity is large in this regime, scaling as $\mathcal{O}(dT)$. However, by choosing L and H differently, both quantities can be made to scale subpolynomially in T. In particular, with $L = 2^{\sqrt{\log T}}$ (which corresponds to $H = \sqrt{\log T}$ to ensure $L^H = T$), we balance the L and $(\log(L))^{\mathcal{O}(H)}$ terms in the query complexity, which both scale as $2^{\widetilde{\mathcal{O}}(\sqrt{\log(T)})}$. With this choice, both the sample and query complexity scale as $2^{\widetilde{\mathcal{O}}(\sqrt{\log T})} \cdot d$, yielding Theorem 3.3.

Algorithm 7 `AutoLearn.warmup`$(D_{\text{input}} \| T, \delta)$: Learning semiautomata via curriculum

Input: Class of semiautomaton transitions Π over state space S and alphabet Σ,
Number of steps of semiautomaton simulation, T,
Target failure probability δ,
Dataset of instances $D_{\text{input}} = \{\boldsymbol{x}^i = (s_0^i, \boldsymbol{w}^i)\}_{i=1}^n$.
Hyperparameters: $L \in \mathbb{N}$ such that $T \pmod L \equiv 0$.
Instantiate: $\tau = T/L$,
$k = C \log(L)$ for a large constant $C > 0$,
Split D_{input} into k equal parts, $\{D_{\text{input}}^j : j \in \{0, \cdots, k-1\}\}$.
for $j = 0, 1, \cdots, k-1$, **do**
  $D_{\text{out}}^j \leftarrow \emptyset$.
  Let $\Pi_j \leftarrow \{\widehat{\pi}_\tau^0, \cdots, \widehat{\pi}_\tau^{j-1}\}$. ▶ By convention, $\Pi_0 = \emptyset$
  for instances $\boldsymbol{x} = (s_0, \boldsymbol{w}) \in D_{\text{input}}^j$ **do**
    $D_{\boldsymbol{x}} = \texttt{Split}(\boldsymbol{x} \| L)$. ▶ Defined in Algorithm 4
    $O \leftarrow \texttt{InvSampling}(D_{\boldsymbol{x}} \| \Pi_j, k)$. ▶ Defined in Algorithm 3
    If $O \neq \perp$, update $D_{\text{out}}^j \leftarrow D_{\text{out}}^j \cup \{O\}$
  Train model $\widehat{\pi}^j \leftarrow \texttt{NTP}_{\texttt{Full-CoT}}(D_{\text{out}}^j \| \tau, \delta/2k)$
Return: $\widehat{\pi} = \widetilde{\pi}^{\circ L}$, where $\widetilde{\pi} = \texttt{Plu}(\{\widehat{\pi}_\tau^j : j \in \{0, \cdots, k-1\}\})$. ▶ `Plu`$(\cdot)$ returns the plurality of the models in its argument

F Analyzing a Simpler Variant of Algorithm 1: `AutoLearn.warmup`

This section presents a complete description and analysis for the simplified version of `AutoLearn` sketched in Appendix E, which uses only a single level of recursion. The algorithm, which we denote by `AutoLearn.warmup`, decomposes length-T instances into length-τ ones, where $\tau = T/L$ and $L > 1$ is a parameter to be specified. Then, a sequence of models $\widehat{\pi}^0, \cdots, \widehat{\pi}^{k-1}$ are trained to solve these length-τ instances, which are then aggregated using majority voting and composed L-fold to get a model which solves length-T instances. Each $\widehat{\pi}^j$ is trained by querying full CoTs on length-τ instances and carrying out next-token prediction (Definition 2.2), which we refer to as $\texttt{NTP}_{\texttt{Full-CoT}}$ (Algorithm 8). A pictorial depiction is provided in Figure 4, and a formal description of `AutoLearn.warmup` is provided within Algorithm 7.

The purpose of analyzing this variant of `AutoLearn` is to expose and understand the mechanism by which inverted sampling (Algorithm 3) converts longer length-T instances (i.e., the dataset D_{input}) into shorter ones of length $\tau = T/L$ (i.e., the datasets D_{out}^j for $j = 0, \cdots, k-1$). Our main guarantee, established below, shows that `AutoLearn.warmup` achieves sample complexity scaling as $\widetilde{\mathcal{O}}(d)$ and query complexity $\widetilde{\mathcal{O}}(d\sqrt{T})$ in order to achieve constant error on length-T instances.

Theorem F.1 (Guarantee for `AutoLearn.warmup`)**.** `AutoLearn.warmup` *(Algorithm 7) for square $T \geqslant 4$, with $L = \sqrt{T}$, draws a total of,*

$$n_{\texttt{sample}} = \mathcal{O}\big(d \log(|S|) \log(1/\delta) \cdot \log^9(T)\big)$$

instances from ρ, and makes $n_{\texttt{query}} = \mathcal{O}(n_{\texttt{sample}}\sqrt{T})$ calls to the `iCoT`$(\cdot)$ *oracle to return a model $\widehat{\pi}$ such that with probability $1 - \delta$,*

$$\Pr_{\boldsymbol{x} \sim \rho}\big[\widehat{\pi}(\boldsymbol{x}) \neq \pi_T^\star(\boldsymbol{x})\big] \leqslant \frac{1}{4}$$

Proof. The proof sketch of this result and all necessary lemmas are introduced in Appendix F.1. The final proof invoking these lemmas is presented in Appendix F.3. □

F.1 Proof Sketch of Theorem F.1

In this section, we sketch the proof of Theorem F.1, first introducing some relevant notation. We will let $\widehat{\pi}$ denote the model returned by Algorithm 7.

Algorithm 8 $\texttt{NTP}_{\texttt{Full-CoT}}(D\|\tau,\delta)$: Full-CoT next-token prediction (Joshi et al., 2025)

Learn a model solving length-τ instances by labeling all intermediate states and running the NTP oracle.

Input: Dataset of length-τ instances $D = \{\boldsymbol{x}^i = (s_0^i, \boldsymbol{w}_{1:\tau}^i)\}_{i=1}^n$,
Supervision oracle $\texttt{iCoT}(\cdot)$, NTP oracle $\texttt{NTP}(\cdot)$ for Π,
Failure probability δ.

Instantiate: $D_{\text{step}} \leftarrow \emptyset$.

for $i = 1, \cdots, n$ **do**

  for $t = 0, \cdots, \tau - 1$ **do**

    Query intermediate state: $s_{t+1}^i \leftarrow \texttt{iCoT}\big((s_0^i, \boldsymbol{w}_{1:t+1}^i)\big)$.

    $D_{\text{step}} \leftarrow D_{\text{step}} \cup \big\{(s_t^i, w_{t+1}^i) \mapsto s_{t+1}^i\big\}$.

Return: $\widehat{\pi} \leftarrow \texttt{NTP}(D_{\text{step}})$.

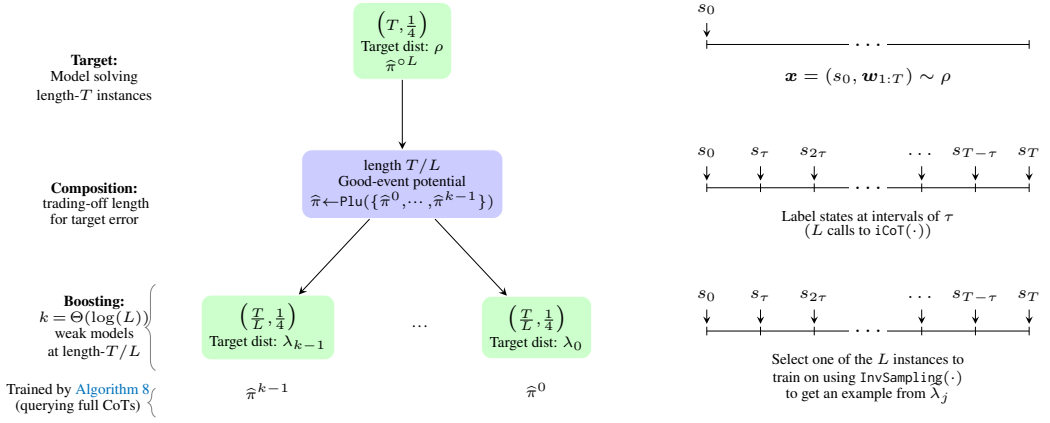


Figure 4: A depiction of `AutoLearn.warmup`. The notation (T, ε) indicates the length of instances being solved and the desired error tolerance under the indicated target law, respectively. Actual training samples follow $\widehat{\lambda}_j$; change of measure transfers weak learning to λ_j.

Notation. Recall the definition of $\rho_{t_1:t_2}$ which is the distribution of $(s_{t_1-1}, \boldsymbol{w}_{t_1:t_2})$ for $(s_0, \boldsymbol{w}_{1:T}) = \boldsymbol{x} \sim \rho$ and where $s_{t_1-1} = \pi^\star_{t_1-1}(\boldsymbol{x})$. Let σ denote the joint distribution of $\underline{X} = (X_0, \cdots, X_{L-1})$, where each $X_i = (s_t, \boldsymbol{w}_{t+1:t+\tau})$, for $t = \tau i$ and $\boldsymbol{x} = (s_0, \boldsymbol{w}_{1:T}) \sim \rho$. Note that $\underline{X} \sim \sigma$ is a measurable function of the initial state s_0 and the word $\boldsymbol{w}_{1:T}$, and can be constructed from the instance $\boldsymbol{x} = (s_0, \boldsymbol{w}_{1:T}) \sim \rho$ by querying $\texttt{iCoT}(\cdot)$ at L timesteps ($t = \tau, 2\tau, \cdots, T$). Let $\boldsymbol{X} = S \times \Sigma^\tau$ denote the space of length-τ instances. Note that the support of σ is contained in $\boldsymbol{X}^L$. *Note that the distributions σ and ρ, while closely related, are different from one another.* We also define the filtration $\mathcal{H}_{j-1}$ which captures all the randomness up until just before the dataset D_{out}^j is collected in iteration j (i.e., the randomness in the construction of the models $\widehat{\pi}^0, \cdots, \widehat{\pi}^{j-1}$ and the randomness in the instances $\boldsymbol{x} \sim \rho$ which contribute to the datasets $D_{\text{out}}^0, \cdots, D_{\text{out}}^{j-1}$). In particular, $\mathcal{H}_{k-1}$ denotes all the random variables involved in the execution of `AutoLearn.warmup`.

With this, we let $\overline{\sigma}$ denote the distribution $\frac{1}{L}\sum_{i=0}^{L-1} \sigma(X_i = \cdot)$, which is the uniform mixture over marginals of σ, which is identical to $\overline{\rho}$. Thus, our earlier discussion in Observation 3.5 demonstrating the central role played by $\overline{\rho}$ in enabling compositional generalization applies to $\overline{\sigma}$ as well. All-in-all, the purpose of introducing additional notation in the form of σ is to make it easier to write down the distributions over length-τ instances on which the weak learners $\widehat{\pi}^j$ are trained.

First, we define a sequence of distributions, $\{\overline{\rho}_j\}_{j=0}^{k-1}$ supported on $\boldsymbol{X}$ obtained by applying a tilt to $\overline{\sigma}$ (or equivalently $\overline{\rho}$),

$$\overline{\rho}_j(\cdot) \propto \Big(\frac{1}{L}\sum_{i=0}^{L-1} \sigma(X_i = \cdot)\Big) w_j(\cdot) = \overline{\sigma}(\cdot)\, w_j(\cdot) \tag{11}$$

where $w_j(\cdot)$ is the rank-based weight function defined in Eq. (2). $\overline{\rho}_j$ is the idealized "good distribution" under which the weak model in iteration j should be trained as prescribed by boosting-by-filtering (Freund, 1995). Sampling from $\overline{\rho}_j$ is prohibitive, a point which we discuss more in Section I.2.2. To this end, we define an alternate sequence of distributions which are easier to sample from, denoted $\{\lambda_j\}_{j=0}^{k-1}$,

$$\lambda_j(\cdot) \propto \overline{\sigma_{\mathcal{K}_j}}(\cdot)\, w_j(\cdot), \text{ where, } \overline{\sigma_{\mathcal{K}_j}}(\cdot) = \Big(\frac{1}{L}\sum_{i=0}^{L-1} \Pr\big(X_i = \cdot \mid (\underline{X}, \boldsymbol{\eta}_{0:j}) \in \mathcal{K}_j \text{ and } \mathcal{H}_{j-1}\big)\Big) \tag{12}$$

where the probability is calculated under $\underline{X} = (X_i)_{i=0}^{L-1} \sim \sigma$ and $\boldsymbol{\eta}_j = (\eta_{j,i})_{i=0}^{L-1} \sim \mathrm{Unif}([0,1])^{\otimes L}$, is a set of i.i.d. uniform random variables, with $\boldsymbol{\eta}_{0:j} = (\boldsymbol{\eta}_{j'})_{j'=0}^{j}$ and $\boldsymbol{\eta} = \boldsymbol{\eta}_{0:k-1}$. This definition hinges on the sequence of "good" events $(\mathcal{K}_j)_{j=0}^{k-1}$, such that $\mathcal{K}_j \subseteq \boldsymbol{X}^L \times [0,1]^{(j+1)L}$, which we defer the definition of to later in Eq. (21). Finally, define the sequence of random variables $\{R_j\}_{j=0}^{k-1}$ which are measurable functions of $\underline{X} \sim \sigma$,

$$R_j = \sum_{i=0}^{L-1} w_j(X_i), \text{ where, } w_j(X_i) = \alpha^{j,k}_{\mathrm{rank}_j(X_i)} \text{ and } \mathrm{rank}_j(X_i) = \sum_{\ell=0}^{j-1} \mathbb{I}(\widehat{\pi}^\ell_\tau(X_i) = \pi^\star_\tau(X_i)), \tag{13}$$

where $\alpha^{j,k}_r$ is defined in Eq. (2). R_j will be referred to as the *regression* in iteration j. The name "regression" is used to denote the fact that R_j is a measure of how much the accuracy of the plurality of the models $\widehat{\pi}^0, \cdots, \widehat{\pi}^{j-1}$ can possibly worsen by, when the new model $\widehat{\pi}^j$ is added to the ensemble.

Additional Notation. In order to state our results succinctly, we introduce some additional notation to make it easier to write down expressions involving events on different spaces. In particular for events $\mathcal{A}$ defined on $\boldsymbol{X}^L \times [0,1]^{k_A L}$ and $\mathcal{B}$ defined on $\boldsymbol{X}^L \times [0,1]^{k_B L}$ with $k_A \geqslant k_B$, we can define intersections by considering the smallest subset of $(\underline{X}, \boldsymbol{\eta})$ with respect to which both events are measurable. Namely,

$$\mathcal{A} \cap \mathcal{B} \iff \big\{(\underline{X}, \boldsymbol{\eta}_{0:k_A-1}) : (\underline{X}, \boldsymbol{\eta}_{0:k_A-1}) \in \mathcal{A} \text{ and } (\underline{X}, \boldsymbol{\eta}_{0:k_B-1}) \in \mathcal{B}\big\}. \tag{14}$$

Similarly, for a third event $\mathcal{C} \subseteq \boldsymbol{X}^L \times [0,1]^{k_C L}$ with $k_C \geqslant k_A$, containment statements of the form,

$$\mathcal{C} \subseteq \mathcal{A} \cap \mathcal{B} \iff \mathcal{C} \subseteq \big\{(\underline{X}, \boldsymbol{\eta}_{0:k_C-1}) : (\underline{X}, \boldsymbol{\eta}_{0:k_A-1}) \in \mathcal{A} \text{ and } (\underline{X}, \boldsymbol{\eta}_{0:k_B-1}) \in \mathcal{B}\big\}. \tag{15}$$

are defined by defining both sides with respect to the minimal subset of $(\underline{X}, \boldsymbol{\eta})$ with respect to which all events are measurable. A similar logic can be used to parse statements of the form $(\underline{X}, \boldsymbol{\eta}) \in \mathcal{K}_j$ by viewing the LHS as the singleton event in $\boldsymbol{X}^L \times [0,1]^{kL}$ and defining membership via containment: $\{(\underline{X}, \boldsymbol{\eta})\} \subseteq \mathcal{K}_j$.

We first decompose the test error of the model $\widehat{\pi}$ returned by `AutoLearn.warmup` by a potential function argument, to track how the test error of the aggregated ensemble improves as more models are added.

Lemma F.2 (Test-error decomposition)**.** *Let $\widehat{\pi}$ denote the model returned by* `AutoLearn.warmup`*. Fix a fresh test tuple independent of training. For a nested sequence of events, each $\mathcal{K}_j$ determined by the history $\mathcal{H}_{j-1}$ and test variables, $\mathcal{K}_{k-1} \subseteq \cdots \subseteq \mathcal{K}_0 \subseteq \boldsymbol{X}^L \times [0,1]^{kL}$, the prediction error of $\widehat{\pi}$ under ρ can be upper bounded as,*

$$\begin{aligned}
&\Pr_{\boldsymbol{x}\sim\rho}\big[\widehat{\pi}(\boldsymbol{x}) \neq \pi^\star_T(\boldsymbol{x})\big] \\
&\leqslant L \cdot \beta^{0,k}_0 + \sum_{j=0}^{k-1} (\mathtt{err}_j - \mathtt{err}_\star) \cdot \mathbb{E}[R_j \cdot \mathbb{I}((\underline{X}, \boldsymbol{\eta}) \in \mathcal{K}_j) \mid \mathcal{H}_{k-1}] \\
&\qquad + \Pr((\underline{X}, \boldsymbol{\eta}) \notin \mathcal{K}_{k-1} \mid \mathcal{H}_{k-1})
\end{aligned}$$

Recall that, $\beta^{j,k}_r$ is the weight function defined across Eqs. (2) and (3), $\mathtt{err}_\star = \frac{1}{4}$, while $\mathtt{err}_j$ is the test error of $\widehat{\pi}^j$ under the distribution $\lambda_j \propto \overline{\sigma_{\mathcal{K}_j}}(\cdot) w_j(\cdot)$ where w_j is defined in Eq. (13),

$$\mathtt{err}_j = \Pr_{X\sim\lambda_j}\big[\widehat{\pi}^j_\tau(X) \neq \pi^\star_\tau(X)\big].$$

Proof. The proof of this result is presented in Appendix I.1. □

Note that Lemma F.2 allows the events $\mathcal{K}_j$ to be defined on a subset of $\boldsymbol{X}^L \times [0,1]^{kL}$ and the containment $\mathcal{K}_j \subseteq \mathcal{K}_{j-1}$ is defined via Eq. (15). Overall this result decomposes the test error of $\widehat{\pi}$ into 3 terms; the remainder of the proof of Theorem F.1 is to show that, with high probability, each of these terms is upper bounded by a constant.

The proof of Lemma F.2 introduces a potential function Φ_j which tracks the performance of the plurality of the models $\widehat{\pi}^0, \cdots, \widehat{\pi}^{j-1}$ trained before step j. The term $L \cdot \beta_0^{0,k}$ bounds the initial value of this potential, Φ_0. The quantity $\beta_0^{0,k}$ decays exponentially with k, and so, as long as $k = \Omega(\log(L))$, the first term is bounded by a sufficiently small absolute constant, say $\frac{1}{16}$, which is within the target error guarantee of Theorem F.1. This bound on k should be viewed as a requirement on the number of weak models to be trained for the initial potential term $L\beta_0^{0,k}$ to be small.

The remaining terms in Lemma F.2 are easiest to interpret when $\mathcal{K}_{k-1} = \cdots = \mathcal{K}_0 = \boldsymbol{X}^L \times [0,1]^{kL}$, where they reduce to $\sum_{j=0}^{k-1}(\texttt{err}_j - \texttt{err}_\star) \cdot \mathbb{E}[R_j \mid \mathcal{H}_{k-1}]$. This expression brings out the role of the regression R_j and a clean tradeoff to establish: in iterations where generating training examples from λ_j is easy, $\texttt{err}_j$ is likely to be smaller than the threshold $\texttt{err}_\star = \frac{1}{4}$, resulting in $(\texttt{err}_j - \texttt{err}_\star) \cdot \mathbb{E}[R_j \mid \mathcal{H}_{k-1}] \leqslant 0$. In the iterations where sampling from λ_j is expensive, we need to argue that R_j is also likely to be small. The interpretation of such a result is that, in the iterations where sampling from λ_j is expensive, there is sufficient leeway in the plurality of the existing models that adding an arbitrary $\widehat{\pi}^j$ to the ensemble does not hurt the error significantly.

Toward establishing such a guarantee, we first define an "abort" event which captures whether the dataset collected in iteration j is sufficiently large. This is a proxy for whether sampling from λ_j is easy, and will be a sufficient condition to capture whether $\texttt{err}_j \leqslant \texttt{err}_\star$. We first introduce some notation in order to define this event. Let,

$$p_j = \Pr_{\boldsymbol{x}\sim\rho}\big(\texttt{InvSampling}(D_{\boldsymbol{x}} \parallel \Pi_j, k) \neq \perp \;\big|\; \mathcal{H}_{j-1}\big), \text{ where, } D_{\boldsymbol{x}} = \texttt{Split}(\boldsymbol{x} \| L) \tag{16}$$

denote the probability that an instance $\boldsymbol{x} \sim \rho$ contributes a shorter instance O to the dataset D_{out}^j through split-and-inverted-sampling (cf. Algorithms 3 and 4). It is a short calculation to derive the formula,

$$p_j = \Pr(|\mathcal{I}_j| \geqslant 1 \mid \mathcal{H}_{j-1}) = \mathbb{E}\left[1 - \prod_{i=0}^{L-1}\left(1 - \frac{\alpha_{\text{rank}_j(X_i)}^{j,k}}{\alpha_{\max}^{j,k}}\right) \,\middle|\, \mathcal{H}_{j-1}\right] \tag{17}$$

where $\mathcal{I}_j = \left\{0 \leqslant i \leqslant L-1 : \eta_{j,i} \leqslant \frac{w_j(X_i)}{\alpha_{\max}^{j,k}}\right\}$ denotes the random set of accepted indices in a run of InvSampling (Algorithm 3) when applied to $D_{\boldsymbol{x}}$ for $\boldsymbol{x} \sim \rho$ in the j^{th} iteration; $\underline{X} = (X_i)_{i=0}^{L-1}$ is obtained by decomposing the length-T instance $\boldsymbol{x}$ into L length-τ instances.[13] For completeness, we prove Eq. (17) later via the chain of equations in Eq. (61) to Eq. (62). We also define a slight modification of p_j, which captures the same acceptance probability for split-and-inverted-sampling, but conditioned on the event $\mathcal{K}_{j-1}$,

$$\widetilde{p}_j = \Pr\left(|\mathcal{I}_j| \geqslant 1 \mid \mathcal{K}_{j-1}, \mathcal{H}_{j-1}\right), \text{ and define } p_j^\star = \frac{1}{1024k^{5/2}}. \tag{18}$$

While p_j above captures the probability that a sample $\boldsymbol{x} \sim \rho$ is accepted by split-and-inverted-sampling, we define the abort event in terms of $\widetilde{p}_j$. In particular, this implies that there may be iterations j where the dataset D_{out}^j is large (i.e., p_j is large), but the iteration still aborts because the effective dataset collected (as determined by $\widetilde{p}_j$) is too small. Define,

$$\texttt{ABORT}[j] = \left\{\widetilde{p}_j \leqslant p_j^\star\right\} \tag{19}$$

F.1.1 Analyzing terms in the test-error decomposition of Lemma F.2

Our main argument to analyze the main summation $\sum_{j=0}^{k-1}(\texttt{err}_j - \texttt{err}_\star) \cdot \mathbb{E}[R_j \cdot \mathbb{I}((\underline{X}, \boldsymbol{\eta}) \in \mathcal{K}_j) \mid \mathcal{H}_{k-1}]$ in Lemma F.2 is shown next. We will split our analysis of the main summation into two parts,

[13] i.e., define $\underline{X} = (X_j)_{j=0}^{L-1}$ with $X_j = (s_{j\tau}, \boldsymbol{w}_{j\tau+1:(j+1)\tau})$, is constructed from $\boldsymbol{x} = (s_0, \boldsymbol{w}_{1:T})$.

one across iterations that abort, and the other across iterations that don't abort. The high-level idea is to show that in any iteration j which does not abort, $\texttt{err}_j \leqslant \texttt{err}_\star$ with high probability, which means that the corresponding term in the summation is non-positive; in any iteration j which aborts, the regression R_j is small with moderate probability, and using the event $\mathcal{K}$ to avoid the potentially long tails of this random variable. Throughout the sketch, we will emphasize the conditions required to define the event $\mathcal{K}$, before ultimately defining it.

Aborted iterations. An aborted iteration has small $\widetilde{p}_j$ by definition; this is an analytical event, not an instruction to halt. We use this to show, in Lemma F.7, that the regression R_j is typically small: it will only ever exceed some threshold $\mathfrak{r}_j$ with moderately low probability $q_j \lesssim \frac{1}{k}$. Suppose the event $\mathcal{K}_j$ satisfies the following condition in any iteration j,

$$(\underline{X}, \boldsymbol{\eta}) \in \mathcal{K}_j \implies \big\{\texttt{ABORT}[j] \text{ and } R_j > \mathfrak{r}_j\big\}^c. \tag{C1}$$

Then, in any iteration j which aborts, we can bound the corresponding term in the summation in Lemma F.2 by,

$$\begin{aligned}(\texttt{err}_j - \texttt{err}_\star) \cdot \mathbb{E}[R_j \cdot \mathbb{I}((\underline{X}, \boldsymbol{\eta}) \in \mathcal{K}_j) \mid \mathcal{H}_{k-1}] &\leqslant \mathbb{E}[R_j \cdot \mathbb{I}(R_j \leqslant \mathfrak{r}_j) \mid \mathcal{H}_{k-1}] \\ &\leqslant \mathfrak{r}_j.\end{aligned}$$

The $\mathfrak{r}_j$'s are shown to satisfy the upper bound, $\sum_{j=0}^{k-1} \mathfrak{r}_j \leqslant \frac{1}{8}$ in Lemma F.8. This means that the contribution to the summation across all aborted iterations is bounded by a constant. The role of the failure probability on the regression upper bound, q_j, will appear later in showing that the (yet to be defined) sequence of events $\mathcal{K}_0, \cdots, \mathcal{K}_{k-1}$ satisfies with high probability,

$$\Pr((\underline{X}, \boldsymbol{\eta}) \notin \mathcal{K}_{j-1} \mid \mathcal{H}_{j-1}) \leqslant \frac{1}{16} \tag{C2}$$

With $j = k$, this is the final term in the error decomposition in Lemma F.2 and is established in Lemma F.4. The same lower bound on the probability of each $\mathcal{K}_j$ will also be useful in showing guarantees for the weak models, $\widehat{\pi}^j$, trained in non-aborted iterations. This is discussed as a part of the next section.

