Learning through Internalization

Nikolaos Tsilivis
New York University
nt2231@nyu.edu

Nirmit Joshi
Toyota Technological
Institute at Chicago

Marko Medvedev
University of Chicago

Julia Kempe
New York University

Nathan Srebro
Toyota Technological
Institute at Chicago

Abstract

We study internalization processes, by which neural-network-based systems absorb an explicit computational procedure into their own weights, and how they facilitate learning. We investigate how transformers internalize the simulation of semiautomata by internalizing chain-of-thought (CoT) tokens, which classes of semiautomata are harder to internalize, and expose the flip side of internalization, that is, a progressive degradation of out-of-distribution performance. We then provide the first provable analysis of successful internalization: for the task of learning parities, we show that a simplified one-layer transformer provably first learns the target with explicit CoT supervision and then internalizes the autoregressive generation as CoT tokens are progressively removed, learning to directly compute the parity. Learning such a representation directly from data without CoT supervision is computationally hard. Finally, we discuss how learning through internalization relates to the *Positive Distribution Shift* phenomenon recently introduced by Medvedev et al. [2026].

1 Introduction

Humans often find it difficult to learn some tasks directly. When a teenager wants to learn how to drive a car, they benefit from a teacher who meticulously explains the many intermediate steps of the process and does not ask them to start practicing driving right off. Driving students begin by learning each of these subtasks (e.g. the role of each pedal, how to press the right button or clutch, how to steer the wheel) and only later do they become ready to operate the vehicle on the road. In fact, only after repetition and effort do some manage to drive without actively thinking about it. For these people, driving has become automatic, and they do not have to always think about all the intermediate steps. They have internalized their execution and have mastered driving, which was the task they ultimately cared about; all the other subtasks were merely stepping stones to it. And while doing things faster does not always mean doing them better, since doing things automatically can make people more prone to mistakes under varying conditions, we still recognize this internalization process as something positive and desirable. The whole process (explicit subtask learning, composition and finally internalization) permits the learning of difficult tasks that seem otherwise impenetrable, while the final stage of internalization allows humans to reduce execution time, enabling them to move on to other pursuits.

In machine learning, there have been many instances where a neural network has benefited from such a process. We define **internalization** as the process by which a neural-network-based system that initially implements a procedure over many steps has its weights reconfigured so that it implements the same procedure in far fewer steps. For example, neural-network based systems can learn how to play board games by internalizing the output of an expensive next-move search in their own weights, as in Samuel's work on checkers [Samuel, 1959] or Google's AlphaGo Zero [Silver et al., 2017].

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

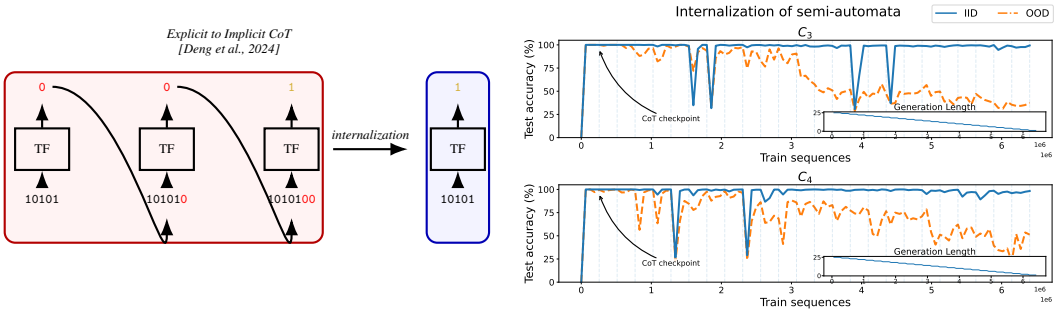


Figure 1: **Internalization of chain-of-thought in autoregressive transformers.** *Left*: An illustration of the outcome of the internalization process. *Right*: Two examples of internalization of semiautomaton simulation over $T = 25$ steps. As the transformer internalizes more chain-of-thought tokens, its out-of-distribution test accuracy (induced by a different sampling process for input tokens) degrades. Each dashed lines corresponds to one additional token dropped from the chain-of-thought.

More recently, several works have explored internalization schemes for machine reasoning with autoregressive large language models (LLMs) [Deng et al., 2023, 2024, Yu et al., 2024]. While training LLMs to solve complex problems "directly", i.e., without chain-of-thought (CoT) supervision, might often be difficult, these works present an alternative: a transformer first learns to autoregressively generate answers based on step-by-step CoT supervision, and is then later optimized to internalize the outcome of the lengthy generation in its weights, thereby generating responses with fewer, or perhaps even just one, forward pass (Figure 1, left). In particular, Deng et al. [2024] introduced a curriculum method for the internalization part that progressively removes CoT tokens during training, and showed that the whole process (explicit CoT training and then gradual internalization) can be very effective in arithmetic tasks, such as number multiplication, which otherwise appear impenetrable if attempted to be learned directly.

While this approach appears powerful and has seen success on several tasks [Deng et al., 2024], other works have reported mixed results on more sophisticated reasoning problems [Yu et al., 2024, Lin et al., 2025b], with Lin et al. [2025b] identifying that "internalized reasoning" can overfit to specific patterns and may fail to generalize more broadly. Motivated by the promising empirical results on internalization of CoT, in our work we pose and attempt to answer several questions regarding internalization. We ask:

Which tasks are easy to internalize, and which tasks are difficult? How do network representations evolve during internalization? Does internalization lead to representations that mirror explicit reasoning but unfold vertically across network layers, or does it instead lead to shortcut solutions that exploit the model's parallel computation? What are the principles that enable efficient learning through internalization?

Our contribution. First, in Section 2, we define what constitutes an internalization process throughout the paper, and explain how internalization can sometimes be used for learning purposes. In Section 3, we empirically study internalization of CoT in a canonical sequential reasoning setting: simulating a semiautomaton for T timesteps and outputting its final state. Then, in Section 4, we provide a theoretical result on the sparse parity task that shows that internalizing CoT can enable a model to learn representations that are computationally hard to find directly. Finally, in Section 5, we explore connections between learning through internalization and a recently introduced *Positive Distribution Shift* framework [Medvedev et al., 2026]. Our main results are summarized below.

- We identify several *method specific factors*, such as architecture or curriculum choices, that affect the probability of successful internalization. For example, we observe that, across different semiautomata tasks, internalization of CoT is consistently more successful with wider rather than deeper architectures at a fixed parameter budget. This perhaps contradicts the mental picture that implicit reasoning tokens are "represented vertically" in the network post-internalization.
- We identify *task specific factors* that control whether a semiautomaton is easy or hard to internalize. In particular, amongst the class of modular counter semiautomata those with

prime order appear to be the most difficult to internalize. We conjecturally relate this to the ability of transformers to compactly represent counter semiautomata whose order has small prime-power factors (Theorem 3.1).

- We expose the flip side of internalization: much like humans, transformers that internalize operations often tend to perform worse in OOD settings (Figure 1, right). In particular, the internalization process for semiautomata can lead the model to learn shortcut solutions [Liu et al., 2023], which can generalize worse than the representation outputted by CoT training.
- We provide the first provable analysis of learning through internalization: for the sparse parity task, we prove that a stylized one-layer transformer with linear attention and a fixed nonlinear MLP first learns the target parity of hidden support, using explicit CoT, and then internalizes this computation as CoT tokens are removed one at a time (Theorem 4.1), thereby directly computing the parity. Learning such a representation directly from data without internalization of CoT is computationally hard.

Finally, throughout the paper, and in greater detail in Section 5, we formally explain how learning through internalization provides a viable strategy for sidestepping the hardness of "direct" learning. We explicitly connect this perspective to the Positive Distribution Shift phenomenon [Medvedev et al., 2026]. Motivated by this, we explore a new internalization scheme for Multi Layer Perceptrons that enables learning hard parity functions (Section 5). For autoregressive transformers, we further empirically demonstrate that training on a single mixture distribution of "short" and "long" CoTs may already suffice for learning, without requiring an explicit curriculum that progressively removes tokens (Appendix F).

2 Internalization Processes

We first define what we call an internalization process throughout the paper. We then describe the main phenomenon of study: instead of learning a distribution directly, we first learn a separate distribution with a "slow" system and then internalize the "slow" system within a "faster" one.

Let Σ be an alphabet, $\mathcal{X} \subseteq \Sigma^*$ be an input space, Θ be an encoding (parameter) space, and let $\mathcal{Y} \subseteq \Sigma^*$ be an output space. Let $\mathcal{H} = \{h_\theta | \theta \in \Theta\} \subseteq \mathcal{Y}^{\mathcal{X}}$ be a hypothesis class of functions each indexed by $\theta \in \Theta$, and, for any $\theta \in \Theta$, let $\mathfrak{h}_\theta$ denote an algorithm that given input $x \in \mathcal{X}$ computes $h_\theta(x)$. Furthermore, for any $\theta \in \Theta$, denote the runtime of $\mathfrak{h}_\theta$ on input $x \in \mathcal{X}$ by $\mathrm{Time}_{\mathfrak{h}_\theta}(x)$. Let $P : \mathcal{Y}^{\mathcal{X}} \to \mathcal{Y}^{\mathcal{X}}$ be a procedure that, given access to a hypothesis h_θ, produces a new mapping $P(h_\theta) : \mathcal{X} \to \mathcal{Y}$. For any $\theta \in \Theta$, let $\mathfrak{P}(\mathfrak{h}_\theta)$ be an algorithm that given input $x \in \mathcal{X}$ computes $P(h_\theta)(x)$ and denote its runtime by $\mathrm{Time}_{\mathfrak{P}(\mathfrak{h}_\theta)}(x)$. We assume that $\mathfrak{P}(\mathfrak{h}_\theta)$ has only query access to $\mathfrak{h}_\theta$ and does not know $\mathcal{H}$ nor can it modify θ.

Definition 1 (Internalization)**.** *Fix $\theta \in \Theta$. If there exists $\theta' \in \Theta$ with $\theta' \neq \theta$ such that*

1. ***Output agreement:***
$$h_{\theta'}(x) = P(h_\theta)(x) \qquad \forall x \in \mathcal{X}. \tag{1}$$
2. *$\mathfrak{h}_{\theta'}$* ***is faster than*** *$\mathfrak{P}(\mathfrak{h}_\theta)$: For all $x \in \mathcal{X}$, it holds:*
$$\mathrm{Time}_{\mathfrak{h}_{\theta'}}(x) < \mathrm{Time}_{\mathfrak{P}(\mathfrak{h}_\theta)}(x), \tag{2}$$

then we say that $\mathfrak{h}_{\theta'}$ ***internalizes*** *$\mathfrak{P}(\mathfrak{h}_\theta)$.*

In words, if we have a program (e.g., a neural network) $\mathfrak{h}_\theta$ and some other system that uses this program (potentially many times) to produce a new program $\mathfrak{P}(\mathfrak{h}_\theta)$, then we say that $\mathfrak{h}_{\theta'}$ internalizes $\mathfrak{P}(\mathfrak{h}_\theta)$ if there exists a new, fast, program $\mathfrak{h}_{\theta'}$, induced by a hypothesis $h_{\theta'}$ belonging to the same class of hypotheses $\mathcal{H}$ as the original h_θ, that agrees with the previous slow $\mathfrak{P}(\mathfrak{h}_\theta)$ on all inputs.

While the above definition and example ground our discussion of internalization (see also Appendix B for further discussion), they do not explain how to internalize, nor do they concern themselves with learning any of the hypotheses. Next, we describe how an "indirect" approach to learning (learning of the slow inference system and then internalization) can sometimes be used instead of "direct" learning of a hard distribution.

Let $\mathcal{X}$ be an input space, let $\mathcal{Y}$ be an output space, let $f^\star : \mathcal{X} \to \mathcal{Y}$ be an unknown target function, and let $\mathcal{D}$ be a distribution over $\mathcal{X}$. For a hypothesis $h : \mathcal{X} \to \mathcal{Y}$, define its error by $L_{\mathcal{D}}(h) :=$

$\mathbb{P}_{x\sim\mathcal{D}}[h(x) \neq f^\star(x)]$. The usual end-to-end learning approach is to collect samples $(x, f^\star(x))$ with $x \sim \mathcal{D}$ and use them to find a hypothesis $h_\theta \in \mathcal{H}$ with small error $L_\mathcal{D}(h_\theta)$. However, for many natural hypothesis classes and target functions, such as parities, this direct learning problem may be computationally difficult [Valiant, 1984, Kearns and Valiant, 1994].

An alternative to such an "end-to-end" approach would be to capitalize on internalization: instead of trying to learn the final map directly, one may first learn a predictor with slower inference time $P(h_\theta) : \mathcal{X} \to \mathcal{Y}$, induced by an $h_\theta \in \mathcal{H}$, with small error $L_\mathcal{D}(P(h_\theta))$. This slower inference system may be easier to obtain because we can exploit additional supervision or search. The second step is to then find a hypothesis $h_{\theta'} \in \mathcal{H}$ that internalizes $P(h_\theta)$. Thus, the original goal of "end-to-end" learning can be split into two: learning of a "slower" system and then internalization.

Recently, Medvedev et al. [2026] formalized the intuition that a *positive* shift in the distribution can make such "end-to-end" learning problems tractable. In the terminology of Medvedev et al. [2026], the sequence of distributions forming the internalization curriculum of Deng et al. [2024] constitutes a positive distribution shift of the end-to-end distribution associated with the hard reasoning problems they study. We return to this point in Section 5.

3 Internalizing Semiautomata with Transformers

In this section, we consider a specific case of learning through internalization. In particular, we investigate whether transformers can internalize the simulation of *semiautomata*.

A semiautomaton consists of a state space, an input space, and a transition function that, given the current state and an input symbol, determines the next state. We are interested in predicting the state of the semiautomaton after T transition steps, specified by inputs $\sigma_1, \ldots, \sigma_T$ and an initial state q_0, and consider various types of training that differ in the type of supervision they provide beyond the input tokens:

1. *End-to-end* (E2E) training, where only the final state q_T is appended to the input,
2. *Chain-of-thought* (CoT) training, where sequences contain all the intermediate states $q_1, \ldots, q_{T-1}$ together with the initial and final states,
3. *Internalization* via a curriculum, where we first perform CoT training and then progressively start omitting intermediate states from the training sequences [Deng et al., 2024]. We consider internalization via a *left* and a *right* curriculum [Deng et al., 2024], which differ in whether the CoT starts getting amputated from the beginning or the end of the sequences, as well as an *inductive* curriculum that we introduce, which trains on sequences with progressively larger leaps between states.

In our experiments, we mainly consider modular counter semiautomata C_n, which have inputs and states from $\mathbb{Z}_n = \{0, \ldots, n-1\}$ and the state at time $t+1$ is computed as $q_{t+1} = (q_t + \sigma_t) \bmod n$. We also consider the non-commutative semiautomaton associated with the permutation group S_3. The input distribution draws each symbol uniformly at random from the input alphabet. To probe out-of-distribution generalization during training, we also consider distributions that bias the total sum of the input tokens. See Appendix E for more details and formal definitions.

Simulating semiautomata presents a clean setting for studying the internalization of CoT in transformers. Recall that, during internalization, we start training with explicit CoT containing $T-1$ intermediate states/sequential reasoning steps. Such a CoT sequence can be compactly represented by a *single* transformer layer (attention on the current state and the corresponding input symbol, while the MLP implements the transition function). However, the transformer needs to change its representation during internalization and could, in principle, converge to either of two qualitatively different ones: either to a representation that mirrors explicit reasoning but is unfolded vertically across $\Theta(T)$ transformer layers, or to a shallower representation that exploits the parallel computation inherent to the transformer. Indeed, we know that such shallow *shortcut* solutions, utilizing only $\Theta(\log T)$ transformer layers of reasonable width, exist for all semiautomata [Liu et al., 2023]. We ask: which type of representation is favored by SGD-type internalization? Can we internalize with shallow transformers, or does successful internalization of T steps demand $\Theta(T)$ transformer layers to begin with?

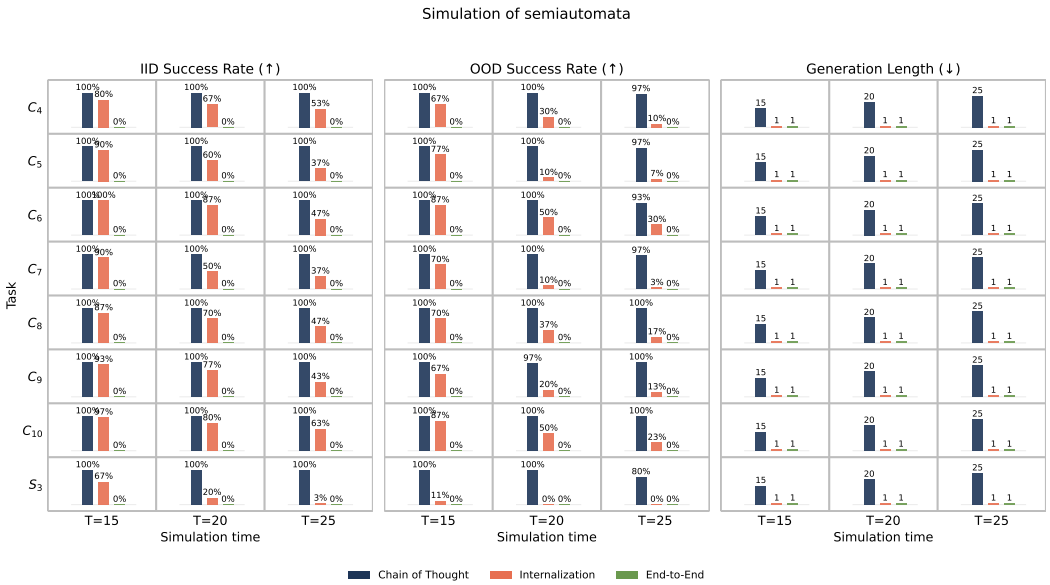


Figure 2: **Comparison between chain-of-thought (CoT), end-to-end (E2E) and (left) internalization training for semiautomata simulation with transformers.** We report in-distribution (ID) and out-of-distribution (OOD) success rates that are defined as the percentage of runs that resulted in >95% test accuracy (ID or OOD, respectively), as well as the length of the generated responses.

3.1 Internalizing chain of thought over model width

We first present results of training relatively shallow transformers (depth $L = 4$) compared to the simulation time $T \in \{15, 20, 25\}$ on a range of different semiautomata (Figure 2). We compare CoT and E2E training with internalization via left curriculum. In our experiments, the number of possible input sequences far exceeds the number of training sequences, so we can rule out learning via "memorization". We found that internalization runs were effectively binary in achieving solid test accuracy[1] or being on par to random chance, and for this reason we opted to report success rates (fraction of runs with test accuracy > 95%) instead of aggregated accuracy numbers. We ran each configuration for 30 seeds – see Appendix E for details. We list our observations:

- **E2E training fails.** We observe that transformers are not able to learn to simulate semiautomata with E2E training. The end-to-end tasks exhibit long-range dependencies that appear difficult to learn even for relatively "simple" local operations, such as $\mathrm{mod}n$ addition.
- **CoT training succeeds.** On the other hand, the task becomes easy with chain-of-thought training. There, the model leverages the additional supervision and generalizes on novel inputs; both in-distribution and out-of-distribution. Indeed, we observe very small drops (if any) from ID to OOD success rates across tasks for CoT training. The downside, of course, is the high inference time, since the model has to generate all the intermediate states before generating the final one.
- **Internalization succeeds with shallow transformers.** Remarkably, we find that the internalization process with the left curriculum often produces networks that generalize perfectly in distribution, despite them not generating any intermediate tokens; this holds even for non-commutative semiautomata. In fact, the internalization process can succeed even with depth-1 transformers (see, for example, Figure 4), demonstrating that an ability to explicitly represent all the intermediate reasoning steps is not necessary for "implicit" reasoning in the model. However, we observe a tradeoff: the OOD success rates of the internalization runs are significantly lower than those of CoT training. We return to this point in Section 3.2.