Non-aborted iterations. If iteration j does not abort, we show in Lemma F.6 that with high probability, $\texttt{err}_j \leqslant \texttt{err}_\star$. By extension, this implies that the corresponding term in the summation in Lemma F.2 $(\texttt{err}_j - \texttt{err}_\star) \cdot \mathbb{E}[R_j \cdot \mathbb{I}((\underline{X}, \boldsymbol{\eta}) \in \mathcal{K}_j) \mid \mathcal{H}_{k-1}] \leqslant 0$. Recall $\texttt{err}_j = \Pr_{X \sim \lambda_j}\big[\widehat{\pi}^j_\tau(X) \neq \pi^\star_\tau(X)\big]$ itself is the test error of $\widehat{\pi}^j$ under the distribution of instances $\lambda_j \propto \overline{\sigma_{\mathcal{K}_j}}(\cdot) w_j(\cdot)$, a distribution which depends on the event $\mathcal{K}_j$. Letting $\widehat{\lambda}_j$ denote the distribution over instances which examples in the i.i.d. dataset D^j_{out} follow, Lemma F.6 is proved by showing that the following condition,

$$\left\| \frac{\lambda_j}{\widehat{\lambda}_j} \right\|_\infty \leqslant \mathcal{O}\left(\frac{k \log(L)}{\widetilde{p}_j}\right) \tag{C3}$$

is satisfied for all iterations j which do not abort, as long as $\mathcal{K}_j$ is defined appropriately in terms of $\mathcal{K}_{j-1}$ and $\mathcal{U}_j$ (Lemma F.3). From the definition of an aborted iteration, we know that $\widetilde{p}_j$ is at least $p^\star_j$, which implies a density ratio upper bound scaling as $k^{\mathcal{O}(1)} \log(L)$. Note finally, that examples in the distribution D^j_{out} come from λ_j, and the acceptance probability of split-and-inverted-sampling to accept $\boldsymbol{x} \sim \rho$ to generate a sample from this distribution is p_j; noticing that,

$$p_j = \Pr\big(|\mathcal{I}_j| \geqslant 1 \mid \mathcal{H}_{j-1}\big) \geqslant \Pr\big(|\mathcal{I}_j| \geqslant 1 \mid \mathcal{K}_{j-1}, \mathcal{H}_{j-1}\big) \cdot \Pr\big(\mathcal{K}_{j-1} \mid \mathcal{H}_{j-1}\big) \gtrsim \widetilde{p}_j$$

where the last inequality uses (C2).

Finally, `AutoLearn.warmup` uses next-token prediction to learn the model $\widehat{\pi}^j$ on this dataset. Using prior guarantees for next-token prediction (Joshi et al., 2025) and a change-of-measure argument using (C3), we have learning guarantees for $\widehat{\pi}^j$ under λ_j.

The existence of an event $\mathcal{K}_j$ establishing (C1), (C2) and (C3) is established in Lemmas F.3 and F.4. The above analysis across aborted and non-aborted iterations combines to simplify the summation in Lemma F.2 to the following inequality: with high probability,

$$\Pr_{\boldsymbol{x} \sim \rho}\big[\widehat{\pi}(\boldsymbol{x}) \neq \pi^\star_T(\boldsymbol{x})\big]$$

$$\leqslant L \cdot \beta_0^{0,k} + \sum_{j=0}^{k-1} \mathfrak{r}_j + \Pr((\underline{X}, \boldsymbol{\eta}) \notin \mathcal{K}_{k-1} \mid \mathcal{H}_{k-1}) \leqslant \frac{1}{16} + \frac{1}{8} + \frac{1}{16} = \frac{1}{4} \tag{20}$$

The final inequality follows from the bound on $\sum_{j=0}^{k-1} \mathfrak{r}_j \leqslant \frac{1}{8}$, and choosing k to be sufficiently large (as $\Theta(\log(L))$), which gives $L \cdot \beta_0^{0,k} \leqslant \frac{1}{16}$. This uses a bound which shows that $\beta_0^{0,k}$ decays exponentially fast in k (Lemma F.9).

F.2 Proving Theorem F.1: relevant lemmata

Having established the broader program, we introduce the lemmas mentioned above. We will focus on the non-aborted iterations first, and then discuss the aborted iterations next.

F.2.1 Non-aborted iterations $\left(\widetilde{p}_j > p_j^\star\right)$.

In iterations j which do not abort, we show that $\widehat{\pi}^j$ achieves low test error under λ_j with high probability, establishing a weak learning guarantee. Since the proof relies heavily on the $\mathcal{K}_j$ events, we first define these events before proving that they satisfy the structural conditions laid out in (C1) to (C3).

Definition of $(\mathcal{K}_j)_{j=0}^{k-1}$. For each $0 \leqslant j \leqslant k-1$, the event $\mathcal{K}_j \subseteq \boldsymbol{X}^L \times [0,1]^{(j+1)L}$, we construct, is a function of $\underline{X} \sim \sigma$ and $\boldsymbol{\eta}_{0:j}$; recall $\boldsymbol{\eta} = (\boldsymbol{\eta}_{j'} : 0 \leqslant j' \leqslant k-1)$, where $\boldsymbol{\eta}_{j'} = (\eta_{j',i})_{i=0}^{L-1} \sim \mathrm{Unif}([0,1])^{\otimes L}$ is a sequence of i.i.d. random variables. The $\mathcal{K}_j$ events will be constructed recursively: for all $0 \leqslant j \leqslant k-1$,

$$\mathcal{K}_j \leftarrow \texttt{ConstructEvent}(\mathcal{U}_j \cap \mathcal{K}_{j-1} \,\|\, c, \sigma, \mathcal{H}_{j-1}) \text{ where } c = \frac{1}{32k}, \tag{21}$$

with $\mathcal{K}_{-1} = \boldsymbol{X}^L$. The definition of `ConstructEvent` is deferred to later (Lemma F.3), and we define the $(\mathcal{U}_j)_{j=0}^{k-1}$ events next. Each $\mathcal{U}_j \subseteq \boldsymbol{X}^L$ is a *random* event which is a measurable function of $\mathcal{H}_{j-1}$, and is defined as,

$$\mathcal{U}_j = \left\{\widetilde{p}_j \leqslant p_j^\star \text{ and } R_j > \mathfrak{r}_j\right\}^c. \tag{22}$$

Here, the regression, R_j was defined earlier in Eq. (13) and the thresholds $\mathfrak{r}_j > 0$ are defined later in Eq. (31), and the conditional acceptance probability $\widetilde{p}_j$ is defined in Eq. (18).

In order to complete the definition of the $\mathcal{K}_j$'s, we still need to define the `ConstructEvent` algorithm which defines $\mathcal{K}_j$ in terms of $\mathcal{U}_j$ and $\mathcal{K}_{j-1}$. In the subsequent lemma, we show the existence of such an algorithm which enables the resulting $\mathcal{K}_j$'s to satisfy the structural constraints in (C1) to (C3).

Lemma F.3 ($\mathcal{K}_j$'s satisfy (C1) and (C3))**.** *Fix some $0 \leqslant j \leqslant k-1$, and recall that $\widehat{\lambda}_j$ denotes the distribution over instances the i.i.d. dataset D_{out}^j is sampled from. Consider any filtration $(\mathcal{H}_j)_{j=0}^{k-1}$ and sequence of events, $(\mathcal{U}_j)_{j=0}^{k-1}$ such that $\mathcal{U}_j \subseteq \boldsymbol{X}^L$ and is a measurable function of $\mathcal{H}_{j-1}$. Fix some constant $c \in (0,1)$. There exists an algorithm $\texttt{ConstructEvent}(\cdot \,\|\, c, \sigma, \mathcal{H}_{k-1})$ which iteratively constructs a sequence of events $\mathcal{K}_j \subseteq \boldsymbol{X}^L \times [0,1]^{(j+1)L}$ as:*

$$\mathcal{K}_j \leftarrow \texttt{ConstructEvent}(\mathcal{U}_j \cap \mathcal{K}_{j-1} \,\|\, c, \sigma, \mathcal{H}_{j-1})$$

where $\mathcal{K}_{-1} = \boldsymbol{X}^L$. The $\mathcal{K}_j$ events satisfy the following conditions:

$$\mathcal{K}_j \subseteq \mathcal{U}_j \cap \mathcal{K}_{j-1} \quad \text{and} \quad \Pr((\underline{X}, \boldsymbol{\eta}) \notin \mathcal{K}_j \mid (\underline{X}, \boldsymbol{\eta}) \in \mathcal{U}_j \cap \mathcal{K}_{j-1}, \mathcal{H}_{j-1}) \leqslant c \tag{23}$$

Finally, the distribution $\lambda_j(\cdot) \propto \overline{\sigma_{\mathcal{K}_j}}(\cdot) w_j(\cdot)$ over $\boldsymbol{X}$ introduced in Eq. (12), which depends on $\mathcal{K}_j$ satisfies the following condition,

$$\left\| \frac{\lambda_j}{\widehat{\lambda}_j} \right\|_\infty \lesssim \frac{c^{-1} \log(L)}{\Pr\left(\mathcal{U}_j \cap \mathcal{K}_{j-1} \,|\, |\mathcal{I}_j| \geqslant 1, \mathcal{H}_{j-1}\right)} \tag{24}$$

Here, $\mathcal{I}_j$, defined in Eq. (17), is the set of accepted indices in a run of split-and-inverted-sampling (Algorithms 3 and 4) for $\boldsymbol{x} \sim \rho$ in the j^{th} iteration.

Proof. The proof of this lemma is discussed in Appendix I.2. `ConstructEvent`$(\cdot)$ is defined in Eq. (67). □

Interpretation of Lemma F.3. In order to interpret this result, consider the special case where $k = 1, \mathcal{U}_0 = \boldsymbol{X}^L$ and c is small (we remove the j subscript on $\mathcal{K}_j, w_j$ for succinctness). In this setting, `ConstructEvent` creates an event $\mathcal{K} \subseteq \boldsymbol{X}^L \times [0,1]^L$ such that, (a) $\Pr(\mathcal{K}) \geqslant 1 - c$ is a high probability event, and furthermore, (b) $\left\|\frac{\nu}{\widehat{\nu}}\right\|_\infty \lesssim c^{-1}\log(L)$ where ν is the distribution $\propto \overline{\sigma_{\mathcal{K}}}(\cdot)w(\cdot)$ and $\widehat{\nu}$ is the distribution over inputs realized by split-and-inverted-sampling for $\boldsymbol{x} \sim \rho$. We will focus on this special case and discuss it further in Appendix I.2, showing that this guarantee is an exponential improvement over the best guarantee achievable if the event $\mathcal{K}$ was forced to equal $\boldsymbol{X}^L \times [0,1]^L$.

While Lemma F.3 gives us a bound on the density ratio between λ_j and $\widehat{\lambda}_j$, the term in the denominator depends on the quantity $\Pr\big(\mathcal{U}_j \cap \mathcal{K}_{j-1} \mid |\mathcal{I}_j| \geqslant 1, \mathcal{H}_{j-1}\big)$ which is a-priori different from the bound we hinted at earlier in (C3), involving $\widetilde{p}_j$. Our main argument will be to show that this term cannot be too small in an iteration which does not abort. Observe that when $\widetilde{p}_j > p_j^\star$, the event $\mathcal{U}_j$ is satisfied, and therefore,

$$\begin{aligned}\Pr(\mathcal{U}_j \cap \mathcal{K}_{j-1} \mid |\mathcal{I}_j| \geqslant 1, \mathcal{H}_{j-1}) &= \Pr(\mathcal{K}_{j-1} \mid |\mathcal{I}_j| \geqslant 1, \mathcal{H}_{j-1}) \qquad (25)\\ &\geqslant \Pr(|\mathcal{I}_j| \geqslant 1 \text{ and } \mathcal{K}_{j-1} \mid \mathcal{H}_{j-1})\\ &\geqslant \Pr(|\mathcal{I}_j| \geqslant 1 \mid \mathcal{K}_{j-1}, \mathcal{H}_{j-1}) \cdot \Pr(\mathcal{K}_{j-1} \mid \mathcal{H}_{j-1}) \geqslant \frac{15}{16} \cdot \widetilde{p}_j \qquad (26)\end{aligned}$$

where we use (C2) (which we establish next in Lemma F.4) to lower bound $\Pr(\mathcal{K}_{j-1} \mid \mathcal{H}_{j-1})$. By definition of the iteration j not aborting, we know that $\widetilde{p}_j > p_j^\star$ is lower bounded. And furthermore, by the definition of $p_j^\star$ as well as the manner in which we designed the sequence of events $\mathcal{K}_j$ in Eq. (21) choosing $c = \frac{1}{32k}$, we have the density ratio bound,

$$\left\|\frac{\lambda_j}{\widehat{\lambda}_j}\right\|_\infty \lesssim k^{7/2}\log(L) \qquad (27)$$

Next we establish that the $\mathcal{K}_j$ events satisfy the condition (C2), showing that it covers a moderate amount of mass.

Lemma F.4 ($\mathcal{K}_j$'s satisfy (C2))**.** *For all* $0 \leqslant j \leqslant k-1$, $\Pr(\mathcal{K}_j \mid \mathcal{H}_j) = \Pr(\mathcal{K}_j \mid \mathcal{H}_{j-1}) \geqslant \frac{15}{16}$.

Proof. The proof of this result is discussed in Appendix I.3. □

Lemma F.4 is proved by an iterative argument which peels off the contribution of the previous iterations $j' \leqslant j$ to $\mathcal{K}_j$ one step at a time. Note that these events depend on the behavior of the regression R_j across aborted iterations, and so the tools required to prove this result are discussed later in Section F.2.2.

While Lemma F.4 is used to arrive at the density ratio bound in Eq. (27), this result also gives us a handle on $\Pr(\mathcal{K}_{k-1} \mid \mathcal{H}_{k-1})$, which appears in the breakdown of the test-loss of $\widehat{\pi}$ in Lemma F.2.

Finally, note that while Eq. (27) gives us a bound on the density ratio between λ_j and $\widehat{\lambda}_j$ in iterations which don't abort, we still need to argue that the learner can generate sufficiently many samples from the latter distribution (i.e., a sufficiently large dataset D_{out}^j) to learn from in these iterations. This will follow from the fact that a lower bound on $\widetilde{p}_j$ in the non-aborted iteration, *also certifies a lower bound on* p_j *in these iterations.* Namely,

$$\widetilde{p}_j = \Pr(|\mathcal{I}_j| \geqslant 1 \mid \mathcal{K}_{j-1}, \mathcal{H}_{j-1}) \leqslant \frac{\Pr(|\mathcal{I}_j| \geqslant 1 \text{ and } \mathcal{K}_{j-1} \mid \mathcal{H}_{j-1})}{\Pr(\mathcal{K}_{j-1} \mid \mathcal{H}_{j-1})} \leqslant \frac{p_j}{(15/16)} \qquad (28)$$

Where the last inequality is by Lemma F.4 and the definition of p_j. All in all, via a concentration argument, this implies that in iterations which don't abort, the dataset D_{out}^j is unlikely to be small, and is at least $|D_{\text{input}}|/k^{\mathcal{O}(1)}$ in size. This is shown formally in the following lemma.

Lemma F.5 (Large $\widetilde{p}_j \implies$ round j receives a large dataset)**.** *Suppose for a large constant $C_1 > 0$,*

$$|D_{\text{input}}| \geqslant C_1 k^7 \log(L) \cdot (d \log\Big(e\tau |S| k^{7/2} \log L\Big) + \log(k/\delta)). \tag{29}$$

If $\widetilde{p}_j > p_j^\star$ in iteration j, then $\Pr\big(|D_{\text{out}}^j| < C_2 k^{7/2} \log(L) \cdot (d \log\big(e\tau |S| k^{7/2} \log L\big) + \log(k/\delta)) \mid \mathcal{H}_{j-1}\big) \leqslant \frac{\delta}{2k}$.

Proof. The proof of this lemma is discussed in Appendix I.4. □

Finally, we invoke the density ratio bound in Eq. (27), along with the sufficiently large size of the dataset D_{out}^j in any non-aborted iteration j (Lemma F.5) to establish weak learning guarantees for $\widehat{\pi}^j$. Recall from `AutoLearn.warmup`, that $\widehat{\pi}^j$ is trained via next-token prediction on D_{out}^j, which is a dataset of length-τ instances drawn from $\widehat{\lambda}_j$. The prior work of (Joshi et al., 2025) established learning guarantees for next-token prediction, and via a change-of-measure argument to transfer learning guarantees under $\widehat{\lambda}_j$ to λ_j (facilitated by the density ratio bound in Lemma F.3), this implies learning guarantees for $\widehat{\pi}^j$ under the distribution λ_j. This is shown in the following lemma.

Lemma F.6 (Weak learning guarantee for $\widehat{\pi}^j$)**.** *Assume the input-size bound Eq. (29). Consider any iteration j where $\widetilde{p}_j > p_j^\star$. With probability at least $1 - \frac{\delta}{k}$, the model $\widehat{\pi}^j$ trained in iteration j satisfies,*

$$\Pr_{X \sim \lambda_j}\left[\widehat{\pi}_\tau^j(X) \neq \pi_\tau^\star(X)\right] = \texttt{err}_j \leqslant \texttt{err}_\star = \frac{1}{4}$$

Proof. The proof of this result is discussed in Appendix I.5. □

With this, we complete the analysis in the non-aborted iterations. Next we move on to the aborted iterations.

F.2.2 Aborted iterations $\left(\widetilde{p}_j \leqslant p_j^\star\right)$

In the aborted iterations, $\widetilde{p}_j$ may be too small, so we are no longer guaranteed a small bound on the density ratio between λ_j and $\widehat{\lambda}_j$ through Lemma F.3. In such iterations, our argument will be to show that the regression R_j, which controls the test-error of $\widehat{\pi}$ (cf. Lemma F.2) is small with moderate probability. In particular, from its definition in Eq. (18), and the structure of $\mathcal{I}_j$ as being a sum of independent indicators,

$$\widetilde{p}_j = \Pr\big(|\mathcal{I}_j| \geqslant 1 \mid \mathcal{H}_{j-1}, \mathcal{K}_{j-1}\big) = \mathbb{E}\left[1 - \prod_{i=0}^{L-1}\left(1 - \frac{\alpha_{\text{rank}_j(X_i)}^{j,k}}{\alpha_{\max}^{j,k}}\right) \,\middle|\, \mathcal{K}_{j-1}, \mathcal{H}_{j-1}\right] \tag{30}$$

On the other hand, we have $R_j = \sum_{i=0}^{L-1} \alpha_{\text{rank}_j(X_i)}^{j,k}$. When $\widetilde{p}_j$ is small, we can show a moderate probability upper bound on R_j via an application of Markov's inequality. This implies that in iterations which abort, the regression is likely to be small. In order to state the bound on the regression, we define the threshold,

$$\mathfrak{r}_j \triangleq \frac{\alpha_{\max}^{j,k}}{16\sqrt{k}} \tag{31}$$

Lemma F.7 (Aborted iterations have low regression)**.** *If $\widetilde{p}_j \leqslant p_j^\star$, then, $\Pr\big(R_j > \mathfrak{r}_j \mid \mathcal{K}_{j-1}, \mathcal{H}_{j-1}\big) \leqslant \frac{1}{32k^2}$.*

Proof. The proof of this result is given in Appendix I.6. □

While the above result shows that R_j is small with moderate probability, the bound on the error of $\widehat{\pi}$ in Lemma F.2 relies on the *expected* regression, which is a-priori not guaranteed to be small. This is handled by the fact that we are guaranteed $(\underline{X}, \boldsymbol{\eta}) \in \mathcal{K}_j \implies \{\widetilde{p}_j \leqslant p_j^\star \text{ and } R_j > \mathfrak{r}_j\}^c$. This means that the regression in Lemma F.2 in aborted iterations is gated by the $\mathfrak{r}_j$ threshold, which is small, and we avoid the tails of R_j. We introduce a short lemma proving that the cumulative regression threshold, $\sum_{j=0}^{k-1} \mathfrak{r}_j$ is indeed small, necessary to complete this argument.

Lemma F.8 (Bound on cumulative regression). $\sum_{j=0}^{k-1} \mathfrak{r}_j \leqslant \frac{1}{8}$.

Proof. The proof of this lemma is similar to that of (Freund, 1995, Lemma 3.9). Noting that for $\mathtt{err}_\star = \frac{1}{4}$, the mode of the binomial PMF gives for $0 \leqslant j \leqslant k-2$, $\alpha_{\max}^{j,k} \leqslant \frac{1}{\sqrt{2\pi \cdot \frac{3}{16}}} \cdot \frac{1}{\sqrt{k-j-1}} \leqslant \frac{0.95}{\sqrt{k-j-1}}$, while for $\alpha_{\max}^{k-1,k} \leqslant 1$. As a consequence,

$$\sum_{j=0}^{k-1} \mathfrak{r}_j \leqslant \frac{1}{16\sqrt{k}} \left(\sum_{j=0}^{k-2} \frac{0.95}{\sqrt{k-j-1}} + 1 \right) \leqslant \frac{1}{8}.$$

□

Having established all the necessary tools, we are ready to furnish a proof of Theorem F.1 via bounding the test-error decomposition for $\widehat{\pi}$ in Lemma F.2.

F.3 Proof of Theorem F.1

We prove accuracy, sample complexity and query complexity guarantees for `AutoLearn.warmup` below.

Accuracy bound. Let $\mathcal{J}_{\mathsf{ABORT}} \triangleq \left\{0 \leqslant j \leqslant k-1 : \widetilde{p}_j \leqslant p_j^\star\right\}$ denote the set of aborted iterations, which is $\mathcal{H}_{k-2}$ measurable. By the test-error decomposition in Lemma F.2,

$$\begin{aligned}
&\Pr_{\boldsymbol{x}\sim\rho}\left[\widehat{\pi}(\boldsymbol{x}) \neq \pi_T^\star(\boldsymbol{x})\right] \\
&\leqslant L \cdot \beta_0^{0,k} + \sum_{j=0}^{k-1} (\mathtt{err}_j - \mathtt{err}_\star) \cdot \mathbb{E}[R_j \cdot \mathbb{I}((\underline{X}, \boldsymbol{\eta}) \in \mathcal{K}_j) \mid \mathcal{H}_{k-1}] \\
&\quad + \Pr((\underline{X}, \boldsymbol{\eta}) \notin \mathcal{K}_{k-1} \mid \mathcal{H}_{k-1}) \\
&\overset{(a)}{\leqslant} \frac{1}{16} + \sum_{j=0}^{k-1} (\mathtt{err}_j - \mathtt{err}_\star) \cdot \mathbb{E}[R_j \cdot \mathbb{I}((\underline{X}, \boldsymbol{\eta}) \in \mathcal{K}_j) \mid \mathcal{H}_{k-1}] + \frac{1}{16} \\
&\leqslant \frac{1}{8} + \sum_{j \in \mathcal{J}_{\mathsf{ABORT}}} \mathbb{E}[R_j \cdot \mathbb{I}((\underline{X}, \boldsymbol{\eta}) \in \mathcal{K}_j) \mid \mathcal{H}_{k-1}] \\
&\quad + \sum_{j \notin \mathcal{J}_{\mathsf{ABORT}}} (\mathtt{err}_j - \mathtt{err}_\star) \cdot \mathbb{E}[R_j \cdot \mathbb{I}((\underline{X}, \boldsymbol{\eta}) \in \mathcal{K}_j) \mid \mathcal{H}_{k-1}] \\
&\leqslant \frac{1}{8} + \sum_{j \in \mathcal{J}_{\mathsf{ABORT}}} \mathbb{E}[R_j \cdot \mathbb{I}(\{\widetilde{p}_j \leqslant p_j^\star \text{ and } R_j > \mathfrak{r}_j\}^c) \mid \mathcal{H}_{k-1}] + L \sum_{j \notin \mathcal{J}_{\mathsf{ABORT}}} (\mathtt{err}_j - \mathtt{err}_\star)_+ \\
&\leqslant \frac{1}{8} + \sum_{j \in \mathcal{J}_{\mathsf{ABORT}}} \mathfrak{r}_j + L \sum_{j \notin \mathcal{J}_{\mathsf{ABORT}}} (\mathtt{err}_j - \mathtt{err}_\star)_+ \\
&\overset{(b)}{\leqslant} \frac{1}{4} + L \sum_{j \notin \mathcal{J}_{\mathsf{ABORT}}} (\mathtt{err}_j - \mathtt{err}_\star)_+,
\end{aligned}$$

where in (a) we use the choice of $k = \Theta(\log(L))$ and the exponential decay of $\beta_0^{0,k}$ as a function of k (cf. Lemma F.9), the upper bound on $\Pr((\underline{X}, \boldsymbol{\eta}) \notin \mathcal{K}_{k-1} \mid \mathcal{H}_{k-1})$ in Lemma F.4, and the definition of $\mathcal{K}_j$. And in (b) we invoke Lemma F.8, showing $\sum_{j=0}^{k-1} \mathfrak{r}_j \leqslant \frac{1}{8}$. In any iteration j where $\mathtt{ABORT}[j]$ does not occur, by Lemma F.6, with probability $1 - \frac{\delta}{k}$, $\mathtt{err}_j \leqslant \mathtt{err}_\star$. This implies, with probability at least $1 - \delta$,

$$\Pr_{\boldsymbol{x}\sim\rho}\left[\widehat{\pi}(\boldsymbol{x}) \neq \pi_T^\star(\boldsymbol{x})\right] \leqslant \frac{1}{4}.$$

Bound on sample complexity of `AutoLearn.warmup`. The bound on the size of the instance dataset required to establish the above guarantee for the depth-1 construction is demonstrated in Lemma F.5 and is,

$$|D_{\text{input}}| \geqslant \log^8(T) \cdot \left[d \log(eT|S|) + \log(T/\delta)\right],$$

where we set $L = \sqrt{T}$ and $k = C\log(L)$ for a sufficiently large constant C. This gives the desired bound on the size of the dataset of instances in Theorem F.1.

Bound on query complexity of `AutoLearn.warmup`. Observe that on every instance in the dataset D_{input}, `AutoLearn.warmup` queries $\texttt{iCoT}(\cdot)$ exactly $T/\tau = L$ times within the $\texttt{Split}(\cdot \,\|\, L)$ subroutine. On the other hand, to train the weak model $\widehat{\pi}^j$, `AutoLearn.warmup` makes a call to $\texttt{NTP}_{\texttt{Full-CoT}}\big(D^j_{\text{out}} \,\|\, \tau, \delta/2k\big)$ which queries $\texttt{iCoT}(\cdot)$ τ times per instance in D^j_{out}. Summation across the invocations of this subroutine for $j = 0, \cdots, k-1$, the overall query complexity is upper bounded by,

$$\sum_{j=0}^{k-1} \tau |D^j_{\text{out}}| + L|D_{\text{input}}| \leqslant (\tau + L)|D_{\text{input}}| = 2\sqrt{T}|D_{\text{input}}|.$$

where in the last equation we plug in the choice $L = \tau = \sqrt{T}$. This gives the desired bound on the query complexity in Theorem F.1.

Lemma F.9 (Exponential decay of $\beta_0^{0,k}$)**.** *There exists an absolute constant $c > 0$ such that, $\beta_0^{0,k} \leqslant e^{-ck}$.*

Proof. Consider a set of k biased coins, $Z_1, \cdots, Z_k$, with probability of heads equal to $1 - \texttt{err}_\star = \frac{3}{4}$. It is a short calculation to see that $\beta_0^{0,k}$ equals the probability that at most $\frac{k}{2}$ of them come up heads. Since the expected number of heads is $\frac{3k}{4}$, by standard concentration arguments (say, Hoeffding's inequality), the statement of the lemma is established. □

G Analysis of `AutoLearn` (Algorithm 2): Proof of Theorem C.2

This section presents the full analysis of `AutoLearn`; for first-time readers, we recommend first reading the analysis of the simplified depth-1 version of `AutoLearn` in Appendix F before proceeding to this section. We prove the following generalized version of Theorem C.2, which states the guarantee in terms of the sample complexity of the weak learner for next-state/token data, $\texttt{Base}(\cdot)$, without assuming it satisfies a "generic" rate: $n_{\texttt{weak}}(\delta') \leqslant \texttt{comp}(\Pi) \cdot \log(1/\delta')$.

Theorem G.1 (Theorem C.2 for general weak learners)**.** *For any $\delta' \in (0,1)$, let $\texttt{Base}(\cdot \,\|\, \delta')$ be any algorithm which learns an unknown $\pi^\star \in \Pi$ to constant error from i.i.d. next-state data (Definition C.1). Let $n_{\texttt{weak}}(\delta')$ denote its sample complexity to achieve error $\frac{1}{4}$ with probability at least $1 - \delta'$.*

Let $\varepsilon, \delta \in (0,1)$ and suppose $H = \sqrt{\log_2(T)} \in \mathbb{N}$. Suppose $\texttt{AutoLearn}(\cdot \,\|\, T, \varepsilon, \delta)$ (Algorithm 2) is invoked using the weak learner $\texttt{Base}$ on a dataset of $n_{\texttt{sample}}$ i.i.d. instances drawn from ρ, and queries the $\texttt{iCoT}(\cdot)$ oracle $n_{\texttt{query}}$ times. The sample and query complexity required by the algorithm for the resulting model $\widehat{\pi} : S \times \Sigma^T \to S$ to achieve $\Pr_{\boldsymbol{x}\sim\rho}\big[\widehat{\pi}(\boldsymbol{x}) \neq \pi^\star_T(\boldsymbol{x})\big] \leqslant \varepsilon$ with probability at least $1 - \delta$ are upper bounded by,

$$n_{\texttt{sample}}, n_{\texttt{query}} \leqslant h_\varepsilon(T) \cdot \left[\frac{n_{\texttt{weak}}(\delta/h_\varepsilon(T)) \vee \log(h_\varepsilon(T)/\delta)}{\varepsilon}\right],$$

where $h_\varepsilon(T) = 2^{C\sqrt{\log(T)}\cdot\log\log(T)} \cdot \log^2(1/\varepsilon)$ for an absolute constant $C > 0$. Furthermore, $n_{\texttt{base}} \leqslant h_\varepsilon(T)$, where $n_{\texttt{base}}$ is the number of invocations of $\texttt{Base}$.

Proof. The main proof is discussed in Appendix G.3. □

Theorem C.2 itself follows from this result by setting $n_{\texttt{weak}}(\delta') = \texttt{comp}(\Pi)\log(1/\delta')$ and simplifying.

`AutoLearn` is an invocation of $\texttt{AutoLearn}_{H+\frac{1}{2}}$ with an appropriately chosen decomposition and branching schedule. $\texttt{AutoLearn}_{H+\frac{1}{2}}$ itself follows a tree structure where each instance of $\texttt{AutoLearn}_h(\cdot \,\|\, t, \delta)$ invokes $\texttt{AutoLearn}_{h-\frac{1}{2}}(\cdot \,\|\, t, \delta)$ k_h times. This is depicted in Figure 5. Prior to this we introduce some notation.

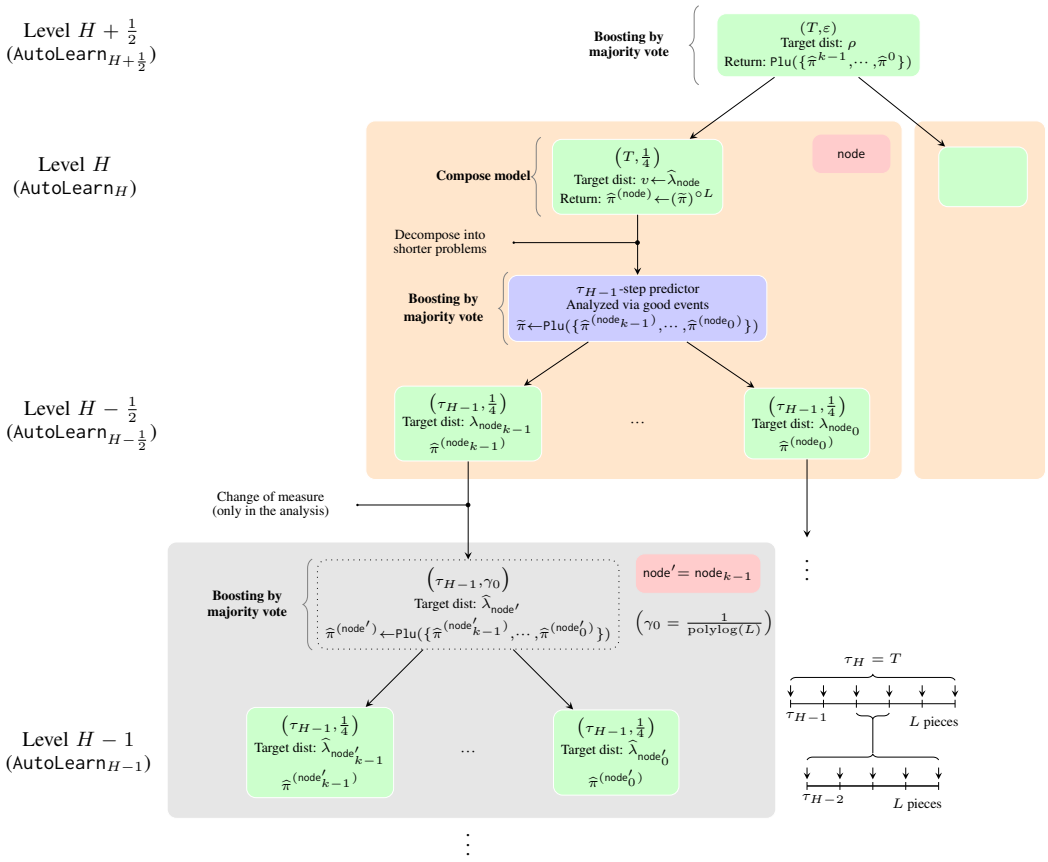


Figure 5: The recursive call of invocations within structure of AutoLearn when the target error is ε: each green box corresponds to a call to AutoLearn$_h(\cdot)$ for some $h \in \{0, \frac{1}{2}, 1, \ldots, H+\frac{1}{2}\}$. From each integer level h to the next one, instances decrease by a factor $L_h = L$ in length.

Notation. For $h \in \mathbb{Z}$, let $\boldsymbol{X}_h = S \times \Sigma^{\tau_h}$ denote the space of τ_h-length instances. Let $\boldsymbol{X}_{h+\frac{1}{2}} = \boldsymbol{X}_h$ for all such h.

Tree diagram corresponding to the recursive structure of Algorithm 1. The recursive curriculum proceeds with AutoLearn$_h$ calling k_h instances of AutoLearn$_{h-\frac{1}{2}}$. The diagram in Figure 5 arranges these invocations into a tree: the root corresponds to a call of AutoLearn$_{H+\frac{1}{2}}$ and going down the tree recursively invokes AutoLearn$_H$, AutoLearn$_{H-\frac{1}{2}}$ and so on. The number of nodes at any level h is $\prod_{r=h+1/2,h+1,\ldots,H+1/2} k_r$ (the empty product at the root is one). We let "node" denote a node in the tree, let Par(node) denote the parent of node, Ch(node) denote the ordered set of children of node[14] and Sib(node) $=$ Ch(Par(node)) denote its siblings. Models are *trained* only at nodes at level 0 of the tree, but are hierarchically combined to define models induced at intermediate nodes of the tree. We also refer to a node node being *resolved* to indicate that the model corresponding to node has been learned. This is only possible if all level-0 nodes in the induced subtree with node as root have been trained. Furthermore, the construction can be thought of as resolving nodes in the tree in a depth-first fashion: at any level h, the instantiations of AutoLearn$_h$ can only be resolved when the ones "prior" to it (in the depth-first ordering) at the same level have been resolved. We use the notation $\mathcal{T}$ to collect the set of all nodes in the tree, and $\mathcal{T}_h$ to denote the set of all nodes at a given level $h \in \{0, \frac{1}{2}, 1, \cdots, H+\frac{1}{2}\}$. root denotes the singular node in $\mathcal{T}_{H+\frac{1}{2}}$ (i.e., the root node of $\mathcal{T}$).

[14]Note that the children of a node are not interchangeable

Each node is associated with a collection of objects we define next. For any $h \in \{0, \frac{1}{2}, 1, \cdots, H+\frac{1}{2}\}$ and $\mathsf{node} \in \mathcal{T}_h$, let $\widehat{\pi}^{(\mathsf{node})}$ denote a model associated with it, defined recursively as,

$$\widehat{\pi}^{(\mathsf{node})} = \widehat{\pi}^{\circ L_h}, \text{ where, } \widehat{\pi} \leftarrow \mathtt{Plu}\big(\{\widehat{\pi}^{(\mathsf{node}')} : \mathsf{node}' \in \mathtt{Ch}(\mathsf{node})\}\big). \tag{32}$$

where $\widehat{\pi}^{(\mathsf{node})}$ for $\mathsf{node} \in \mathcal{T}_0$ are induced by invocations of $\mathtt{Base}$. Next, as we carried out in Lemma F.6, define a sequence of distributions for each node. For node at level h, let j denote its index within its ordered set of siblings, $\mathtt{Sib}(\mathsf{node})$ and let the first j siblings of node be denoted by the set $\mathtt{Sib}_{<j}(\mathsf{node})$. Then, for $X \in \boldsymbol{X}_h$,

$$\begin{aligned} w_{\mathsf{node}}(X) &= \alpha^{j,k}_{\mathrm{rank}(X\|\mathsf{node})}, \text{ where,} \\ \mathrm{rank}(X\|\mathsf{node}) &= \sum_{\mathsf{node}' \in \mathtt{Sib}_{<j}(\mathsf{node})} \mathbb{I}(\widehat{\pi}^{(\mathsf{node}')}(X) = \pi^\star_{\tau_h}(X)) \in [0, j] \end{aligned} \tag{33}$$

and $\alpha^{j,k}_r$ is defined in Eq. (2). The rank of a node captures how many of its earlier siblings collect any reward on X. Finally, for any $\mathsf{node} \in \mathcal{T}$ we use the notation D_{node} to indicate the dataset of instances which the corresponding invocation of $\mathtt{AutoLearn}_h$ processes. First, we prove a short lemma which argues that for any node, D_{node} is composed of independent instances.

Lemma G.2 (Independence of per-node datasets)**.** *Conditioned on the history before its collection and on its size, each per-node dataset D_{node} consists of i.i.d. examples from its history-dependent accepted-sample distribution.*

Proof. The root input is i.i.d. At each node, condition on the earlier epochs and models, leaving the current epoch fresh. The same fixed sampling kernel acts independently on its examples and returns at most one sub-instance per example. Conditional on the number accepted, the accepted examples are i.i.d. from the kernel's conditional output law. Truncating to a fixed size preserves this property. Applying this argument recursively proves the claim; unconditional independence across adaptive epochs is not asserted. □

With this, for node at level h or $h + \frac{1}{2}$, we let $\widehat{\lambda}_{\mathsf{node}}$ denote the distribution over instances of length τ_h from which the elements of D_{node} are sampled. At level $h = H + \frac{1}{2}$, we have that, $\widehat{\lambda}_{\mathtt{root}} = \rho$. With this notation introduced, we are ready to state two inductive assumptions on the correctness of invocations of $\mathtt{AutoLearn}_h$ at level h (for integer and non-integer values of h), which together show that with high probability, the depth $h + 1$ version succeeds conditioned on this. Define the following sequences recursively,

$$\text{If } h \in \mathbb{N},\ n^\star_{\mathtt{sample}}(h + \tfrac{1}{2}, 2k_{h+\frac{1}{2}}\delta) = n^\star_{\mathtt{sample}}(h, \delta) \cdot 2k_{h+\frac{1}{2}} e^{ck_{h+\frac{1}{2}}},$$

$$\text{And, } n^\star_{\mathtt{sample}}(h + 1, 2k_{h+1}\delta) = n^\star_{\mathtt{sample}}(h + \tfrac{1}{2}, \delta) \cdot C_3 k_{h+1}^{7/2} \log(L_{h+1}) \tag{34}$$

$$\text{And, } \quad n^\star_{\mathtt{query}}(r, 2k_r\delta) = L_r n^\star_{\mathtt{sample}}(r, 2k_r\delta) + k_r n^\star_{\mathtt{query}}(r - \tfrac{1}{2}, \delta), \quad r \in \{\tfrac{1}{2}, 1, \ldots\}. \tag{35}$$

where $n^\star_{\mathtt{sample}}(0, \delta) = n^\star_{\mathtt{query}}(0, \delta) = \big[n_{\mathsf{weak}}(\delta) \vee 8\log(e/\delta)\big]$, and $C_3 > 0$ is a large absolute constant, while in the definition of $n^\star_{\mathtt{sample}}$, c is chosen to be the same constant as in the exponent of $\beta_0^{0,k}$ in Lemma F.9. With the choices of decomposition schedule $(L_h)_{h\geqslant 0}$ and branching schedule $(k_h)_{h\geqslant 0}$ as described in Algorithm 2, we get that, for $h \leqslant H$,

$$n^\star_{\mathtt{sample}}(h, \delta),\ \frac{n^\star_{\mathtt{query}}(h, \delta)}{1 + 2hL} \leqslant (\log(T))^{\mathcal{O}(h)} \cdot \Big[n_{\mathsf{weak}}(\delta/(\log(T))^{\mathcal{O}(h)}) \vee \log\Big((\log T)^{\mathcal{O}(h)}/\delta\Big)\Big]$$

and for $h = H + \frac{1}{2}$,

$$n^\star_{\mathtt{sample}}(H + \tfrac{1}{2}, \delta),\ n^\star_{\mathtt{query}}(H + \tfrac{1}{2}, \delta) \leqslant \frac{h_\varepsilon(T)}{\varepsilon} \cdot \big[n_{\mathsf{weak}}\big(\delta/h_\varepsilon(T)\big) \vee \log(h_\varepsilon(T)/\delta)\big].$$

where $h_\varepsilon(T) = 2^{C\sqrt{\log(T)}\log\log(T)} \cdot \log^2(1/\varepsilon)$. The proof of Theorem G.1 relies on establishing inductive guarantees on the sample and query complexity needed at each level of the algorithm. This is separated into guarantees for integer levels, and for half-integer levels in the two hypotheses below.

Hypothesis$_{h+\frac{1}{2}}(\delta)$

Consider any $\mathsf{node} \in \mathcal{T}_{h+\frac{1}{2}}$ and semiautomata simulation instances of length τ_h over the target distribution over instances, $\widehat{\lambda}_{\mathsf{node}}$. When invoked on a dataset of independent instances drawn from $\widehat{\lambda}_{\mathsf{node}}$ of size $n_{\mathsf{sample}}(h+\frac{1}{2},\delta) \geqslant n^{\star}_{\mathsf{sample}}(h+\frac{1}{2},\delta)$, $\mathtt{AutoLearn}_{h+\frac{1}{2}}$ returns a deterministic model $\widehat{\pi}$ such that with probability $1-\delta$,

$$\Pr_{X\sim\widehat{\lambda}_{\mathsf{node}}}\left(\widehat{\pi}(X) \neq \pi^{\star}_{\tau_h}(X)\right) \leqslant 3\sqrt{k_{h+\frac{1}{2}}}\cdot e^{-ck_{h+\frac{1}{2}}}.$$

where $c>0$ is the constant in the exponent of $\beta_0^{0,k}$ in Lemma F.9. In the process, the number of queries made to $\mathtt{iCoT}(\cdot)$ is upper bounded by $n^{\star}_{\mathsf{query}}(h+\frac{1}{2},\delta)$.

Hypothesis$_{h+1}(\delta)$

Consider any $\mathsf{node} \in \mathcal{T}_{h+1}$ and semiautomata simulation instances of length τ_{h+1} over the target distribution over instances, $\widehat{\lambda}_{\mathsf{node}}$. When invoked on a dataset of independent instances drawn from $\widehat{\lambda}_{\mathsf{node}}$ of size at least, $n_{\mathsf{sample}}(h+1,\delta) \geqslant n^{\star}_{\mathsf{sample}}(h+1,\delta)$, then the invocation $\mathtt{AutoLearn}_{h+1}$ corresponding to node returns a deterministic model $\widehat{\pi}$ such that with probability $1-\delta$,

$$\Pr_{X\sim\widehat{\lambda}_{\mathsf{node}}}\left(\widehat{\pi}(X) \neq \pi^{\star}_{\tau_{h+1}}(X)\right) \leqslant \frac{1}{4}.$$

The number of queries made to $\mathtt{iCoT}(\cdot)$ is at most $n^{\star}_{\mathsf{query}}(h+1,\delta)$.

Lemma G.3 (Aggregation steps)**.** *For any* $h \in \mathbb{Z}$, $\text{Hypothesis}_h(\delta) \implies \text{Hypothesis}_{h+\frac{1}{2}}(2k_{h+\frac{1}{2}}\delta)$.

Proof. This lemma is restated in more detail and proved in Lemma G.8. □

Lemma G.4 (Aggregation+composition steps)**.** *Consider any value of* $h \in \{0,1,\cdots,H-1\}$ *and suppose* $k_{h+1} = \log(L) = c_2\sqrt{\log(T)}$ *and* $k_{h+\frac{1}{2}} = c_3\log\log(T)$ *for appropriate constants* $c_2, c_3 > 0$*. Then, we have the recursion,* $\text{Hypothesis}_{h+\frac{1}{2}}(\delta) \implies \text{Hypothesis}_{h+1}(2k_{h+1}\delta)$.

Proof. The base case (sample and query complexity of $\text{Hypothesis}_0(\delta')$ at $h=0$) are directly implied by the definition of the sample and query complexity of $\mathtt{Base}(\cdot\,\|\,\delta')$ in terms of $n_{\mathtt{weak}}(\delta')$. This lemma is restated in more detail and proved in Lemma G.15. □

Of the two claims, the first one is easier to show, since it is a direct consequence of boosting-by-filtering (Freund, 1995). In particular, since there is no composition step involved in going from level h to level $h+\frac{1}{2}$, the distributional shift is only due to the boosting step, and the analysis of this is standard. Next, we sketch how to bootstrap Hypothesis_h into $\text{Hypothesis}_{h+\frac{1}{2}}$, implying the statement of Lemma G.3.

G.1 Analysis of half-integer steps: Proof of Lemma G.3

In this section we analyze Lemma G.3. We fix a level $h \in \mathbb{Z}$ and a particular $\mathsf{node} \in \mathcal{T}_{h+\frac{1}{2}}$. We let $\widehat{\pi}^0,\cdots,\widehat{\pi}^{k-1}$ denote the k models returned by $\mathtt{AutoLearn}_h$, corresponding to nodes in $\mathtt{Ch}(\mathsf{node})$ (and which are invoked on distributions $\widehat{\lambda}_{\mathsf{node}'}$ for $\mathsf{node}' \in \mathtt{Ch}(\mathsf{node})$). Let us index the nodes in $\mathtt{Ch}(\mathsf{node})$ as $(\mathsf{node}_0,\cdots,\mathsf{node}_{k-1})$. To prove Lemma G.3, we translate the analysis of $\mathtt{AutoLearn.warmup}$ (cf. Theorem F.1) for the case of $L=1$ and setting $\boldsymbol{X} \leftarrow \boldsymbol{X}_h$. Since $L=1$, some parts of the analysis greatly simplify.

First we begin by introducing some notation before jumping into the proof of the result. Let $(\mathcal{H}_{\mathsf{node}_j})_{j=0}^{k-1}$ denote the filtration where $\mathcal{H}_{\mathsf{node}_{j-1}}$ captures all randomness up until just before the dataset D_{node_j} is collected: this includes the randomness in the construction of the models $\widehat{\pi}^{(\mathsf{node}_0)},\cdots,\widehat{\pi}^{(\mathsf{node}_{j-1})}$ corresponding to nodes $\mathsf{node}_0,\cdots,\mathsf{node}_{j-1} \in \mathtt{Ch}(\mathsf{node})$, as well as the

randomness in the instances from D_{node} which contribute to the datasets $D_{\mathsf{node}_0}, \cdots, D_{\mathsf{node}_{j-1}}$. Define the following distribution over $\boldsymbol{X}_h$,

$$\lambda_{\mathsf{node}_j}(\cdot) \propto \widehat{\lambda}_{\mathsf{node}}(\cdot) w_{\mathsf{node}_j}(\cdot), \tag{36}$$

where $w_{\mathsf{node}_j}(\boldsymbol{x}) = \alpha_r^{j,k}$ for $r = \mathrm{rank}(\boldsymbol{x} \| \mathsf{node}_j)$. Recall that $\mathrm{rank}(\cdot \| \mathsf{node}_j)$ (Eq. (33)) captures the number of models among $\{\widehat{\pi}^0, \cdots, \widehat{\pi}^{j-1}\}$ which correctly predict the terminal state of the instance $X \in \boldsymbol{X}_h$. First, we write down a test-error decomposition, similar to the one in Lemma F.2, where we break down the accuracy of the model returned by an instantiation of $\mathtt{AutoLearn}_{h+\frac{1}{2}}$ (i.e., corresponding to some node) into the accuracy of the models returned by the instances of $\mathtt{AutoLearn}_h$ it spawns (i.e., corresponding to $\mathsf{Ch}(\mathsf{node})$).

For the purpose of this section, we will denote $k \leftarrow k_{h+\frac{1}{2}}$ and $L \leftarrow L_{h+\frac{1}{2}} = 1$ to simplify notation.

Lemma G.5 (Test-error decomposition)**.** *Let $\widehat{\pi}^{(\mathsf{node})}$ denote the model returned by the instantiation of $\mathtt{AutoLearn}_{h+\frac{1}{2}}$ corresponding to $\mathsf{node} \in \mathcal{T}_{h+\frac{1}{2}}$. Let $\widehat{\pi}^{(\mathsf{node}_j)}$ denote the model returned by $\mathtt{AutoLearn}_h$ corresponding to each $\mathsf{node}_j \in \mathsf{Ch}(\mathsf{node})$. Then,*

$$\Pr_{X \sim \widehat{\lambda}_{\mathsf{node}}}\left[\widehat{\pi}^{(\mathsf{node})}(X) \neq \pi^\star_{\tau_h}(X)\right] \leqslant \beta_0^{0,k} + \sum_{j=0}^{k-1} (\mathtt{err}_j - \mathtt{err}_\star) \cdot \mathbb{E}_{X \sim \widehat{\lambda}_{\mathsf{node}}}\left[\alpha^{j,k}_{\mathrm{rank}(X \| \mathsf{node}_j)} \mid \mathcal{H}_{\mathsf{node}_{k-1}}\right]$$

Here, $\beta_r^{j,k}$ is defined in Eq. (2), and $\mathtt{err}_j$ is the test error of $\widehat{\pi}^{(\mathsf{node}_j)}$ under the distribution $\lambda_{\mathsf{node}_j}$,

$$\mathtt{err}_j = \Pr_{X \sim \lambda_{\mathsf{node}_j}}\left[\widehat{\pi}^{(\mathsf{node}_j)}(X) \neq \pi^\star_{\tau_h}(X)\right]$$

Proof. Since there is no composition step involved, this result is immediate from the proof of Lemma F.2 by the choice $L = 1$, $\mathcal{K} = \boldsymbol{X}^L$, and $\rho \leftarrow \widehat{\lambda}_{\mathsf{node}}$, $\sigma \leftarrow \widehat{\lambda}_{\mathsf{node}}$ and $\lambda_j \leftarrow \lambda_{\mathsf{node}_j}$. □

Since there is no composition involved in going from step h to $h + \frac{1}{2}$, the learner has a sampling oracle for $\widehat{\lambda}_{\mathsf{node}}$, which can be used to generate samples from $\lambda_{\mathsf{node}_j}$ by rejection sampling (which coincides with inverted sampling when $L = 1$). In order to analyze the error decomposition in Lemma G.5 first we define events which track whether the instances of $\mathtt{AutoLearn}_h$ corresponding to nodes in $\mathsf{Ch}(\mathsf{node})$ are invoked on sufficiently large datasets to satisfy the inductive guarantee we assume for nodes at level h. To this end, define for $0 \leqslant j \leqslant k-1$,

$$\mathtt{ABORT}_h[\mathsf{node}_j] = \left\{p_{\mathsf{node}_j} \leqslant p_j^\star\right\}, \text{ where, } p_j^\star = e^{-ck}$$

where $c > 0$ is the constant in Lemma F.9. With this, the next lemma simply uses the definition of $\mathrm{Hypothesis}_h$ to argue that if any node $\mathsf{node}_j \in \mathsf{Ch}(\mathsf{node})$ does not abort, we can get a weak learning guarantee for node_j.

Lemma G.6 (Weak learner guarantee for $\mathtt{AutoLearn}_h$)**.** *Assume the inductive hypothesis $\mathrm{Hypothesis}_h(\delta)$. Conditioned on the event that $\mathtt{ABORT}_h[\mathsf{node}_j]$ is false, with probability at least $1 - 2\delta$, the model $\widehat{\pi}^{(\mathsf{node}_j)}$ returned by node_j satisfies,*

$$\Pr_{X \sim \lambda_{\mathsf{node}_j}}\left[\widehat{\pi}^{(\mathsf{node}_j)}(X) \neq \pi^\star_{\tau_h}(X)\right] = \mathtt{err}_j \leqslant \mathtt{err}_\star \triangleq \frac{1}{4}$$

Proof. By Lemma G.7, the accepted dataset has size at least $n^\star_{\mathtt{sample}}(h, \delta)$ except with probability δ. Conditional on its size and the earlier history it is i.i.d.; $\mathrm{Hypothesis}_h(\delta)$ contributes a further failure probability δ. □

Next we argue about the probability that any of the nodes in $\mathsf{Ch}(\mathsf{node})$ abort. Define,

$$p_{\mathsf{node}_j} = \Pr_{\boldsymbol{x} \sim \widehat{\lambda}_{\mathsf{node}}}\left(\mathtt{InvSampling}(\{\boldsymbol{x}\} \parallel \Pi_j, k) \neq \perp\right), \text{ where, } \Pi_j = \left\{\widehat{\pi}^{(\mathsf{node}_0)}, \cdots, \widehat{\pi}^{(\mathsf{node}_{j-1})}\right\}.$$

This is the probability with which an instance $\boldsymbol{x} \sim \widehat{\lambda}_{\mathsf{node}}$ is accepted by split-and-inverted-sampling (Algorithms 3 and 4). Next we show that if p_{node_j} is large, then the instantiation of $\mathtt{AutoLearn}_h$ corresponding to node_j is likely to be instantiated with a large dataset, implying that node_j indeed returns a weak learner with high probability.

Lemma G.7 (Truthful aborts at half-levels). *Suppose* $\texttt{AutoLearn}_{h+\frac{1}{2}}(\cdot \,\|\, \tau_{h+\frac{1}{2}}, 2k\delta)$ *(cf. Algorithm 1) is invoked on a dataset of at least* $n^\star_{\texttt{sample}}(h+\frac{1}{2}, 2k\delta) = 2ke^{ck} \cdot n^\star_{\texttt{sample}}(h,\delta)$ *instances where* $c > 0$ *is the constant in the exponent of* $\beta_0^{0,k}$ *in Lemma F.9. Then, for any index* j *such that* $p_{\texttt{node}_j} \geqslant e^{-ck}$, $\Pr\big(|D_{\texttt{node}_j}| < n^\star_{\texttt{sample}}(h,\delta) \mid \mathcal{H}_{\texttt{node}_{j-1}}\big) \leqslant \delta$.