Width helps more than depth. We next study the role of depth in internalization.

[1] In this section, test accuracy refers to final-state accuracy after autoregressive generation.

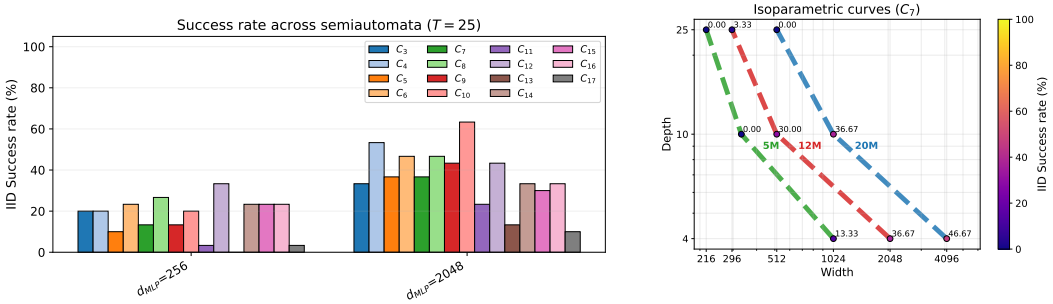


Figure 3: *Left:* **The probability of internalization depends on the order of the cyclic semiautomaton.** We show IID success rates across various counter semiautomata for two different width configurations (depth is fixed to $L = 4$). We observe that increasing the MLP width of the transformer improves the odds of successful internalization, but to a lesser extent for C_n with n being a prime number. *Right:* **The probability of internalization increases with width rather than depth.** We show in-distribution success rates across various transformer configurations. Each line corresponds to an isoparametric curve (scales: 5M, 12M, and 20M parameters).

In Figure 3 (right), we show internalization results for simulating the C_7 semiautomaton over $T = 25$ steps where we vary the depth (number of layers) and width (maximum of embedding dimension and MLP width) of the transformer while keeping the number of total parameters (approximately) fixed. We observe that, across different parameter scales, increasing depth ($4 \to 10 \to 25$) is less beneficial for internalization than increasing width. In fact, deeper models ($L = 25$) do not appear well suited for internalization. See also Figure 7 for similar plots on C_{11} and C_{13}. These findings support the hypothesis that the intermediate states/reasoning steps are internalized "horizontally" over the width of the model rather than being explicitly represented "vertically" over its depth.

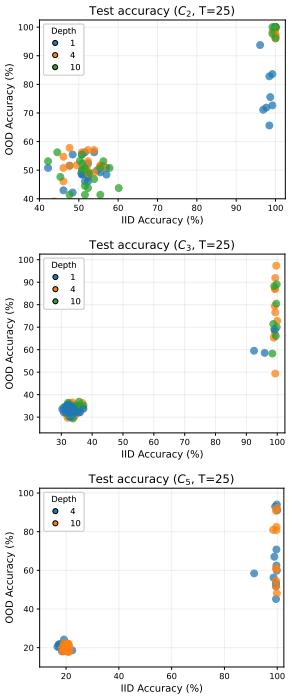


Figure 4: Scatter plots of test accuracy at the end of internalization for several seeds across transformers with various depths.

3.2 The flip side of internalization: worse OOD performance

Despite the previous positive internalization results with shallow transformers, we observe what appears to be the flip side of internalization: the decline of OOD performance. Indeed, Figure 2 shows that across tasks and simulation times, internalized runs are far less probable to have good OOD test accuracy in comparison to transformers produced by CoT training.

In Figure 1 (right), we show two internalization runs for C_3 and C_4, respectively, that demonstrate how the OOD performance progressively gets worse as we remove more tokens from the chain-of-thought supervision during internalization.

Furthermore, in Figure 4, we present a scatter plot of internalization runs for various random seeds, showing the final test accuracy (in-distribution and out-of-distribution) of the transformer on C_2, C_3, and C_5. For all three tasks, we observe that succeeding in internalizing in distribution does not necessarily imply good out-of-distribution performance. For C_2, we observe that increasing depth has a positive effect on OOD performance, while for C_3 and C_5 this OOD weakness persists even for deeper networks. These results imply that in our settings transformers fail to exactly internalize in the strict sense of Definition 1.

We hypothesize that the decline in OOD performance during internalization occurs because the shallow transformer is forced to implement the $\mathrm{mod}n$ operation over more than two input tokens (as initially done in the CoT representation), and this may be difficult for a single MLP to robustly learn from uniformly sampled sequences.

3.3 Prime C_n are harder to internalize

Semiautomata such as C_2, C_7, C_{18} all share the same state space structure, but they differ in the transition function, i.e., the local operation rule, which depends on the order of the semiautomaton. The exact value of the order does not seem to affect whether a transformer can learn under CoT or E2E training. But, could the "complexity" of the local operation matter for successful internalization?

In Figure 3 (left), we plot success rates on all counter semiautomata from C_3 all the way to C_{17} for two different width configurations. We observe a noticeable drop in the probability of internalization for semiautomata C_n, where n is a prime number. For example, semiautomata C_{11} and C_{13} appear to be more difficult to internalize than C_{14}, C_{15} and C_{16}, even though they have a comparable number of possible inputs and states. This difference persists even for wider networks (MLP width: $256 \rightarrow 2048$). Why might this be the case?

In order to internalize a semiautomaton, the transformer needs at the very least to be able to represent the final end-to-end mapping from T inputs and initial state to final state. In the regime we are studying, where the number of layers is much smaller than the horizon of the simulation, prior work has established that generic semiautomata with $|Q|$ states can be represented by a transformer of $O\left(|Q|^2\right)$ MLP width [Liu et al., 2023]. While this suggests that a transformer might need larger width to internalize counter semiautomata of larger order, it unfortunately tells us very little on how to differentiate between semiautomata whose orders differ only sligthly, e.g., C_{10} vs C_{11}.

We show next that a transformer can represent some counter semiautomata C_n much more compactly than what was known before, using an MLP width that depends on the sizes of the *prime-power factors* (prime factors with their multiplicity taken into account) of n. In particular, this implies that C_n, where n has small prime-power factors, can be represented efficiently by a small transformer.

Theorem 3.1 (Representation of C_n, $n = \prod_{j=1}^{k} p_j^{v_j}$, with $O(\log T)$ depth and $O(\sum_{j=1}^{k} p_j^{v_j})$ width; informal)**.** *Consider a counter semiautomaton C_n, $n = \prod_{j=1}^{k} p_j^{v_j}$. Then, for any T, there exists a transformer that simulates C_n over T steps using $O\left(\log T\right)$ layers, embedding dimension $O\left(k\right)$ and MLP width $O\left(\sum_{j=1}^{k} p_j^{v_j}\right)$.*

See Theorem C.3 in the appendix for a formal statement and proof. The proof exploits the fact that addition of two numbers $\mathrm{mod} n$ can be performed "in parallel" in the basis of each prime-power factor of n due to the Chinese Remainder Theorem. Theorem C.3 implies that if n is a smooth number, i.e, $n = \prod_{i=1}^{k} p_i$, $p_i = O(1)$, then C_n can be simulated using an MLP of width $\sum_{j=1}^{k} p_j = O(\log n)$. For this class of counter semiautomata, this construction implies an exponential improvement on the size of the transformer, compared to the (general) result of Liu et al. [2023].

Based on this result, we conjecture that semiautomata such as C_{10} and C_{12} are easier to learn through internalization than C_{11} because the size of their final representation is smaller and hence they require fewer samples to learn. Speculating even further, we could hypothesize that the more "parallelizable" the local operation of a semiautomaton is, the easier it might be to internalize. We defer a deeper investigation of this fascinating question to future work.

We discuss additional optimization and architectural choices and how they they affect the internalization of semiautomata in Appendix E.

4 Provable Internalization of Parities in 1-layer Transformer with Linear Attention

In this section, we present our main theoretical result: a simplified one-layer transformer can provably learn a hidden parity with explicit CoT supervision and then internalize the computation, outputting the parity in a single generation step (Theorem 4.1).

We study the problem of learning a hidden parity over an unknown support $S_\star = \{i_1, \ldots, i_k\} \subseteq [d]$, where the input $x = (x_1, \ldots, x_d) \sim \mathrm{Unif}(\{0,1\}^d)$, and the final target is $y = \bigoplus_{i \in S_\star} x_i = x_{i_1} \oplus \cdots \oplus x_{i_k}$. This class is hard to learn in the statistical query model [Kearns, 1998], which is often viewed as capturing limitations of gradient-based learning, and is also believed to be hard to learn in the noisy PAC setting. Kim and Suzuki [2024] showed that, with CoT supervision, transformers

can provably learn parity efficiently, but the learned model still explicitly generates the CoT tokens autoregressively at inference time. Here, we show that this computation can also be internalized: after a CoT-shortening curriculum (from left), the model learns to directly output the parity. This gives a provable example where learning through internalization enables learning a task that is computationally hard for "direct" learning.

CoT and Curriculum. In addition to the final label, we use a CoT sequence given by the prefix parities $z_t = x_{i_1} \oplus \cdots \oplus x_{i_t}$, $t \in [k]$, so that $z_k = y$. The curriculum begins with the full sequence $a^{[1]} = (x_1, \ldots, x_d, z_1, \ldots, z_k)$ and progressively removes tokens from left such that in Phase t, the input sequence is

$$a^{[t]} = (x_1, \ldots, x_d, z_t, z_{t+1}, \ldots, z_k). \tag{3}$$

After all CoT tokens are removed, the model must directly output the parity given input $(x_1, \ldots, x_d)$.

Architecture. We analyze a simplified one-layer linear-attention transformer, similar in spirit to prior work of Kim and Suzuki [2024]. The architecture uses a single attention head that collapses the key-query parameterization into a single matrix W_{QK}, and fixes the value matrix and MLP. For a content sequence $a = (a_1, \ldots, a_D) \in \mathbb{R}^D$, we represent each token in dimension $D + 1$, with one scalar content coordinate and D one-hot positional coordinates:

$$X(a) = \begin{pmatrix} a_1 & 1 & 0 & \cdots & 0 \\ a_2 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_D & 0 & 0 & \cdots & 1 \end{pmatrix} \in \mathbb{R}^{D \times (D+1)}.$$

The attention scores are bilinear (without softmax attention) that have been also popularized as a variant to softmax [e.g., Katharopoulos et al., 2020, Shen et al., 2021]:

$$S = X(a) W_{QK} X(a)^\top,$$

where $W_{QK} \in \mathbb{R}^{(D+1)\times(D+1)}$. We fix the value matrix to be the identity, $W_V = I_{D+1}$, and apply the causal mask to obtain the attention output

$$H(X(a)) = (M \odot X(a) W_{QK} X(a)^\top) X(a),$$

where $M_{ij} = \mathbf{1}\{j \le i\}$. For any position i, the fixed MLP reads the first coordinate of the residual stream $H_i(X(a)) \in \mathbb{R}^{D+1}$, which corresponds to the content coordinate, and applies a nonlinearity $\sigma : \mathbb{R} \to \mathbb{R}$:

$$f_W(a)_i = \sigma((H_i(X(a)))_1).$$

We choose σ to be a piecewise-linear parity activation, equal to 0 on even integers and 1 on odd integers, with linear interpolation between consecutive integers; see Equation (23). Following the same simplification as in Kim and Suzuki [2024], we assume that W_{QK} ignores the scalar content coordinate and is parameterized only through the positional block:

$$W_{QK} = \begin{pmatrix} 0 & 0 \\ 0 & W \end{pmatrix}, \qquad W \in \mathbb{R}^{D \times D}.$$

See Appendix D.1 for how this choice greatly simplifies the architecture: for any position $i \in [D]$,

$$f_W(a)_i = \sigma(H_i(X(a))_1) = \sigma\left(\sum_{j \le i} W_{ij} a_j\right). \tag{4}$$

Training Algorithm. We now specify how the model is trained in plain language, with pseudocode deferred to Algorithm 2. In Phase 1, we use the full teacher-forced input sequence $a^{[1]} = (x_1, \ldots, x_d, z_1, \ldots, z_k)$ from Equation (3) and train on an empirical i.i.d. batch for the following population objective:

$$L_1(W) := \frac{1}{2}\mathbb{E}_a\left[\sum_{l=1}^{k}\left(f_W(a^{[1]})_{d+l-1} - a^{[1]}_{d+l}\right)^2\right] = \frac{1}{2}\mathbb{E}_a\left[\sum_{l=1}^{k}\left(f_W(a^{[1]})_{d+l-1} - z_l\right)^2\right].$$

This amounts to CoT training on the output part of the sequence, as in supervised fine tuning (SFT). In subsequent phases $t = 2, \ldots, k$, we use the left-amputated sequence $a^{[t]} = (x_1, \ldots, x_d, z_t, \ldots, z_k)$ from Equation (21) and, after drawing a fresh batch, train on the empirical version of the following objective:

$$L_t(W) = \frac{1}{2}\mathbb{E}_a\left[\left(f_W(a^{[t]})_d - a^{[t]}_{d+1}\right)^2\right] = \frac{1}{2}\mathbb{E}_a\left[\left(f_W(a^{[t]})_d - z_t\right)^2\right].$$

Note that we only train on the first output position for stages $t = 2, \ldots, k$. A fresh batch of size B is drawn in each phase, and we train on the same batch for T iterations within that phase, with step size η. At the end of each phase, we round the entries of W to the nearest integer.[2] Finally, we use an initialization scheme with mostly zero entries, except $W^{(0)}_{j,j} = -1$ for $j = d + l$ for all $l \in [k-1]$. This encodes the bias that the model should attend to the token at position $d + l$, since in Phase 1, we know that $z_{l+1} = z_l \oplus x_{i_{l+1}}$, and attending to z_l is important.[3] We formally state this training algorithm in Algorithm 2 and show the following performance guarantee.

Theorem 4.1. *There exist universal constants $C, c > 0$ such that running Algorithm 2 with*

$$\eta = \frac{c}{d}, \qquad T = \lceil Cd \rceil, \qquad B \geq Cd^3 \log \frac{kT}{\delta},$$

satisfies the following with probability at least $1 - \delta$.

- *After Phase 1, the model $W^{(1)}$ computes all explicit CoT tokens autoregressively: for every $l \in [k]$ and for every $x \in \{0,1\}^d$,*

$$f_{W^{(1)}}\left(\texttt{append}(x, f_{W^{(1)}}(x)[d+1 : d+l-1])\right)_{d+l-1} = x_{i_1} \oplus \cdots \oplus x_{i_l}\,. \tag{5}$$

- *At the end of the k^{th} phase of the curriculum, the final returned model $f_{W^{(k)}}$ has internalized $P\left(f_{W^{(1)}}\right) = f^{\text{e2e}-k}_{W^{(1)}}$ (in the sense of Definition 1 and Equation (9)) and, in particular, it holds for every $x \in \{0,1\}^d$*

$$f_{W^{(k)}}(x)_d = \bigoplus_{i \in S_\star} x_i. \tag{6}$$

The overall runtime of the algorithm is $\text{poly}(d, \log(1/\delta))$, and it finds a representation $W^{(k)}$ that directly computes the target parity in one forward pass. For the proof of Theorem 4.1, see Appendix D.

Remark 4.2. *We note that the guarantee in Theorem 4.1 ensures exact internalization for all inputs $x \in \{0,1\}^d$, whereas in Section 3.2, we report poor OOD performance on related tasks. While the theoretical result involves simplifications, we attribute the main source of this mismatch to the fixed choice of the MLP for local operations. In our empirical study, these local operations also need to be adjusted, which we believe is primarily responsible for the OOD performance degradation. We leave it to future work to theoretically prove internalization results that exhibit worse OOD performance.*

5 Learning through Internalization and Positive Distribution Shift

Finally, in this section, we introduce and study a novel internalization scheme that enables learning hard functions with feed forward neural networks that is inspired by internalization of chain-of-thought. Here, the slower system does not involve multiple calls to the same neural network, as in the autoregressive case, nor does it search for better outputs according to some fixed rule, but rather maintains an additional set of weights that operate on *hints* provided together with the input. The role of the internalization process is then to transfer the computation onto the weights that operate on the original input space, by relying less and less on hints. This perspective allow us to

[2] Such rounding was also used in Kim and Suzuki [2024], though for a different technical reason, and is also used empirically to reduce memory storage requirements [Wu et al., 2020, Jacob et al., 2018] and for efficient fine-tuning [Dettmers et al., 2022]. In our analysis, it seems to be needed only for a minor technical reason, and the entries before rounding are already close to integers. Experiments show full success without any rounding; cf. Figure 6.

[3] We can also work with the all-zero initialization $W^{(0)} = 0$, in which case the final internalization guarantee in Equation (6) from Theorem 4.1 still holds. This is because the final output at the d^{th} position involves only the d^{th} row $W_{d,:}$, and by Equation (4), it does not depend on the initialization of other rows. Initializing these entries with -1 is only needed for the guarantee of outputting explicit CoT in Equation (5).

connect learning through internalization with the Positive Distribution Shift framework introduced in Medvedev et al. [2026]. Motivated by this connection, we show that learning through a curriculum-based internalization procedure can often be replaced by a more straigthforward mixture learning procedure. We validate this in both the feed forward and the autoregressive setting in Appendix E.1.s

Let k denote the size of the unknown support of the parity function, as in Section 4. In this section, we consider added label noise in the target (we flip the final label with probability η). First, we augment the d-bit input $x = (x_1, \ldots, x_d)$, which is uniformly distributed, with $k-1$ hints z_i of intermediate computations to obtain $\widetilde{x} = (x_1, \ldots, x_d, z_1, \ldots, z_{k-1})$. The t-th hint, $t \in [k-1]$, is the partial parity of the first t indices of the support, i.e., $z_t = \oplus_{j=1}^{t} x_{i_j}$. The augmented inputs are similar to the CoT tokens of the previous section. During the first stage of training, we learn the mapping $\widetilde{x} = (x_1, \ldots, x_d, z_1, \ldots, z_{k-1}) \mapsto y$ by training a ReLU neural network of depth L: $g(\widetilde{x}; \{W_L, b_L, \ldots, W_1, b_1\}) = W_L\, \sigma(W_{L-1}\, \sigma(\cdots \sigma(W_1 \widetilde{x} + b_1) \cdots) + b_{L-1}) + b_L$. Then, we take only the subnetwork of g that operates on the first d bits $h_\theta : \{0,1\}^d \mapsto \mathbb{R}$, $h_\theta(x) = W_L\, \sigma(W_{L-1}\, \sigma(\cdots \sigma(W_1[:, :d]x + b_1) \cdots) + b_{L-1}) + b_L$, as our predictor.

Similar to the autoregressive case and the curriculum of Deng et al. [2024], we now devise a curriculum that teaches this subnetwork to internalize the parity computation as performed initially by g. In particular, at the l'th stage, $l = 2, \ldots, k-1$, we continue training g on mappings of the form $\left(x_1, \ldots, x_d, z_1, \ldots, z_{k-1-l+1}, \underbrace{0, \ldots, 0}_{l-1}\right) \mapsto y$. That is, we progressively zero-out some of the hints provided to the model, while keeping the target fixed. At the final stage of the curriculum, we train on data with all hints zeroed out, i.e. $(x_1, \ldots, x_d, 0, \ldots, 0) \mapsto y$.

We show that this strategy indeed succeeds in learning parity target functions. In Figure 10, we show experiments for $d = 25, k = 12, \eta = 0.1$ and a 2-layer ReLU network. Note that this scenario falls within Definition 1 if we define: (a) the "slow" system $P(h_{\theta_1})(x) := g\left(\left(x, x_{i_1}, \ldots, \oplus_{j=1}^{k-1} x_{i_j}\right); \left\{W_L^{[1]}, b_L^{[1]}, \ldots, W_1^{[1]}, b_1^{[1]}\right\}\right)$, where $W_L^{[1]}, b_L^{[1]}, \ldots, W_1^{[1]}, b_1^{[1]}$ are the weights at the end of the first training stage, and (b) the "fast" system $h_{\theta_{k-1}}(x) := W_L^{[k-1]}\, \sigma\Big(W_{L-1}^{[k-1]}\, \sigma\Big(\cdots \sigma\Big(W_1^{[k-1]}[:, :d]x + b_1^{[k-1]}\Big) \cdots\Big) + b_{L-1}^{[k-1]}\Big) + b_L^{[k-1]}$, where $W_L^{[k-1]}, b_L^{[k-1]}, \ldots, W_1^{[k-1]}, b_1^{[k-1]}$ are the weights at the end of the last stage of the curriculum.

6 Discussion

We saw how internalization can be a viable strategy not only for reducing inference costs while maintaining strong performance, but also for enabling the learning of difficult tasks that otherwise appear hard to learn through a more "direct", end-to-end approach. We investigated such internalization processes in autoregressive transformers and developed a better understanding of their strengths and shortcomings. Already for counter automata, we observed that, while learning without CoT is hard and learning with explicit CoT supervision is easy, the success of internalization varies across tasks and training methods (Figure 2). This suggests that understanding internalization requires accounting for new factors that are captured neither by the intrinsic difficulty of directly learning the end-to-end task nor by the ease of learning with step-by-step supervision.

While the focus of this paper has primarily been on internalizing autoregressive chain-of-thought reasoning, there have been many other instances in which neural networks have benefited from similar processes, as already discussed in the introduction. One such example is AlphaGo Zero [Silver et al., 2017], a neural-network-based system equipped with a tree-search procedure that could suggest next moves and evaluate board positions (see Example B.4, where we formalize this). Arguably, Samuel [1959]'s AI checkers player also learned through a similar internalization process.

In general, as AI inference costs continue to increase, the internalization of expensive scaffolded procedures will likely become increasingly common. While we have explained how this approach can be beneficial both for reducing computation time and for facilitating learning, some of our findings suggest that internalization may at times have negative and unexpected effects on the broader capabilities of a system.

Acknowledgments. NT would like to acknowledge useful discussions with Eran Malach and Elisabetta Cornacchia. JK thanks the Simons Foundation for support through the Collaborative Grant "The Physics of Learning and Neural Computation". NT acknowledges support from a grant by the Coefficient Giving. NT and JK acknowledge support by the NSF through NRT Award 1922658.

References

Emmanuel Abbe, Elisabetta Cornacchia, and Aryo Lotfi. Provable advantage of curriculum learning on parity targets with mixed inputs. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

Emmanuel Abbe, Elisabetta Cornacchia, Jan Hazla, and Donald Kougang-Yombi. Learning high-degree parities: The crucial role of the initialization. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*, 2025.

Avrim Blum, Merrick L. Furst, Jeffrey C. Jackson, Michael J. Kearns, Yishay Mansour, and Steven Rudich. Weakly learning DNF and characterizing statistical query learning using fourier analysis. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing*, pages 253–262. ACM, 1994. doi: 10.1145/195058.195147.

Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *Journal of the ACM*, 50(4):506–519, 2003. doi: 10.1145/792538.792543.

Enric Boix-Adserà. Towards a theory of model distillation. *CoRR*, abs/2403.09053, 2024.

Xiao Chen, Sihang Zhou, Ke Liang, Xiaoyu Sun, and Xinwang Liu. Skip-thinking: Chunk-wise chain-of-thought distillation enable smaller language models to reason better and faster. *arXiv preprint arXiv:2505.18642*, 2025.

Jeffrey Cheng and Benjamin Van Durme. Compressed chain of thought: Efficient reasoning through dense representations. *arXiv preprint arXiv:2412.13171*, 2024.

Elisabetta Cornacchia and Elchanan Mossel. A mathematical model for curriculum learning for parities. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, Proceedings of Machine Learning Research, pages 6402–6423. PMLR, 2023.

Yuntian Deng, Kiran Prasad, Roland Fernandez, Paul Smolensky, Vishrav Chaudhary, and Stuart M. Shieber. Implicit chain of thought reasoning via knowledge distillation. *CoRR*, abs/2311.01460, 2023. doi: 10.48550/ARXIV.2311.01460.

Yuntian Deng, Yejin Choi, and Stuart M. Shieber. From explicit cot to implicit cot: Learning to internalize cot step by step. *CoRR*, abs/2405.14838, 2024. doi: 10.48550/ARXIV.2405.14838. URL `https://doi.org/10.48550/arXiv.2405.14838`.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in neural information processing systems*, 35: 30318–30332, 2022.

Surbhi Goel, Varun Kanade, Adam Klivans, and Justin Thaler. Reliably learning the ReLU in polynomial time. In *Proceedings of the 2017 Conference on Learning Theory*, volume 65 of *Proceedings of Machine Learning Research*, pages 1004–1042. PMLR, 2017.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2024.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015. URL `http://arxiv.org/abs/1503.02531`.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2704–2713, 2018.

Nirmit Joshi, Gal Vardi, Adam Block, Surbhi Goel, Zhiyuan Li, Theodor Misiakiewicz, and Nathan Srebro. A theory of learning with autoregressive chain of thought. In Nika Haghtalab and Ankur Moitra, editors, *The Thirty Eighth Annual Conference on Learning Theory, 30-4 July 2025, Lyon, France*, Proceedings of Machine Learning Research, pages 3161–3212. PMLR, 2025.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, 2020.

Michael J. Kearns. Efficient noise-tolerant learning from statistical queries. *J. ACM*, 45(6):983–1006, 1998.

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

Juno Kim and Taiji Suzuki. Transformers provably solve parity efficiently with chain of thought. *CoRR*, abs/2410.08633, 2024. doi: 10.48550/ARXIV.2410.08633. URL `https://doi.org/10.48550/arXiv.2410.08633`.

Tianhe Lin, Jian Xie, Siyu Yuan, and Deqing Yang. Implicit reasoning in transformers is reasoning through shortcuts. *CoRR*, abs/2503.07604, 2025a. doi: 10.48550/ARXIV.2503.07604.

Tianhe Lin, Jian Xie, Siyu Yuan, and Deqing Yang. Implicit reasoning in transformers is reasoning through shortcuts. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 9470–9487, 2025b.

Bingbin Liu, Jordan T. Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. Transformers learn shortcuts to automata. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*, 2023.

Tengxiao Liu, Qipeng Guo, Xiangkun Hu, Jiayang Cheng, Yue Zhang, Xipeng Qiu, and Zheng Zhang. Can language models learn to skip steps? In *Advances in Neural Information Processing Systems*, 2024.

Marko Medvedev, Kaifeng Lyu, Zhiyuan Li, and Nathan Srebro. Shift is good: Mismatched data mixing improves test performance. *CoRR*, abs/2510.25108, 2025.

Marko Medvedev, Idan Attias, Elisabetta Cornacchia, Theodor Misiakiewicz, Gal Vardi, and Nathan Srebro. Positive distribution shift as a framework for understanding tractable learning, 2026. URL `https://arxiv.org/abs/2602.08907`.

Arthur L. Samuel. Some studies in machine learning using the game of checkers. *IBM J. Res. Dev.*, 3 (3):210–229, 1959.

Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. Codi: Compressing chain-of-thought into continuous space via self-distillation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2025.

Zhuoran Shen, Mingyuan Zhang, Haiyu Zhao, Shuai Yi, and Hongsheng Li. Efficient attention: Attention with linear complexities. In *Winter Conference on Applications of Computer Vision*, 2021.

Itamar Shoshani and Ohad Shamir. Hardness of learning fixed parities with neural networks. *CoRR*, abs/2501.00817, 2025. doi: 10.48550/arXiv.2501.00817.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy P. Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nat.*, 550(7676):354–359, 2017.

Leslie G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, 1984.

Hao Wu, Patrick Judd, Xiaojie Zhang, Mikhail Isaev, and Paulius Micikevicius. Integer quantization for deep learning inference: Principles and empirical evaluation. *arXiv preprint arXiv:2004.09602*, 2020.

Ping Yu, Jing Xu, Jason Weston, and Ilia Kulikov. Distilling system 2 into system 1. *CoRR*, abs/2407.06023, 2024.

A Related Work

Internalized and compressed reasoning. Recent works have explored reducing the cost of explicit CoT by replacing, compressing, or skipping reasoning tokens using latent reasoning states, compressed CoT, or step-skipping objectives [Hao et al., 2024, Shen et al., 2025, Cheng and Van Durme, 2024, Liu et al., 2024, Chen et al., 2025]. These works primarily propose mechanisms for making reasoning traces shorter or latent, focusing on the inference speed aspect of internalization. Lin et al. [2025a] made observations similar to ours on the poor OOD generalization of internalized CoT representations in arithmetic tasks with LLMs.

Positive Distribution Shift. Deng et al. [2024], Yu et al. [2024] show that internalized reasoning can lead autoregressive networks to learn representations that are otherwise difficult to discover through "direct" training. In particular, Deng et al. [2024] devise a curriculum-based internalization method that achieves this. A similar phenomenon has previously been observed in feedforward networks for computationally hard function classes [Abbe et al., 2023, Cornacchia and Mossel, 2023]. In particular, Abbe et al. [2023], Cornacchia and Mossel [2023] show that a carefully chosen curriculum helps a two-layer MLP learn a hidden parity in a way that bypasses traditional computational lower bounds. More generally, and as we explain further throughout the paper, we view these phenomena as falling under the broader learning phenomenon of *Positive Distribution Shift*, recently identified by Medvedev et al. [2026]. The statistical advantage of a distribution shift was first observed in Medvedev et al. [2025].

B Internalization processes: discussion and examples

Next, we provide an additional definition that seems to better align with internalization in machine learning practice as well as some examples of an internalization process. In particular, relaxing Definition 1, we define an approximate internalization process with respect to some underlying input distribution and loss function.

Definition 2 (Distributional Internalization)**.** *Let $\mathcal{D}$ be a distribution over $\mathcal{X}\times\mathcal{Y}$ and let $l : \mathcal{Y}\times\mathcal{Y} \to \mathbb{R}$ be a measurable*[4] *loss function. Fix $\theta \in \Theta$ and let $\varepsilon > 0$. If there exists $\theta' \in \Theta$ with $\theta' \neq \theta$ such that*

1. ***Approximate risk agreement:***

$$\left|\mathbb{E}_{(x,y)\sim\mathcal{D}}\left[l\left(h_{\theta'}(x), y\right)\right] - \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[l\left(P(h_\theta)(x), y\right)\right]\right| \leq \varepsilon. \tag{7}$$

2. $\mathfrak{h}_{\theta'}$ ***is faster than*** $\mathfrak{P}(\mathfrak{h}_\theta)$***:*** *For all $x \in \mathcal{X}$, it holds:*

$$\mathrm{Time}_{\mathfrak{h}_{\theta'}}(x) < \mathrm{Time}_{\mathfrak{P}(\mathfrak{h}_\theta)}(x), \tag{8}$$

*then we say that $\mathfrak{h}_{\theta'}$ $(\varepsilon, \mathcal{D})$-**distributionally internalizes** $\mathfrak{P}(\mathfrak{h}_\theta)$.*

Most internalization processes we consider in this work fall within this definition, as the internalization is performed using some training data following a particular distribution. Some remarks are in order that apply to both definitions.

Remark B.1. *We assume that the initial slow program and the final fast one are induced by hypotheses belonging to the same class. We believe this is analogous to the human case, where a single brain is reconfigured during internalization. This choice also distinguishes internalization from other notions previously studied in machine learning, such as model distillation [Hinton et al., 2015], which is typically viewed as distilling a source class $\mathcal{F}$ into a target class $\mathcal{G}$ [Boix-Adserà, 2024].*

Remark B.2. *In our definition, we prohibit the program $\mathfrak{P}(\mathfrak{h}_\theta)$ from interacting with $\mathcal{H}$ or updating θ, and we allow it only query access to $\mathfrak{h}_\theta$. That is, $\mathfrak{P}(\mathfrak{h}_\theta)$ is an inference-type algorithm. Without these assumptions, the definition would permit the slow program to be an iterative algorithm initialized at $\mathfrak{h}_\theta$, such as gradient descent initialized at θ, which is a more general notion than the one we seek to isolate in this work. Note that internalization, as defined here, needs not be a learning process. It does not assume access to a training dataset, nor do we evaluate absolute performance with respect to a test distribution.*

[4] We also assume that all hypotheses $h_\theta \in \mathcal{H}$ and all predictors $P(h_\theta)$ are measurable and that the written expectations are finite.

Next, we present three examples of internalization.

Example B.3 (Internalizing autoregressive chain-of-thought)**.** We follow the notation of Joshi et al. [2025]. Let Σ be a finite token alphabet, let $\mathcal{X} = \Sigma^*$ be the space of prompts, and let $\mathcal{Y} = \Sigma$ be the space of output tokens. Let $\mathcal{H} = \{h_\theta | \theta \in \Theta\} \subseteq \mathcal{Y}^{\mathcal{X}}$ be a class of next-token generators indexed by $\theta \in \Theta$ for some set Θ.

Fix a $\theta \in \Theta$. For the generator h_θ, define its apply-and-append map $\bar{h}_\theta : \mathcal{X} \to \mathcal{X}$ by $\bar{h}_\theta(x) := \texttt{append}\big(x, h_\theta(x)\big)$. For a fixed generation length $T \in \mathbb{N}_+$, define

$$P(h_\theta)(x) = h_\theta^{\mathrm{e2e}-T}(x) := \underbrace{\bar{h}_\theta \circ \bar{h}_\theta \ldots \circ \bar{h}_\theta}_{T \text{ times}}(x)[-1], \tag{9}$$

where $z[-1]$ denotes the last token of a string z. This is the mapping $P(h_\theta)$ from prompt to final token, ignoring the generated chain-of-thought. Also, define corresponding programs $\mathfrak{h}_\theta, \mathfrak{P}(\mathfrak{h}_\theta)$ for functions $h_\theta, P(h_\theta)$, respectively. Assume that the runtime of $\mathfrak{h}_\phi$ on a prompt depends only on the prompt length and not on ϕ, that is $\mathrm{Time}_{\mathfrak{h}_\phi}(x) = \mathrm{Time}_{\mathfrak{h}_\theta}(x)$ for all x, ϕ, θ, and that $\mathfrak{P}(\mathfrak{h}_\theta)$ computes $P(h_\theta)(x)$ from an input x by performing the T calls to $\mathfrak{h}_\theta$ explicitly.

Suppose there exists $\theta' \neq \theta$ such that

$$h_{\theta'}(x) = P(h_\theta)(x), \qquad \forall x \in \mathcal{X}. \tag{10}$$

Let $r(n)$ denote the runtime of $\mathfrak{h}_\phi$ on a prompt of length n. Then, we have $\mathrm{Time}_{\mathfrak{h}_{\theta'}}(x) = r(|x|)$, whereas the explicit autoregressive implementation satisfies $\mathrm{Time}_{\mathfrak{P}_T(\mathfrak{h}_\theta)}(x) \geq \sum_{t=0}^{T-1} r(|x| + t)$. Hence, if $T > 1$ and $r(n) > 0$ for all n, then

$$\mathrm{Time}_{\mathfrak{h}_{\theta'}}(x) < \mathrm{Time}_{\mathfrak{P}_T(\mathfrak{h}_\theta)}(x). \tag{11}$$

Thus, according to Definition 1, $\mathfrak{h}_{\theta'}$ *internalizes* $\mathfrak{P}(\mathfrak{h}_\theta)$. In words, the updated generator $h_{\theta'}$ produces in one forward pass exactly the same final answer that the original generator h_θ would have produced after T autoregressive steps.

Example B.4 (Internalizing self-play)**.** We formalize the setting of AlphaGo Zero [Silver et al., 2017]. Let S be a set of states and A be a finite set of actions. Let $\texttt{enc}_S : S \to \Sigma^\star$ be an encoding of states using an alphabet Σ, and let $\texttt{enc}_Y : \Delta(A) \times [0, 1] \to \Sigma^\star$ be an encoding of tuples of distributions over actions and state values. Let $\mathcal{X} \subseteq \Sigma^\star$ and $\mathcal{Y} \subseteq \Sigma^\star$ be the input and output spaces. Let $\mathcal{H} = \{h_\theta | \theta \in \Theta\} \subseteq \mathcal{Y}^{\mathcal{X}}$ be a hypothesis class of functions that map (encoded) states to an (encoded) tuple of distribution over actions and winning probability. In Silver et al. [2017], this class was realized by a convolutional neural network with two separate heads predicting each element of the output tuple. Assume for simplicity that each state s contains the information of which player is currently playing and this information is encoded in the last character of $\texttt{enc}_S(s)$.

Fix a $\theta \in \Theta$. Let $\mathfrak{P}_{\mathrm{SP}}(\mathfrak{h}_\theta)$ be a self-play program that receives as input an input $x \in \mathcal{X}$ and computes $P_{\mathrm{SP}}(h_\theta)(x)$. This program performs the following steps:

- Computes an updated distribution $\pi(\cdot|x) \in \Delta(A)$ over actions by performing a search over the tree of possible game continuations of the current state, using a fixed rule TREESEARCH.
- Samples[5] a next move from $\pi(\cdot|x)$ and advances the board.
- Repeats until it reaches a terminal state.
- Returns an (encoded) tuple with the updated distribution of the first round (that corresponds to the input state x) and a bit that indicates whether the player playing at the input state x ended up winning.

See Algorithm 1 for a pseudocode.

Suppose there exists $\theta' \neq \theta$ such that

$$h_{\theta'}(x) = P_{\mathrm{SP}}(h_\theta)(x) \qquad \forall x \in \mathcal{X}. \tag{16}$$

If we assume that each call to $\mathfrak{h}_\phi$ has the same cost for all $\phi \in \Theta$, then, according to Definition 1, $\mathfrak{h}_{\theta'}$ internalizes $\mathfrak{P}_{\mathrm{SP}}(\mathfrak{h}_\theta)$. Note that the algorithm of AlphaGo Zero specified a loss that tried to do exactly that: find weights θ' so that $\mathfrak{h}_{\theta'}$ would match $\mathfrak{P}_{\mathrm{SP}}(\mathfrak{h}_\theta)$ on all states visited during a game.

[5] Our definition does not formally allow randomized algorithms. In order to accommodate this, we can embed, for example, the random seed ρ as part of the input x.

Algorithm 1 The slow self-play program $\mathfrak{P}_{\mathrm{SP}}(\mathfrak{h}_\theta)$.

Require: A program $\mathfrak{h}_\theta$ and an initial encoded state $x \in \mathcal{X}$. A tree-search program TREESEARCH. A board advancement program BOARD.

$s_1 \leftarrow x$.
$t \leftarrow 1$.
$\hat{\pi} \leftarrow \mathfrak{h}_\theta[0]$.
while s_t is not terminal **do**
  Compute

$$\pi_t(\cdot \mid s_t) \leftarrow \textsc{TreeSearch}(\mathfrak{h}_\theta)(s_t). \tag{12}$$

  if $t = 1$ **then**
    $\hat{\pi} \leftarrow \pi_t(\cdot \mid s_t)$.
  end if
  Sample a next move

$$a_t \sim \pi_t(\cdot \mid s_t). \tag{13}$$

  Advance the game state

$$s_{t+1} \leftarrow \textsc{Board}(s_t, a_t). \tag{14}$$

  $t \leftarrow t + 1$.
end while
Let $s_{\mathrm{final}} \leftarrow s_t$.
Define the terminal outcome from the perspective of the player to move at state x

$$z(x) := \mathbb{1}\left\{\textsc{Winner}(s_{\mathrm{final}}) = \textsc{Player}(x)\right\}. \tag{15}$$

return $\mathtt{enc}_Y\left((\hat{\pi}, z(x))\right)$.