Proof. The proof of this result is identical to that of Lemma F.5, where we use the Chernoff bound to control the deviations of n' coins each having probability of heads at least $p_{\texttt{node}_j}$. □

The complement case is when $p_{\texttt{node}_j}$ is small and $\texttt{node}_j$ is no longer likely to succeed. Similar to in the analysis of `AutoLearn.warmup` where we established an explicit formula for p_j, $p_{\texttt{node}_j}$ satisfies the equation,

$$p_{\texttt{node}_j} = \mathbb{E}_{X\sim\widehat{\lambda}_{\texttt{node}}}\left[\frac{\alpha^{j,k}_{\mathrm{rank}(X\|\texttt{node}_j)}}{\alpha^{j,k}_{\max}} \,\middle|\, \mathcal{H}_{\texttt{node}_{j-1}}\right]$$

Which corresponds to setting $L = 1$ in Eq. (17). In particular, combining with Lemma G.7, the test error decomposition in Lemma G.5, the upper bound on $\beta_0^{0,k}$ from Lemma F.9 and simplifying, we arrive at the following result which implies Lemma G.3.

Lemma G.8 (Analysis of aggregation step). *Let* $\widehat{\pi}^{(\texttt{node})}$ *denote the model returned by the instantiation of* $\texttt{AutoLearn}_{h+\frac{1}{2}}$ *corresponding to some* $\texttt{node} \in \mathcal{T}_{h+\frac{1}{2}}$. *Then, assuming the hypothesis* $\mathrm{Hypothesis}_h(\delta)$, *we have that with probability at least* $1 - 2k\delta$,

$$\Pr_{X\sim\widehat{\lambda}_{\texttt{node}}}\left[\widehat{\pi}^{(\texttt{node})}(X) \neq \pi^\star_{\tau_h}(X)\right] \leqslant 3\sqrt{k}\cdot e^{-ck}$$

Further, under $\mathrm{Hypothesis}_h(\delta)$, *this guarantee is achieved if* $\texttt{AutoLearn}_{h+\frac{1}{2}}$ *draws* $n^\star_{\texttt{sample}}(h+\frac{1}{2}, 2k\delta)$ *samples from* $\widehat{\lambda}_{\texttt{node}}$. *The number of queries made to* $\texttt{iCoT}(\cdot)$ *is at most* $n^\star_{\texttt{query}}(h+\frac{1}{2}, 2k\delta)$.

Proof. We first prove the bound on the accuracy, and subsequently show the recursion on the sample and query complexity.

Bound on accuracy. Conditioned on the event that $\texttt{ABORT}_h[\texttt{node}_j]$ is false for some j, we have $\texttt{err}_j \leqslant \texttt{err}_\star$ except with probability 2δ. Taking a union bound over non-aborted iterations and bounding the remaining terms by their acceptance thresholds, by Lemma G.5, and using the definition of $p_{\texttt{node}_j}$, w.p. at least $1 - 2k\delta$,

$$\begin{aligned}\Pr_{X\sim\widehat{\lambda}_{\texttt{node}}}\left[\widehat{\pi}^{(\texttt{node})}(X) \neq \pi^\star_{\tau_h}(X)\right] &\leqslant \beta_0^{0,k} + \sum_{j=0}^{k-1} e^{-ck}\cdot\alpha^{j,k}_{\max} \\ &\overset{(i)}{\leqslant} e^{-ck} + 2\sqrt{k}\cdot e^{-ck} \\ &\leqslant 3\sqrt{k}\cdot e^{-ck}\end{aligned}$$

In (i), we use Lemmas F.8 and F.9 (the bound on $\sum_{j=0}^{k-1}\alpha^{j,k}_{\max}$ is implied by the bound on $\sum_{j=0}^{k-1}\mathfrak{r}_j$). The bound on the number of samples drawn required from $\widehat{\lambda}_{\texttt{node}}$ is derived in Lemma G.7.

Bound on query and sample complexity of $\texttt{AutoLearn}_{h+\frac{1}{2}}$. $\texttt{AutoLearn}_{h+\frac{1}{2}}$ must be invoked on a dataset of size $n^\star_{\texttt{sample}}\big(h+\frac{1}{2}, 2k\delta\big) = n^\star_{\texttt{sample}}(h,\delta)\cdot 2ke^{ck}$ to be able to invoke Lemma G.7. On the other hand, the query complexity of $\texttt{AutoLearn}_{h+\frac{1}{2}}$ is upper bounded by,

$$n^\star_{\texttt{query}}\big(h+\tfrac{1}{2}, 2k\delta\big) = n^\star_{\texttt{sample}}(h+\tfrac{1}{2}, 2k\delta) + k\cdot n^\star_{\texttt{query}}(h,\delta). \tag{37}$$

The $n^\star_{\texttt{sample}}(h+\frac{1}{2}, 2k\delta)$ term accounts for labeling the terminal state in each invocation of `InvSampling` - note that $\texttt{AutoLearn}_{h+\frac{1}{2}}$ processes a dataset of at most $n^\star_{\texttt{sample}}(h+\frac{1}{2}, 2k\delta)$ instances (cf. Line 4 of Algorithm 1). The $k\cdot n^\star_{\texttt{query}}(h,\delta)$ term arises as the cost of training the k children invocations of $\texttt{AutoLearn}_h$. □

G.2 Proof of Lemma G.4

This is the more challenging of the two induction steps (Lemmas G.3 and G.4), since we have to deal with the composition step. Fortunately, the analysis is quite similar to the one we carried out for `AutoLearn.warmup`. To draw the parallel most easily, in this section we will denote the branching factor as $k \leftarrow k_{h+1}$, and the decomposition factor as $L \leftarrow L_{h+1}$.

We define a sequence of events $\mathcal{K}_{\mathsf{node}_0}, \cdots \mathcal{K}_{\mathsf{node}_{k-1}} \subseteq \boldsymbol{X}_h^L \times [0,1]^{kL}$ which play the same role as the $\mathcal{K}_0, \cdots, \mathcal{K}_{k-1}$ events in Lemma F.2; here $\mathcal{K}_{\mathsf{node}_j} \subseteq \boldsymbol{X}_h^L \times [0,1]^{(j+1)L}$. We index the nodes in $\mathsf{Ch}(\mathsf{node})$ as $(\mathsf{node}_0, \cdots, \mathsf{node}_{k-1})$. To prove Lemma G.4, we translate the analysis of `AutoLearn.warmup` (cf. Theorem F.1) by setting $\boldsymbol{X} \leftarrow \boldsymbol{X}_h$, $\rho \leftarrow \widehat{\lambda}_{\mathsf{node}}$. In correspondence with this choice, we denote $\underline{\widehat{\lambda}_{\mathsf{node}}}$ as the distribution obtained from $\widehat{\lambda}_{\mathsf{node}}$ by the same operation which takes $\rho \mapsto \sigma$, by converting it into a distribution over $(S \times \Sigma^{\tau_h})^L$ from a distribution over $S \times \Sigma^{\tau_{h+1}}$ by labeling every τ_h-th state via `iCoT`. On the other hand, the equivalent of λ_j in Lemma F.3 is the following distribution over $\boldsymbol{X}_h$,

$$\lambda_{\mathsf{node}_j}(\cdot) \propto \left(\frac{1}{L} \sum_{i=0}^{L-1} \Pr\big(X_i = \cdot \mid (\underline{X}, \boldsymbol{\eta}) \in \mathcal{K}_{\mathsf{node}_j}, \mathcal{H}_{\mathsf{node}_{j-1}}\big) \right) w_{\mathsf{node}_j}(\cdot), \tag{38}$$

where the probability is computed over $X = (X_0, \cdots, X_{L-1}) \sim \underline{\widehat{\lambda}_{\mathsf{node}}}$ and $\boldsymbol{\eta} \sim \mathrm{Unif}([0,1])^{\otimes kL}$, and recall $\mathcal{H}_{\mathsf{node}_{j-1}}$ captures all randomness until just before the dataset D_{node_j} is collected. The definition of the events $\mathcal{K}_{\mathsf{node}_j}$ is deferred to below Eq. (39), while the weight function $w_{\mathsf{node}_j}(X) = \alpha_r^{j,k}$ for $r = \mathrm{rank}(X \| \mathsf{node}_j)$ is as defined in Eq. (33). For $\underline{X} = (X_0, \cdots, X_{L-1}) \sim \underline{\lambda_{\mathsf{node}}}$, we analogously define the regression of node_j as the random variable,

$$R_{\mathsf{node}_j} = \sum_{i=0}^{L-1} \alpha^{j,k}_{\mathrm{rank}(X_i \| \mathsf{node}_j)}.$$

We first present a decomposition of the test error analogous to Lemma F.2 in the analysis of `AutoLearn.warmup`. Prior to doing so, we mention that expressions involving events on different spaces are interpreted as described in Eqs. (14) and (15).

Lemma G.9 (Test-error decomposition)**.** *Let $\widehat{\pi}^{(\mathsf{node})}$ denote the model returned by the instantiation of $\mathtt{AutoLearn}_{h+1}$ corresponding to $\mathsf{node} \in \mathcal{T}_{h+1}$. Let $\widehat{\pi}^{(\mathsf{node}_j)}$ denote the model returned by $\mathtt{AutoLearn}_{h+\frac{1}{2}}$ corresponding to $\mathsf{node}_j \in \mathsf{Ch}(\mathsf{node})$. For any nested sequence of events determined by the pre-round histories and fresh independent test variables, $\mathcal{K}_{\mathsf{node}_{k-1}} \subseteq \cdots \subseteq \mathcal{K}_{\mathsf{node}_0} \subseteq \boldsymbol{X}_h^L \times [0,1]^{kL}$,*

$$\begin{aligned}
&\Pr_{X \sim \underline{\widehat{\lambda}_{\mathsf{node}}}}\big[\widehat{\pi}^{(\mathsf{node})}(X) \neq \pi^\star_{\tau_{h+1}}(X)\big] \\
&\leqslant L \cdot \beta_0^{0,k} + \sum_{j=0}^{k-1} (\mathtt{err}_j - \mathtt{err}_\star) \cdot \mathbb{E}\big[R_{\mathsf{node}_j} \cdot \mathbb{I}((\underline{X}, \boldsymbol{\eta}) \in \mathcal{K}_{\mathsf{node}_j}) \mid \mathcal{H}_{\mathsf{node}_{k-1}}\big] \\
&\quad + \Pr\big((\underline{X}, \boldsymbol{\eta}) \notin \mathcal{K}_{\mathsf{node}_{k-1}} \mid \mathcal{H}_{\mathsf{node}_{k-1}}\big)
\end{aligned}$$

where, $\underline{X} \sim \underline{\widehat{\lambda}_{\mathsf{node}}}$, $\boldsymbol{\eta} \sim \mathrm{Unif}([0,1])^{\otimes kL}$, the weight function $\beta_r^{j,k}$ is defined in Eqs. (2) and (3) while $\mathtt{err}_j$ is the test error of $\widehat{\pi}^{(\mathsf{node}_j)}$ under the distribution $\lambda_{\mathsf{node}_j}$, $\mathtt{err}_j = \Pr_{X \sim \lambda_{\mathsf{node}_j}}\big[\widehat{\pi}^{(\mathsf{node}_j)}(X) \neq \pi^\star_{\tau_h}(X)\big]$.

Proof. The proof of this result follows from the statement of Lemma F.2 by setting $\rho \leftarrow \widehat{\lambda}_{\mathsf{node}}$, which corresponds to setting $\sigma \leftarrow \underline{\widehat{\lambda}_{\mathsf{node}}}$ and $\lambda_j \leftarrow \lambda_{\mathsf{node}_j}$, and $\mathcal{K}_j \leftarrow \mathcal{K}_{\mathsf{node}_j}$ and $\mathcal{H}_j \leftarrow \mathcal{H}_{\mathsf{node}_j}$. □

The prediction error of $\widehat{\pi}^{(\mathsf{node}_j)}$ is measured under $\lambda_{\mathsf{node}_j}$ in the definition of $\mathtt{err}_j$, but the corresponding invocation of $\mathtt{AutoLearn}_{h+\frac{1}{2}}$ trains on data drawn from a different distribution $\widehat{\lambda}_{\mathsf{node}_j}$. Our next result shows that these two distributions satisfy a density ratio bound when the event $\mathcal{K}_{\mathsf{node}_j}$ is chosen appropriately.

Lemma G.10 (Density ratio bound at depth $h + \frac{1}{2}$)**.** *Fix some $0 \leqslant j \leqslant k-1$, and recall that $\widehat{\lambda}_{\mathsf{node}_j}$ denote the distribution over instances the i.i.d. dataset D_{node_j} is sampled from. Consider*

the filtration $(\mathcal{H}_{\mathsf{node}_j})_{j=0}^{k-1}$ *and sequence of events,* $(\mathcal{U}_{\mathsf{node}_j})_{j=0}^{k-1}$ *such that* $\mathcal{U}_{\mathsf{node}_j} \subseteq \boldsymbol{X}_h^L$ *and is a measurable function of* $\mathcal{H}_{\mathsf{node}_{j-1}}$*. Fix some constant* $c \in (0,1)$*. Construct a sequence of events* $\mathcal{K}_{\mathsf{node}_j} \subseteq \boldsymbol{X}_h^L \times [0,1]^{(j+1)L}$ *via:*

$$\mathcal{K}_{\mathsf{node}_j} \leftarrow \mathtt{ConstructEvent}(\mathcal{U}_{\mathsf{node}_j} \cap \mathcal{K}_{\mathsf{node}_{j-1}} \,\|\, c, \underline{\widehat{\lambda}}_{\mathsf{node}}, \mathcal{H}_{\mathsf{node}_{j-1}})$$

where $\mathcal{K}_{\mathsf{node}_{-1}} = \boldsymbol{X}_h^L$ *and* $\mathtt{ConstructEvent}(\cdot)$ *is defined in Lemma F.3. Then, for each* $0 \leqslant j \leqslant k-1$*,*

$$\mathcal{K}_{\mathsf{node}_j} \subseteq \mathcal{U}_{\mathsf{node}_j} \cap \mathcal{K}_{\mathsf{node}_{j-1}} \quad \textit{and} \quad \Pr(\mathcal{K}_{\mathsf{node}_j} \mid \mathcal{U}_{\mathsf{node}_j} \cap \mathcal{K}_{\mathsf{node}_{j-1}}, \mathcal{H}_{\mathsf{node}_{j-1}}) \geqslant 1 - c$$

And the following density ratio bound is satisfied,

$$\left\| \frac{\lambda_{\mathsf{node}_j}}{\widehat{\lambda}_{\mathsf{node}_j}} \right\|_\infty \lesssim \frac{c^{-1}\log(L)}{\Pr\left(\mathcal{U}_{\mathsf{node}_j} \cap \mathcal{K}_{\mathsf{node}_{j-1}} \mid |\mathcal{I}_{\mathsf{node}_j}| \geqslant 1, \mathcal{H}_{\mathsf{node}_{j-1}}\right)}$$

Here, $\mathcal{I}_{\mathsf{node}_j}$ *is the set of accepted indices in a run of split-and-inverted-sampling (Algorithms 3 and 4) for* $\boldsymbol{x} \sim \underline{\widehat{\lambda}}_{\mathsf{node}}$ *(i.e.,* $\underline{X} \sim \underline{\widehat{\lambda}}_{\mathsf{node}}$*).*

Proof. The statement and proof of this result mirrors that of Lemma F.3. The relationship between $\underline{\lambda}_{\mathsf{node}}, \lambda_{\mathsf{node}_j}, \underline{\widehat{\lambda}}_{\mathsf{node}_j}$ is identical to the relationship between σ, λ_j and $\widehat{\lambda}_j$ in the earlier lemma. □

The density ratio bound from this lemma allows the transfer of learning guarantees under $\underline{\widehat{\lambda}}_{\mathsf{node}_j}$ to $\lambda_{\mathsf{node}_j}$ via a change-of-measure argument in iterations where $\Pr\left(\mathcal{U}_{\mathsf{node}_j} \cap \mathcal{K}_{\mathsf{node}_{j-1}} \mid |\mathcal{I}_{\mathsf{node}_j}| \geqslant 1, \mathcal{H}_{\mathsf{node}_{j-1}}\right)$ is sufficiently large. As in the analysis of `AutoLearn.warmup`, we define marginal and conditional acceptance probability of split-and-inverted-sampling (similar to $\widetilde{p}_j$ in Eq. (18)), and a corresponding abort event to track which child invocations of $\mathtt{AutoLearn}_h$ are invoked on sufficiently large effective datasets. For $0 \leqslant j \leqslant k-1$, define,

$$\begin{aligned}\widetilde{p}_{\mathsf{node}_j} &= \Pr(|\mathcal{I}_{\mathsf{node}_j}| \geqslant 1 \mid \mathcal{K}_{\mathsf{node}_{j-1}}, \mathcal{H}_{\mathsf{node}_{j-1}}) \\ &= \mathbb{E}\left[1 - \prod_{i=0}^{L-1}\left(1 - \frac{\alpha^{j,k}_{\mathrm{rank}(X_i \| \mathsf{node}_j)}}{\alpha^{j,k}_{\max}}\right) \,\middle|\, \mathcal{K}_{\mathsf{node}_{j-1}}, \mathcal{H}_{\mathsf{node}_{j-1}}\right]\end{aligned}$$

Note that $\widetilde{p}_{\mathsf{node}_j}$ is closely related to $p_{\mathsf{node}_j} = \Pr(|\mathcal{I}_{\mathsf{node}_j}| \geqslant 1 \mid \mathcal{H}_{\mathsf{node}_{j-1}})$, which is the probability that split-and-inverted-sampling accepts $\boldsymbol{x} \sim \underline{\widehat{\lambda}}_{\mathsf{node}}$. With this in place, we define,

$$\mathtt{ABORT}_{h+\frac{1}{2}}[\mathsf{node}_j] = \left\{\widetilde{p}_{\mathsf{node}_j} \leqslant p_j^\star\right\}$$

where $p_j^\star$ is as defined earlier in Eq. (18). With this in place, we explicitly instantiate the sequence of $\mathcal{K}_{\mathsf{node}_j}$ events. First, we define,

$$\mathcal{U}_{\mathsf{node}_j} = \left\{\widetilde{p}_{\mathsf{node}_j} \leqslant p_j^\star \text{ and } R_{\mathsf{node}_j} \geqslant \mathfrak{r}_j\right\}^c, \tag{39}$$

where the threshold $\mathfrak{r}_j$ is as defined earlier in Eq. (31). $\mathcal{U}_{\mathsf{node}_j}$ captures the event that the regression is controlled across all aborted iterations. This definition is the extension of the version we defined earlier in the analysis of `AutoLearn.warmup` in Eq. (22). With this definition of $\mathcal{U}_{\mathsf{node}_j}$, $\mathcal{K}_{\mathsf{node}_j}$ is defined recursively as,

$$\mathcal{K}_{\mathsf{node}_j} \leftarrow \mathtt{ConstructEvent}(\mathcal{U}_{\mathsf{node}_j} \cap \mathcal{K}_{\mathsf{node}_{j-1}} \,\|\, c, \underline{\widehat{\lambda}}_{\mathsf{node}}, \mathcal{H}_{\mathsf{node}_{j-1}}) \text{ with } c = \frac{1}{32k},$$

with $\mathcal{K}_{\mathsf{node}_{-1}} = \boldsymbol{X}_h^L$.

In the following lemma, we control the total probability mass of $\mathcal{K}_{\mathsf{node}_j}$.

Lemma G.11 (Mass on good events)**.** *For all* $0 \leqslant j \leqslant k-1$, $\Pr(\mathcal{K}_{\mathsf{node}_j} \mid \mathcal{H}_{\mathsf{node}_{j-1}}) \geqslant \frac{15}{16}$.

Proof. The proof of this result is identical to that of Lemma F.4. □

With this in place, in any iteration j which does not abort, i.e., $\widetilde{p}_{\mathsf{node}_j} > p_j^\star$, following the analysis we described earlier in Eq. (25) to Eq. (26), allows lower bounding

$\Pr\big(\mathcal{U}_{\mathsf{node}_j} \cap \mathcal{K}_{\mathsf{node}_{j-1}} \mid |\mathcal{I}_{\mathsf{node}_j}| \geqslant 1, \mathcal{H}_{\mathsf{node}_{j-1}}\big)$ by $\Omega(\widetilde{p}_{\mathsf{node}_j})$. This implies that in any such iteration we have the density ratio bound,

$$\left\| \frac{\lambda_{\mathsf{node}_j}}{\widehat{\lambda}_{\mathsf{node}_j}} \right\|_\infty \lesssim k^{7/2} \log(L) \tag{40}$$

Furthermore, in any non-aborted iteration j, we also show that the size of the dataset D_{node_j} passed on to the child node node_j to train a model via $\mathtt{AutoLearn}_{h+\frac{1}{2}}$ is sufficiently large.

Lemma G.12 (Large $p_{\mathsf{node}_j} \implies \mathsf{node}_j$ instantiated with a large dataset)**.** *Fix any $0 \leqslant j \leqslant k-1$ such that $\widetilde{p}_{\mathsf{node}_j} > p^\star_j$. If $\mathtt{AutoLearn}_{h+1}$ is invoked on a dataset with at least $C_2 k^{7/2} \log(L) \cdot n^\star_{\mathtt{sample}}(h+\frac{1}{2}, \delta)$ instances for sufficiently large $C_2 > 0$, then $\Pr(|D_{\mathsf{node}_j}| < n^\star_{\mathtt{sample}}(h+\frac{1}{2}, \delta) \mid \mathcal{H}_{\mathsf{node}_{j-1}}) \leqslant \delta$.*

Proof. The proof of this result follows the same approach as that of Lemma F.5, where we use a Chernoff bound to control the deviations of $|D_{\mathsf{node}_j}|$ which is an independent subsample of a $1/k$ fraction of D_{node} with probability p_{node_j}, and noticing that $p_{\mathsf{node}_j} \geqslant \Omega(p^\star_j)$ in non-aborted iterations (cf. Eq. (28)). Using the definition of $p^\star_j$ completes the proof. □

Next, we use the density ratio bound in Eq. (40), along with the sufficiently large size of the dataset D_{node_j} in any non-aborted iteration j (as implied by Lemma G.12) to establish weak learning guarantees for $\widehat{\pi}^{(\mathsf{node}_j)}$. By the induction hypothesis at the previous depth ($\mathrm{Hypothesis}_{h+\frac{1}{2}}$), and a change of measure argument, we next show that the model $\widehat{\pi}^{(\mathsf{node}_j)}$ satisfies a weak learning guarantee under the distribution $\lambda_{\mathsf{node}_j}$.

Lemma G.13 (Weak learning guarantee at depth $h+\frac{1}{2}$)**.** *Consider any iteration j such that $\widetilde{p}_{\mathsf{node}_j} > p^\star_j$. Assume $\mathrm{Hypothesis}_{h+\frac{1}{2}}(\delta)$ and furthermore that $k_{h+1}, \log(L_{h+1}) \leqslant \log^{\mathcal{O}(1)}(T)$. Suppose $k_{h+\frac{1}{2}} = c_1 \log\log(T)$ for a sufficiently large constant c_1, then with probability at least $1-2\delta$, the model $\widehat{\pi}^{(\mathsf{node}_j)}$ returned by node_j satisfies,*

$$\Pr_{X \sim \lambda_{\mathsf{node}_j}}\big[\widehat{\pi}^{(\mathsf{node}_j)}(X) \neq \pi^\star_{\tau_h}(X)\big] = \mathtt{err}_j \leqslant \mathtt{err}_\star = \frac{1}{4}$$

Proof. The proof of this result uses the density ratio bound we derived in Equation (40), the accuracy guarantee implied by $\mathrm{Hypothesis}_{h+\frac{1}{2}}(\delta)$, and a change-of-measure argument. In particular, if $\mathtt{ABORT}_{h+\frac{1}{2}}[\mathsf{node}_j]$ does not occur, then with probability at least $1-\delta$, the instance of $\mathtt{AutoLearn}_{h+\frac{1}{2}}$ corresponding to node_j is invoked on a dataset of size at least $n^\star_{\mathtt{sample}}(h+\frac{1}{2}, \delta)$ (cf. Lemma G.12). By $\mathrm{Hypothesis}_{h+\frac{1}{2}}(\delta)$, with probability at least $1-\delta$, the model $\widehat{\pi}^{(\mathsf{node}_j)}$ returned by this node satisfies,

$$\Pr_{X \sim \widehat{\lambda}_{\mathsf{node}_j}}\big[\widehat{\pi}^{(\mathsf{node}_j)}(X) \neq \pi^\star_{\tau_h}(X)\big] \leqslant 3\sqrt{k_{h+\frac{1}{2}}} \cdot e^{-ck_{h+\frac{1}{2}}}.$$

By the density ratio bound in Eq. (40), with probability at least $1-2\delta$,

$$\begin{aligned}
&\Pr_{X \sim \lambda_{\mathsf{node}_j}}\big[\widehat{\pi}^{(\mathsf{node}_j)}(X) \neq \pi^\star_{\tau_h}(X)\big] \\
&\leqslant \mathcal{O}\big(k_{h+1}^{7/2} \log(L_{h+1})\big) \cdot \Pr_{X \sim \widehat{\lambda}_{\mathsf{node}_j}}\big[\widehat{\pi}^{(\mathsf{node}_j)}(X) \neq \pi^\star_{\tau_h}(X)\big] \leqslant \frac{1}{4}.
\end{aligned}$$

for $k_{h+1} = \log(L) = \log^{\mathcal{O}(1)}(T)$, choosing $k_{h+\frac{1}{2}} = \Omega(\log\log(T))$ with a sufficiently large implicit constant gives the final inequality. □

Next we analyze the aborted iterations, i.e., $\widetilde{p}_{\mathsf{node}_j} \leqslant p^\star_j$. In these iterations, we analyze the regression R_{node_j} and show that if p_{node_j} is small, then R_{node_j} is also likely to be small.

Lemma G.14 (Aborted iterations have low regression)**.** *Define $\mathfrak{r}_j$ as in Eq. (31). Supposing that $\widetilde{p}_{\mathsf{node}_j} \leqslant p^\star_j$, then, $\Pr_{\underline{X} \sim \underline{\widehat{\lambda}_{\mathsf{node}}}}\big(R_{\mathsf{node}_j} \geqslant \mathfrak{r}_j \mid \mathcal{K}_{\mathsf{node}_{j-1}}, \mathcal{H}_{\mathsf{node}_{j-1}}\big) \leqslant \frac{1}{32k^2}$.*

Proof. The proof of this structural result is identical to the analysis of Lemma F.7, connecting $\widetilde{p}_{\mathsf{node}_j}$ and R_{node_j} via Markov's inequality. □

Finally, we combine these results to prove Lemma G.4. We restate this result in slightly more detail below.