Example B.5 (Internalizing checkers self-play)**.** We explain informally how the learning-via-generalization algorithm of Samuel [1959] fits within our definition of internalization. The parametric model is a linear model with polynomial features $h_\theta(s) = \sum_j \theta_j \phi_j(s)$, where s is a state (or an encoding of the state). The slow system runs a minimax search using this linear model for evaluation at the leaves. For example, for a ply of size 2, the slow system is defined as $P(h_\theta)(x) = \max_{a \in A(s)} \min_{b \in A(s_a)} h_\theta(s_{ab})$, where A returns the set of possible moves for a given state. The linear model is trained to internalize the result of this $\max - \min$ evaluation into its weights.

C Representation of C_n semiautomata with transformers

We present here the proof of our representational result on counter semiautomata.

Definition 3 (Counter $\mathrm{mod}n$ semiautomaton)**.** *Let $n \in \mathbb{N}_+$. We define the $\mathrm{mod}n$ counter semiautomaton C_n as the tuple $C_n = (Q, \Sigma, \delta_{\bmod n})$, where $Q = \Sigma = \mathbb{Z}_n := \{0, 1, \ldots, n-1\}$, and $\delta_{\bmod n}(a, b) := (a + b) \bmod n$ for all $a, b \in \mathbb{Z}_n$.*

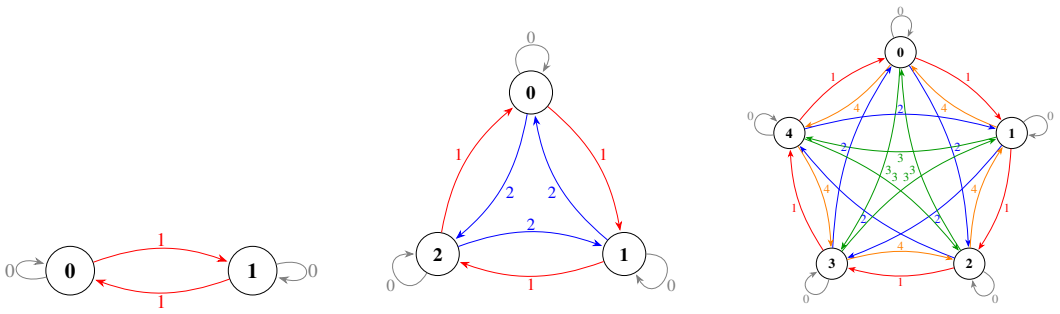
See Figure 5 for an illustration of the counter semiautomata C_2, C_3, C_5.

C.1 Transformer architecture

We consider a decoder-only transformer with L layers and H heads whose outputs are summed together. We are only interested in the last output of the transformer operating on a sequence. We denote the model by $f_{\mathrm{TF}} : \mathbb{R}^{d_{\mathrm{embd}} \times (T+1)} \times \Theta \to \mathbb{R}^{d_{\mathrm{embd}}}$ and it is defined as an alternating composition of Multi-Layer-Perceptron and self-attention layers:

$$f_{\mathrm{TF}} = \pi_{\mathrm{last-token}} \circ f^{(L)}_{\mathrm{MLP}} \circ f^{(L)}_{\mathrm{attn}} \circ \ldots \circ f^{(1)}_{\mathrm{MLP}} \circ f^{(1)}_{\mathrm{attn}}, \tag{17}$$

where $\pi_{\mathrm{last-token}}(X) = X_{:,T+1}$ simply collects the last token's representation. The input $X \in \mathbb{R}^{d_{\mathrm{embd}} \times (T+1)}$ should be understood as an embedding of the input sequence through an embedding layer $E : \Sigma^T \times Q \to \mathbb{R}^{d_{\mathrm{embd}} \times (T+1)}$. The output of the transformer gets decoded through an un-embedding layer $U : \mathbb{R}^{d_{\mathrm{embd}}} \to Q$.


0
1
0
1
0
1
0
1
2
0
1
2
2
2
1
0
0
1
2
3
4
0
1
2
3
4
0
1
2
3
4

We will also make use of the following lemma on approximating modular addition with two-layer MLPs.

Lemma C.2 (Robust ReLU implementation of modular addition)**.** *Fix $N \in \mathbb{N}_+$ and $\alpha \in (0, 1/16]$. There is a scalar one-hidden-layer ReLU network $A_N : \mathbb{R}^2 \to \mathbb{R}$ with width at most $4N + 2$ and weights bounded by $C_N = O(N)$, for a universal implicit constant, such that for all $a, b \in \mathbb{Z}_N$ and all $u, v \in \mathbb{R}$ satisfying*

$$|u - a| \le \alpha, \qquad |v - b| \le \alpha,$$

one has

$$A_N(u, v) = (a + b) \bmod N.$$

Proof. It is enough to construct a continuous piecewise-affine function $g_N : \mathbb{R} \to \mathbb{R}$ such that

$$g_N(x) = s \bmod N \qquad \text{whenever} \qquad |x - s| \le 2\alpha,$$

for every integer $s \in \{0, \dots, 2N - 2\}$; then set $A_N(u, v) = g_N(u + v)$.

We construct g_N by making it constant on each interval $[s - 2\alpha, s + 2\alpha]$, with value $s \bmod N$, and linearly interpolating between consecutive constant intervals. Since $4\alpha \le 1/4$, the interpolation gaps have length at least $3/4$. The largest jump in the target values is at most $N - 1$, so all slopes are $O(N)$. The resulting function has at most $2(2N - 1) \le 4N$ "breakpoints". Every continuous piecewise-affine function with K breakpoints can be written as

$$c_0 + c_1 x + \sum_{r=1}^{K} c_r \max(x - \theta_r, 0),$$

so it is representable by a one-hidden-layer ReLU network of width $K + 2 \le 4N + 2$ (the linear term can be represented by two ReLU neurons). The claimed weight bound follows from the slope bound and the construction. □

C.3 Main result

We present next our result on how the semiautomaton C_n, $n = \prod_{j=1}^{m} p_j^{v_j}$ can be represented by a transformer of $\Theta(\log T)$ layers and $\Theta\left(\sum_{j=1}^{m} p_j^{v_j}\right)$ MLP width.

Theorem C.3 (Simulation of C_n, $n = \prod_{j=1}^{m} p_j^{v_j}$, with $\Theta(\log T)$ depth and $\Theta(\sum_{j=1}^{m} p_j^{v_j})$ width)**.** *Let $n \ge 2$, $q_0 \in \mathbb{Z}_n$ and $T \in \mathbb{N}_+$. Let $n = \prod_{j=1}^{m} p_j^{v_j}$ be its prime factorization, and set $N_j = p_j^{v_j}$ and $N_{\max} = \max_{1 \le j \le m} N_j$. There exists a depth-$\lceil \log_2(T + 1) \rceil$ Transformer f_{TF} which, together with the embedding and the un-embedding layers, implements $\delta^{(T)}_{\bmod n}(\sigma_1, \dots, \sigma_T, q_0) = q_0 + \sum_{t=1}^{T} \sigma_t \pmod n$. The construction has embedding dimension $d_{\mathrm{embd}} = 2m + 2$, MLP width $4 \sum_{j=1}^{m} N_j + 4 + 2m$, $H = 2$ heads each of dimension $m + 2$, and uses a two-layer ReLU MLP. Moreover, all weights have ℓ_∞ norm at most $C \max\{N_{\max}, T\sqrt{\log(TN_{\max})}\}$ for a universal constant $C > 0$.*

Proof. The proof introduces two basic changes from the generic $O(\log T)$-depth construction of Liu et al. [2023]. First, instead of embedding each input symbol $\sigma \in \mathbb{Z}_n$ to its transition map $\delta_{\bmod n}(\cdot, \sigma) : \mathbb{Z}_n \to \mathbb{Z}_n$, we use the prime-power factorization embedding

$$\sigma \mapsto (\sigma \bmod N_1, \dots, \sigma \bmod N_m), \qquad N_j = p_j^{v_j}.$$

By the Chinese Remainder Theorem, $x \bmod n$ is uniquely determined by $(x \bmod N_1, \dots, x \bmod N_m)$. Hence the MLP only needs to implement m parallel modular additions, one modulo each N_j. Second, the initial state q_0 is explicitly fed in the input, hence we require one additional attention layer compared to Liu et al. [2023] general construction.

As in the construction of Liu et al. [2023], we prepend T dummy zero tokens. Thus the transformer is run on the padded sequence

$$y = (y_1, \dots, y_M) := (0, \dots, 0, \sigma_1, \dots, \sigma_T, q_0), \qquad M := 2T + 1.$$

Set $L := \lceil \log_2(T + 1) \rceil$ and $\Delta_M := 1 - \cos(2\pi/M)$. Choose

$$\eta := \min\left\{\frac{1}{16 N_{\max}}, \frac{\Delta_M}{12L}, \frac{1}{16}\right\}, \qquad \beta := \frac{2}{\Delta_M} \log\left(\frac{2M}{\eta}\right).$$

The quantity β is the attention-score scale used in the definition of the query-key matrices.

Embedding. Let $d := 2m + 2$ and $k := m + 2$. For each position $i \in [M]$, define the positional vector

$$p_i := \left(\cos \frac{2\pi i}{M}, \sin \frac{2\pi i}{M}\right)^\top \in \mathbb{R}^2.$$

The initial embedding is

$$X_i^{(0)} := \left(y_i \bmod N_1, \ldots, y_i \bmod N_m, \underbrace{0, \ldots, 0}_{m \text{ auxiliary coordinates}}, p_i^\top\right)^\top \in \mathbb{R}^{2m+2}.$$

Let $r_j^{(\ell)}(i)$ denote the first-block residue coordinate modulo N_j at position i after the ℓth MLP.

We use the following coordinate extraction matrices. For

$$x = (a_1, \ldots, a_m, b_1, \ldots, b_m, z_1, z_2)^\top \in \mathbb{R}^{2m+2},$$

define

$$E_1 x := (a_1, \ldots, a_m)^\top, \qquad E_2 x := (b_1, \ldots, b_m)^\top, \qquad E_p x := (z_1, z_2)^\top.$$

Thus $E_1, E_2 \in \mathbb{R}^{m \times d}$ and $E_p \in \mathbb{R}^{2 \times d}$. Let $I_1 := E_1^\top$, $I_2 := E_2^\top$, and $I_p := E_p^\top$ be the corresponding insertion matrices. Therefore $I_1 u$ inserts $u \in \mathbb{R}^m$ into the first residue block, $I_2 u$ inserts it into the auxiliary residue block, and $I_p v$ inserts $v \in \mathbb{R}^2$ into the final two positional coordinates.

The attention matrices. At layer $\ell \in \{1, \ldots, L\}$, set the stride $s_\ell := 2^{\ell-1}$. Since $L = \lceil \log_2(T + 1) \rceil$, we have $s_\ell \le 2^{L-1} \le T$.

The construction uses two heads as in the construction of Liu et al. [2023]. The first head attends to itself. Its query, key, value, and output matrices are

$$W_Q^{(\ell,1)} := \sqrt{\beta} \begin{pmatrix} 0_{m \times d} \\ E_p \end{pmatrix}, \qquad W_K^{(\ell,1)} := \sqrt{\beta} \begin{pmatrix} 0_{m \times d} \\ E_p \end{pmatrix},$$

and

$$W_V^{(\ell,1)} := \begin{pmatrix} E_1 \\ E_p \end{pmatrix}, \qquad W_C^{(\ell,1)} := \begin{pmatrix} I_1 & I_p \end{pmatrix}.$$

These matrices have dimensions $W_Q^{(\ell,1)}, W_K^{(\ell,1)}, W_V^{(\ell,1)} \in \mathbb{R}^{k \times d}$ and $W_C^{(\ell,1)} \in \mathbb{R}^{d \times k}$. For tokens X_i, X_j, the attention score is

$$X_i^\top (W_Q^{(\ell,1)})^\top W_K^{(\ell,1)} X_j = \beta \, (E_p X_i)^\top (E_p X_j).$$

Moreover,

$$W_C^{(\ell,1)} W_V^{(\ell,1)} X_j = I_1 E_1 X_j + I_p E_p X_j,$$

so the self head copies the first residue block and the positional coordinates.

The second head looks back s_ℓ positions. Let $R_{-s_\ell} \in \mathbb{R}^{2 \times 2}$ be the rotation matrix

$$R_{-s_\ell} := \begin{pmatrix} \cos(2\pi s_\ell / M) & \sin(2\pi s_\ell / M) \\ -\sin(2\pi s_\ell / M) & \cos(2\pi s_\ell / M) \end{pmatrix},$$

so that $R_{-s_\ell} p_i = p_{i - s_\ell}$ with indices being interpreted cyclically. Define

$$W_Q^{(\ell,2)} := \sqrt{\beta} \begin{pmatrix} 0_{m \times d} \\ R_{-s_\ell} E_p \end{pmatrix}, \qquad W_K^{(\ell,2)} := \sqrt{\beta} \begin{pmatrix} 0_{m \times d} \\ E_p \end{pmatrix},$$

and

$$W_V^{(\ell,2)} := \begin{pmatrix} E_1 \\ 0_{2 \times d} \end{pmatrix}, \qquad W_C^{(\ell,2)} := \begin{pmatrix} I_2 & 0_{d \times 2} \end{pmatrix}.$$

Thus

$$X_i^\top (W_Q^{(\ell,2)})^\top W_K^{(\ell,2)} X_j = \beta \, (R_{-s_\ell} E_p X_i)^\top (E_p X_j),$$

and

$$W_C^{(\ell,2)} W_V^{(\ell,2)} X_j = I_2 E_1 X_j.$$

Therefore the second head copies the first residue block of the attended token into the auxiliary residue block.

The ideal hard-attention computation. First replace each causal softmax row by its argmax one-hot vector. We later justify this replacement using Lemma C.1 and robustness of the MLP.

Under hard attention, the first head attends from position i to position i. For every non-dummy position $i \geq T+1$, the second head attends from position i to position $i - s_\ell$. Therefore the summed two-head output at position $i \geq T+1$ is

$$Y_i^{(\ell)} = \left(r_1^{(\ell-1)}(i), \ldots, r_m^{(\ell-1)}(i), r_1^{(\ell-1)}(i-s_\ell), \ldots, r_m^{(\ell-1)}(i-s_\ell), p_i^\top\right)^\top.$$

On the first T dummy positions, the second head may attend arbitrarily when $i - s_\ell < 1$, but this does not affect the residue computation.

The MLP maps

$$(a_1, \ldots, a_m, b_1, \ldots, b_m, z_1, z_2)^\top$$

to

$$\left(A_{N_1}(a_1, b_1), \ldots, A_{N_m}(a_m, b_m), \underbrace{0, \ldots, 0}_{m \text{ auxiliary coordinates}}, z_1, z_2\right)^\top,$$

where each A_{N_j} is the robust modular-adder network from Lemma C.2 with robustness parameter $\alpha = 1/16$. The two identity coordinates z_1, z_2 are implemented by $z = \mathrm{ReLU}(z) - \mathrm{ReLU}(-z)$, using two hidden units per coordinate. Hence the MLP width is at most $4\sum_{j=1}^m N_j + 2m + 4$.

For hard attention, this gives the exact recurrence, for every non-dummy position $i \geq T+1$,

$$r_j^{(\ell)}(i) = r_j^{(\ell-1)}(i) + r_j^{(\ell-1)}(i - s_\ell) \pmod{N_j}.$$

By induction on ℓ,

$$r_j^{(\ell)}(i) = \sum_{u=i-2^\ell+1}^{i} y_u \pmod{N_j}$$

for every non-dummy position $i \geq T+1$, with the convention $y_u = 0$ for $u < 1$. The base case $\ell = 0$ is the embedding. For the induction step, the current term $r_j^{(\ell-1)}(i)$ gives the window $[i - 2^{\ell-1} + 1, i]$. The shifted term is at $k = i - s_\ell = i - 2^{\ell-1}$. If $k \geq T+1$, the induction hypothesis gives the window $[i - 2^\ell + 1, i - 2^{\ell-1}]$. If $k \leq T$, then the shifted position is dummy and its residue is zero; the same shifted window also has zero sum, because all its indices are at most T or below 1. Thus the two windows combine to $[i - 2^\ell + 1, i]$.

At the last position $M = 2T + 1$ and depth L, the window length is $2^L \geq T + 1$. Therefore the window ending at M contains all nonzero semantic tokens $\sigma_1, \ldots, \sigma_T, q_0$ and may contain only dummy zeros before them. Hence

$$r_j^{(L)}(M) = q_0 + \sum_{t=1}^{T} \sigma_t \pmod{N_j}$$

for every $j \in [m]$.

Softmax approximation. It remains to justify replacing hard attention by ordinary causal softmax attention with finite weights. The two heads are analyzed by the same argument.

For the ideal positional vectors, if the target position is $t^\star$, then $p_{t^\star}^\top p_{t^\star} = 1$, while for every other position $j \neq t^\star$,

$$p_{t^\star}^\top p_j \leq \cos\frac{2\pi}{M} = 1 - \Delta_M.$$

Thus the ideal unscaled dot-product gap is Δ_M.

Because the positional coordinates are copied through softmax attention, they are not exact after the first layer (note that this is because we do not use a residual connection in our theoretical construction). However, we prove the following invariant. After layer ℓ, all first-block residue coordinates are exact, all auxiliary residue coordinates are zero, and each positional vector differs from its ideal p_i by at most $\ell\eta$ in Euclidean norm.

The invariant is true at $\ell = 0$. Suppose it holds at layer $\ell - 1$. Since $(\ell - 1)\eta \le L\eta \le \Delta_M/12$, the perturbation of any positional dot product is at most $3L\eta$, and the perturbation of the difference between a target score and a non-target score is at most $6L\eta \le \Delta_M/2$. Therefore, after multiplying by β, every relevant causal row has argmax gap at least

$$\gamma = \frac{\beta \Delta_M}{2}.$$

For the first head this holds for every row. For the second head this holds for every non-dummy row $i \ge T + 1$, whose target is $i - s_\ell$. On dummy rows, all causal residue values are zero, so no selection guarantee is needed.

Applying Lemma C.1 to a causal row of length at most M gives

$$\|\texttt{CausalSoftmax}(A)_{i,1:i} - e_{t^\star}\|_1 \le 2Me^{-\beta\Delta_M/2} \le \eta$$

by the definition of β.

Consequently, every copied residue coordinate is within $\eta N_{\max} \le 1/16$ of the corresponding exact residue. The MLP modular adders from Lemma C.2 therefore output the exact modular sums. This proves exactness of the first residue block and zeroing of the auxiliary residue block at layer ℓ.

For positional coordinates, the first head outputs an η-accurate convex combination around the target position. More explicitly, if the previous positional error was at most $(\ell - 1)\eta$, then

$$\left\| \sum_{j \le i} w_j E_p X_j^{(\ell-1)} - p_i \right\|_2 \le (\ell - 1)\eta + \eta = \ell\eta,$$

where w is the first head's attention row and $\|w - e_i\|_1 \le \eta$. The MLP passes the two positional coordinates through the identity. Thus, the invariant holds at layer ℓ.

By induction, the finite-softmax transformer has the same exact residue coordinates as the hard-attention transformer after every MLP layer.

Decoding. At the last token, the first m coordinates are exactly

$$\left(q_0 + \sum_{t=1}^{T} \sigma_t \bmod N_1, \ldots, q_0 + \sum_{t=1}^{T} \sigma_t \bmod N_m \right)^\top.$$

The unembedding U applies the inverse Chinese remainder map, yielding the unique element of $\mathbb{Z}_n$ with these residues. Therefore

$$U\, f_{\mathrm{TF}}(E(\sigma_1, \ldots, \sigma_T, q_0)) = q_0 + \sum_{t=1}^{T} \sigma_t \pmod n.$$

Finally, the nonzero entries in the attention matrices are bounded by $\sqrt{\beta}$ or 1, and the MLP weights are bounded by $C_{\mathrm{MLP}} N_{\max}$ for a universal constant C_{MLP}. Since $\Delta_M^{-1} = O(T^2)$ and $\eta^{-1} = O(T^2 L N_{\max})$, we have $\sqrt{\beta} = O(T\sqrt{\log(TN_{\max})})$. Hence all weights have ℓ_∞ norm at most $C \max\{N_{\max}, T\sqrt{\log(TN_{\max})}\}$ for a universal constant $C > 0$. This completes the proof. □

D Learning Parity in 1-Layer Transformer with Linear Attention

We first start by specifying the architecture, curriculum, and details of the algorithm, and we then proceed to the proof.

D.1 Architecture, Left Removal Curriculum, and Algorithm

Causal linear attention architecture. Let D be the maximum sequence length, which will be $d + k$ for our setup in internalizing parities. Our architecture, as specified below, is parameterized by a single attention head score matrix $W \in \mathbb{R}^{D \times D}$. We will fix other components of the architecture such as MLP weights, and value matrix. Each of the D tokens is represented in dimension $D + 1$. The

first coordinate is a scalar content coordinate, and the remaining D coordinates are one-hot positional encodings.

Given a content vector $a = (a_1, \ldots, a_D) \in \mathbb{R}^D$, we form the token matrix

$$X(a) = \begin{pmatrix} a_1 & 1 & 0 & \cdots & 0 \\ a_2 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_D & 0 & 0 & \cdots & 1 \end{pmatrix} \in \mathbb{R}^{D \times (D+1)}.$$

Equivalently, the i-th token is

$$X_i(a) = (a_i, e_i) \in \mathbb{R}^{1+D},$$

where $e_i \in \mathbb{R}^D$ is the i-th standard basis vector.

We use a single attention head without softmax normalization. The query-key scores are bilinear:

$$S = XW_{QK}X^\top.$$

Here $W_{QK} \in \mathbb{R}^{(D+1)\times(D+1)}$. Equivalently, if $Q = XW_Q$ and $K = XW_K$, then $QK^\top = XW_QW_K^\top X^\top$, and we collapse the parameterization $W_{QK} := W_QW_K^\top$ into a single parameter. We always fix the value matrix to be the identity,

$$W_V = I_{D+1}.$$

We constrain W_{QK} to ignore the scalar content coordinate and to be causal in the positional block. Namely, we write

$$W_{QK} = \begin{pmatrix} 0 & 0 \\ 0 & W \end{pmatrix}, \quad \text{where} \quad W \in \mathbb{R}^{D \times D},$$

is an unconstrained trainable matrix. The first row and column of W_{QK} are zero, so the bilinear query-key scores depend only on the positional coordinates. For tokens $i, j \in [D]$, this gives

$$X_i(a)^\top W_{QK} X_j(a) = (a_i, e_i)^\top \begin{pmatrix} 0 & 0 \\ 0 & W \end{pmatrix} (a_j, e_j) = e_i^\top W e_j = W_{ij}.$$

Hence

$$X(a)W_{QK}X(a)^\top = W.$$

Causality is enforced by applying the lower-triangular mask

$$M \in \{0,1\}^{D\times D}, \qquad M_{ij} = \begin{cases} 1, & j \le i, \\ 0, & j > i. \end{cases}$$

Thus token i may attend only to tokens at positions $j \le i$, including itself. The masked score matrix is

$$M \odot X(a)W_{QK}X(a)^\top = M \odot W,$$

where $\odot$ denotes entrywise multiplication. Since we fix the value matrix to be the identity, the causal linear-attention output is

$$H(X(a)) = \left(M \odot X(a)W_{QK}X(a)^\top\right) X(a) = (M \odot W)X(a).$$

Output of Attention on shorter input length The same construction is consistent under restriction to shorter sequences. If we use the same input representation but restrict the sequence to the first $D' \le D$ tokens, then the restricted token matrix is

$$X_{[D']}(a) = X(a)_{1:D',:} \in \mathbb{R}^{D'\times(D+1)}.$$

The bilinear score matrix becomes

$$X_{[D']}(a)W_{QK}X_{[D']}(a)^\top = W_{[D']\times[D']},$$

the upper-left $D' \times D'$ restriction of W. After applying the $D' \times D'$ causal mask $M^{(D')}$, defined by $M^{(D')}_{ij} = \mathbf{1}\{j \le i\}$ the corresponding causal attention output is

$$H_{[D']}(X(a)) = \left(M^{(D')} \odot W_{[D']\times[D']}\right) X_{[D']}(a).$$

Thus restricting the input to its first D' tokens simply restricts the positional score matrix W to its first D' rows and columns.

Final output with fixed MLP For each output position $i \in [D]$, we define the scalar prediction

$$\widehat{y}_i := f_W(a)_i = \sigma((H_i(X(a)))_1)\,.$$

This can be viewed as applying a fixed MLP/readout to $H_i(X(a))_1 \in \mathbb{R}$, followed by the scalar nonlinearity $\sigma : \mathbb{R} \to \mathbb{R}$. Equivalently, the vector of predictions is

$$\widehat{y} = f_W(a) = (\sigma((H_1(X(a)))_1)\,, \ldots, \sigma((H_D(X(a)))_1)) \in \mathbb{R}^D.$$

For any position i,

$$H_i(X(a)) = \sum_{j \le i} W_{ij} X_j(a), \quad \text{and hence} \quad (H_i(X(a)))_1 = \sum_{j \le i} W_{ij} a_j\,.$$

D.2 Curriculum for Internalizing the Input

Curriculum for learning parities We now describe how the above architecture is used for learning parities. The input is $x = (x_1, \ldots, x_d) \in \{0,1\}^d$, and the CoT initially consists of k binary tokens $z = (z_1, \ldots, z_k) \in \{0,1\}^k$. Thus the full content vector has length $D = d + k$, and is given by

$$a^{[1]} = (x_1, \ldots, x_d, z_1, \ldots, z_k) \in \{0,1\}^{d+k}.$$

We use the CoT tokens as intermediate supervision. In the full sequence, the token z_i appears at position $d + i$. As to be specified, the prediction at a position is computed from the tokens up to that position. We train the model to predict the CoT entries $z_1, \ldots, z_k$ at the corresponding positions $d + 1, \ldots, d + k$, in the first phase. The model learns to compute parity, by *explicitly* outputting the entire CoT, in the first phase. Equivalently, for each $i \in [k]$, the prediction at position $d + i$ is trained against the target z_i.

Phases of curriculum The curriculum proceeds by progressively shortening the CoT. In the first phase, the model is trained on the full sequence $(x_1, \ldots, x_d, z_1, \ldots, z_k)$. In the next phase, we remove the first CoT token and train on $(x_1, \ldots, x_d, z_2, \ldots, z_k)$. In Phase t for $1 \le t \le k$: we remove the first $t - 1$ CoT tokens and train on the shortened sequence

$$\text{Input Sequence in Phase } t: \qquad a^{[t]} = (x_1, \ldots, x_d, z_t, z_{t+1}, \ldots, z_k).$$

At the end of k phases the input sequence only contains $(x_1, \ldots, x_d)$. The desired goal is that, by the end of such a curriculum, the model learns to directly produce z_k using our architecture.

Output of the Architecture In the first phase ($t = 1$), the content vector is

$$a^{[1]} = (x_1, \ldots, x_d, z_1, \ldots, z_k) \in \{0,1\}^{d+k}.$$

For every CoT position $d + i$, where $0 \le i \le k$, the output of the architecture

$$\widehat{y}^{[1]}_{d+i} = f_W(a^{[1]})_{d+i} = \sigma\left(\sum_{j \le d+i} W_{d+i,j} a^{[1]}_j\right).$$

In Phase t, similarly, the content vector has CoT tokens removed and is

$$a^{[t]} = (x_1, \ldots, x_d, z_t, z_{t+1}, \ldots, z_k) \in \{0,1\}^{d+k-t+1}. \tag{21}$$

Since the first CoT token in phase t is now z_t, the output for $0 \le i \le k - t + 1$ is

$$\widehat{y}^{[t]}_{d+i} = f_W(a^{[t]})_{d+i} = \sigma\left(\sum_{j \le d+i} W_{d+i,j} a^{[t]}_j\right). \tag{22}$$

At the end of final (k^{th}) phase, after all the CoT tokens have been removed, the content vector and the output at d^{th} position are respectively given by

$$a^{[k+1]} = (x_1, \ldots, x_d) \quad \text{and} \quad \widehat{y}^{[k+1]}_d = \sigma\left(\sum_{j=1}^{d} W_{d,j} x_j\right).$$

Fixed non-linear activation We use a fixed piecewise-linear activation $\sigma : \mathbb{R} \to \mathbb{R}$ that agrees with parity on the integers: $\sigma(m) = m \mod 2$ for $m \in \mathbb{Z}$. Between consecutive integers, σ is defined by linear interpolation. Equivalently, for $s \in [m, m+1]$,

$$\sigma(s) = \begin{cases} s - m, & m \text{ even}, \\ 1 - (s - m), & m \text{ odd}. \end{cases} \tag{23}$$

At integer breakpoints, σ is not differentiable. We fix the left-derivative convention:

$$\sigma'(m) = \lim_{\epsilon \downarrow 0} \frac{\sigma(m) - \sigma(m - \epsilon)}{\epsilon} = \begin{cases} -1, & m \text{ even}, \\ +1, & m \text{ odd}. \end{cases} \tag{24}$$

Away from integer points, σ' is the usual derivative.

Algorithm 2 INTERNALIZE-PARITY via CoT Shortening

Require: Input dimension d, CoT length k, step size η, per-phase number of steps T, batch size B
Require: Training distribution over $(x, z_1, \ldots, z_k) \in \{0,1\}^d \times \{0,1\}^k$

Initialize $W^{(0)} \leftarrow 0^{(d+k)\times(d+k)}$ and then reset the rows for phase-one next-token CoT prediction:

$$W^{(0)}_{d+l-1,d+l-1} \leftarrow -1 \qquad \text{for all } l \in \{2, \ldots, k\}.$$

for $t = 1, \ldots, k$ **do**
  Sample a phase-t batch $\mathcal{B}_t = \{(x^{(b)}, z_1^{(b)}, \ldots, z_k^{(b)})\}_{b=1}^B$
  For each $b \in [B]$, form the shortened phase-t input

$$a^{[t,b]} = (x_1^{(b)}, \ldots, x_d^{(b)}, z_t^{(b)}, \ldots, z_k^{(b)}).$$

  if $t = 1$ **then**
    The empirical loss in Phase-1 is next-token CoT loss

$$\widehat{L}_1(W) = \frac{1}{2B} \sum_{b=1}^{B} \sum_{l=1}^{k} \left(f_W(a^{[1,b]})_{d+l-1} - a^{[1,b]}_{d+l} \right)^2.$$

  else
    The empirical loss is only on d^{th} token

$$\widehat{L}_t(W) = \frac{1}{2B} \sum_{b=1}^{B} \left(f_W(a^{[t,b]})_d - a^{[t,b]}_{d+1} \right)^2 = \frac{1}{2B} \sum_{b=1}^{B} \left(f_W(a^{[t,b]})_d - z_t^{(b)} \right)^2.$$

  end if
  Make T updates starting $W \leftarrow W^{(t-1)}$ on empirical Phase-t loss $\widehat{L_t}(\cdot)$.
  for $s = 1, \ldots, T$ **do**
    Update the row corresponding to the first CoT/output position:

$$W \leftarrow W - \eta \nabla_W \widehat{L}_t(W)$$

  end for
  Let $W^{(t)}$ be the final iterate of Phase t, and round entries to nearest integers $W^{(t)} \leftarrow \texttt{Round}(W^{(t)})$.
end for
return $W^{(k)}$

D.3 Analysis

We prove Theorem 4.1 for Algorithm 2. Throughout, the support coordinates $i_1, \ldots, i_k$ are distinct, $1 \le k \le d$, and $x \sim \text{Unif}(\{0,1\}^d)$. Write

$$z_0(x) = 0, \qquad z_t(x) = \bigoplus_{h=1}^{t} x_{i_h}, \quad t \in [k].$$

The activation σ is the piecewise-linear interpolation of parity on $\mathbb{Z}$. At an integer m, gradients use its left derivative,

$$\sigma'_-(m) = 2\sigma(m) - 1 \in \{-1, 1\}. \tag{25}$$

At all other arguments, they use its usual derivative. We use the loss normalization in the algorithm: each supervised row contributes a batch average with factor $1/(2B)$, and Phase 1 sums these contributions without an additional average over rows.

Recall that the output at position j is

$$f_W(a)_j = \sigma\left(\sum_{h \le j} W_{jh} a_h\right). \tag{26}$$

Consequently, the supervised rows have disjoint trainable parameters. Updating all of W is therefore equivalent to updating each supervised row on its own loss; unsupervised rows and causally masked entries have zero gradient. Let $W^{(t,s)}$ denote the iterate after s updates in Phase t, with $W^{(t,0)} = W^{(t-1)}$, and let $W^{(t)}$ be the result of rounding $W^{(t,T)}$ coordinatewise to the nearest integer.

For row d, the available features are x in every phase. For the row predicting z_l in Phase 1, the available features are

$$q_l(x) = (x_1, \ldots, x_d, z_1, \ldots, z_{l-1}) \in \{0,1\}^{d+l-1}, \qquad l \in [k], \tag{27}$$

where $q_1(x) = x$. We first analyze a common row-learning problem for these feature vectors, and then apply it to all supervised rows and phases.

Quadratic regions of the squared parity loss. The squared loss remains quadratic on an interval centered at the target bit, including across a breakpoint where the prediction matches that bit. This observation allows us to control the actual gradient-descent trajectory.

Lemma D.1 (Quadratic loss identity)**.** *For $m \in \mathbb{Z}$, $p \in \{0,1\}$, and $|v - p| \le 1$,*

$$g_{m,p}(v) := \frac{1}{2}\big(\sigma(m - v) - \sigma(m - p)\big)^2 = \frac{1}{2}(v - p)^2. \tag{28}$$

With the prescribed derivative convention for σ, the chain-rule derivative equals $v - p$ for $|v - p| < 1$, and equals $-p$ at $v = 0$.

Proof. For every integer n and $|h| \le 1$, the definition of σ gives

$$|\sigma(n + h) - \sigma(n)| = |h|.$$

Taking $n = m - p$ and $h = p - v$, then squaring, proves the identity. Differentiation gives the interior derivative. At $v = p$, the residual vanishes, so the prescribed chain rule also gives zero.

At $v = 0$, we have $\sigma(m) - \sigma(m - p) = p\,\sigma'_-(m)$. Thus the prescribed chain-rule derivative is

$$-\big(\sigma(m) - \sigma(m - p)\big)\sigma'_-(m) = -p\big(\sigma'_-(m)\big)^2 = -p.$$

□

Second moments of the available features. We first establish the matrix bounds needed to analyze the regression updates. Recall that row d uses $q_1(x) = x$ in every curriculum phase. The additional rows trained in Phase 1 use

$$q_l(x) = (x_1, \ldots, x_d, z_1, \ldots, z_{l-1}), \qquad l = 2, \ldots, k.$$

Thus l specifies which CoT token is being predicted in Phase 1; it does not denote the curriculum phase.

Fix one of these feature vectors $q = q_l$, and let $N = d + l - 1$. Its population and empirical second moments are

$$\Sigma_q := \mathbb{E}[q(x)q(x)^\top], \qquad \widehat{\Sigma}_q := \frac{1}{B}\sum_{b=1}^{B} q(x^{(b)})q(x^{(b)})^\top,$$

where the training examples are independent and uniform on $\{0,1\}^d$.

For $l \geq 2$, the only repeated feature is $z_1 = x_{i_1}$. To calculate the second moment, let

$$\phi(x) = \begin{cases} x, & l = 1, \\ (x_1, \ldots, x_d, z_2, \ldots, z_{l-1}), & l \geq 2, \end{cases}$$

and write $n = \dim(\phi)$. Let $R \in \{0,1\}^{N \times n}$ restore the removed copy: it inserts $\phi_{i_1} = x_{i_1}$ at position $d+1$, leaving the other entries in their original order. For $l = 1$, take $R = I_d$. In both cases, $q = R\phi$.

The repeated features receive identical gradient updates. We therefore need a positive eigenvalue bound only on the subspace containing these updates:

$$U := \operatorname{range}(R) = \begin{cases} \mathbb{R}^d, & l = 1, \\ \{v \in \mathbb{R}^N : v_{i_1} = v_{d+1}\}, & l \geq 2. \end{cases}$$

Every sample vector belongs to U, and every gradient update is a linear combination of sample vectors. The following lemma bounds the population second moment on U and its empirical approximation.

Lemma D.2 (Feature geometry and concentration)**.** *For the feature maps above,*

$$\Sigma_q = \frac{1}{4} R(I_n + \mathbf{1}\mathbf{1}^\top)R^\top. \tag{29}$$

Both Σ_q and $\widehat{\Sigma}_q$ annihilate $U^\perp$ and map into U. Moreover,

$$\lambda_{\min}(\Sigma_q|_U) \geq \frac{1}{4}, \qquad \|\Sigma_q\|_{\mathrm{op}} \leq N, \qquad \|\widehat{\Sigma}_q\|_{\mathrm{op}} \leq N.$$

For every $\varepsilon > 0$,

$$\Pr\left[\|\widehat{\Sigma}_q - \Sigma_q\|_{\mathrm{op}} > \frac{\varepsilon}{\sqrt{N}}\right] \leq 2N^2 \exp\left(-\frac{2B\varepsilon^2}{N^3}\right). \tag{30}$$

Proof. The entries of ϕ are parities on distinct nonempty supports. Under the uniform input distribution, each is a fair bit and any two are independent. Consequently,

$$\mathbb{E}[\phi\phi^\top] = \frac{1}{4}(I_n + \mathbf{1}\mathbf{1}^\top).$$

Since $q = R\phi$, this proves (29).

Every sample vector belongs to U, so both second-moment matrices map into U and vanish on $U^\perp$. Moreover,

$$\Sigma_q \succeq \frac{1}{4} RR^\top.$$

The nonzero eigenvalues of $RR^\top$ are those of $R^\top R$. This latter matrix is diagonal, with an entry equal to the number of copies of each feature: one for every feature, except two for the duplicated feature when $l \geq 2$. Thus every nonzero eigenvalue is at least one, giving $\lambda_{\min}(\Sigma_q|_U) \geq 1/4$. The upper bounds follow from $\|q(x)\|_2^2 \leq N$.

For concentration, fix i, j and set $Y_b = q_i(x^{(b)})q_j(x^{(b)})$. These variables are independent across b and belong to $[0,1]$. The scalar two-sided Hoeffding inequality gives

$$\Pr\Big[|(\widehat{\Sigma}_q - \Sigma_q)_{ij}| > t\Big] \leq 2e^{-2Bt^2}.$$

Taking a union bound over the N^2 entries and setting $t = \varepsilon/N^{3/2}$ yields

$$\Pr\left[\max_{i,j} |(\widehat{\Sigma}_q - \Sigma_q)_{ij}| > \frac{\varepsilon}{N^{3/2}}\right] \leq 2N^2 \exp\left(-\frac{2B\varepsilon^2}{N^3}\right).$$

Finally, $\|A\|_{\mathrm{op}} \leq N \max_{i,j} |A_{ij}|$ for every $N \times N$ matrix A, proving (30). □

Population and empirical regression trajectories. We now study the regression trajectories used to analyze the addition of one new input bit. We fix one row-learning task:

$$(q, r) = \begin{cases} (x, i_t), & \text{for row } d \text{ in curriculum Phase } t, \\ (q_l, i_l), & \text{for the row predicting } z_l \text{ in Phase } 1, \quad l \geq 2. \end{cases}$$

Thus x_r is the new input bit to be learned, and q is the vector available to that row. The first d entries of q are the raw inputs, so $q_r(x) = x_r$. Write $N = \dim(q)$.