Lemma G.15 (Analysis of composition step)**.** *Let* $\widehat{\pi}^{(\mathsf{node})}$ *denote the model returned by the instantiation of* $\mathtt{AutoLearn}_{h+1}$ *corresponding to some* $\mathsf{node} \in \mathcal{T}_{h+1}$*. Then, assuming* $L_{h+1} = L = 2^{\sqrt{\log_2(T)}}$*,* $k_{h+1} = k = c_1\sqrt{\log(T)}$ *and* $k_{h+\frac{1}{2}} = c_2 \log\log(T)$ *for appropriate* $c_1, c_2 > 0$*, as well as* $\mathrm{Hypothesis}_{h+\frac{1}{2}}(\delta)$*, with probability at least* $1 - 2\delta k_{h+1}$*,*

$$\Pr_{X\sim\widehat{\lambda}_{\mathsf{node}}}\left[\widehat{\pi}^{(\mathsf{node})}(X) \neq \pi^\star_{\tau_{h+1}}(X)\right] \leqslant \frac{1}{4}$$

Assuming $\mathrm{Hypothesis}_{h+\frac{1}{2}}(\delta)$*, this guarantee is achieved as long as* $\mathtt{AutoLearn}_{h+1}$ *draws,*

$$n^\star_{\mathsf{sample}}(h+1, 2k\delta) = n^\star_{\mathsf{sample}}(h+\tfrac{1}{2}, \delta)\cdot C_2 k_{h+1}^{7/2}\log(L_{h+1})$$

samples from $\widehat{\lambda}_{\mathsf{node}}$*, for some large constant* $C_2 > 0$*. In the process, the number of queries made to* $\mathtt{iCoT}(\cdot)$ *is upper bounded by,*

$$n^\star_{\mathsf{query}}(h+1, 2k\delta) = L_{h+1}\cdot n^\star_{\mathsf{sample}}(h+1, 2k\delta) + k_{h+1}\cdot n^\star_{\mathsf{query}}(h+\tfrac{1}{2}, \delta).$$

Proof. We first prove the bound on the accuracy, and subsequently show the recursion on the sample and query complexity.

Bound on accuracy. Let $\mathcal{J}_{\mathsf{ABORT}}[\mathsf{node}] \triangleq \left\{0 \leqslant j \leqslant k-1 : \widetilde{p}_{\mathsf{node}_j} \leqslant p^\star_j\right\}$ denote the set of aborted iterations, which is $\mathcal{H}_{\mathsf{node}_{k-2}}$ measurable. By the test-error decomposition in Lemma G.9, the choice of $k = \Omega(\log(L))$ (along with the upper bound on $\beta_0^{0,k}$ in Lemma F.9) and the lower bound we prove in Lemma G.11 showing $\Pr(\mathcal{K}_{\mathsf{node}_{k-1}} \mid \mathcal{H}_{\mathsf{node}_{k-1}}) \geqslant \frac{15}{16}$,

$$\begin{aligned}
&\Pr_{X\sim\widehat{\lambda}_{\mathsf{node}}}\left[\widehat{\pi}^{(\mathsf{node})}(X) \neq \pi^\star_{\tau_{h+1}}(X)\right] \\
&\leqslant \frac{1}{16} + \sum_{j=0}^{k-1}(\mathsf{err}_j - \mathsf{err}_\star)\cdot\mathbb{E}\left[R_{\mathsf{node}_j}\cdot\mathbb{I}((\underline{X}, \boldsymbol{\eta}) \in \mathcal{K}_{\mathsf{node}_j}) \mid \mathcal{H}_{\mathsf{node}_{k-1}}\right] + \frac{1}{16} \qquad (41)\\
&\leqslant \frac{1}{8} + \sum_{j\in\mathcal{J}_{\mathsf{ABORT}}}\mathbb{E}[R_{\mathsf{node}_j}\cdot\mathbb{I}\left(\{\widetilde{p}_{\mathsf{node}_j} \leqslant p^\star_{\mathsf{node}_j} \text{ and } R_{\mathsf{node}_j} > \mathfrak{r}_j\}^c\right) \mid \mathcal{H}_{\mathsf{node}_{k-1}}] \\
&\quad + L\sum_{j\notin\mathcal{J}_{\mathsf{ABORT}}}(\mathsf{err}_j - \mathsf{err}_\star)_+ \\
&\leqslant \frac{1}{8} + \sum_{j\in\mathcal{J}_{\mathsf{ABORT}}}\mathfrak{r}_j + L\sum_{j\notin\mathcal{J}_{\mathsf{ABORT}}}(\mathsf{err}_j - \mathsf{err}_\star)_+ \\
&\leqslant \frac{1}{4} + L\sum_{j\notin\mathcal{J}_{\mathsf{ABORT}}}(\mathsf{err}_j - \mathsf{err}_\star)_+ \qquad (42)
\end{aligned}$$

where the last inequality uses Lemma F.8, which shows that $\sum_{j=0}^{k-1}\mathfrak{r}_j \leqslant \frac{1}{8}$. By Lemma G.13, with probability $1-2\delta$, in any iteration $j \notin \mathcal{J}_{\mathsf{ABORT}}[\mathsf{node}]$, $\mathsf{err}_j \leqslant \mathsf{err}_\star$. A union bound completes the bound on accuracy.

Query and sample complexity of $\mathtt{AutoLearn}_{h+1}$. The sample complexity of $\mathtt{AutoLearn}_{h+1}$ is $n_{\mathsf{sample}}(h+1, 2k\delta) = n^\star_{\mathsf{sample}}(h+\frac{1}{2}, \delta)\cdot C_2 k^{7/2}\log(L)$ as required by Lemma G.12. The query complexity of $\mathtt{AutoLearn}_{h+1}$ is then bounded as,

$$n^\star_{\mathsf{query}}(h+1, 2k\delta) = L\cdot n^\star_{\mathsf{sample}}(h+1, 2k\delta) + k\cdot n^\star_{\mathsf{query}}(h+\tfrac{1}{2}, \delta). \qquad (43)$$

where the $L\cdot n^\star_{\mathsf{sample}}(h+1, 2k\delta)$ term accounts for the query cost of invoking $\mathtt{Split}(D_{\boldsymbol{x}}\|L)$ and the $k\cdot n^\star_{\mathsf{query}}(h+\frac{1}{2}, \delta)$ term for training the k children via $\mathtt{AutoLearn}_{h+\frac{1}{2}}$. □

With this, we are ready to furnish a proof of the final recursive guarantee.

G.3 Final Recursive Guarantee: Proof of Theorem G.1

Correctness guarantee. By Lemmas G.3 and G.4, we know that $\text{Hypothesis}_{H+\frac{1}{2}}(\delta_{H+\frac{1}{2}})$ is true where $\delta_{H+\frac{1}{2}} = \delta \cdot \prod_{h' \in \{\frac{1}{2},1,\cdots,H+\frac{1}{2}\}} 2k_{h'}$. With $k_{H+\frac{1}{2}}$ chosen as k_ε and redefining $\delta \leftarrow \delta / \prod_{h' \in \{\frac{1}{2},1,\cdots,H+\frac{1}{2}\}} 2k_{h'}$, we get that with probability at least $1-\delta$,

$$\Pr_{\boldsymbol{x} \sim \widehat{\lambda}_{\texttt{root}}} [\widehat{\pi}(\boldsymbol{x}) \neq \pi^\star_T(\boldsymbol{x})] \leqslant \varepsilon.$$

Noting that $\widehat{\lambda}_{\texttt{root}}$ is the distribution over $D_{\texttt{node}}$ with $\texttt{node} = \texttt{root}$, which is simply the initial distribution over instances, ρ, this proves the correctness guarantee for AutoLearn. Note that $\prod_{h' \in \{\frac{1}{2},1,\cdots,H+\frac{1}{2}\}} 2k_{h'} \leqslant h_\varepsilon(T) = (\log(T))^{\mathcal{O}(H)} \cdot \log^2(1/\varepsilon)$. The confidence parameter passed recursively to each base call is δ/P, where $P = \prod_{r=1/2,1,\ldots,H+1/2} 2k_r \leqslant h_\varepsilon(T)$.

Sample and query complexity. By unrolling the induction guarantees in Lemmas G.3 and G.4, and using the definition of $\text{Hypothesis}(H+\frac{1}{2},\delta)$, the sample complexity requirement of AutoLearn resolves to, $n_{\texttt{sample}}(H+\frac{1}{2},\delta) \geqslant n^\star_{\texttt{sample}}(H+\frac{1}{2},\delta)$ where, plugging in the choices of $(k_h)_{h \geqslant 0}$ and $(L_h)_{h \geqslant 0}$,

$$n^\star_{\texttt{sample}}(H+\tfrac{1}{2},\delta) \leqslant \frac{h_\varepsilon(T)}{\varepsilon} \cdot \left[n_{\texttt{weak}}\big(\delta/h_\varepsilon(T)\big) \vee \log(h_\varepsilon(T)/\delta) \right]$$

where $h_\varepsilon(T) = (\log(T))^{\mathcal{O}(H)} \cdot \log^2(1/\varepsilon)$ upper bounds $\prod_{h' \in \{\frac{1}{2},1,\cdots,H+\frac{1}{2}\}} 2k_{h'}$. The query complexity upper bound also resolves to the same quantity,

$$n^\star_{\texttt{query}}(H+\tfrac{1}{2},\delta) \leqslant \frac{h_\varepsilon(T)}{\varepsilon} \cdot \left[n_{\texttt{weak}}\big(\delta/h_\varepsilon(T)\big) \vee \log(h_\varepsilon(T)/\delta) \right]$$

For completeness, unrolling the recurrences more precisely gives $N \leqslant A_T k_\varepsilon e^{ck_\varepsilon} [n_{\texttt{weak}}(\delta/P) \vee 8\log(eP/\delta)]$, where $A_T = (\log T)^{\mathcal{O}(H)}$. The query recurrence gives $Q/N \leqslant 1 + \sum_r L_r \leqslant 1 + (2H+1)L$. This extra factor is absorbed in $2^{\mathcal{O}(\sqrt{\log T}\log\log T)}$, but must be retained in the intermediate query bound. Also $\log P = \mathcal{O}(H \log\log T + \log\log(e/\varepsilon))$. Using $k_\varepsilon e^{ck_\varepsilon} = O(\varepsilon^{-1}\log^{3/2}(e/\varepsilon))$ proves the stated h_ε bound and, for a generic weak learner, Theorem C.2 with $\log^2(e/\varepsilon)$. This completes the proof of Theorem G.1.

H Analysis of AutoLearn.RL (Algorithm 6): Proof of Theorem D.3

AutoLearn.RL is an invocation of $\texttt{AutoLearn}_1$, but where the base learner is instantiated as $\texttt{Base}(\cdot \,\|\, \delta') \equiv \texttt{RLFineTune}\big(\cdot \,\big\|\, B, \varepsilon_0, \delta'\big)$ (cf Algorithm 5) where $\varepsilon_0^{-1} \asymp k_{\frac{1}{2}}^{7/2} \log\big(L_{\frac{1}{2}}\big)$, with decomposition schedule $L_{\frac{1}{2}} = L \triangleq T/B$ and $L_1 = 1$, and branching schedule $k_{\frac{1}{2}} = C\log(T/B)$ and $k_1 = k_\varepsilon$ for a sufficiently large absolute constant $C > 0$.

We inherit the tree notation from the previous section to arrange the recursive calls of $\texttt{AutoLearn}_h$. Here, the tree $\mathcal{T}$ has levels $\{0, \frac{1}{2}, 1\}$, with $\mathcal{T}_1 = \{\texttt{root}\}$. For $h \in \{\frac{1}{2}, 1\}$ and each $\texttt{node} \in \mathcal{T}_h$, let $\texttt{Ch}(\texttt{node}) = (\texttt{node}_0, \ldots, \texttt{node}_{k_h - 1})$ denote the ordered collection of k_h children of node, each lying at level $h - \frac{1}{2}$, and set

$$\mathcal{T}_{h-\frac{1}{2}} = \bigcup_{\texttt{node} \in \mathcal{T}_h} \texttt{Ch}(\texttt{node}).$$

For $1 \leqslant j \leqslant k_h$, write $\texttt{Ch}_{<j}(\texttt{node}) = \{\texttt{node}_0, \ldots, \texttt{node}_{j-1}\}$ for the first j children of node. In particular, $\mathcal{T}_{\frac{1}{2}} = \texttt{Ch}(\texttt{root})$ and $|\mathcal{T}_{\frac{1}{2}}| = k_1$, while $|\mathcal{T}_0| = k_1 k_{\frac{1}{2}}$. As before, $D_{\texttt{node}}$ denotes the dataset processed by the invocation of $\texttt{AutoLearn}_h$ corresponding to node, and $\widehat{\lambda}_{\texttt{node}}$ denotes the distribution from which the elements of $D_{\texttt{node}}$ are sampled; in particular, $\widehat{\lambda}_{\texttt{root}} = \rho$. If $\texttt{node} \in \mathcal{T}_0$, then $\widehat{\pi}^{(\texttt{node})}$ is the model returned by Base. If $\texttt{node} \in \mathcal{T}_h$ for $h \in \{1, \frac{1}{2}\}$, define

$$\widehat{\pi}^{(\texttt{node})} = \big(\widetilde{\pi}^{(\texttt{node})}\big)^{\circ L_h}, \text{ where, } \widetilde{\pi}^{(\texttt{node})} = \texttt{Plu}\big(\{\widehat{\pi}^{(\texttt{node}')} : \texttt{node}' \in \texttt{Ch}(\texttt{node})\}\big)$$

Thus the root has k_1 level-$\frac{1}{2}$ children which it aggregates with $L_1 = 1$, while each level-$\frac{1}{2}$ node returns the L-fold composition of $k_{\frac{1}{2}}$ models which solve length-B instances.

H.1 Guess-and-Check: Simulating $\texttt{iCoT}(\cdot)$ via π^{ref} and $\mathcal{V}$

The $\texttt{iCoT}(\cdot)$ oracle is used in Algorithm 1 in two places: first, to compute the block-boundary states needed by $\texttt{Split}(\cdot \,\|\, L)$, and second, to compute the ranks used by $\texttt{InvSampling}(\cdot)$. In Algorithm 6, the $\texttt{iCoT}(\cdot)$ oracle is not available to the learner. Instead, the learner has access to a reference model π^{ref} and an outcome verifier $\mathcal{V}$. Block-boundary labels are simulated by guess-and-check. At the root, ranks are computed directly from verifier feedback on the learned models' predictions, so full terminal labels are unnecessary.

Observation H.1 (Simulating $\texttt{iCoT}(\cdot)$ via guess-and-check)**.** *Fix any length-B instance $\boldsymbol{x} \in S \times \Sigma^B$. By rolling out π^{ref} on $\boldsymbol{x}$ and querying $\mathcal{V}$ to verify each candidate terminal state, the correct terminal state $\pi^\star_B(\boldsymbol{x})$ can be identified with probability at least $1-\delta'$ using at most $\mathcal{O}(C^{(B)}_{\mathsf{ans}} \log(1/\delta'))$ calls to π^{ref} and $\mathcal{V}$.*

The proof is immediate from the outcome coverage assumption: each independent rollout produces the correct terminal state with probability at least $(C^{(B)}_{\mathsf{ans}})^{-1}$. Therefore, for a length-T instance, the boundary states required by $\texttt{Split}(\boldsymbol{x} \| L)$ can be identified by applying Observation H.1 sequentially to the $L = T/B$ consecutive length-B blocks, where the blocks are processed sequentially, utilizing the terminal state of each instance as the starting state for the next block. This costs $\mathcal{O}(LC^{(B)}_{\mathsf{ans}} \log(L/\delta'))$ verifier queries and $\mathcal{O}(TC^{(B)}_{\mathsf{ans}} \log(L/\delta'))$ reference-model state generations per instance. We will refer to this approach as $\texttt{GuessAndCheck}(\cdot \,\|\, T, B, \delta')$.

Similarly, to compute ranks in $\texttt{InvSampling}$, for a child node_j of a level-$\frac{1}{2}$ node node, the learner evaluates $\widehat{\pi}^{(\mathsf{node}')}(\boldsymbol{x})$ for $\mathsf{node}' \in \mathsf{Ch}_{<j}(\mathsf{node})$ and queries $\mathcal{V}(\boldsymbol{x}, \widehat{\pi}^{(\mathsf{node}')}(\boldsymbol{x}))$ to test whether this terminal prediction is correct. This requires at most $k_{\frac{1}{2}}$ verifier queries per length-B sub-instance.

H.2 Analysis of the $h = \frac{1}{2}$ Level: Boosting with Composition

We first analyze a node $\mathsf{node} \in \mathcal{T}_{\frac{1}{2}}$, which is one of the k_1 invocations of $\texttt{AutoLearn}_{\frac{1}{2}}$ within $\texttt{AutoLearn.RL}$. The analysis closely follows the proof of $\texttt{AutoLearn}$ in Lemma G.4 with $\tau = B$, and with $\mathsf{NTP}_{\texttt{Full-CoT}}$ replaced by $\texttt{RLFineTune}$. Calls to $\texttt{iCoT}(\cdot)$ are simulated as described in Observation H.1. In this section, we will let k denote $k_{\frac{1}{2}}$ for succinctness.

We define an abstract sequence of events $\mathcal{K}_{\mathsf{node}_j}$ for $0 \leqslant j \leqslant k-1$, which will be useful in defining the distributions under which we instantiate weak learning guarantees for $\widehat{\pi}^0, \cdots, \widehat{\pi}^{k-1}$. We translate the analysis of $\texttt{AutoLearn.warmup}$ to this setting, with the translation $\boldsymbol{X} \to S \times \Sigma^B$ and $\rho \to \widehat{\lambda}_{\mathsf{node}}$. In correspondence with this choice, let $\underline{\widehat{\lambda}_{\mathsf{node}}}$ denote the joint distribution of $\underline{X} = (X_0, \ldots, X_{L-1})$, where $X_i = (s_{Bi}, \boldsymbol{w}_{Bi+1:B(i+1)})$ and $s_{Bi} = \pi^\star_{Bi}(\boldsymbol{x})$ for $\boldsymbol{x} = (s_0, \boldsymbol{w}_{1:T}) \sim \widehat{\lambda}_{\mathsf{node}}$. Similar to how we defined it earlier in Eq. (38) in the analysis of $\texttt{AutoLearn}$, $\lambda_{\mathsf{node}_j}$ is defined as the following distribution over $S \times \Sigma^B$,

$$\lambda_{\mathsf{node}_j}(\cdot) \propto \left(\frac{1}{L} \sum_{i=0}^{L-1} \Pr\left(X_i = \cdot \mid (\underline{X}, \boldsymbol{\eta}) \in \mathcal{K}_{\mathsf{node}_j}, \mathcal{H}_{\mathsf{node}_{j-1}}\right) \right) w_{\mathsf{node}_j}(\cdot), \tag{44}$$

where the probability is computed under $\underline{X} \sim \underline{\widehat{\lambda}_{\mathsf{node}}}$ and $\boldsymbol{\eta} \sim \mathrm{Unif}([0,1])^{\otimes kL}$. The $\mathcal{K}_{\mathsf{node}_j}$ events are defined later, below Eq. (47), while the weight function $w_{\mathsf{node}_j}(X)$ equals $\alpha^{j,k}_r$ with $r = \mathrm{rank}(X \| \mathsf{node}_j) = \sum_{\ell<j} \mathbb{I}(\widehat{\pi}^{(\mathsf{node}_\ell)}(X) = \pi^\star_B(X))$. For $\underline{X} = (X_0, \ldots, X_{L-1}) \sim \underline{\widehat{\lambda}_{\mathsf{node}}}$, we define the regression of node_j as,

$$R_{\mathsf{node}_j} = \sum_{i=0}^{L-1} \alpha^{j,k}_{\mathrm{rank}(X_i \| \mathsf{node}_j)}.$$

As a consequence of the test error breakdown in Lemma G.9, we have the following decomposition: for any sequence of events $\mathcal{K}_{\mathsf{node}_{k-1}} \subseteq \cdots \subseteq \mathcal{K}_{\mathsf{node}_0} \subseteq (S \times \Sigma^B)^L \times [0,1]^{kL}$,

$$\Pr_{X \sim \widehat{\lambda}_{\mathsf{node}}}\left[\widehat{\pi}^{(\mathsf{node})}(X) \neq \pi^\star_T(X)\right]$$

$$\leqslant L \cdot \beta_0^{0,k} + \sum_{j=0}^{k-1} (\mathtt{err}_j - \mathtt{err}_\star) \cdot \mathbb{E}\big[R_{\mathsf{node}_j} \cdot \mathbb{I}((\underline{X}, \boldsymbol{\eta}) \in \mathcal{K}_{\mathsf{node}_j}) \mid \mathcal{H}_{\mathsf{node}_{k-1}}\big] + \Pr\big((\underline{X}, \boldsymbol{\eta}) \notin \mathcal{K}_{\mathsf{node}_{k-1}} \mid \mathcal{H}_{\mathsf{node}_{k-1}}\big) \tag{45}$$

where probabilities and expectations are over $\underline{X} \sim \widehat{\lambda}_{\underline{\mathsf{node}}}$, $\boldsymbol{\eta} \sim \mathrm{Unif}([0,1])^{\otimes kL}$, the weight function $\beta_r^{j,k}$ is defined in Eqs. (2) and (3), while $\mathtt{err}_\star = \frac{1}{4}$ and $\mathtt{err}_j$ is the test error of the model $\widehat{\pi}^{(\mathsf{node}_j)}$ returned by `Base` corresponding to $\mathsf{node}_j \in \mathsf{Ch}(\mathsf{node})$ under the distribution $\lambda_{\mathsf{node}_j}$. That is,

$$\mathtt{err}_j = \Pr_{X \sim \lambda_{\mathsf{node}_j}}\big[\widehat{\pi}^{(\mathsf{node}_j)}(X) \neq \pi_B^\star(X)\big]. \tag{46}$$

Note that the model $\widehat{\pi}^{(\mathsf{node}_j)}$ is trained under i.i.d. examples drawn from the distribution $\widehat{\lambda}_{\mathsf{node}_j}$, but the prediction error $\mathtt{err}_j$ above is calculated under the distribution $\lambda_{\mathsf{node}_j}$. In the sequel, we will show the $\mathcal{K}_{\mathsf{node}_j}$ events can be chosen in such a way that the two distributions satisfy a bound on the density ratio. To this end, define,

$$\mathcal{U}_{\mathsf{node}_j} = \big\{\widetilde{p}_{\mathsf{node}_j} \leqslant p_j^\star \text{ and } R_{\mathsf{node}_j} \geqslant \mathfrak{r}_j\big\}^c \tag{47}$$

where $\mathfrak{r}_j$ is defined in Eq. (31). This event captures the regression across all aborted iterations not growing to be too large, and extends the definitions we introduced earlier (e.g., Eq. (39)) in the analysis of `AutoLearn.warmup` and `AutoLearn`. The event $\mathcal{K}_{\mathsf{node}_j}$ is obtained recursively via,

$$\mathcal{K}_{\mathsf{node}_j} \leftarrow \mathtt{ConstructEvent}\big(\mathcal{U}_{\mathsf{node}_j} \cap \mathcal{K}_{\mathsf{node}_{j-1}} \mid c, \widehat{\lambda}_{\underline{\mathsf{node}}}, \mathcal{H}_{\mathsf{node}_{j-1}}\big) \text{ where } c = \frac{1}{32k}$$

with $\mathcal{K}_{\mathsf{node}_{-1}} = (S \times \Sigma^B)^L$ (extended by independent test coins), where $\mathtt{ConstructEvent}(\cdot)$ was defined earlier in Lemma G.10. With this definition, the $\mathcal{K}_{\mathsf{node}_j}$ events inherit the property we established earlier in Lemma G.11: for all $0 \leqslant j \leqslant k-1$,

$$\Pr(\mathcal{K}_{\mathsf{node}_j} \mid \mathcal{H}_{\mathsf{node}_{j-1}}) \geqslant \frac{15}{16}. \tag{48}$$

By following the same argument we used to derive Eq. (40), in any iteration j which does not abort, i.e., $\widetilde{p}_{\mathsf{node}_j} > p_j^\star$, where $p_j^\star$ was defined in Eq. (18), we have,

$$\left\| \frac{\lambda_{\mathsf{node}_j}}{\widehat{\lambda}_{\mathsf{node}_j}} \right\|_\infty \lesssim k^{7/2} \log(L). \tag{49}$$

Define $n_{\mathsf{weak}}^{\mathrm{RL}}(\delta)$ as any upper bound on the sample complexity of `RLFineTune` (Proposition D.2) at the target error $\varepsilon_0 = \frac{1}{C_1 k^{7/2} \log(L)}$, say,

$$n_{\mathsf{weak}}^{\mathrm{RL}}(\delta) = \mathcal{O}\Big(k^{7/2} \log(L) \cdot \Big(d \log\Big(eB|S|C_{\mathsf{seq}}^{(B)}\Big) \log(ek \log L) + \log(e/\delta)\Big)\Big). \tag{50}$$

where C_1 is sufficiently large.

Lemma H.2 (Large $\widetilde{p}_{\mathsf{node}_j} \implies \mathsf{node}_j$ instantiated with a large dataset)**.** *Suppose, $|D_{\mathsf{node}}| \geqslant n_{\mathsf{sample}}^\star(\frac{1}{2}, \delta) = C_1 k^{7/2} \log(L) \cdot n_{\mathsf{weak}}^{\mathrm{RL}}(\delta/2k)$ and consider any iteration j which does not abort, i.e., $\widetilde{p}_{\mathsf{node}_j} > p_j^\star$. Then, with probability $1 - \frac{\delta}{2k}$, $|D_{\mathsf{node}_j}| \geqslant n_{\mathsf{weak}}^{\mathrm{RL}}(\delta/2k)$.*

Proof. The proof is identical to that of Lemma G.12. □

Next, we establish weak learning guarantees for the model $\widehat{\pi}^j$ in iterations j which did not abort. Recall that `AutoLearn.RL` uses `RLFineTune` (Algorithm 5), which is trained to solve length-τ instances in the datasets D_{node_j} for $j = 0, \cdots, k-1$. Rajaraman et al. (2026) established guarantees for `RLFineTune`, and via a change-of-measure argument to transfer learning guarantees under $\widehat{\lambda}_{\mathsf{node}_j}$ to $\lambda_{\mathsf{node}_j}$ (facilitated by the density ratio bound in Eq. (49)), this implies learning guarantees for $\widehat{\pi}^{(\mathsf{node}_j)}$ under the distribution $\lambda_{\mathsf{node}_j}$, which is precisely $\mathtt{err}_j$.

Lemma H.3 (Weak learning guarantee for `RLFineTune`)**.** *Fix a child $\mathsf{node}_j \in \mathsf{Ch}(\mathsf{node})$. With probability at least $1 - \frac{\delta}{k}$, in any iteration j which does not abort, $\mathtt{err}_j \leqslant \mathtt{err}_\star = \frac{1}{4}$, where $\mathtt{err}_j$ is defined in Eq. (46) and $\mathtt{err}_\star = \frac{1}{4}$.*

Proof. By Equation (49), $\|\lambda_{\mathsf{node}_j}/\widehat{\lambda}_{\mathsf{node}_j}\|_\infty \lesssim k^{7/2}\log(L)$. In order for $\widehat{\pi}^{(\mathsf{node}_j)}$ to achieve error $\frac{1}{4}$ under $\lambda_{\mathsf{node}_j}$, it suffices for it to achieve prediction error at most $\varepsilon_0 = (C_1 k^{7/2}\log(L))^{-1}$ under $\widehat{\lambda}_{\mathsf{node}_j}$ as long as $C_1 > 0$ is sufficiently large. This is guaranteed by Proposition D.2 and noting that by Lemma H.2, with probability at least $1 - \frac{\delta}{2k}$, $|D_{\mathsf{node}_j}| \geqslant n^{\mathrm{RL}}_{\mathsf{weak}}(\delta/2k)$ meets the required sample threshold for `RLFineTune` to achieve error ε_0 with probability at least $1 - \frac{\delta}{2k}$. □

With these guarantees in place, we can finally derive a guarantee for $\widehat{\pi}^{(\mathsf{node})}$.