In both cases, x_r occurs only once in q. For $q = x$, this is immediate. For $q = q_l$, the only repeated feature is $x_{i_1} = z_1$, whereas the new bit is x_{i_l} with $i_l \neq i_1$. Consequently, $e_r \in U$: it either belongs to the full space $U = \mathbb{R}^d$, or has zero entries at both duplicate positions.

The desired regression coefficient vector is e_r, since $\langle e_r, q(x) \rangle = x_r$. We compare two auxiliary sequences, both initialized at zero. The population sequence uses the population second moment:

$$u_{s+1} = u_s - \eta \Sigma_q (u_s - e_r), \qquad u_0 = 0. \tag{31}$$

The empirical sequence uses the second moment of the fixed training batch:

$$\widehat{u}_{s+1} = \widehat{u}_s - \eta \widehat{\Sigma}_q (\widehat{u}_s - e_r), \qquad \widehat{u}_0 = 0. \tag{32}$$

These are the population and empirical gradient-descent updates for predicting x_r by the linear prediction $\langle u, q(x) \rangle$.

Our goal is to bound the prediction error

$$\langle u, q(x) \rangle - x_r = \langle u - e_r, q(x) \rangle.$$

An error strictly smaller than one allows us to apply the quadratic-loss identity. We first prove this bound for the population sequence, then show that the empirical sequence stays close enough to preserve it.

Lemma D.3 (Population prediction margin)**.** *For either choice of (q, r) above, suppose $0 < \eta \leq 1/N$. Then, for every $s \geq 0$,*

$$\|u_s - e_r\|_2 \leq 1, \qquad |\langle u_s - e_r, q(x) \rangle| \leq (1 - \eta/4)^s \quad \textit{for every } x \in \{0, 1\}^d. \tag{33}$$

Proof. By Theorem D.2, the eigenvalues of $\Sigma_q|_U$ belong to $[1/4, N]$. Since $u_0 - e_r = -e_r \in U$,

$$u_{s+1} - e_r = (I - \eta \Sigma_q)(u_s - e_r)$$

is a contraction on U. This proves the norm bound.

For the prediction bound, recall that $q = R\phi$, where ϕ lists the distinct features. Define

$$b_s := R^\top u_s, \qquad A_s := \sum_j b_{s,j}.$$

The vector b_s gives the total coefficient on each distinct feature: when a feature appears twice, its coefficient in b_s is the sum of the two coefficients in u_s. In particular,

$$\langle u_s, q(x) \rangle = b_s^\top \phi(x).$$

The first d entries of ϕ are still $x_1, \ldots, x_d$, so the target bit x_r also has index r in ϕ.

Let μ_j be the number of copies of ϕ_j in q, and write

$$\Lambda := R^\top R = \operatorname{diag}(\mu_1, \ldots, \mu_n).$$

Each μ_j is one or two. The target occurs once, so $\mu_r = 1$, and $\sum_j \mu_j = N$. The population recursion and the second-moment formula give

$$b_{s+1} = b_s - \frac{\eta}{4} \Lambda \big(b_s - e_r + (A_s - 1)\mathbf{1} \big). \tag{34}$$

Here e_r in this equation is the rth basis vector in $\mathbb{R}^n$.

We first show that the coefficients remain nonnegative and their sum remains at most one. Both properties hold at $s = 0$. If they hold at step s, then

$$b_{s+1,j} = (1 - \eta\mu_j/4)b_{s,j} + \frac{\eta\mu_j}{4}\big(1 - A_s + \mathbf{1}\{j = r\}\big) \geq 0.$$

Also, summing the updates gives

$$1 - A_{s+1} = \left(1 - \frac{\eta(N+1)}{4}\right)(1 - A_s) + \frac{\eta}{4}\sum_j (\mu_j - 1)b_{s,j} \geq 0.$$

The coefficients in these expressions are nonnegative because $\eta \leq 1/N$. Induction therefore gives $b_{s,j} \geq 0, A_s \leq 1$ for every s. The coefficient on the target bit satisfies

$$b_{s+1,r} = (1 - \eta/4)b_{s,r} + \frac{\eta}{4}(2 - A_s) \geq (1 - \eta/4)b_{s,r} + \eta/4.$$

Since $b_{0,r} = 0$, it follows that

$$b_{s,r} \geq 1 - (1 - \eta/4)^s.$$

These coefficient bounds imply the prediction bound on every input. If $x_r = 0$, then

$$0 \leq b_s^\top \phi(x) \leq A_s - b_{s,r} \leq 1 - b_{s,r}.$$

If $x_r = 1$, then

$$b_{s,r} \leq b_s^\top \phi(x) \leq A_s \leq 1.$$

In either case,

$$|\langle u_s, q(x)\rangle - x_r| \leq 1 - b_{s,r} \leq (1 - \eta/4)^s,$$

as required. □

The population prediction error is therefore strictly below one after every positive number of steps. We next transfer this property to the empirical sequence. Since the batch is reused, all empirical updates depend on the same matrix $\widehat{\Sigma}_q$. It suffices to control this matrix once.

Fix $\varepsilon_0 = 1/128$, and define

$$\mathcal{E}(q) := \left\{ \|\widehat{\Sigma}_q - \Sigma_q\|_{\mathrm{op}} \leq \frac{\varepsilon_0}{\sqrt{N}} \right\}. \tag{35}$$

Lemma D.4 (Empirical prediction margin and convergence)**.** *For either choice of (q, r) above, suppose $0 < \eta \leq 1/N$ and $\mathcal{E}(q)$ holds. Then, for every $s \geq 0$,*

$$\|\widehat{u}_s - u_s\|_2 \leq \frac{\varepsilon_0}{\sqrt{N}} \min\{s\eta, 8\}. \tag{36}$$

Consequently, for every $s \geq 1$ and every input x,

$$|\langle \widehat{u}_s - e_r, q(x)\rangle| < 1. \tag{37}$$

Moreover, for every $T \geq 0$,

$$\|\widehat{u}_T - e_r\|_2 \leq (1 - \eta/8)^T. \tag{38}$$

Proof. On $\mathcal{E}(q)$, the minimum eigenvalue of $\widehat{\Sigma}_q|_U$ is at least

$$\frac{1}{4} - \frac{\varepsilon_0}{\sqrt{N}} \geq \frac{1}{8}.$$

Its maximum eigenvalue is at most N by Theorem D.2. Hence

$$\|(I - \eta\widehat{\Sigma}_q)|_U\|_{\mathrm{op}} \leq 1 - \eta/8.$$

To compare the two sequences, set $\Delta_s := \widehat{u}_s - u_s$. Subtracting their recursions gives

$$\Delta_{s+1} = (I - \eta\widehat{\Sigma}_q)\Delta_s - \eta(\widehat{\Sigma}_q - \Sigma_q)(u_s - e_r).$$

All vectors in this recursion belong to U. Using $\|u_s - e_r\|_2 \le 1$ and $\Delta_0 = 0$, we obtain

$$\|\Delta_s\|_2 \le \frac{\varepsilon_0}{\sqrt{N}}\,\eta \sum_{j=0}^{s-1}(1-\eta/8)^j \le \frac{\varepsilon_0}{\sqrt{N}}\min\{s\eta, 8\}.$$

This proves (36).

Since $\|q(x)\|_2 \le \sqrt{N}$, the population prediction bound gives

$$\begin{aligned}|\langle \widehat{u}_s - e_r, q(x)\rangle| &\le |\langle u_s - e_r, q(x)\rangle| + \|\Delta_s\|_2\|q(x)\|_2 \\ &\le e^{-s\eta/4} + \varepsilon_0 \min\{s\eta, 8\}.\end{aligned}$$

This is strictly less than one for every $s \ge 1$. Indeed, writing $t = s\eta$, if $0 < t \le 1$, then

$$1 - e^{-t/4} \ge t/8 > \varepsilon_0 t.$$

If $t \ge 1$, then

$$1 - e^{-t/4} \ge 1 - e^{-1/4} > 1/8 > 8\varepsilon_0.$$

Thus (37) holds for every input.

Finally, the empirical error itself satisfies

$$\widehat{u}_{s+1} - e_r = (I - \eta\widehat{\Sigma}_q)(\widehat{u}_s - e_r).$$

Since $\widehat{u}_0 - e_r = -e_r \in U$ has norm one, contraction on U gives (38). □

Exact recovery of one row. We now identify actual gradient descent on the parity loss with the auxiliary empirical regression trajectory. The next lemma applies to any integer-valued initial row, provided its target is obtained by adding the parity contribution of one uniquely occurring feature.

Lemma D.5 (One-row recovery)**.** *Fix (q, r) as in the preceding regression lemmas, and let $N = \dim(q)$. Fix $w_0 \in \mathbb{Z}^N$, and suppose the target is*

$$y(x) = \sigma(\langle w_0, q(x)\rangle - x_r)\,.$$

Run gradient descent from w_0 on the fixed-batch loss

$$\widehat{\ell}(w) = \frac{1}{2B}\sum_{b=1}^{B}\big(\sigma(\langle w, q(x^{(b)})\rangle) - y(x^{(b)})\big)^2$$

using $0 < \eta \le 1/N$ and the specified breakpoint convention. On $\mathcal{E}(q)$, its iterates satisfy

$$w_s = w_0 - \widehat{u}_s \qquad (s \ge 0), \tag{39}$$

where $\widehat{u}_s$ is defined by (32)*. In particular, if $(1-\eta/8)^T < 1/2$, then*

$$\mathsf{Round}(w_T) = w_0 - e_r. \tag{40}$$

Proof. For a batch example, set $m_b = \langle w_0, q(x^{(b)})\rangle \in \mathbb{Z}$ and $p_b = q_r(x^{(b)}) \in \{0, 1\}$. Suppose $w_s = w_0 - \widehat{u}_s$. Its scalar prediction on this example is $\sigma(m_b - \langle \widehat{u}_s, q(x^{(b)})\rangle)$, and its target is $\sigma(m_b - p_b)$.

At $s = 0$, Theorem D.1 gives the regression gradient under the stipulated breakpoint convention. At every $s \ge 1$, Theorem D.4 gives

$$\Big|\langle \widehat{u}_s, q(x^{(b)})\rangle - p_b\Big| < 1.$$

Thus Theorem D.1 again identifies the selected gradient with the quadratic regression gradient. Applying the chain rule for $w = w_0 - u$ yields

$$\nabla_w \widehat{\ell}(w_s) = -\frac{1}{B}\sum_{b=1}^{B}\big(\langle \widehat{u}_s, q(x^{(b)})\rangle - p_b\big)q(x^{(b)}) = -\widehat{\Sigma}_q(\widehat{u}_s - e_r).$$

Consequently,

$$w_{s+1} = w_s + \eta\widehat{\Sigma}_q(\widehat{u}_s - e_r) = w_0 - \widehat{u}_{s+1}.$$

Starting from $w_0 = w_0 - \widehat{u}_0$, induction proves (39).

By Theorem D.4,

$$\|w_T - (w_0 - e_r)\|_\infty = \|\widehat{u}_T - e_r\|_\infty \le \|\widehat{u}_T - e_r\|_2 \le (1-\eta/8)^T.$$

If this is less than $1/2$, nearest-integer rounding recovers the integer vector $w_0 - e_r$ exactly. □

Proof of the main theorem.

Proof of Theorem 4.1. We may take $c = 1/4$, so $\eta = 1/(4d)$, and choose a sufficiently large universal constant C. Since every available feature vector has dimension $N \le 2d$, this step size satisfies $\eta \le 1/N$. For $C \ge 64$ and $T = \lceil Cd \rceil$, we also have

$$(1 - \eta/8)^T \le \exp\left(-\frac{T}{32d}\right) \le e^{-2} < \frac{1}{2}. \tag{41}$$

First we construct one event that covers every row-learning task. For Phase 1, require $\mathcal{E}(q_l)$ on its batch for all $l \in [k]$. For each Phase $t \ge 2$, require $\mathcal{E}(q_1)$ on that phase's batch. There are $2k - 1$ row–batch pairs. Denote the intersection of their events by $\mathcal{E}_{\text{all}}$. By Theorem D.2, $N \le 2d$, and $\varepsilon_0 = 1/128$, a union bound gives

$$\Pr[\mathcal{E}_{\text{all}}^c] \le 16kd^2 \exp\left(-\frac{B}{65536d^3}\right). \tag{42}$$

No independence between the events for different rows on the same batch is needed. Under the theorem's batch-size assumption, any fixed $C \ge 2^{17}$ makes the right-hand side at most

$$16kd^2 \left(\frac{\delta}{kT}\right)^2 \le \frac{16\delta^2}{kC^2} \le \delta.$$

Hence $\Pr[\mathcal{E}_{\text{all}}] \ge 1 - \delta$. All the following statements are deterministic on this event.

Phase-1 learning and autoregressive CoT computation. The row predicting z_1 is row d. Its feature vector is $q_1 = x$, its initial weights are zero, and $z_1 = \sigma(-x_{i_1})$. By Theorem D.5 and (41), rounding at the end of Phase 1 gives

$$W^{(1)}_{d,1:d} = -e_{i_1}.$$

For $l \ge 2$, consider row $d + l - 1$. Its feature vector is q_l, and its initialized coefficient on the last feature z_{l-1} is -1, with all other coefficients zero. Its initial logit is therefore $-z_{l-1}(x)$. Since

$$z_l(x) = \sigma(-z_{l-1}(x) - x_{i_l}),$$

this is another instance of Theorem D.5, with target feature x_{i_l}. This feature occurs once: the only possible duplicate in q_l is $z_1 = x_{i_1}$, and $i_l \ne i_1$. Consequently the rounded weights satisfy

$$W^{(1)}_{d+l-1,1:d} = -e_{i_l}, \qquad W^{(1)}_{d+l-1,d+l-1} = -1, \qquad 2 \le l \le k, \tag{43}$$

with every other coefficient in these rows equal to zero. All causal coefficients were allowed to vary during training; the initialized CoT coefficient is recovered by rounding. Masked entries and unsupervised rows remain at their initialized integer values.

Define the autoregressive rollout recursively by

$$\widehat{z}_l(x) = f_{W^{(1)}}(x, \widehat{z}_1(x), \ldots, \widehat{z}_{l-1}(x))_{d+l-1}, \qquad l \in [k],$$

with an empty generated prefix for $l = 1$. The recovered first row gives $\widehat{z}_1(x) = \sigma(-x_{i_1}) = z_1(x)$. If $\widehat{z}_{l-1}(x) = z_{l-1}(x)$, then (43) gives

$$\widehat{z}_l(x) = \sigma(-x_{i_l} - z_{l-1}(x)) = z_l(x).$$

Induction proves exact generation of every CoT token for every Boolean input, as asserted in the first claim of the theorem.

Internalization across phases. We show that the first prediction row satisfies

$$W^{(t)}_{d,1:d} = -\sum_{h=1}^{t} e_{i_h}, \qquad t \in [k]. \tag{44}$$

The Phase-1 analysis establishes the base case. Suppose (44) holds at the end of Phase $t - 1$. In Phase t, row d still has feature vector $q_1 = x$, initial weights $w_0 = -\sum_{h<t} e_{i_h}$, and target

$$z_t(x) = \sigma(\langle w_0, x \rangle - x_{i_t}).$$

All input features occur once. Applying Theorem D.5 on the Phase-t event and using (41) yields

$$W^{(t)}_{d,1:d} = w_0 - e_{i_t} = -\sum_{h=1}^{t} e_{i_h}.$$

This completes the induction. Although each gradient step updates the full matrix, the loss in these phases depends only on row d, so all other rows have zero gradient and remain unchanged.

At the end of Phase k, for every $x \in \{0,1\}^d$,

$$f_{W^{(k)}}(x)_d = \sigma\left(-\sum_{h=1}^{k} x_{i_h}\right) = \bigoplus_{h=1}^{k} x_{i_h} = z_k(x).$$

The Phase-1 rollout also gives $f^{\mathrm{e2e}-k}_{W^{(1)}}(x) = \widehat{z}_k(x) = z_k(x)$. Thus, under the internalization map in the theorem,

$$f_{W^{(k)}}(x)_d = P(f_{W^{(1)}})(x) = f^{\mathrm{e2e}-k}_{W^{(1)}}(x) \qquad \text{for every } x \in \{0,1\}^d.$$

Both claims hold on the same event $\mathcal{E}_{\mathrm{all}}$, whose probability is at least $1-\delta$. Finally, taking B at the displayed polynomial threshold, $k \le d$ and $T = O(d)$ give runtime polynomial in d and $\log(1/\delta)$. □

D.4 Experiments for Learning Parity with Simplified Transformer

We also experimentally see the learning dynamics by training the same transformer architecture as in Algorithm 2 on the parity task, but without the rounding step. We set $d = 20$ and $k = 9$, use batch size $B = 1024$, step size $\eta = 10^{-3}$. Rather than switching phases after a fixed iteration budget T, we advance to the next phase once the training loss falls below a prescribed threshold. The learned attention matrices are reported in Figure 6. Consistent with the theoretical mechanism, the attention weights progressively concentrate on the relevant coordinates, and by the end of the curriculum the model selects the desired coordinates from the hidden parity support.

E Internalizing semiautomata: experimental details and additional figures

In this section, we provide more experimental details and results on internalizing the simulation of semiautomata. We describe our learning setting in more detail.

Setup A semiautomaton $\mathcal{A}$ is a tuple $\mathcal{A} = (Q, \Sigma, \delta)$ where Q is a set of *states*, Σ is an input alphabet and $\delta : Q \times \Sigma \to Q$ denotes a *transition function*. We are interested in predicting the state of the semiautomaton after T transition steps, starting from a given initial state $q_0 \in Q$. Equivalently, we are interested in learning the *extended* transition function $\delta^{(t)} : Q \times \Sigma^T \to Q$. We consider a uniform distribution over input symbols $\sigma_1, \ldots, \sigma_T \sim_{iid} \mathrm{Unif}(\Sigma)$. Each input sequence is concatenated with a fixed initial state q_0, that is $z = (\sigma_1, \ldots, \sigma_T, q_0) \in \Sigma^T \times Q$. The learning goal is to find a predictor $h : \Sigma^T \times Q \to Q$ that minimizes the probability of predicting the wrong final state:

$$L_{\mathcal{A}}(h) = \mathbb{P}_{\sigma_1,\ldots,\sigma_T}\left[h(\sigma_1, \ldots, \sigma_T, q_0) \neq \delta^{(T)}(q_0, \sigma_1 \ldots \sigma_T)\right]. \tag{45}$$

Autoregressive training We consider various training procedures that differ in the type and structure of the training data:

- *Chain-of-thought* (CoT) training: We consider sequences $z = (\sigma_1, \ldots, \sigma_T, q_0, q_1, \ldots, q_T)$ that contain all the intermediate states between input and final state, where $q_t = \delta(q_{t-1}, \sigma_t,) = \delta^{(t)}(q_0, \sigma_1 \ldots \sigma_t)$ for $t \in [T]$.
- *End-to-end* (E2E) training: Only the final state $q_T \in Q$ is appended to the input sequence.
- *Internalization via left curriculum*: We consider T stages of a curriculum, where at each stage t we use data of the form $z^{(t)} = (\sigma_1, \ldots, \sigma_T, q_0, q_t, \ldots, q_T)$. That is, we omit the first $t-1$ states from the sequence.