Lemma H.4 (Constant-accuracy guarantee for $h = \frac{1}{2}$ learners)**.** *Suppose* $\mathsf{node} \in \mathcal{T}_{\frac{1}{2}}$ *is invoked on a dataset* D_{node} *consisting of at least* $n^\star_{\mathsf{sample}}(\frac{1}{2},\delta)$ *independent instances drawn from* $\widehat{\lambda}_{\mathsf{node}}$*, where* $n^\star_{\mathsf{sample}}(\frac{1}{2},\delta)$ *is defined in Lemma H.2. The associated model* $\widehat{\pi}^{(\mathsf{node})} : S \times \Sigma^T \to S$ *satisfies, with probability at least* $1-\delta$*,*

$$\Pr_{\boldsymbol{x}\sim\widehat{\lambda}_{\mathsf{node}}}\left[\widehat{\pi}^{(\mathsf{node})}(\boldsymbol{x}) \neq \pi^\star_T(\boldsymbol{x})\right] \leqslant \frac{1}{4}.$$

Furthermore, the query complexity in achieving this guarantee is upper bounded by,

$$\widetilde{\mathcal{O}}\left(n^\star_{\mathsf{sample}}(\tfrac{1}{2},\delta)\left(\frac{TC^{(B)}_{\mathsf{ans}}}{B} + C^{(B)}_{\mathsf{seq}}\right)\right),$$

and computational cost $\widetilde{\mathcal{O}}\left(n^\star_{\mathsf{sample}}(\frac{1}{2},\delta)\cdot\left(TC^{(B)}_{\mathsf{ans}} + BC^{(B)}_{\mathsf{seq}}\right)\right)$.

Proof. Let $\mathcal{J}_{\mathsf{ABORT}}[\mathsf{node}] = \{0 \leqslant j \leqslant k-1 : \widetilde{p}_{\mathsf{node}_j} \leqslant p^\star_j\}$ denote the set of aborted iterations. Combining Eq. (45) and Lemma H.3, we arrive at the following sequence of inequalities,

$$\begin{aligned}
&\Pr_{\boldsymbol{x}\sim\widehat{\lambda}_{\mathsf{node}}}\left[\widehat{\pi}^{(\mathsf{node})}(\boldsymbol{x}) \neq \pi^\star_T(\boldsymbol{x})\right] \\
&\leqslant L\cdot\beta_0^{0,k} + \sum_{j=0}^{k-1}(\mathtt{err}_j - \mathtt{err}_\star)\cdot\mathbb{E}\left[R_{\mathsf{node}_j}\cdot\mathbb{I}((\underline{X},\boldsymbol{\eta})\in\mathcal{K}_{\mathsf{node}_j}) \mid \mathcal{H}_{\mathsf{node}_{k-1}}\right] \\
&\qquad + \Pr((\underline{X},\boldsymbol{\eta})\notin\mathcal{K}_{\mathsf{node}_{k-1}} \mid \mathcal{H}_{\mathsf{node}_{k-1}}) \\
&\overset{(a)}{\leqslant} \frac{1}{8} + \sum_{j=0}^{k-1}(\mathtt{err}_j - \mathtt{err}_\star)\cdot\mathbb{E}\left[R_{\mathsf{node}_j}\cdot\mathbb{I}((\underline{X},\boldsymbol{\eta})\in\mathcal{K}_{\mathsf{node}_j}) \mid \mathcal{H}_{\mathsf{node}_{k-1}}\right] \\
&\overset{(b)}{\leqslant} \frac{1}{8} + \sum_{j\in\mathcal{J}_{\mathsf{ABORT}}}\mathfrak{r}_j + L\sum_{j\notin\mathcal{J}_{\mathsf{ABORT}}}(\mathtt{err}_j - \mathtt{err}_\star)_+
\end{aligned}$$

where (a) uses Eq. (48) and the choice of $k = \Theta(\log L)$, which bounds $L\beta_0^{0,k} \leqslant 1/16$ (cf. Lemma F.9), and (b) uses the same analysis we carried out in Eq. (41) to Eq. (42). By Lemma F.8 and Lemma H.3, with probability at least $1-\delta$,

$$\Pr_{\boldsymbol{x}\sim\widehat{\lambda}_{\mathsf{node}}}\left[\widehat{\pi}^{(\mathsf{node})}(\boldsymbol{x}) \neq \pi^\star_T(\boldsymbol{x})\right] \leqslant \frac{1}{8} + \frac{1}{8} = \frac{1}{4}.$$

Sample and query complexity, and computational cost. The sample complexity at the $h = \frac{1}{2}$ depth is $n^\star_{\mathsf{sample}}(\frac{1}{2},\delta)$ from Lemma H.2. For the bounds on the query complexity and computational cost, simulating block boundaries on $n^\star_{\mathsf{sample}}(\frac{1}{2},\delta)$ instances contributes a total of $\widetilde{\mathcal{O}}(n^\star_{\mathsf{sample}}(\frac{1}{2},\delta)\cdot(T/B)C^{(B)}_{\mathsf{ans}})$ verifier queries and $\widetilde{\mathcal{O}}(n^\star_{\mathsf{sample}}(\frac{1}{2},\delta)\cdot TC^{(B)}_{\mathsf{ans}})$ reference-model state generations. The calls to `RLFineTune` use $\widetilde{\mathcal{O}}(n^\star_{\mathsf{sample}}(\frac{1}{2},\delta))$ accepted length-B training instances in total, which contributes $\widetilde{\mathcal{O}}(n^\star_{\mathsf{sample}}(\frac{1}{2},\delta)\cdot C^{(B)}_{\mathsf{seq}})$ verifier queries and requires $\widetilde{\mathcal{O}}(n^\star_{\mathsf{sample}}(\frac{1}{2},\delta)\cdot C^{(B)}_{\mathsf{seq}})$ length-B rollouts from π^{ref}. Combining both bounds results in the bound on query complexity and computational cost. □

H.3 Analysis of the $h = 1$ Level: Boosting to Error ε

We next analyze the root node of $\mathcal{T}$. Since $L_1 = 1$, this node only carries out aggregation and no composition. `root` invokes the constant-accuracy guarantee from Lemma H.4 at each node in

$\mathcal{T}_{\frac{1}{2}}$, on a sequence of reweighted distributions over length-T instances, and aggregates the resulting models by plurality vote. Because $L_1 = 1$, conditional on the previous models and accepted count, `InvSampling` produces *exact* i.i.d. samples from the tilted distribution $\lambda_{\mathtt{root}_j} \propto \rho \cdot w_{\mathtt{root}_j}$, so no change-of-measure or good-event truncation is needed (in contrast with the half level). The analysis exactly mirrors the no-composition aggregation step of Lemma G.8. For the purpose of this section, we will let k_1 be denoted as k.

Throughout, we write $\mathtt{root}_j$ for the j^{th} child of the root and set $\rho = \widehat{\lambda}_{\mathtt{root}}$. The models $\{\widehat{\pi}^{(\mathtt{root}_j)} : 0 \leqslant j < k\}$ produced by the half-level instances at the children of the root play the role of the weak learners at this level. For $X \sim \rho$, the rank $\mathrm{rank}(X \| \mathtt{root}_j)$ counts how many of the previous models $\widehat{\pi}^{(\mathtt{root}_0)}, \ldots, \widehat{\pi}^{(\mathtt{root}_{j-1})}$ correctly predict $\pi^\star_T(X)$. On the other hand, the weight $w_{\mathtt{root}_j}(X) = \alpha^{j,k}_{\mathrm{rank}(X\|\mathtt{root}_j)}$ defines the distribution `InvSampling` samples from (via rejection sampling from ρ) in the j^{th} child node of `root`. In particular, since $L_1 = 1$, `InvSampling` samples i.i.d. from,

$$\widehat{\lambda}_{\mathtt{root}_j}(\cdot) \propto \rho(\cdot)\, w_{\mathtt{root}_j}(\cdot),$$

The marginal acceptance probability of `InvSampling` at round j is, by specializing Eq. (17) to $L = 1$,

$$p_{\mathtt{root}_j} = \mathbb{E}_{X\sim\rho}\left[\frac{\alpha^{j,k}_{\mathrm{rank}(X\|\mathtt{root}_j)}}{\alpha^{j,k}_{\max}} \,\middle|\, \mathcal{H}_{\mathtt{root}_{j-1}}\right], \tag{51}$$

where $\mathcal{H}_{\mathtt{root}_{j-1}}$ is all the randomness within the algorithm up until just before the dataset $D_{\mathtt{root}_j}$ is collected. $p_{\mathtt{root}_j}$ is the probability with which an instance $\boldsymbol{x} \sim \widehat{\lambda}_{\mathtt{root}}$ is accepted by split-and-inverted-sampling (Algorithms 3 and 4). For each child $\mathtt{root}_j \in \mathtt{Ch}(\mathtt{root})$, define the abort event,

$$\mathtt{ABORT}_1[\mathtt{root}_j] = \left\{p_{\mathtt{root}_j} \leqslant p^\star_j\right\}, \text{ where, } p^\star_j = e^{-ck} \tag{52}$$

where $c > 0$ is the constant in Lemma F.9. Firstly, as a consequence of the test-error decomposition we proved earlier in Lemma G.5, with $\mathtt{node} \leftarrow \mathtt{root}$ and $\widehat{\lambda}_{\mathtt{node}} \leftarrow \rho$, the model $\widehat{\pi}^{(\mathtt{root})}$ satisfies,

$$\Pr_{\boldsymbol{x}\sim\rho}\left[\widehat{\pi}^{(\mathtt{root})}(\boldsymbol{x}) \neq \pi^\star_T(\boldsymbol{x})\right] \leqslant \beta^{0,k}_0 + \sum_{j=0}^{k-1}(\mathtt{err}_j - \mathtt{err}_\star)\cdot \mathbb{E}_{X\sim\rho}\left[\alpha^{j,k}_{\mathrm{rank}(X\|\mathtt{root}_j)}\right], \tag{53}$$

where $\mathtt{err}_\star = \frac{1}{4}$ and $\mathtt{err}_j = \Pr_{X\sim\widehat{\lambda}_{\mathtt{root}_j}}\left[\widehat{\pi}^{(\mathtt{root}_j)}(X) \neq \pi^\star_T(X)\right]$. First we argue that in any iteration which doesn't abort, the dataset $D_{\mathtt{root}_j}$ must be sufficiently large.

Lemma H.5 (Truthful aborts at `root`). *Suppose $|D_{\mathtt{root}}| \geqslant n^\star_{\mathtt{sample}}(1, 2k\delta) \triangleq 2ke^{ck} \cdot n^\star_{\mathtt{sample}}(\frac{1}{2}, \delta)$, where $c > 0$ is the constant in the exponent of $\beta^{0,k}_0$ in Lemma F.9. Then, for any iteration j such that $p_{\mathtt{root}_j} \geqslant e^{-ck}$, $\Pr\left(|D_{\mathtt{root}_j}| < n^\star_{\mathtt{sample}}(\frac{1}{2}, \delta) \mid \mathcal{H}_{\mathtt{root}_{j-1}}\right) \leqslant \delta$.*

Proof. The proof of this result follows the same structure as Lemma G.7. □

As a consequence of this lemma, in the iterations that don't abort, the corresponding instantiations of $\mathtt{AutoLearn}_{\frac{1}{2}}$ are invoked on a dataset $D_{\mathtt{root}_j}$ containing at least $n^\star_{\mathtt{sample}}(\frac{1}{2}, \delta)$ samples.

Lemma H.6 (Weak learner guarantee for $h = \frac{1}{2}$). *Consider any j such that $p_{\mathtt{root}_j} \geqslant p^\star_j$. With probability at least $1 - 2\delta$, the model $\widehat{\pi}^{(\mathtt{root}_j)}$ returned by $\mathtt{root}_j$ satisfies,*

$$\Pr_{X\sim\lambda_{\mathtt{root}_j}}\left[\widehat{\pi}^{(\mathtt{root}_j)}(X) \neq \pi^\star_T(X)\right] = \mathtt{err}_j \leqslant \mathtt{err}_\star \triangleq \frac{1}{4}$$

Proof. Identical to Lemma G.6. If $\mathtt{ABORT}_1[\mathtt{root}_j]$ is false, then the instance of $\mathtt{AutoLearn}_{\frac{1}{2}}$ at $\mathtt{root}_j$ is invoked with at least $n^\star_{\mathtt{sample}}(\frac{1}{2}, \delta)$ accepted instances from $\widehat{\lambda}_{\mathtt{root}_j}$. As a consequence of Lemma H.4, we have that, $\Pr_{X\sim\lambda_{\mathtt{root}_j}}\left[\widehat{\pi}^{(\mathtt{root}_j)}(X) \neq \pi^\star_T(X)\right] \leqslant \frac{1}{4}$ with probability at least $1 - \delta$. □

Combining these lemmas, we establish a guarantee for the model $\pi^{(\mathtt{root})}$.

Lemma H.7 (Guarantee at `root`). *Fix $\varepsilon \in (0, 1)$. Let $\widehat{\pi} \leftarrow \widehat{\pi}^{(\mathtt{root})}$ denote the model returned by the instantiation of $\mathtt{AutoLearn}_1$ corresponding to `root`, i.e., the output of `AutoLearn.RL`. Suppose $k = k_\varepsilon$. Then, with probability at least $1 - 2k\delta$, $\Pr_{X\sim\widehat{\lambda}_{\mathtt{root}}}\left[\widehat{\pi}(X) \neq \pi^\star_T(X)\right] \leqslant \varepsilon$. Furthermore, this guarantee is achieved if `root` is invoked on a dataset of $n^\star_{\mathtt{sample}}(1, 2k\delta)$ samples from $\widehat{\lambda}_{\mathtt{root}}$.*

Proof. Conditioned on the event that $\texttt{ABORT}_1[\texttt{root}_j]$ is false for some j, with probability at least $1-2\delta$, we have that $\texttt{err}_j \leqslant \texttt{err}_\star$ (Lemma H.6). Noting that in any iteration j where $\texttt{ABORT}_1[\texttt{root}_j]$ is true, $p_{\texttt{root}_j} \leqslant p^\star_j$. Using the definition of $p_{\texttt{root}_j}$ and by a union bound, w.p. at least $1-2k\delta$,

$$\begin{aligned}\Pr_{X\sim\widehat{\lambda}_{\texttt{root}}}\left[\widehat{\pi}^{(\texttt{root})}(X) \neq \pi^\star_T(X)\right] &\leqslant \beta_0^{0,k} + \sum_{j=0}^{k-1} e^{-ck}\cdot\alpha_{\max}^{j,k} \\ &\overset{(a)}{\leqslant} e^{-ck} + 2\sqrt{k}\cdot e^{-ck} \\ &\leqslant 3\sqrt{k}\cdot e^{-ck}\end{aligned}$$

In (a), we use Lemmas F.8 and F.9 (the bound on $\sum_{j=0}^{k-1}\alpha_{\max}^{j,k}$ is implied by the bound on $\sum_{j=0}^{k-1}\mathfrak{r}_j$). Finally, using the definition of k_ε completes the proof. □

H.4 Proof of Theorem D.3

Recall that $\texttt{AutoLearn.RL}$ is the invocation of $\texttt{AutoLearn}_1$ at $\texttt{root}$ with the base learner $\texttt{RLFineTune}$. The accuracy guarantee for $\texttt{AutoLearn.RL}$ is immediate: Lemma H.7, applied at $\texttt{root}$ with $\widehat{\lambda}_{\texttt{root}} = \rho$ shows that the model $\widehat{\pi}^{(\texttt{root})}$ returned by $\texttt{AutoLearn.RL}$ satisfies $\Pr_{\boldsymbol{x}\sim\rho}\left[\widehat{\pi}^{(\texttt{root})}(\boldsymbol{x}) \neq \pi^\star_T(\boldsymbol{x})\right] \leqslant \varepsilon$ with probability at least $1-2k_1\delta$, and redefining $\delta \leftarrow \delta/(4k_1)$ gives failure probability at most $\delta/2$ in the ideal process with exact block labels. It remains to bound the sample, query, and computational complexity of $\texttt{AutoLearn.RL}$.

Couple the actual process to this ideal process using the same data and sampling coins. There are at most $M_G = k_1 n^\star_{\texttt{sample}}(1/2, \delta/(4k_1))$ guess-and-check invocations, since every child truncates its input before splitting it. Conditional on any successful past, the next invocation fails with probability at most δ_1 by the uniform block-coverage assumption. A union bound gives total simulation failure at most $\delta/2$. Until such a failure the two processes coincide, so the total learning failure is at most δ. This argument does not condition the i.i.d. training data on joint simulation success. The intermediate lemmas above concern the ideal process; their displayed costs include the capped simulation costs.

Bound on sample complexity. By Lemma H.7, $\texttt{root}$ requires a dataset of $n^\star_{\texttt{sample}}(1,\delta/2) = \widetilde{\mathcal{O}}(d/\varepsilon)$ instances drawn from ρ, resulting in the upper bound on sample complexity of $\texttt{AutoLearn.RL}$.

Bound on query complexity. Verifier queries are incurred by $\texttt{AutoLearn.RL}$ in two places. First, at $\texttt{root}$, the aggregation step of $\texttt{AutoLearn}_1$ uses $\mathcal{V}$ to compute the rank of each of the $n^\star_{\texttt{sample}}(1,\delta/2)$ length-T instances with respect to the k_1 child models, contributing at most $k_1 n^\star_{\texttt{sample}}(1,\delta/2)$ verifier queries in total. Second, at each child $\texttt{node} \in \mathcal{T}_{\frac{1}{2}}$, by Lemma H.4, each invocation of $\texttt{AutoLearn}_{\frac{1}{2}}$ incurs a burn-in query cost of

$$\widetilde{\mathcal{O}}\left(\frac{dTC^{(B)}_{\texttt{ans}}}{B} + dC^{(B)}_{\texttt{seq}}\right) = \frac{n_{\texttt{burn-in}}}{B},$$

Summing over the $k_1 = \widetilde{\mathcal{O}}(1)$ children in $\mathcal{T}_{\frac{1}{2}}$ and combining with the contribution from $\texttt{root}$ gives,

$$n_{\texttt{query}} \leqslant \widetilde{\mathcal{O}}\left(\frac{d}{\varepsilon}\right) + \frac{n_{\texttt{burn-in}}}{B}.$$

Note that these bounds are stated for when the failure probability is $2k_1\delta$. Replacing $\delta \leftarrow \delta/(4k_1)$ results in a bound which is larger by a factor of $\log(2k_1) = \widetilde{\mathcal{O}}(1)$.

Bound on computational cost. The computational cost of $\texttt{AutoLearn.RL}$ likewise decomposes into a contribution at $\texttt{root}$ and contributions at the children $\texttt{node} \in \mathcal{T}_{\frac{1}{2}}$. At $\texttt{root}$, evaluating each of the k_1 length-T child models on each of the $\widetilde{\mathcal{O}}(d/\varepsilon)$ instances costs $\widetilde{\mathcal{O}}(dT/\varepsilon)$ state computations.

At each child node $\in \mathcal{T}_{\frac{1}{2}}$, the invocation of $\texttt{AutoLearn}_{\frac{1}{2}}$ incurs a burn-in computational cost of $\widetilde{\mathcal{O}}(dTC^{(B)}_{\texttt{ans}} + dBC^{(B)}_{\texttt{seq}}) = n_{\texttt{burn-in}}$ by Lemma H.4. Summing over the $k_1 = \widetilde{\mathcal{O}}(1)$ children gives

$$n_{\texttt{comp}} \leqslant \widetilde{\mathcal{O}}\left(\frac{dT}{\varepsilon}\right) + n_{\texttt{burn-in}}.$$

Note that, as in the analysis of the query complexity, this bound is stated for when the failure probability is $2k_1\delta$. Replacing $\delta \leftarrow \delta/(4k_1)$ results in a bound which is larger by a factor of $\log(k_1) = \widetilde{\mathcal{O}}(1)$. Combining the accuracy, sample, query, and computational bounds completes the proof of Theorem D.3.

I Proofs of Lemmas

I.1 Upper Bounding Prediction Error via Potentials: Proof of Lemma F.2

In this section, we will produce an upper bound which decomposes the error of the model returned by the depth-1 construction into the regression accumulated across iterations which abort. In order to establish this lemma, we will first introduce a potential function which will serve as a proxy to bounding the accuracy of the plurality model. For $0 \leqslant j \leqslant k$, define the potential function,

$$\Phi_j = \sum_{i=0}^{L-1} \mathbb{E}\left[\beta^{j,k}_{\mathrm{rank}_j(X_i)} \cdot \mathbb{I}((\underline{X}, \boldsymbol{\eta}) \in \mathcal{K}_{j-1}) \mid \mathcal{H}_{k-1}\right]$$

Here, $\underline{X} \sim \sigma$ and $\boldsymbol{\eta} = (\boldsymbol{\eta}_j)_{j=0}^{k-1}$ where $\boldsymbol{\eta}_j = (\eta_{j,i})_{i=0}^{L-1}$. The proof of Lemma F.2 follows by first showing how the potential function Φ_k relates to the test loss of the final model returned by $\texttt{AutoLearn.warmup}$, $\widehat{\pi}$. Then, we show how Φ_k can be analyzed by writing it down as $\Phi_0 + \sum_{j=0}^{k-1}(\Phi_{j+1} - \Phi_j)$ and bounding the successive differences.

Lemma I.1 (Potential function decomposition)**.** *Let $\widehat{\pi}$ denote the model returned by* $\texttt{AutoLearn.warmup}$*. Then,*

$$\Pr_{\boldsymbol{x}\sim\rho}\left[\widehat{\pi}(\boldsymbol{x}) \neq \pi^\star_T(\boldsymbol{x})\right] \leqslant \Phi_k + \Pr((\underline{X}, \boldsymbol{\eta}) \notin \mathcal{K}_{k-1} \mid \mathcal{H}_{k-1}).$$

Proof. The proof of this result is deferred to Appendix J.2. □

Next we show how the potential difference $\Phi_{j+1} - \Phi_j$ evolves, by relating it to the $\texttt{err}_j$ error terms. The following lemma is adapted from (Freund, 1995, Lemma 3.7), with a subtle modification to account for the nested $\mathcal{K}_j$ events.

Lemma I.2 (Recursive bound on potentials)**.** *Let $\texttt{err}_j$ be as defined in the statement of Lemma F.6. The potential function Φ_j satisfies the following recurrence relation,*

$$\Phi_{j+1} - \Phi_j \leqslant (\texttt{err}_j - \texttt{err}_\star) \cdot \mathbb{E}\left[R_j \cdot \mathbb{I}((\underline{X}, \boldsymbol{\eta}) \in \mathcal{K}_j) \mid \mathcal{H}_{k-1}\right]$$

Proof. The proof of this result is deferred to Appendix J.3. □

I.1.1 Proof of Lemma F.2

By cascading Lemma I.2 across j and combining with Lemma I.1, we get,

$$\begin{aligned}
&\Pr_{\boldsymbol{x}\sim\rho}\left[\widehat{\pi}(\boldsymbol{x}) \neq \pi^\star_T(\boldsymbol{x})\right] \\
&\leqslant \Phi_k + \Pr((\underline{X}, \boldsymbol{\eta}) \notin \mathcal{K}_{k-1} \mid \mathcal{H}_{k-1}) \\
&\leqslant \Phi_0 + \sum_{j=0}^{k-1} (\texttt{err}_j - \texttt{err}_\star) \cdot \mathbb{E}\left[R_j \cdot \mathbb{I}((\underline{X}, \boldsymbol{\eta}) \in \mathcal{K}_j) \mid \mathcal{H}_{k-1}\right] \\
&\qquad + \Pr((\underline{X}, \boldsymbol{\eta}) \notin \mathcal{K}_{k-1} \mid \mathcal{H}_{k-1}).
\end{aligned}$$

Noting that $\Phi_0 = \sum_{i=0}^{L-1} \mathbb{E}\left[\beta^{0,k}_{\mathrm{rank}_0(X_i)} \mid \mathcal{H}_{k-1}\right] = L \cdot \beta_0^{0,k}$ completes the proof.

I.2 Sampling Approximately from λ_j: Proof of Lemma F.3

In this section, we will prove Lemma F.3 by studying the underlying problem of sampling from a reweighted mixture of marginals (Definition E.1) which captures its more general structure.

The problem of sampling from the reweighted mixture of marginals posits that the learner has sampling access to a distribution μ over some product space $\boldsymbol{X}^L$, with the objective of approximately generating samples from the distribution $\nu(\cdot) \propto \frac{1}{L}\sum_{i=0}^{L-1} \mu(X_i = \cdot)w(\cdot)$ for some weight function $w : \boldsymbol{X} \to [0,1]$. For appropriately chosen w, ν captures the idealized distribution to train weak learners under when boosting is used to train a model with small error under $\overline{\mu}$. When connected back to Lemma F.3, μ corresponds to the distribution σ over tuples (of size L) of instances of length τ, while $\nu(\cdot)$ corresponds to a reweighted version of the distribution $\overline{\rho}(\cdot) = \frac{1}{L}\sum_{i=0}^{L-1} \sigma(X_i = \cdot)$. We use $\underline{X} = (X_0, \cdots, X_{L-1})$ to denote samples from μ. The first part of this section will focus on the case where there is a single weight function and the goal is to sample from its reweighted mixture of marginals, ν.

What does it mean to sample from ν *approximately*? It is worth stopping to ask what kind notion of approximation we are willing to tolerate to sample from ν. In the context of boosting, the learner aims to train a model which achieves constant (say, $\frac{1}{4}$) error under ν. To achieve such a guarantee, it suffices for the learner to generate a sample from some other distribution $\widehat{\nu}$ which satisfies the following *approximate coverage* guarantee:

$$\Pr_{X\sim\nu}\left(\frac{\nu(X)}{\widehat{\nu}(X)} \geqslant C_{\mathsf{cov}}\right) \leqslant \delta_{\mathsf{cov}} \tag{54}$$

In particular, under the above assumption, as long as $\delta_{\mathsf{cov}} < \frac{1}{4}$, a model which achieves error $\varepsilon = C_{\mathsf{cov}}^{-1}(\frac{1}{4} - \delta_{\mathsf{cov}})$ under the distribution over inputs $\widehat{\nu}$ also achieves error $\frac{1}{4}$ under the distribution ν. The requirement in Eq. (54) is natural, and in fact (via an application of Markov's inequality) implied by closeness of ν and $\widehat{\nu}$ in any f-divergence as long as f grows faster than a linear function. In particular, a bound on $D_{\mathrm{KL}}\left(\nu \,\middle\|\, \widehat{\nu}\right)$ implies Eq. (54) with appropriately chosen C and δ_C.

Sampling from a distribution with approximate coverage to ν is only a sufficient condition for transferring error guarantees under $\widehat{\nu}$ to under ν. Since we aim to train models via boosting, it may suffice to use a weaker notion of approximation, and we discuss a candidate in the next paragraph. The main reason for doing so is not merely convenience: in Lemma I.6 we establish a *strong lower bound* to achieve the guarantee in Eq. (54) unless C_{cov} scales as $\mathrm{poly}(L)$ or the sampling algorithm uses $\mathrm{poly}(L)$ samples from μ.