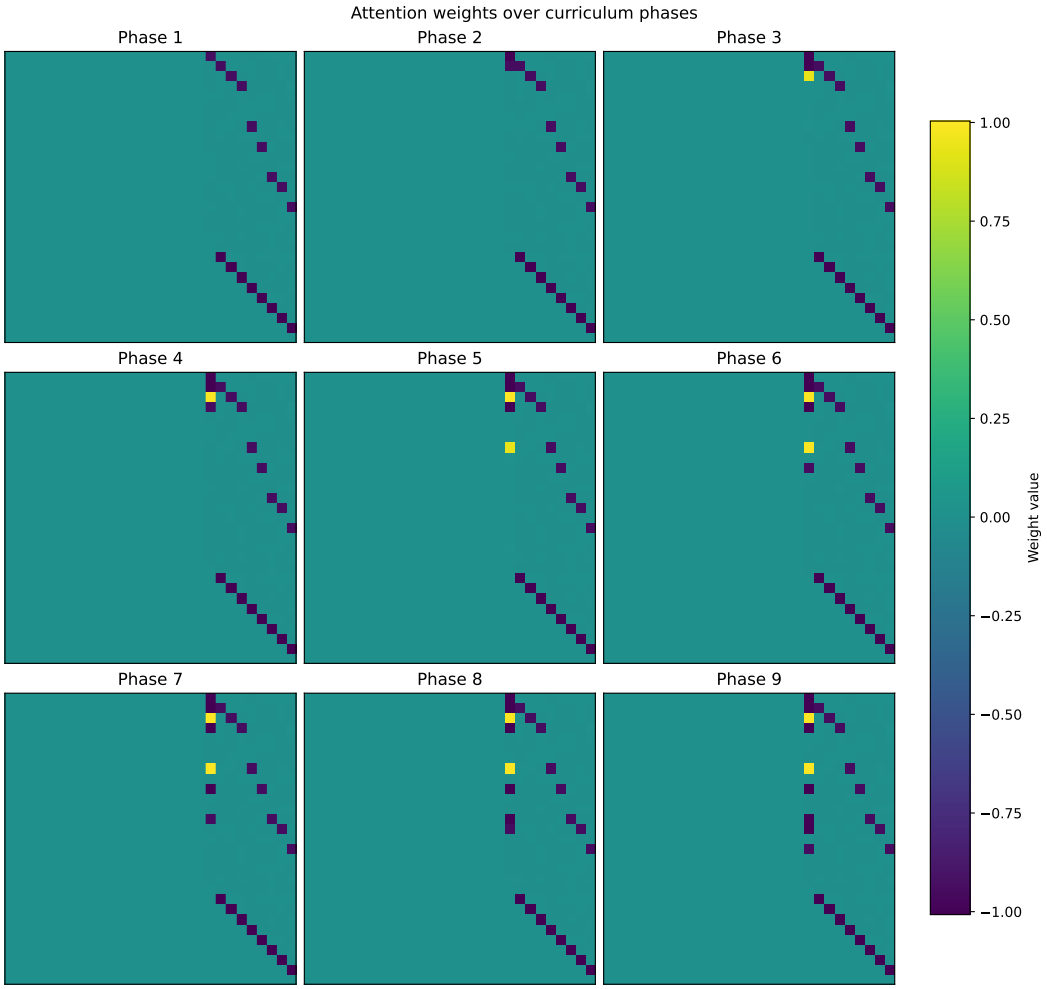


Figure 6: Evolution of the learned attention weights during curriculum training for the parity task. Each panel shows the attention-weight matrix after one internalization phase, where the model is trained after progressively removing one CoT token. The visualization suggests that the same architecture gradually reorganizes its attention weights to recover the hidden parity support as the explicit CoT is removed.

- *Internalization via right curriculum*: Similarly, we consider T stages of a curriculum, where at each stage t we use data of the form $z^{(t)} = (\sigma_1, \ldots, \sigma_T, q_0, q_1, \ldots, q_{T-t}, q_T)$. In other words, we omit the last $t-1$ states from the sequence (starting from state q_{T-1}).
- *Internalization via inductive curriculum*: We consider T stages of a curriculum, where at each stage t we use data with a chain of thought that contains leaps of length t, i.e., $z^{(t)} = (\sigma_1, \ldots, \sigma_T, q_0, q_t, q_{2t}, \ldots, q_{kt})$ where $k = \arg\max\{n \in \mathbb{N} : nt \leq T\}$. Note that the last token might or might not be the final state q_T in some of the stages.

We train with a next-token prediction $\log$ loss applied only at the state positions of each sequence (i.e., the predictable part of the sequence). We start all curricula at stage 1 (when the data contain the full history of states $q_1, \ldots, q_T$) and increase the stage after a fixed number of iterations.

For counter automata, the alphabet Σ and state space Q are the same, that is $\Sigma = Q = \mathbb{Z}_n$. We encode input symbols and state symbols using different symbols in our transformer implementation. For S_3, the input symbols are 4 permutations: identity, transposition $1 \leftrightarrow 2$, transposition $2 \leftrightarrow 1$ and a shift to the left by 1. We encoded each state using their Lehmer rank. We always append an EOS token at the end of the sequences. For OOD evaluation on C_n, $n > 2$, each input token is sampled independently as $\sigma_t \sim \text{Binomial}(n-1, p)$, where n is the size of the input space and p is

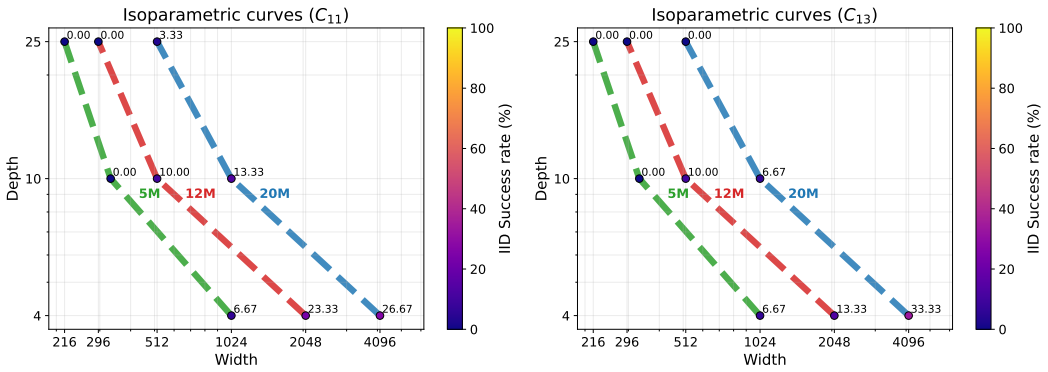


Figure 7: **Internalization success rates across isoparametric curves for C_{11} and C_{13}.** We observe that increasing width, rather than depth, improves the chances of internalization.

a parameter. For the parity semiautomaton C_2, OOD is measured by simply changing the bias of sampling '1' vs '0' from 0.5 to p. Unless stated otherwise, we always use $p = 0.8$ which induces a distribution concentrated on larger symbols and larger total sequence sums.

Experimental details We run experiments using Huggingface's transformers library. The architecture is a GPT2-type architecture with absolute positional embeddings, normalization layers, residual connections and uses the GeLU activation function. We use the AdamW optimizer with constant learning rate $3 \cdot 10^{-4}$ and parameters $(\beta_1, \beta_2) = (0.9, 0.95)$, where the reduced momentum coefficients enable faster adaptation to the changing loss between curriculum stages. We reset the optimizer's state in the beginning of each curriculum stage.

Each run for a counter semiautomaton completes in $50,000$ training iterations of online training with 128 freshly sampled sequences per iteration. All curriculum stages consist of an equal amount of training iterations equal to $50,000/T$, where T is the automaton simulation time. The experiments on S_3 run for $200,000$ training iterations. We deem an internalization run as *successful* if in-distribution test accuracy is above 95% at the end of training. This is how we compute successs rates across the section.

The main set of results of Figure 2 consists of 30 random seeds combination of semiautomaton and simulation length. The depth of the network is set to 4, the embedding dimension is set to 512, the number of heads is set to 128 and the MLP width at each layer is 2048.

Figure 3 (left), which compares success rates across different semiautomata for $T = 25$, presents simulations with networks of depth 4, embedding dimension 512, number of heads 128, and MLP width 256 and 2048.

Figure 3 (right) and Figure 7 show success rates across varying width and depth configurations for C_7 and C_{11}, C_{13}, respectively. In these figures, width is defined as the maximum of the embedding dimension and MLP width. We evaluate three parameter-matched architecture families:

$$\text{20M} : [(4, 512, 4096), (10, 512, 1024), (25, 344, 512)],$$
$$\text{12M} : [(4, 376, 1024), (10, 332, 128), (25, 216, 64)],$$
$$\text{5M} : [(4, 512, 2048), (10, 448, 512), (25, 296, 256)],$$

where each tuple denotes $(\text{depth}, \text{embedding dimension}, \text{MLP width})$. In all cases, we use per-head-dimension 4 which determines the number of heads for each configuration.

Figure 4 presents scatter plots of in-distribution (ID) and out-of-distribution (OOD) test accuracies at the end of internalization across multiple random seeds. For C_2, we use an embedding dimension of 64 with 32 attention heads under the inductive curriculum. For C_3, the depth-1 configuration uses embedding dimension 2048, MLP width 8192, and 512 attention heads, also trained with the inductive curriculum. The depth-4 and depth-10 configurations for both C_3 and C_5 use embedding dimension 512 and 128 attention heads, and are trained using the left curriculum. We report depth-1 results with the inductive curriculum because it consistently achieved stronger internalization performance than the left curriculum in shallow architectures.

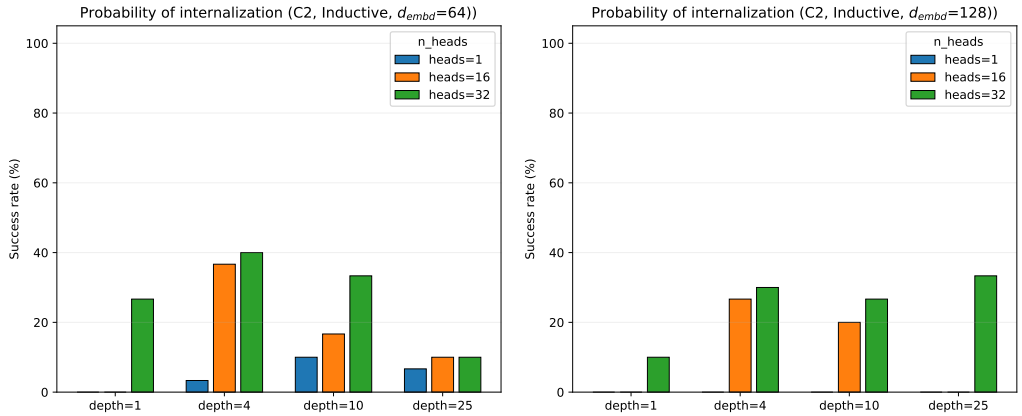


Figure 8: **Increasing the number of attention heads improves the success rate of internalization.** Results are shown for C_2 under the inductive curriculum, with embedding dimensions 64 (left) and 128 (right), across multiple depth configurations. The MLP width is set to four times the embedding dimension.

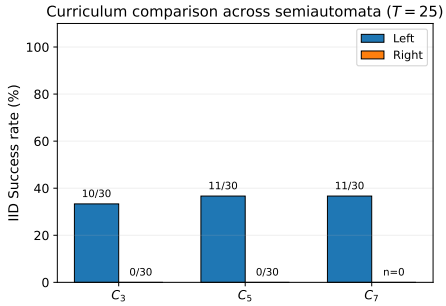


Figure 9: **Comparison across left and right curricula for internalization of C_3, C_5 and C_7.** This is for transformers of depth 4, embedding dimension 512, number of heads set to 128 and MLP width equal to 2048.

More results We found the number of attention heads to be highly important for successful internalization. In particular, for a fixed embedding dimension shared across heads, increasing the number of heads consistently improved the internalization success rate. Figure 8 demonstrates this effect for C_2 across two embedding dimensions and multiple depth configurations.

We found that a curriculum method that starts by removing tokens from the right is strictly worse at internalizing chain-of-thought reasoning than either a left-to-right or an inductive method. For example, in Figure 9, we present success rates across three different semiautomata and observe that, while a left curriculum succeeded in internalizing approximately one-third of the time, the right curriculum failed in every case. Similar observations were made by Deng et al. [2024].

We provide an intuitive explanation for this phenomenon. We believe that this shortcoming is largely attributable to the use of absolute positional embeddings in the architecture. As chain-of-thought tokens are internalized, the MLP – which is shared across positions – must play a dual role: it must implement both the simple one-step operation (computing $\mathrm{mod} n$ over two numbers) and the increasingly complex aggregated operation of addition modulo n over t numbers, where t denotes the current stage of the curriculum. Since our transformer uses absolute positional embeddings, it must use positional information to gate the MLP into one of these two operations. If we internalize from the left, then the position at which the complex operation occurs remains fixed at $T + 1$. In contrast, if we internalize from the right, then this position changes at every stage, creating a difficult moving target for the MLP. We believe that other architectures, such as mixture-of-experts models, may not exhibit such a difference between the two methods.

E.1 Learning Semiautomata with Mixture: Experimental Details

We also consider learning semiautomata from a single mixture distribution over CoT lengths, rather than from an ordered curriculum (see next Section). Let

$$\mathcal{D}_1, \ldots, \mathcal{D}_T$$

denote the stage distributions corresponding to one of the CoT-removal procedures defined above. The mixture distribution is $\mathcal{D}_{\text{mix}} = \frac{1}{T}\sum_{t=1}^{T} \mathcal{D}_t$. In our implementation, at each training step we sample a stage $t \sim \text{Unif}([T])$ and draw the batch from the corresponding stage distribution $\mathcal{D}_t$. Thus, marginally over training, the model is trained on a single distribution supported on examples with different CoT lengths. We use the same number of samples in total for the curriculum and mixture runs so that the comparisons are fair.

The main difference from the curriculum experiments is having additional token for the answer length hints. In the original semiautomata experiments, an example consists of the input symbols, followed by the fixed initial state q_0, followed by the retained state tokens and `EOS`. In the mixture experiments, we insert one additional token after q_0 and before the answer block:

$$(\sigma_1, \ldots, \sigma_T, q_0, \texttt{LEN}_L, q_{j_1}, \ldots, q_{j_K}, \texttt{EOS}).$$

Here $1 \le j_1 < \cdots < j_K = T$, so the final state q_T is always retained, and $L = K + 1$ is the length of the answer block including the trailing `EOS`. We add T new vocabulary tokens

$$\texttt{LEN}_2, \ldots, \texttt{LEN}_{T+1},$$

disjoint from the input tokens, state tokens, and `EOS`. The direct-answer case has $K = 1$, hence $L = 2$, corresponding to the output

$$(q_T, \texttt{EOS}).$$

The token $\texttt{LEN}_L$ tells the model which output format is requested for the current example. The hint is part of the prompt and is not included in the loss; as above, the next-token loss is applied only to the predictable answer/state positions.

At evaluation time, we can either use the natural length hint associated with a mixture component or force a desired answer length. In particular, forcing $\texttt{LEN}_2$ asks the model to answer directly, while forcing larger values of L asks it to generate a longer CoT before the final answer. We report final-state accuracy, where the prediction is the last generated state token before `EOS`. This lets us evaluate the same trained model both in a direct-answer regime and in longer-CoT regimes.

F From Internalization Curricula to Positive Distribution Shift: Learning with a Single Mixture Distribution

In this section, we use the example of Section 5 to explain the connection between learning through internalization and the Positive Distribution Shift framework introduced in Medvedev et al. [2026]. Motivated by this connection, we show that learning through a curriculum-based internalization procedure can often be replaced by a more straigthforward mixture learning procedure. We validate this in both the feed forward and the autoregressive setting.

Recall that in the example of Section 5, our goal is to learn the parity distribution $(\mathcal{D}, f^*) = (\text{Unif}(\{0,1\}^d), x_{i_1} \oplus \cdots \oplus x_{iu_k} \oplus \xi)$ (where ξ represents the label noise) tractably, i.e. find a predictor h_θ which achieves low error $L_{(\mathcal{D},f^*)}(h_\theta)$. Noisy parity is computationally hard to learn - it is known to be hard in the SQ framework [Kearns, 1998, Blum et al., 1994], there is no known tractable PAC learning algorithm for it [Blum et al., 2003, Goel et al., 2017], and even fixed parities are difficult to learn with standard neural-networks trained with standard gradient descent [Shoshani and Shamir, 2025, Abbe et al., 2025]. However, as shown above, the procedure of learning $(\mathcal{D}, f^*)$ from the curriculum $(\mathcal{D}_1, f^*), \ldots, (\mathcal{D}_{k-1}, f^*)$, where $\mathcal{D}_i$ is given by $\widetilde{x} \sim \left(x_1, \ldots, x_d, z_1, \ldots, z_{k-1-l+1}, \underbrace{0, \ldots, 0}_{l-1}\right)$, by the predictor h_θ is (experimentally) tractable. This learning procedure with the curriculum $(\mathcal{D}, f^*) \to \{(D_i, f^*)\}_{i=1}^{k-1} \to h_\theta$ is similar to the target dependent positive distribution shift, *f-PDS*,

introduced in Medvedev et al. [2026], where we modify the training distribution depending on the target (see Appendix G for the definition of f-PDS). However, that type of positive distribution shift only involves a single training distribution (and only a covariate shift on the original input space[6]). In addition, Medvedev et al. [2026] broadly conjecture that a curriculum can be replaced with learning from a single distribution. Motivated by this, in top Figure 10, we show (in green) that we can replace the curriculum for learning parities with MLPs from Section 5 with a mixture distribution $\mathcal{D}_{\text{mix}} = \frac{1}{k}\sum_{l=1}^{k-1} \mathcal{D}_l$, i.e. we can efficiently learn $(\mathcal{D}, f^*)$ using the subnetwork of the network that learns" $(\mathcal{D}_{\text{mix}}, f^*)$. Therefore, $(\mathcal{D}_{\text{mix}}, f^*)$ represents a (generalized[7]) f-PDS for learning $(\mathcal{D}, f^*)$, which can be constructed intuitively and tractably from f^* and $\mathcal{D}$: to learn $(\mathcal{D}, f^*)$, train g on $(\mathcal{D}_{\text{mix}}, f^*)$ then take the subnetwork h_θ that operates on the first d bits as our subnetwork.

Inspired by the PDS view, we ask whether the autoregressive learning of semiautomata simulation from Section 3 using a curriculum can also be replaced by a single distribution. Indeed, in Figure 10, we show that we can learn semiautomata by training on a single mixture distribution of curriculum stages, with added hints for answer length (see Appendix E.1 for more details). This presents an alternative to learning through internalization for sidestepping the hardness of learning.

Benefits of Learning from a Mixture: Direct Answering and OOD There are three main benefits to learning semiautomata with an autoregressive model from a mixture, as in Figure 10, over training with a curriculum as in Section 3. First, learning from a mixture removes the need to choose the curriculum schedule details and parameters, which can impact success of learning. Conversely, learning from a single mixture distribution is simpler. Second, a mixture distribution with CoTs of varying lengths more closely models the structure of natural data where some examples contain long explanations and other contain short explanations. Our results indicate that this heterogeneity of length of answers can itself act as a positive distribution shift for learning hard tasks. Third, with the answer hint, a single model trained on the mixture supports multiple inference budgets. This lets us force the model to answer directly or let it generate a longer CoT, and as Figure 10 shows, the mixture trained model can recover better OOD performance in the full-CoT regime, while the curriculum trained model does not.