I.2.1 A weakening of approximate coverage

Rather than requiring the learner to satisfy Eq. (54), we will consider the following weakened version. Let $\widetilde{\mu}$ denote any distribution on $\boldsymbol{X}^L$ such that $\mathrm{TV}\left(\mu, \widetilde{\mu}\right)$ is at most c. Define $\widetilde{\nu} \propto \widetilde{\overline{\mu}}(\cdot)w(\cdot)$ where $\widetilde{\overline{\mu}}$ is the mixture of marginals of $\widetilde{\mu}$. Define the following weakened version of approximate coverage,

$$\text{There exists } \widetilde{\mu} \text{ such that } \mathrm{TV}\left(\mu, \widetilde{\mu}\right) \leqslant c \text{ and } \Pr_{X\sim\widetilde{\nu}}\left(\frac{\widetilde{\nu}(X)}{\widehat{\nu}(X)} \geqslant C_{\mathsf{cov}}\right) \leqslant \delta_{\mathsf{cov}}. \tag{55}$$

In particular, this relaxation only requires a sampling algorithm to achieve approximate coverage with respect to the reweighted mixture of marginals of *any distribution $\widetilde{\mu}$ which is close to μ*. Note that the operation of taking the mixture of marginals and tilting is highly "non-Lipschitz", so even though μ and $\widetilde{\mu}$ are close to one another, ν and $\widetilde{\nu}$ may be far apart.

In order to realize the above style of guarantee, we will construct $\widetilde{\mu}$ by considering restrictions/conditionals of μ unto some high probability set. In particular consider some events $\mathcal{K} \subseteq \boldsymbol{X}^L \times [0,1]^L$ and define,

$$\mu_{\mathcal{K}}(\cdot) = \Pr(\underline{X} = \cdot \mid (\underline{X}, \boldsymbol{\eta}) \in \mathcal{K}) \text{ and } \overline{\mu_{\mathcal{K}}}(\cdot) = \frac{1}{L}\sum_{i=0}^{L-1} \Pr(X_i = \cdot \mid (\underline{X}, \boldsymbol{\eta}) \in \mathcal{K})$$

where $\boldsymbol{\eta} \sim \mathrm{Unif}([0,1])^{\otimes L}$ is a sequence of i.i.d. uniform random variables. If $\mathcal{K} = \boldsymbol{X}^L \times [0,1]^L$ is total, $\mu_{\mathcal{K}} = \mu$. The first result we establish in this section shows that when $\mathcal{K}$ is chosen carefully, the guarantee in Eq. (55) can be achieved with $\widetilde{\nu} \leftarrow \nu_{\mathcal{K}} \propto \overline{\mu_{\mathcal{K}}}(\cdot)w(\cdot)$, with $\delta_{\mathsf{cov}} = 0$ and $C_{\mathsf{cov}} = \mathcal{O}(c^{-1}\log(L))$, for any $c > 0$ bounding the TV distance between μ and $\mu_{\mathcal{K}}$.

Lemma I.3 (Approximate sampling from $\nu_{\mathcal{K}}$)**.** *Fix $L \geqslant 2$, a distribution μ over $\boldsymbol{X}^L$, and a non-negative weight function $w : \boldsymbol{X} \to [0,1]$ with $p_{\mu,w} > 0$. Let $\widehat{\nu}$ be the distribution over $\boldsymbol{X}$ induced by the procedure:*

Procedure I.1

Draw $\underline{X} \sim \mu$ until one is accepted by the following filter: Sample random variables $\boldsymbol{\eta} = (\eta_i)_{i=0}^{L-1} \sim \mathrm{Unif}([0,1])^{\otimes L}$, and define $\mathcal{I} = \left\{i : \eta_i \leqslant \frac{w(X_i)}{\|w\|_\infty},\ 0 \leqslant i \leqslant L-1\right\}$; accept $\underline{X}$ if $|\mathcal{I}| \geqslant 1$ and return X_I for $I \sim \mathrm{Unif}(\mathcal{I})$.

Fix any $c \in (0,1)$. There exists $\mathcal{K} \subseteq \boldsymbol{X}^L \times [0,1]^L$ such that, $\Pr((\underline{X}, \boldsymbol{\eta}) \notin \mathcal{K}) \leqslant c$. Furthermore,

$$\left\| \frac{\nu_{\mathcal{K}}}{\widehat{\nu}} \right\|_\infty \leqslant C_{\mathsf{cov}}, \text{ where } C_{\mathsf{cov}} = \mathcal{O}\left(c^{-1} \cdot \log(L)\right) \tag{56}$$

Finally, the probability that $\underline{X} \sim \mu$ is accepted within Procedure I.1 to generate a sample from $\widehat{\nu}$ is,

$$p_{\mu,w} = \mathbb{E}\left[1 - \prod_{i=0}^{L-1}\left(1 - \frac{w(X_i)}{\|w\|_\infty}\right)\right] \tag{57}$$

Proof. This result is a corollary of Lemma I.5 setting $\mathcal{U} = \boldsymbol{X}^L$. □

Remark I.4. *Note that applying* $\texttt{InvSampling}(D_{\boldsymbol{x}} \| \Pi_j, k)$ *where* $D_{\boldsymbol{x}} \leftarrow \texttt{Split}(\boldsymbol{x}|L)$ *(Algorithms 3 and 4) for $\boldsymbol{x} = (s_0, \boldsymbol{w}_{1:T}) \sim \rho$ is identical to applying Procedure I.1 , with the supplied envelope $M = \alpha^{j,k}_{\max}$, to $\underline{X}$ constructed from $\boldsymbol{x}$ by splitting and labeling the intermediate states using $f^\star$: i.e., $\underline{X} = (X_i)_{i=0}^{L-1}$ where $X_i = (s_{i\tau}, \boldsymbol{w}_{i\tau+1:(i+1)\tau})$ where $s_{i\tau} = f^\star_{i\tau}(\boldsymbol{x})$.*

Note that Procedure I.1 achieves the guarantee in Lemma I.3 using just a single draw from μ if $p_{\mu,w} = 1$. Even in this special case, any sampling algorithm which is forced to achieve approximate coverage Eq. (54) with respect to μ (i.e., Eq. (55) with $c = 0$) cannot in general match the best guarantee achievable if μ is allowed to be slightly perturbed to $\widetilde{\mu}$. We will discuss this further in Section I.2.2.

Lemma I.3 itself is a special case of Lemma I.5 below, which extends this result to the setting where the learner aims to approximately sample from the reweighted mixture of marginals of μ.

Lemma I.5 (Approximate sampling from $\nu_{\mathcal{K}}$)**.** *Fix $L \geqslant 2$, a distribution μ over $\boldsymbol{X}^L$, and a non-negative weight function $w : \boldsymbol{X} \to [0,1]$ with $p_{\mu,w} > 0$. Let $\widehat{\nu}$ be the distribution over $\boldsymbol{X}$ induced by Procedure I.1 . Consider an event $\mathcal{U} \subseteq \boldsymbol{X}^L$ with $\Pr(\mathcal{U}, |\mathcal{I}| \geqslant 1) > 0$, and fix $c \in (0,1)$. There exists $\mathcal{K} \subseteq \boldsymbol{X}^L \times [0,1]^L$ such that,*

$$(\underline{X}, \boldsymbol{\eta}) \in \mathcal{K} \implies \underline{X} \in \mathcal{U} \quad \text{and} \quad \Pr((\underline{X}, \boldsymbol{\eta}) \notin \mathcal{K} \mid \underline{X} \in \mathcal{U}) \leqslant c.$$

Furthermore,

$$\left\| \frac{\nu_{\mathcal{K}}}{\widehat{\nu}} \right\|_\infty \leqslant C_{\mathsf{cov}}, \text{ where } C_{\mathsf{cov}} = \mathcal{O}\left(\frac{c^{-1} \cdot \log(L)}{\Pr(\underline{X} \in \mathcal{U} \mid |\mathcal{I}| \geqslant 1)}\right) \tag{58}$$

Finally, the probability that $\underline{X} \sim \mu$ is accepted within Procedure I.1 to generate a sample from $\widehat{\nu}$ is $p_{\mu,w}$ (cf. Eq. (57)).

Proof. The tuple-based construction and density comparison are proved in Appendix J.4. The acceptance probability is the direct calculation in Eq. (57). □

The extension of this result to the case where there are multiple weight functions, is relatively straightforward and discussed further in Appendix J.5 where we also prove Lemma F.3. In particular Lemma F.3 considers the case we have a sequence of k (potentially random) weight functions $(w_j)_{j=0}^{k-1}$ and events $(\mathcal{U}_j)_{j=0}^{k-1}$, and constructs a sequence of events $(\mathcal{K}_j)_{j=0}^{k-1}$ which enable density ratio bounds akin to Eq. (58).

I.2.2 A SAMPLING LOWER BOUND FOR ACHIEVING APPROXIMATE COVERAGE

In order to further interpret the bound in Lemma I.3, it will be helpful to mention a worst case bound which pertains to the guarantee achievable in the absence of the conditioning event $\mathcal{K}$. We show that sampling from a distribution satisfying approximate coverage (cf. Eq. (54)) to the target distribution $\nu \propto \overline{\mu}(\cdot)w(\cdot)$, even with a modest $\delta_{\mathsf{cov}} = \frac{1}{4}$, either requires $C_{\mathsf{cov}} = \mathrm{poly}(L)$ or requires the sampling algorithm to draw $\mathrm{poly}(L)$ samples from μ, even when considering the ideal scenario where $p_{\mu,w} = 1$.

Lemma I.6 (Worst-case bound when $\mathcal{K}^c = \emptyset$)**.** *Let $\boldsymbol{X} = \{0, 1, \cdots, L+1\}$. There exists a class of distributions $\boldsymbol{\mu} = \{\mu_{p,z} : (p, z) \in [0,1] \times [L]\}$, for $L \geqslant 2$, parameterized by $p \in [0,1]$ supported on $\boldsymbol{X}^L$, as well as a fixed weight function $w : \boldsymbol{X} \to [0,1]$ such that:*

1. *$p_{\mu,w} = 1$ for all $\mu \in \boldsymbol{\mu}$.*
2. *For a sufficiently small absolute constant $c > 0$, consider any sampling algorithm $\mathtt{Alg}$ which draws at most $N = c\sqrt{L}$ samples drawn from the distribution $\mu \in \boldsymbol{\mu}$ to generate a sample from some distribution $\widehat{\nu} \in \Delta_{\{0,1,\cdots,L+1\}}$, and such that $\widehat{\nu}$ is well defined for every base distribution $\mu \in \boldsymbol{\mu}$. Then the distribution $\widehat{\nu}$ realized by $\mathtt{Alg}$ must incur,*
$$\max_{\mu \in \boldsymbol{\mu}} \mathrm{TV}\left(\widehat{\nu}, \nu\right) \geqslant \frac{1}{2}$$
where ν is the distribution $\propto \overline{\mu}(\cdot)w(\cdot)$.
3. *Furthermore, if $\mathtt{Alg}$ draws at most $N = c' L^{1/3}$ samples from $\mu \in \boldsymbol{\mu}$ for a sufficiently small constant $c' > 0$, then the distribution $\widehat{\nu}$ realized by $\mathtt{Alg}$ must incur,*
$$\max_{\mu \in \boldsymbol{\mu}} \Pr_{X \sim \nu}\left(\frac{\nu(X)}{\widehat{\nu}(X)} \geqslant c' L^{1/3}\right) \geqslant \frac{1}{2}$$

Proof. The proof of this result is discussed in Appendix J.6. □

I.3 $\mathcal{K}_j$ IS A HIGH PROBABILITY EVENT: PROOF OF LEMMA F.4

We first introduce an auxiliary lemma which will be helpful to prove Lemma F.4. The following result establishes that the $\mathcal{U}_j$ events occur simultaneously with moderately high probability.

Lemma I.7. *Across all iterations $0 \leqslant j \leqslant k-1$, we have $\Pr(\mathcal{U}_j \mid \mathcal{K}_{j-1}, \mathcal{H}_{j-1}) \geqslant 1 - \frac{1}{32k^2}$.*

Proof. The proof of this result is in Section I.3.1. □

Having established this result, we next show how to prove Lemma F.4. As a consequence of the recursive definition of $\mathcal{K}_j$ which is constructed from $\mathcal{U}_j$ and $\mathcal{K}_{j-1}$, and by Eq. (23),

$$\begin{aligned}
\Pr(\mathcal{K}_j \mid \mathcal{H}_{j-1}) &\geqslant (1-c) \cdot \Pr(\mathcal{U}_j \cap \mathcal{K}_{j-1} \mid \mathcal{H}_{j-1}) \\
&= (1-c) \cdot \Pr(\mathcal{U}_j \mid \mathcal{K}_{j-1}, \mathcal{H}_{j-1}) \cdot \Pr(\mathcal{K}_{j-1} \mid \mathcal{H}_{j-1}) \\
&\overset{(a)}{\geqslant} (1-c) \cdot \left(1 - \frac{1}{32k^2}\right) \cdot \Pr\big(\mathcal{K}_{j-1} \mid \mathcal{H}_{j-2}\big).
\end{aligned}$$

where in (a) we use the fact that $\mathcal{K}_{j-1}$ is a measurable function of $(\underline{X}, \boldsymbol{\eta})$ and $\mathcal{H}_{j-2}$, and Lemma I.7. Plugging in the value of $c = \frac{1}{32k}$, and solving the recursion over k steps gives us: for all $0 \leqslant j \leqslant k-1$,

$$\Pr(\mathcal{K}_j \mid \mathcal{H}_{j-1}) \geqslant \frac{15}{16}.$$

I.3.1 $\mathcal{U}_j$ IS A HIGH PROBABILITY EVENT: PROOF OF LEMMA I.7

By Lemma F.7,

$$\widetilde{p}_j \leqslant p_j^\star \implies \Pr\big(R_j > \mathfrak{r}_j \mid \mathcal{K}_{j-1}, \mathcal{H}_{j-1}\big) \leqslant \frac{1}{32k^2} \tag{59}$$

Simplifying further, this implies for all j,

$$\Pr(\mathcal{U}_j^c \mid \mathcal{K}_{j-1}, \mathcal{H}_{j-1}) = \Pr\big(\widetilde{p}_j \leqslant p_j^\star \text{ and } R_j > \mathfrak{r}_j \mid \mathcal{K}_{j-1}, \mathcal{H}_{j-1}\big)$$
$$= \mathbb{I}(\widetilde{p}_j \leqslant p_j^\star) \cdot \Pr\big(R_j > \mathfrak{r}_j \mid \mathcal{K}_{j-1}, \mathcal{H}_{j-1}\big) \leqslant \frac{1}{32k^2} \tag{60}$$

This completes the proof of the lemma.

I.4 Truthful Aborts: Proof of Lemma F.5

By definition of p_j, and by the structure of the split-and-inverted-sampling pipeline (Algorithms 3 and 4), the size of the dataset $|D_{\text{out}}^j|$ which the model $\widehat{\pi}^j$ is trained on can be expressed as the sum of $n' = |D_{\text{input}}^j| = |D_{\text{input}}|/k$ i.i.d. Bernoulli random variables, each with mean p_j. By an application of the multiplicative Chernoff bound,

$$\Pr\left(|D_{\text{out}}^j| < \frac{n'p_j}{2} \,\middle|\, \mathcal{H}_{j-1}\right) \leqslant \exp\left(-\frac{n'p_j}{8}\right)$$

In a non-aborted iteration, $p_j \geqslant \frac{15}{16} \cdot \widetilde{p}_j > \frac{15}{16} \cdot p_j^\star$ by Eq. (28). By the sufficiently large choice of $|D_{\text{input}}| = n'k$, we have that, (a) $n'p_j \geqslant 8\log(4k/\delta)$, and (b) $n'p_j \gtrsim k^{7/2}\log(L) \cdot (d\log\big(e\tau|S|k^{7/2}\log L\big) + \log(k/\delta))$. By its definition in Eq. (19), this implies,

$$\Pr\left(|D_{\text{out}}^j| < C_2 k^{7/2}\log(L) \cdot (d\log\Big(e\tau|S|k^{7/2}\log L\Big) + \log(k/\delta)) \,\middle|\, \mathcal{H}_{j-1}\right) \leqslant \frac{\delta}{2k}$$

Plugging in n' and $p_j^\star$ completes the proof.

I.5 Training Weak Models: Proof of Lemma F.6

By Lemma F.3, in order to prove an upper bound on the prediction error under λ_j, it suffices to establish learning guarantees under $\widehat{\lambda}_j$ and use a change-of-measure argument. Noting that `AutoLearn.warmup` instantiates the base learner as $\mathsf{NTP}_{\texttt{Full-CoT}}$ (Algorithm 8), we next provide a learner guarantee for this next-token prediction based rule. As a corollary of (Joshi et al., 2025, Theorem B.5), we have the following result.

Proposition I.8. *Given a dataset $D = \{\boldsymbol{x}^i\}_{i=1}^n$ of length τ instances drawn i.i.d. from $\widehat{\lambda}_j$, $\mathsf{NTP}_{\texttt{Full-CoT}}(\cdot)$ queries the $\texttt{iCoT}(\cdot)$ oracle τn times in total to return a model $\widehat{\pi}^{\mathsf{NTP}}$ such that, with probability $1 - \frac{\delta}{2k}$ for any constant $C_0 > 0$,*

$$\Pr_{X\sim\widehat{\lambda}_j}\big[\widehat{\pi}_\tau^{\mathsf{NTP}}(X) \neq \pi_\tau^\star(X)\big] \leqslant \frac{1}{C_0 k^{7/2}\log(L)},$$

as long as for a sufficiently large absolute constant $C_1 > 0$ depending on C_0,

$$n \geqslant C_1 k^{7/2}\log(L) \cdot (d\log\Big(e\tau|S|k^{7/2}\log L\Big) + \log(k/\delta)).$$

Proof. For n length-τ trajectories, the consistency loss depends on at most $n\tau$ next-state inputs. The Natarajan growth bound is at most $(Cn\tau|S|^2/d)^d$ when $n\tau \geqslant d$. The standard realizable double-sampling argument therefore gives terminal error at most η from $n \geqslant C\eta^{-1}[d\log(e\tau|S|/\eta) + \log(1/\zeta)]$ trajectories, with failure probability ζ. Substitute $\eta = (C_0 k^{7/2}\log L)^{-1}$ and $\zeta = \delta/(2k)$. The oracle returns a model consistent with every transition in the observed trajectories, and hence with each observed terminal state. □

I.5.1 Proof of Lemma F.6

Consider any non-aborted iteration j. Note that the model $\widehat{\pi}^j$ is learned via $\mathsf{NTP}_{\texttt{Full-CoT}}$ on the dataset D_{out}^j. By the bound on the density ratio in Eq. (27), and a change-of-measure argument, with probability at least $1 - \frac{\delta}{k}$,

$$\Pr_{X\sim\lambda_j}\big[\widehat{\pi}^j(X) \neq \pi_\tau^\star(X)\big] \leqslant \left\|\frac{\lambda_j}{\widehat{\lambda}_j}\right\|_\infty \cdot \Pr_{X\sim\widehat{\lambda}_j}\big[\widehat{\pi}_\tau^j(X) \neq \pi_\tau^\star(X)\big] \leqslant \frac{1}{4}.$$

The last inequality uses Proposition I.8 with sufficiently large C_0, and Lemma F.5.

I.6 Aborted Iterations Have Low Regression: Proof of Lemma F.7

Prior to proving Lemma F.7, Next, we move onto establishing the moderate probability upper bound on R_j via Markov's inequality. From its definition in Eq. (18),

$$\begin{aligned}
\widetilde{p}_j &= \mathbb{E}\left[1 - \prod_{i=0}^{L-1}\left(1 - \frac{\alpha^{j,k}_{\mathrm{rank}_j(X_i)}}{\alpha^{j,k}_{\max}}\right) \,\middle|\, \mathcal{H}_{j-1}, \mathcal{K}_{j-1}\right] \\
&\geqslant \mathbb{E}\left[1 - \exp\left(-\frac{R_j}{\alpha^{j,k}_{\max}}\right) \,\middle|\, \mathcal{H}_{j-1}, \mathcal{K}_{j-1}\right] \\
&\overset{(a)}{\geqslant} (1 - e^{-\theta}) \cdot \Pr\left(R_j \geqslant \theta \alpha^{j,k}_{\max} \mid \mathcal{H}_{j-1}, \mathcal{K}_{j-1}\right) \\
&\geqslant \frac{\theta}{2} \cdot \Pr\left(R_j \geqslant \theta \alpha^{j,k}_{\max} \mid \mathcal{H}_{j-1}, \mathcal{K}_{j-1}\right)
\end{aligned}$$

where (a) is by an application of Markov's inequality for $0 < \theta \leqslant 1$. The proof concludes by choosing $\theta = \frac{1}{16\sqrt{k}}$ and using the upper bound on $\widetilde{p}_j$ assumed in the statement of the lemma.

I.6.1 Explicit formula for p_j

Below we prove the explicit formula for p_j we described in Eq. (17). Let $\mathcal{I}_j$ denote the (random) set of indices in $\{0, \ldots, L-1\}$ accepted by `InvSampling`, so that $\mathcal{I}_j \neq \emptyset \iff \texttt{InvSampling}(D_{\boldsymbol{x}} \parallel \Pi_j, k) \neq \perp$. This implies that,

$$\begin{aligned}
p_j &= \Pr\big(\texttt{InvSampling}(D_{\boldsymbol{x}} \parallel \Pi_j, k) \neq \perp \;\big|\; \mathcal{H}_{j-1}\big) \qquad (61) \\
&= \Pr(|\mathcal{I}_j| \geqslant 1 \mid \mathcal{H}_{j-1}) \\
&= \mathbb{E}\left[1 - \Pr_{(\eta_i)_{i=0}^{L-1} \sim \mathrm{Unif}([0,1])^{\otimes L}}\left(\forall i \in \{0, \cdots, L-1\},\ \eta_i \geqslant \frac{w_j(X_i)}{\alpha^{j,k}_{\max}}\right) \,\middle|\, \mathcal{H}_{j-1}\right] \\
&= \mathbb{E}\left[1 - \prod_{i=0}^{L-1}\left(1 - \frac{w_j(X_i)}{\alpha^{j,k}_{\max}}\right) \,\middle|\, \mathcal{H}_{j-1}\right]. \qquad (62)
\end{aligned}$$

J Proofs of Auxiliary Results

J.1 Lower Bounds Against Full-CoT Learners: Proof of Proposition C.5

Consider any family of models $\mathcal{G} : S \to S$ with Natarajan dimension d. Construct the semiautomaton class Π on the state space S and $\Sigma = \{-1, +1\}$. Each $\pi \in \Pi$ corresponds to a $g \in \mathcal{G}$ which we will indicate through the mapping π_g and the transition of π_g is defined as,

$$\forall s \in S, \pi_g((s, -1)) = g(s)$$
$$\forall s \in S, \pi_g((s, +1)) = s$$

In other words, π transitions according to $g(\cdot)$ when the input letter is -1, and behaves like the identity mapping when the input letter is $+1$. Suppose the distribution over instances, ρ, is constructed by taking the direct product of any distribution over states $\sigma_0^\star$, and the (deterministic) distribution over symbols $\delta_{-1} \times (\delta_{+1})^{T-1}$. Under this structure, note that almost surely for $s_0 \sim \sigma_0^\star$, $\pi_{g,T}(s_0) = g(s_0)$. Consider any PAC learning algorithm $\texttt{Alg}(\cdot)$ which consumes a dataset of CoTs, $D = \{((s_0, \boldsymbol{w}) \mapsto \pi_{g^\star,1:T}((s_0, \boldsymbol{w})))\}$ for some unknown $\pi_{g^\star} \in \Pi$ and $(s_0, \boldsymbol{w}) \sim \rho$, to return a model $\widehat{\pi}$ satisfying with probability at least $\frac{1}{2}$,

$$\Pr_{(s_0,\boldsymbol{w})\sim\rho}\big(\widehat{\pi}((s_0, \boldsymbol{w})) \neq \pi_{g^\star,T}((s_0, \boldsymbol{w}))\big) \leqslant \frac{1}{4}.$$

This can be rewritten as,

$$\Pr_{s_0\sim\sigma_0^\star}\big(\widehat{\pi}((s_0, \{-1, +1 \cdots, +1\})) \neq g^\star(s_0)\big) \leqslant \frac{1}{4}.$$

This implies that the induced model $\widehat{\pi}\big((\cdot, \{-1, +1, \cdots, +1\})\big)$ is a PAC learner for $\mathcal{G}$ under any distribution over states $\sigma_0^\star$. Furthermore, note that any fully labeled state sequence $(s_0, \boldsymbol{w}) \mapsto \pi_{g^\star,1:T}((s_0, \boldsymbol{w}))$ can be constructed from the singular labeling $s_0 \mapsto g^\star(s_0)$ and vice versa.

Choose $\mathcal{G}$ to realize all binary labelings of $d \geqslant 64$ designated states (using two fixed labels in S), and fix its values elsewhere. Let $\sigma_0^\star$ be uniform on those states and draw the d target bits independently and uniformly. If $N_{\text{label}} \leqslant d/8$ instances are fully labeled, at least $7d/8$ bits remain independent and unknown, even under adaptive selection of which sampled instances to label. Conditional on the observations and the learner's random coins, errors on these states stochastically dominate $\text{Bin}(\lceil 7d/8 \rceil, 1/2)$. A Chernoff bound shows that the probability of at most $d/4$ errors is less than $1/2$. Averaging over targets therefore yields a target for which the required guarantee fails. Hence $N_{\text{label}} = \Omega(d)$ is necessary. All variability in Π lies on its -1 input slice, and restriction to that slice recovers $\mathcal{G}$, so $\text{Ndim}(\Pi) = \text{Ndim}(\mathcal{G}) = d$. Finally, $n_{\texttt{query}} = TN_{\text{label}}$ and $n_{\texttt{sample}} \geqslant N_{\text{label}}$, proving both lower bounds.

J.2 Bounding Prediction Error by Potential Φ_k: Proof of Lemma I.1

Recall that the output of $\texttt{AutoLearn.warmup}$ is the model $\widehat{\pi}^{\circ L}$ where $\widehat{\pi} = \texttt{Plu}\big(\{\widehat{\pi}_\tau^j : j \in \{0, \cdots, k-1\}\}\big)$. By definition of $\beta_r^{k,k}$, we have that,

$$\begin{aligned}
&\Pr_{\boldsymbol{x}\sim\rho}\big[\widehat{\pi}^{\circ L}(\boldsymbol{x}) \neq \pi_T^\star(\boldsymbol{x}) \mid \mathcal{H}_{k-1}\big] \\
&\overset{(a)}{=} \Pr\big(\widehat{\pi}^{\circ L}(\texttt{concat}(\underline{X})) \neq \pi_T^\star(\texttt{concat}(\underline{X})) \mid \mathcal{H}_{k-1}\big) \\
&\leqslant \Pr\big(\exists i \in \{0, \cdots, L-1\} : \widehat{\pi}(X_i) \neq \pi_\tau^\star(X_i) \text{ and } (\underline{X}, \boldsymbol{\eta}) \in \mathcal{K}_{k-1} \mid \mathcal{H}_{k-1}\big) \\
&\qquad + \Pr((\underline{X}, \boldsymbol{\eta}) \notin \mathcal{K}_{k-1} \mid \mathcal{H}_{k-1}) \\
&\leqslant \sum_{i=0}^{L-1} \Pr\big(\widehat{\pi}(X_i) \neq \pi_\tau^\star(X_i) \text{ and } (\underline{X}, \boldsymbol{\eta}) \in \mathcal{K}_{k-1} \mid \mathcal{H}_{k-1}\big) \\
&\qquad + \Pr((\underline{X}, \boldsymbol{\eta}) \notin \mathcal{K}_{k-1} \mid \mathcal{H}_{k-1}) \\
&\overset{(b)}{\leqslant} \sum_{i=0}^{L-1} \sum_{r=0}^{\lfloor k/2 \rfloor} \Pr\big(\text{rank}_k(X_i) = r \text{ and } (\underline{X}, \boldsymbol{\eta}) \in \mathcal{K}_{k-1} \mid \mathcal{H}_{k-1}\big)
\end{aligned}$$

$$
\begin{aligned}
&\quad + \Pr((\underline{X}, \boldsymbol{\eta}) \notin \mathcal{K}_{k-1} \mid \mathcal{H}_{k-1}) \\
&= \sum_{i=0}^{L-1} \mathbb{E}\big[\beta^{k,k}_{\mathrm{rank}_k(X_i)} \cdot \mathbb{I}((\underline{X}, \boldsymbol{\eta}) \in \mathcal{K}_{k-1}) \mid \mathcal{H}_{k-1}\big] \\
&\quad + \Pr((\underline{X}, \boldsymbol{\eta}) \notin \mathcal{K}_{k-1} \mid \mathcal{H}_{k-1}) \\
&= \Phi_k + \Pr\big((\underline{X}, \boldsymbol{\eta}) \notin \mathcal{K}_{k-1} \mid \mathcal{H}_{k-1}\big)
\end{aligned}
$$

where in (a), $\mathsf{concat}(X)$ is defined as taking $X = (X_0, \cdots, X_{L-1}) \in (S \times \Sigma^\tau)^L$ where $X_i = (s^i, \boldsymbol{w}^i)$ and mapping it to $(s^0, \boldsymbol{w}^0, \cdots, \boldsymbol{w}^{L-1})$, which is in $S \times \Sigma^T$, and (b) uses the fact that if $\mathrm{rank}_k(X_i) > \lfloor k/2 \rfloor$, then the plurality vote satisfies $\widehat{\pi}(X_i) = \pi^\star_\tau(X_i)$.

J.3 Recursion for Φ_j: Proof of Lemma I.2

Recall by definition,

$$
\Phi_{j+1} = \sum_{i=0}^{L-1} \sum_{r=0}^{j+1} \beta^{j+1,k}_r \cdot \Pr\big(\mathrm{rank}_{j+1}(X_i) = r \text{ and } (\underline{X}, \boldsymbol{\eta}) \in \mathcal{K}_j \mid \mathcal{H}_{k-1}\big)
$$

Note that $\mathrm{rank}_{j+1}(X_i) = r$ is only possible if $\mathrm{rank}_j(X_i) = r$ or $\mathrm{rank}_j(X_i) = r-1$. Likewise, under the condition $\mathrm{rank}_j(X_i) = r$, then $\mathrm{rank}_{j+1}(X_i) \neq r \implies \mathrm{rank}_{j+1}(X_i) = r+1$. With this, we decompose the above expression as,

$$
\begin{aligned}
\Phi_{j+1} &= \sum_{i=0}^{L-1} \sum_{r=0}^{j} \beta^{j+1,k}_r \cdot \Pr\big((\underline{X}, \boldsymbol{\eta}) \in \mathcal{K}_j \text{ and } \mathrm{rank}_j(X_i) = r \mid \mathcal{H}_{k-1}\big) \\
&\quad - \sum_{i=0}^{L-1} \sum_{r=0}^{j} \beta^{j+1,k}_r \\
&\quad \cdot \Pr\big((\underline{X}, \boldsymbol{\eta}) \in \mathcal{K}_j \text{ and } \mathrm{rank}_{j+1}(X_i) = r+1 \text{ and } \mathrm{rank}_j(X_i) = r \mid \mathcal{H}_{k-1}\big) \\
&\quad + \sum_{i=0}^{L-1} \sum_{r=1}^{j+1} \beta^{j+1,k}_r \\
&\quad \cdot \Pr\big((\underline{X}, \boldsymbol{\eta}) \in \mathcal{K}_j \text{ and } \mathrm{rank}_{j+1}(X_i) = r \text{ and } \mathrm{rank}_j(X_i) = r-1 \mid \mathcal{H}_{k-1}\big) \\
&= \sum_{i=0}^{L-1} \sum_{r=0}^{j} \beta^{j+1,k}_r \cdot \Pr\big((\underline{X}, \boldsymbol{\eta}) \in \mathcal{K}_j \text{ and } \mathrm{rank}_j(X_i) = r \mid \mathcal{H}_{k-1}\big) \\
&\quad - \sum_{i=0}^{L-1} \sum_{r=0}^{j} \big(\underbrace{\beta^{j+1,k}_r - \beta^{j+1,k}_{r+1}}_{\alpha^{j,k}_r}\big) \\
&\quad \cdot \Pr\big((\underline{X}, \boldsymbol{\eta}) \in \mathcal{K}_j \text{ and } \underbrace{\mathrm{rank}_{j+1}(X_i) = r+1 \text{ and } \mathrm{rank}_j(X_i) = r}_{\equiv \left\{\mathrm{rank}_j(X_i) = r \text{ and } \widehat{\pi}^j_\tau(X_i) = \pi^\star_\tau(X_i)\right\}} \mid \mathcal{H}_{k-1}\big)
\end{aligned} \tag{63}
$$

All test variables below are fresh and independent of training. We fix the full training history, so the learned models are fixed; the law of the test tuple conditioned on the past-measurable event $\mathcal{K}_j$ is unchanged by revealing later training data. Let $\sigma_{\mathcal{K}_j}$ denote the distribution $\Pr(\underline{X} = \cdot \mid (\underline{X}, \boldsymbol{\eta}) \in \mathcal{K}_j, \mathcal{H}_{j-1})$. Then, the last expression in Eq. (63) can be further decomposed as,

$$
\begin{aligned}
&\sum_{i=0}^{L-1} \sum_{r=0}^{j} \alpha^{j,k}_r \cdot \Pr\big((\underline{X}, \boldsymbol{\eta}) \in \mathcal{K}_j \text{ and } \mathrm{rank}_j(X_i) = r \text{ and } \widehat{\pi}^j_\tau(X_i) = \pi^\star_\tau(X_i) \mid \mathcal{H}_{k-1}\big) \\
&= \sum_{i=0}^{L-1} \mathbb{E}_{\underline{X} \sim \sigma_{\mathcal{K}_j}}\big[\alpha^{j,k}_{\mathrm{rank}_j(X_i)} \cdot \mathbb{I}\big(\widehat{\pi}^j_\tau(X_i) = \pi^\star_\tau(X_i)\big) \mid \mathcal{H}_{k-1}\big] \cdot \Pr\big((\underline{X}, \boldsymbol{\eta}) \in \mathcal{K}_j \mid \mathcal{H}_{k-1}\big) \\
&\overset{(a)}{=} L \cdot \mathbb{E}_{X \sim \overline{\sigma_{\mathcal{K}_j}}}\big[w_j(X) \cdot \mathbb{I}\big(\widehat{\pi}^j_\tau(X) = \pi^\star_\tau(X)\big) \mid \mathcal{H}_{k-1}\big] \cdot \Pr\big((\underline{X}, \boldsymbol{\eta}) \in \mathcal{K}_j \mid \mathcal{H}_{k-1}\big)
\end{aligned}
$$

$$\overset{(b)}{=} L \cdot \mathbb{E}_{X\sim\overline{\sigma\mathcal{K}_j}}\big[w_j(X)\big] \cdot \Pr_{X\sim\lambda_j}\big(\widehat{\pi}^j_\tau(X) = \pi^\star_\tau(X)\big) \cdot \Pr\big((\underline{X},\boldsymbol{\eta}) \in \mathcal{K}_j \mid \mathcal{H}_{k-1}\big)$$

$$\overset{(c)}{=} L \cdot \mathbb{E}_{X\sim\overline{\sigma\mathcal{K}_j}}\big[\alpha^{j,k}_{\mathrm{rank}_j(X)}\big] \cdot (1-\mathtt{err}_j) \cdot \Pr\big((\underline{X},\boldsymbol{\eta}) \in \mathcal{K}_j \mid \mathcal{H}_{k-1}\big)$$

$$= (1-\mathtt{err}_j) \cdot \sum_{i=0}^{L-1}\sum_{r=0}^{j} \mathbb{E}\big[\alpha^{j,k}_r \cdot \mathbb{I}((\underline{X},\boldsymbol{\eta}) \in \mathcal{K}_j \text{ and } \mathrm{rank}_j(X_i) = r) \mid \mathcal{H}_{k-1}\big]$$

where in (a), recall that $w_j(X) = \alpha^{j,k}_{\mathrm{rank}_j(X)}$ which is measurable with respect to $\mathcal{H}_{j-1}$. In (b), $\lambda_j(\cdot)$ is the distribution proportional to $\overline{\sigma\mathcal{K}_j}(\cdot)w_j(\cdot)$. In equation (c), we use the definition of $\mathtt{err}_j$ from Lemma F.6. We may combine this equation back with Eq. (63), noting that $\alpha^{j,k}_r = \beta^{j+1,k}_r - \beta^{j+1,k}_{r+1}$, $\mathtt{err}_\star = \frac{1}{4}$ and $\mathtt{err}_\star \cdot \beta^{j+1,k}_r + (1-\mathtt{err}_\star)\cdot\beta^{j+1,k}_{r+1} = \beta^{j,k}_r$,

$$\begin{aligned}
\Phi_{j+1} &= \sum_{i=0}^{L-1}\sum_{r=0}^{j}\Big(\mathtt{err}_\star \cdot \beta^{j+1,k}_r + (1-\mathtt{err}_\star)\cdot\beta^{j+1,k}_{r+1}\Big) \\
&\qquad \cdot \Pr\big((\underline{X},\boldsymbol{\eta}) \in \mathcal{K}_j \text{ and } \mathrm{rank}_j(X_i) = r \mid \mathcal{H}_{k-1}\big) \\
&\qquad + (\mathtt{err}_j - \mathtt{err}_\star)\sum_{i=0}^{L-1}\sum_{r=0}^{j}\alpha^{j,k}_r \cdot \Pr\big((\underline{X},\boldsymbol{\eta}) \in \mathcal{K}_j \text{ and } \mathrm{rank}_j(X_i) = r \mid \mathcal{H}_{k-1}\big) \\
&= \sum_{i=0}^{L-1}\sum_{r=0}^{j}\beta^{j,k}_r \cdot \Pr\big((\underline{X},\boldsymbol{\eta}) \in \mathcal{K}_j \text{ and } \mathrm{rank}_j(X_i) = r \mid \mathcal{H}_{k-1}\big) \\
&\qquad + (\mathtt{err}_j - \mathtt{err}_\star)\sum_{i=0}^{L-1}\mathbb{E}\big[\alpha^{j,k}_{\mathrm{rank}_j(X_i)} \cdot \mathbb{I}((\underline{X},\boldsymbol{\eta}) \in \mathcal{K}_j) \mid \mathcal{H}_{k-1}\big] \\
&\overset{(a)}{\leqslant} \sum_{i=0}^{L-1}\sum_{r=0}^{j}\beta^{j,k}_r \cdot \Pr\big((\underline{X},\boldsymbol{\eta}) \in \mathcal{K}_{j-1} \text{ and } \mathrm{rank}_j(X_i) = r \mid \mathcal{H}_{k-1}\big) \\
&\qquad + (\mathtt{err}_j - \mathtt{err}_\star)\sum_{i=0}^{L-1}\mathbb{E}\big[\alpha^{j,k}_{\mathrm{rank}_j(X_i)} \cdot \mathbb{I}((\underline{X},\boldsymbol{\eta}) \in \mathcal{K}_j) \mid \mathcal{H}_{k-1}\big] \\
&= \Phi_j + (\mathtt{err}_j - \mathtt{err}_\star)\cdot\mathbb{E}\big[R_j \cdot \mathbb{I}((\underline{X},\boldsymbol{\eta}) \in \mathcal{K}_j) \mid \mathcal{H}_{k-1}\big].
\end{aligned}$$

where (a) uses the fact that $\mathcal{K}_j \subseteq \mathcal{K}_{j-1} \subseteq \cdots \subseteq \mathcal{K}_0$.

J.4 Density Ratio Bound between $\nu_\mathcal{K}$ and $\widehat{\nu}$: Proof of Lemma I.5

The proof works with any supplied envelope $M \geqslant \|w\|_\infty > 0$, using acceptance probabilities $w(X_i)/M$ (in the BBM application, $M = \alpha^{j,k}_{\max}$). Write $r_i = w(X_i)/M$, $W = \sum_i r_i$, and $A = \{|\mathcal{I}| \geqslant 1\}$. Let $p = \Pr(A)$, $t = \Pr(A \cap \mathcal{U}) > 0$, and $q = t/p$. Conditional on $\underline{X}$, the acceptance indicators are independent Bernoulli variables with means r_i. For any measurable $S \subseteq \boldsymbol{X}$, putting $g_S = \sum_i r_i \mathbb{I}(X_i \in S)$, Jensen's inequality gives

$$\begin{aligned}
\widehat{\nu}(S) &= \frac{1}{p}\mathbb{E}\left[\sum_i r_i \mathbb{I}(X_i \in S)\mathbb{E}\left[\frac{1}{1+\sum_{\ell\neq i}\mathbb{I}(\ell\in\mathcal{I})}\,\middle|\,\underline{X}\right]\right] \\
&\geqslant \frac{1}{p}\mathbb{E}\left[\frac{g_S}{1+W}\right]. \qquad (64)
\end{aligned}$$

In the inner expectation, coordinate i has already been accepted; its coin is independent of the other coins.

Let $m = \max\{1, \lceil \log_2 L \rceil\}$, $\kappa = 1 - e^{-1}$, and $b = c\kappa/(2m)$. Define $B_0 = \{W \leqslant 1\}$ and $B_\ell = \{2^{\ell-1} < W \leqslant 2^\ell\}$ for $1 \leqslant \ell \leqslant m$, and set

$$L_{\mathrm{heavy}} = \{\ell \in [m] : \Pr(A \cap \mathcal{U} \cap B_\ell) \geqslant bt\}, \qquad (65)$$

$$\mathcal{K} = \mathcal{U} \cap \left(B_0 \cup \bigcup_{\ell\in L_{\mathrm{heavy}}} B_\ell\right). \qquad (66)$$

We identify $\mathcal{K}$ with its cylinder extension when auxiliary coins are included. In particular, membership in $\mathcal{K}$ depends on $\underline{X}$, not on the realized acceptance coins. Since $a(\underline{X}) := \Pr(A \mid \underline{X}) = 1 - \prod_i (1 - r_i) \geqslant 1 - e^{-W} \geqslant \kappa$ on every positive bin,

$$\Pr(\mathcal{U} \setminus \mathcal{K}) \leqslant \kappa^{-1} \sum_{\ell \notin L_{\text{heavy}},\, \ell \geqslant 1} \Pr(A \cap \mathcal{U} \cap B_\ell) \leqslant mbt/\kappa \leqslant (c/2)\Pr(\mathcal{U}).$$

This proves the required mass bound.

The definition of the weighted conditional marginals gives $\nu_{\mathcal{K}}(S) = \mathbb{E}[\mathbb{I}(\mathcal{K}) g_S]/D$, where $D = \mathbb{E}[\mathbb{I}(\mathcal{K}) W]$. If a positive bin is heavy, let $J = \max L_{\text{heavy}}$. Then $W \leqslant 2^J$ on $\mathcal{K}$ and $D \geqslant 2^{J-1} \Pr(\mathcal{U} \cap B_J) \geqslant 2^{J-1} bt$. Consequently, by Eq. (64),

$$\nu_{\mathcal{K}}(S) \leqslant \frac{1 + 2^J}{D} \mathbb{E}\left[\frac{g_S}{1 + W}\right] \leqslant \frac{3p}{bt} \widehat{\nu}(S) = \frac{6m}{\kappa c q} \widehat{\nu}(S).$$

If no positive bin is heavy, $\mathcal{K} = \mathcal{U} \cap B_0$. Since $a \leqslant W$, $D \geqslant \Pr(A \cap \mathcal{U} \cap B_0) \geqslant (1 - mb)t \geqslant t/2$. Now $W \leqslant 1$ on $\mathcal{K}$, so $\nu_{\mathcal{K}}(S) \leqslant (4/q)\widehat{\nu}(S)$. Both cases prove Eq. (58), with $1 + \log L$ in place of $\log L$ when $L = 1$. The acceptance probability is Eq. (57).

J.5 Density Ratio Bound between $\nu_{\mathcal{K}_j}$ and $\widehat{\nu}_j$: Proof of Lemma F.3

Fix the history $\mathcal{H}_{j-1}$ preceding round j. The weight w_j and the event $\mathcal{U}_j \cap \mathcal{K}_{j-1}$ are then fixed. Apply the construction in Eqs. (65) and (66) with $\mathcal{U} = \mathcal{U}_j \cap \mathcal{K}_{j-1}$ and weight w_j, under the law of a fresh test tuple and any previous auxiliary coins. Current acceptance coins are independent conditional on this base random object. Define

$$\mathcal{K}_j = \texttt{ConstructEvent}(\mathcal{U}_j \cap \mathcal{K}_{j-1} \mid c, \sigma, \mathcal{H}_{j-1}), \qquad \mathcal{K}_{-1} = \boldsymbol{X}^L, \tag{67}$$

where `ConstructEvent` is precisely the tuple-based construction in Eq. (66). It need not be computed by the learning algorithm. This gives the nesting and conditional mass bound in Eq. (23). Applying Lemma I.5 conditionally gives

$$\left\| \frac{\nu_{\mathcal{K}_j}}{\widehat{\nu}_j} \right\|_\infty \leqslant \mathcal{O}\left(\frac{c^{-1}(1 + \log L)}{\Pr(\mathcal{U}_j \cap \mathcal{K}_{j-1} \mid |\mathcal{I}_j| \geqslant 1, \mathcal{H}_{j-1})} \right). \tag{68}$$

Here $\nu_{\mathcal{K}_j}$ is the original weighted mixture of conditional marginals, not the law of accepted coordinates conditioned on their coins. In the application $L \geqslant 2$, $1 + \log L$ is absorbed into $\mathcal{O}(\log L)$. If the denominator is zero, choose $\mathcal{K}_j = \mathcal{U}_j \cap \mathcal{K}_{j-1}$; its weighted mass is zero and the corresponding regression contribution vanishes. The tilted law may then be assigned arbitrarily. This proves Lemma F.3.

J.6 Sampling Lower Bound: Proof of Lemma I.6

Let $\mathbf{1}$ be the all-ones vector and e_i the coordinate vectors in $\mathbb{R}^L$. For $p \in [0, 1]$ and $z \in [L]$, define

$$\mu_{p,z}(z\mathbf{1}) = p, \qquad \mu_{p,z}((L + 1)e_i) = (1 - p)/L, \quad 0 \leqslant i < L.$$

Take $w(x) = \mathbb{I}(x \in [L+1])$. Every tuple has an accepted coordinate, so $p_{\mu,w} = 1$, and the weighted marginal satisfies

$$\nu_{p,z}(z) = 1 - \nu_{p,z}(L + 1) = \frac{Lp}{1 + (L - 1)p}.$$

The learner knows w and has sample access to the unknown $\mu_{p,z}$. Pad any procedure using at most N draws with unused draws to obtain a fixed-size sample. Conditional on none of these draws being $z\mathbf{1}$, the sample law is independent of z. Consequently some $z \in [L]$ receives conditional output probability at most $1/L$. A union bound on observing a rare tuple gives

$$\widehat{\nu}(z) \leqslant 1/L + Np. \tag{69}$$

For the TV claim, take $p = 4/L$ and $L \geqslant 10$. Then $\nu_{p,z}(z) \geqslant 4/5$, so for $N \leqslant L/20$,

$$\mathrm{TV}\,(\nu_{p,z}, \widehat{\nu}) \geqslant 4/5 - (1/L + 1/5) \geqslant 1/2.$$

In particular this implies the stated $N \leqslant c\sqrt{L}$ claim for a sufficiently small absolute c. For $2 \leqslant L < 10$, choose c so that this condition allows only $N = 0$; taking $p = 1$ and the least likely output z gives TV at least $1 - 1/L \geqslant 1/2$.

For the density-ratio claim take $p = 1/L$. The same argument gives some z with

$$\nu_{p,z}(z) = \frac{L}{2L-1} \geqslant \frac{1}{2}, \qquad \widehat{\nu}(z) \leqslant \frac{N+1}{L}, \qquad \frac{\nu_{p,z}(z)}{\widehat{\nu}(z)} \geqslant \frac{L}{2(N+1)}.$$

If $N \leqslant c' L^{1/3}$, a sufficiently small absolute $c' > 0$ makes the last expression at least $c' L^{1/3}$ for all $L \geqslant 2$. Thus the density-ratio event has $\nu_{p,z}$-probability at least $1/2$, proving the remaining claim. The TV argument in fact proves a stronger linear sample lower bound for this family.

J.7 Improved T Dependency when Online Learning is Possible: Proof of Theorem B.4

If the terminal prediction is wrong, the binary search maintains the invariant that the state at $T_{\min} - 1$ is correct and the state at $T_{\max}$ is incorrect. When both midpoint states are correct, the left endpoint moves right; when the predecessor is incorrect, the right endpoint moves left. Otherwise the algorithm finds an adjacent correct-to-incorrect transition. Thus a wrong terminal prediction always produces a genuine online mistake. For each instance $\boldsymbol{x} = (s_0, \boldsymbol{w}_{1:T})$ in the dataset, $\texttt{Alg}_{\text{on}}$ predicts states $s_1, \cdots, s_T$. The binary search procedure described in Line 7 to 16 of Algorithm 9 satisfies the property that if the while loop is broken out of, the learner must have found a time t such that $s_t \neq s_t^\star$, but $s_{t-1} = s_{t-1}^\star$. If the while loop is never broken out of, then by the invariant above, the prediction of $\texttt{Alg}_{\text{on}}$ (which is a function of the prior instances and label observations) satisfies $s_T = s_T^\star$. On the other hand, each instance where the while loop is broken out of gives an (s, w) tuple such that the prediction s_+ of $\texttt{Alg}_{\text{on}}$ satisfies $\pi^\star(s, w) \neq s_+$ but where s itself was predicted correctly. This is a "true" mistake, where the prediction was not incorrect because the prior state itself was predicted incorrectly. Since $\texttt{Alg}_{\text{on}}$ makes at most $M^\star(\texttt{Alg}_{\text{on}})$ (true) mistakes, we have that,

$$\sum_{m=1}^{n} \mathbb{I}(y_T^m \neq \pi_T^\star(\boldsymbol{x}^m)) \leqslant M^\star(\texttt{Alg}_{\text{on}}).$$

where $y_T^m = \widehat{\pi}_T^m(\boldsymbol{x}^m)$. Taking an expectation on both sides and noting the definition of $\widehat{\pi}$ returned by Algorithm 9, by an online-to-batch argument we get,

$$\mathbb{E}_D \mathbb{E}_{\boldsymbol{x} \sim \rho, y_T \sim \widehat{\pi}(\cdot|\boldsymbol{x})} \left[\mathbb{I}\big(y_T \neq \pi_T^\star(\boldsymbol{x})\big) \right] = \frac{1}{n} \sum_{m=1}^{n} \mathbb{E}\left[\mathbb{I}(\widehat{\pi}_T^m(\boldsymbol{x}^m) \neq \pi_T^\star(\boldsymbol{x}^m)) \right] \leqslant \frac{M^\star(\texttt{Alg}_{\text{on}})}{n}.$$

This shows that $n = \mathcal{O}(M^\star(\texttt{Alg}_{\text{on}})/\varepsilon)$ samples suffices to get expected error ε. For each of the n instances in the input dataset, $\texttt{iCoT}(\cdot)$ is queried at at most $2\lceil \log_2(T+1) \rceil$ timepoints. This implies the relation between the query complexity and sample complexity.

Lemma J.1. *If no $t \in [T]$ exists such that $s_t \neq s_t^\star$, but $s_{t-1} = s_{t-1}^\star$, then $s_T = s_T^\star$.*

Proof. Start from $t = 1$. Since $s_0 = s_0^\star$ by definition, the condition in the statement of the lemma implies that $s_1 = s_1^\star$. Repeating this argument inductively, this implies that for all $t \leqslant T$, $s_t = s_t^\star$, which proves the claim. □

Lemma J.2. *For a class of next-state/token predictors Π from $S \times \Sigma \to S$, $\mathrm{Ldim}(\Pi) \leqslant O(\mathrm{Ndim}(\Pi) \log(e|S||\Sigma|))$ for finite S, Σ.*

Proof. By a counting argument $\mathrm{Ldim}(\Pi) \leqslant \log_2 |\Pi|$, since a complete mistake tree of depth m requires 2^m distinct hypotheses, one for each root-to-leaf path. On the other hand, viewing Π as a multiclass prediction class from $X = S \times \Sigma$ to $Y = S$, the multiclass Sauer–Shelah lemma for Natarajan dimension implies (Natarajan, 1989; Daniely et al., 2015)

$$|\Pi| \leqslant \sum_{i=0}^{d} \binom{|X|}{i} |Y|^{2i} \leqslant \left(\frac{e|X||Y|^2}{d} \right)^d.$$

Simplifying completes the proof. □

Algorithm 9 $\texttt{AutoLearn}_{\text{on}}(D_{\text{input}} \| T)$

Learning semiautomata via curriculum: reduction to online learning

Input: Class of semiautomaton transitions Π over state space S and alphabet Σ,
Number of steps of semiautomaton simulation, T,
Dataset of instances $D_{\text{input}} = \{\boldsymbol{x}^i = (s_0^i, \boldsymbol{w}^i)\}_{i=1}^n$.
Online learning algorithm $\texttt{Alg}_{\text{on}}$ for Π (Definition B.2)
Instantiate: Let $\texttt{Alg}_{\text{on}}$'s initial model be $\widehat{\pi}^1 : S \times \Sigma \to S$ and let $m \leftarrow 1$.
for instance $\boldsymbol{x}^m = (s_0, \boldsymbol{w}_{1:T}) \in D_{\text{input}}$ **do**
  for $t = 1, \cdots, T$ **do**
    Let $s_t \leftarrow \widehat{\pi}^m(s_{t-1}, w_t)$.
  Instantiate $T_{\min} \leftarrow 1, T_{\max} \leftarrow T$.
  while $T_{\max} \geqslant T_{\min}$ **do**
    $t \leftarrow T_{\min} + \left\lfloor \frac{T_{\max} - T_{\min}}{2} \right\rfloor$
    Query $s_t^\star \leftarrow \texttt{iCoT}((s_0, \boldsymbol{w}_{1:t}))$ and $s_{t-1}^\star \leftarrow \texttt{iCoT}((s_0, \boldsymbol{w}_{1:t-1}))$ (use s_0 when $t = 1$).
    if $s_t \neq s_t^\star$ but $s_{t-1} = s_{t-1}^\star$ **then**
      Submit input (s_{t-1}, w_t) to $\texttt{Alg}_{\text{on}}$, observe its prediction (s_t) and reveal the correct label $s_t^\star$ ▶ $\texttt{Alg}_{\text{on}}$ updates itself on the mistake
      break
    else if $s_t = s_t^\star$ and $s_{t-1} = s_{t-1}^\star$ **then**
      $T_{\min} = t + 1$
    else if $s_{t-1} \neq s_{t-1}^\star$ **then**
      $T_{\max} = t - 1$
  Denote the (potentially updated) model of $\texttt{Alg}_{\text{on}}$ as $\widehat{\pi}^{m+1} : S \times \Sigma \to S$
  Update $m \leftarrow m + 1$.
Return: $\widehat{\pi}_T^M$, drawing $M \sim \text{Unif}([n])$ once per test instance.