G Additional Details for Learning Through Internalization as Positive Distribution Shift

G.1 General Positive Distribution Shift

Let $\mathcal{X}$ be the input space, and $(\mathcal{D}, f^*)$ be the target distribution, where $f^* : \mathcal{X} \to \Delta(\mathcal{Y})$ is the target function. We aim to find a predictor h that has low error $L_{(\mathcal{D},f^*)}$ specified by loss $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_{\geq 0}$ (we extend it to a randomized prediction by $\bar{l}(p, q) = \mathbb{E}_{\hat{a}\sim p, a\sim q} l(\hat{a}, a),\ p, q \in \Delta(\mathcal{Y})$). We are allowed to train on a (potentially larger) context space $\mathcal{C}$, which could include added hints, padding, prompts, or other information, and for which there is an extraction map $\pi : \mathcal{C} \to \mathcal{X}$ which maps the context c back to an input $x = \pi(c)$. The label depends only on the input $\pi(c)$. We train on a distribution $\mathcal{D}'_{\mathcal{C}} \in \Delta(\mathcal{C})$ labeled by $f^* \circ \pi$. For a class of models $\mathcal{M}$, we define an efficient transport map $Tr_b : \mathcal{M} \to \Delta(\mathcal{Y})^{\mathcal{X}}$ for an inference budget b. For a hypothesis $h \in \mathcal{H} \subseteq \Delta(\mathcal{Y})^{\mathcal{X}}$, we define $L_{(\mathcal{D},f^*)}(h) = \mathbb{E}_{x\sim\mathcal{D}}\bar{l}(h(x), f^*(x))$ and $L_{(\mathcal{D},f^*)}(\mathcal{H}) = \inf_{h\in\mathcal{H}} L_{(\mathcal{D},f^*)}(h)$. We evaluate the loss of a model $M \in \mathcal{M}$ on the target function accuracy as $L^b_{(\mathcal{D},f^*)}(M) = L_{(\mathcal{D},f^*)}(Tr_b(M)) = E_{x\sim\mathcal{D}}\bar{\ell}\,(Tr_b(M)(x), f^*(x))$.

Definition 4 (Generalized PDS learnability)**.** *A learning rule A generalized-PDS-learns $(\mathcal{D}, f^*)$ and a hypothesis class $\mathcal{H} \subseteq \Delta(\mathcal{Y})^{\mathcal{X}}$ with inference budget b, training context $\mathcal{C}$, training distribution $\mathcal{D}'_{\mathcal{C}}$, extraction map π, model class $\mathcal{M}$, and efficient transport map Tr_b with sample complexity $m(\varepsilon)$ and runtime $T(\varepsilon)$ if for every $\varepsilon > 0$ and for $S = \{(c_i, y_i)\}_{i=1}^{m(\varepsilon)} \sim (\mathcal{D}'_{\mathcal{C}}, f^\star \circ \pi)^{m(\varepsilon)}$, the learner outputs $\hat{M} = A(S) \in \mathcal{M}$ in time at most $T(\varepsilon)$, and*

$$\mathbb{E}_{S,A} L_{(\mathcal{D},f^\star)}\big(\mathrm{Tr}_b(\hat{M})\big) \leq L_{(\mathcal{D},f^\star)}(\mathcal{H}) + \varepsilon.$$

[6] So in that regard, the curriculum considered here does not fall under the strict definition of f-PDS; see Appendix G for details.

[7] Since the input space changes, it is not only a covariate shift; see Appendix G for details.

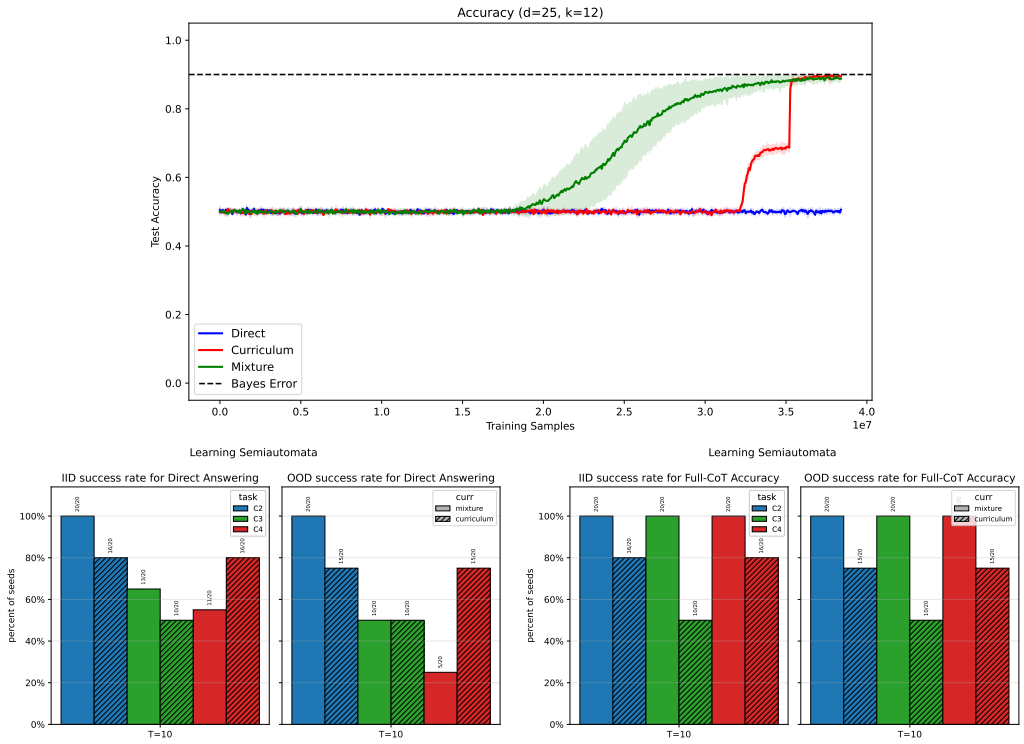


Figure 10: *Top:* **Learning through Internalization in MLPs.** We plot the test accuracy for curriculum and PDS learning of parities described in Section 5 and Appendix F. That is, we plot the dependence of test accuracy on the number of training samples for three different training strategies for learning an unknown noisy parity over $k = \frac{d}{2}$ bits (label noise $\eta = 0.1$) over the input space of d bits expanded with $k-1$ hints $\widetilde{x} = (x_1, \ldots, x_d, z_1, \ldots, z_{k-1})$, with t-th hint being the parity of first t indices of the support, and we plot the unhinted accuracy, i.e. accuracy of predicting y from $(x_1, \ldots, x_d, 0, \ldots, 0)$. We consider (i) direct training (in blue) where all the hints are zeroed out, (ii) curriculum training (in red) where in stage l we have $l-1$ hints zeroed out from the left, and (iii) mixture (in green) over the inputs from the $k-1$ stages of the curriculum, where we draw a stage uniformly at random. We see that direct training (blue) fails to learn, while both the curriculum (red) and the mixture (green) succeed in learning. *Bottom:* **Learning C_n from a single mixture distribution over various CoT lengths with a hint for the length of answer.** We train a transformer on $\mathcal{D}_{\text{mix}} = \frac{1}{T+1}\sum_{l=0}^{T} \mathcal{D}_l$, where $\mathcal{D}_l$ is the distribution over examples containing CoT of length l, and each example includes a hint for the length of the answer. We plot the percentage of seeds that achieve > 0.95 accuracy for (i) on the left, direct answering - the model is given the hint for answer length equal to 1, and (ii) on the right, the *same* model is allowed to generate full CoT. We see that when training with mixture, we can learn to both directly answer and to generate the full CoT with a single model.

Depending on what $\mathcal{C}$ and $\mathcal{D}_{\mathcal{C}}$ (and the elements in its support) are allowed to depend on, we can define generalized-f-PDS and generalized-D(R)S-PAC learnabilities. In particular, for f-PDS, we allow the training distribution $\mathcal{D}'_{\mathcal{C}}$ to depend on the target f^*.

Original PDS. The original PDS setting from Medvedev et al. [2026] is recovered by taking the training and evaluation spaces to be the same. Let $f^* : \mathcal{X} \to \Delta(\mathcal{Y})$ be the target function, let $\mathcal{D} \in \Delta(\mathcal{X})$ be the target input distribution, and let $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$ be the hypothesis class, which is deterministic in this case. Set $\mathcal{C} = \mathcal{X}, \pi = \mathrm{id}_{\mathcal{X}}$. Let $\mathcal{M} = \mathcal{H}$. We only need to consider $b = 0$ and we define $\mathrm{Tr}_b(h) = h$. We take $\mathcal{D}'_{\mathcal{C}} = \mathcal{D}'$ and use zero-one loss, $\ell(\hat{y}, y) = \mathbf{1}\{\hat{y} \neq y\}$. We have

$$L_{(\mathcal{D},f^\star)}(\mathrm{Tr}_b(h)) = \mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{y\sim f(\cdot|x)}\mathbf{1}\{h(x) \neq y\} = L_{\mathcal{D},f}(h).$$

Thus the generalized PDS guarantee becomes

$$\mathbb{E}_{S\sim(D',f)^m} L_{D,f}(A(S)) \leq L_{D,f}(\mathcal{H}) + \varepsilon,$$

which is the original PDS guarantee.

Mixture over hinted parities in MLPs. For the mixture experiments with MLPs in Section 5, we need to take $\mathcal{C}$ to be the expanded input space with the hints, and Tr_b to be the operation that extracts the subnetwork from the bigger network.

We instantiate $\mathcal{X} = \{0,1\}^d$, $\mathcal{D} = \mathrm{Unif}(\{0,1\}^d)$ and $f^*(\cdot \mid x)$ is $\chi_S(x)$ with probability $1-\eta$ and $1-\chi_S(x)$ with probability η, where S is the hidden parity support of size k. The loss is zero one loss $\ell(\hat{y}, y) = \mathbf{1}\{\hat{y} \neq y\}$. We then take $\mathcal{C} = \{0,1\}^{d+k-1}$ and $\pi(c) = (c_1, \ldots, c_d)$. The hints are given by $z_t(x) = \oplus_{j=1}^{t} x_{i_j}$, for $t = 1, \ldots, k-1$ and for $r = 0, \ldots, k-1$ let $c_r(x) = (x_1, \ldots, x_d, z_1(x), \ldots, z_{k-1-r}(x), 0, \ldots, 0) \in \mathcal{C}$. The hypothesis $\mathcal{H}_k$ here is the k-sparse parities. Let $\mathcal{D}_r$ be the distribution of $c_r(x)$ for $x \sim \mathcal{D}$ and take the training distribution to be $\mathcal{D}'_{\mathcal{C}} = \frac{1}{k}\sum_{r=0}^{k-1} \mathcal{D}_r$. The model class $\mathcal{M}$ is the class of expanded input MLPs $g_\theta(\widetilde{x}) = F_\theta(\widetilde{x}) = W_L\sigma(W_{L-1}\sigma(\cdots\sigma(W_1\widetilde{x} + b_1)\cdots) + b_{L-1}) + b_L$. The transport map takes the subetwork $Tr_0(g_\theta)(x) = (W_L\sigma(W_{L-1}\sigma(\cdots\sigma(W_1[:, 1\!:\!d]\, x + b_1)\cdots) + b_{L-1}) + b_L)$. The learning algorithm A here is the standard SGD on the expanded input MLP g_θ.

Then the PDS learning condition is $\mathbb{E}_{S,A} L_{(\mathcal{D},f^\star)}\big(Tr_0(A(S))\big) \leq L_{(\mathcal{D},f^\star)}(\mathcal{H}_k) + \varepsilon = \eta + \varepsilon$.

So our results with MLPs in Section 5 imply generalized-PDS-learnability of the noisy parity task.

CoT mixture for learning semiautomata. For a semiautomaton (Q, Σ, δ) and horizon T, we take the input space $\mathcal{X}$ to be given by elements of the form $(\sigma_1, \ldots, \sigma_T, q_0, \mathrm{LEN}_L)$, i.e.

$$\mathcal{X} = \Sigma^T \times Q \times \{\mathrm{LEN}_2, \ldots, \mathrm{LEN}_{T+1}\}.$$

Let $u = (\sigma_1, \ldots, \sigma_T, q_0)$. Given the answer length $K \in [T]$, if we have CoT state indices $1 \leq j_1 < \cdots < j_K = T$ left after CoT removal, the answer is $a^*_K(u) = (q_{j_1}(u), \ldots, q_{j_K}(u), \mathrm{EOS})$ where $q_t(u) = \delta(q_{t-1}(u), \sigma_t)$ is the state trajectory $t = 1, \ldots, T$. The target function is $f^*(x) = f^*((u, \mathrm{LEN}_K)) = a^*_K(u)$, and the target distribution is the final state evaluation $\mathcal{D} = \mathcal{D}_U \otimes \delta_{\mathrm{LEN}_2}$, where $\mathcal{D}_U$ is the distribution of $u = (\sigma_1, \ldots, \sigma_T, q_0)$. The output space is $\mathcal{Y} = \cup_{L=2}^{T+1}(Q^{L-1} \times \{\mathrm{EOS}\})$. The loss is $l(\hat{a}, a) = 1_{\mathrm{last}(\hat{a}) \neq \mathrm{last}(a)}$ where $\mathrm{last}(a)$ is the last token before EOS in a. The context distribution is given by

$$c_k(x) = (x, q_{j_1}(u), \ldots, q_{j_K}(u), \mathrm{EOS}).$$

Let $\mathcal{D}_K$ be the distribution of $c_K(u)$ for $u \sim \mathcal{D}_U$ and let $\mathcal{D}'_{\mathcal{C}} = \frac{1}{T}\sum_{K=1}^{T} D_K$. Here $\mathcal{M}$ is the set of autoregressive transformers. The direct answering transport map is for $b = 2$, $Tr_2(M)(x)$ given by the distribution of answers over outputs generated from $x = (u, \mathrm{LEN}_2)$ and the full-CoT inference we take $b = T + 1$ analogously. The learning algorithm A here is standard next-token prediction training of a transformer on the mixture distribution. Thus this case also falls under the generalized-PDS learnability definition.

G.2 Learning Through Internalization with MLPs: Experimental Details

All notation follows Section 5. In the experiments, we use the bipolar version of the parity task, which is equivalent to the binary domain: $\oplus$ is represented by multiplication, the original data input

has dimension $d = 25$, and the support size is $k = 12$. After relabeling the support, the target is $y = \prod_{j=1}^{12} x_j$, with $x \in \{\pm 1\}^{25}$. The augmented input has $d + k - 1 = 36$ coordinates: the first 25 are data coordinates, and the remaining 11 are the hint coordinates $z_1, \ldots, z_{11}$.

For implementation, we index stages by $r \in \{0, \ldots, k-1\}$, the number of removed hints. At stage r, the r strongest/rightmost hints are removed and replaced by the neutral point (0 for binary input, 1 for bipolar input)

$$\widetilde{x}^{(r)} = (x_1, \ldots, x_d, z_1, \ldots, z_{k-1-r}, 1, \ldots, 1).$$

To make the task slightly harder and a better proxy for learning, instead of "zeroing out" the hints, we use independent random noise

$$\widetilde{x}^{(r)} = (x_1, \ldots, x_d, z_1, \ldots, z_{k-1-r}, \xi_{k-r}, \ldots, \xi_{k-1}), \qquad \xi_i \sim \mathrm{Unif}\{\pm 1\}.$$

The plot in Figure 10 is shown with independent random noise. We denote the corresponding input distribution by $\mathcal{D}_r$. Thus $\mathcal{D}_0$ is the full-hint distribution, while $\mathcal{D}_{k-1}$ has no informative hints.

We compare three training conditions with matched total compute. The direct baseline trains only on $\mathcal{D}_{k-1}$. The curriculum condition trains sequentially on $\mathcal{D}_0, \mathcal{D}_1, \ldots, \mathcal{D}_{k-1}$, with equal budget per stage. The mixture condition trains on the single distribution $\mathcal{D}_{\mathrm{mix}} = \frac{1}{k} \sum_{r=0}^{k-1} \mathcal{D}_r$. In the implementation of the mixture condition, each minibatch first samples $r \sim \mathrm{Unif}\{0, \ldots, k-1\}$, and then all examples in that minibatch are sampled from $\mathcal{D}_r$. Marginally, this is training on the fixed mixture distribution $\mathcal{D}_{\mathrm{mix}}$, rather than on an ordered curriculum.

All models are evaluated on fresh samples from $\mathcal{D}_{k-1}$, where every hint coordinate is independent noise. This measures performance when no informative hint is available, while keeping the implemented input dimension fixed at $d + k - 1$. When evaluating the formal subnetwork predictor from Section 5, we instead evaluate h_θ directly on $x \in \{\pm 1\}^{25}$.

We use fully connected ReLU MLPs with raw scalar output. The configuration has depth 2 and width 2048, with approximately 4.27M parameters. Here depth counts hidden ReLU layers: an input layer $36 \to w$, followed by depth-1 hidden layers $w \to w$, and a final linear layer $w \to 1$.

We train with mean-squared error against the ± 1 target using plain SGD with learning rate 3×10^{-3}, batch size 64, no momentum, no weight decay, and no learning-rate schedule. Training is online: every minibatch consists of freshly sampled examples. Each run uses 600 epochs with 1000 SGD steps per epoch, for 6×10^5 total SGD steps. In the curriculum condition, this budget is divided evenly across the 12 stages, giving 50 epochs per stage. Accuracy is computed periodically on 5000 fresh evaluation samples, with a prediction counted as correct when its sign agrees with y. Each condition is run with 3 random seeds.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction (Section 1) clearly state the three main contributions: (1) empirical study of internalization of semiautomata identifying method-specific and task-specific factors and OOD degradation (Sections 3.1–3.3), (2) a provable internalization result for parity in a 1-layer transformer (Section 4), and (3) connecting internalization to Positive Distribution Shift. The scope is appropriately qualified (e.g., stylized architectures, specific task families).

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper discusses limitations throughout and in the Discussion (Section 6). Key limitations acknowledged include: the OOD degradation that accompanies internalization (Section 3.2), the conjectural nature of the prime-hardness explanation (Section 3.3), and the stylized nature of the theoretical model (1-layer transformer with linear attention and fixed MLP in Section 4). The empirical study is limited to semiautomata tasks and moderate-scale transformers.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The paper includes several theoretical results. The representation theorem for C_n semiautomata (Theorem C.3) is formally stated with all assumptions and proved in the appendix. Supporting lemmas (Lemma C.1, Lemma C.2) are stated and proved. The provable internalization result for parity (Theorem 4.1) is stated in Section 4 with analysis in the appendix. All theorems are numbered and cross-referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper describes the semiautomata tasks, training procedures (CoT, E2E, left/right/inductive curricula), and architectural choices in the main text (Sections 3–3.3). Full experimental details including optimizer, learning rate, batch size, number of training iterations, and curriculum switching criteria are provided in the appendix.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The paper does not currently provide code or data. The experiments use synthetically generated data (semiautomata simulation and parity tasks), so no external datasets are needed. We plan to release the code upon publication.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

Justification: The main text describes the tasks, curricula, and key architectural choices (depth, width configurations, 30 seeds per configuration). Hyperparameter specifications are provided in the appendix.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The main experimental results (Section 3.1) report success rates computed over 30 random seeds per configuration, which captures variability due to random initialization and data sampling. The paper also presents scatter plots of individual runs (Figure 4) showing the distribution of outcomes. While formal confidence intervals are not reported, the large number of seeds and binary nature of outcomes (success rate $>95\%$ threshold) make the reported success rates informative.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No]

   Justification: All experiments in this paper involve training small-scale models (up to 20M parameters) on synthetic data and can be reproduced on a single consumer GPU within reasonable time. Given the modest computational requirements, we did not report detailed compute budgets.

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This is foundational research on understanding internalization in neural networks. It uses synthetic data (semiautomata, parities), involves no human subjects, and raises no ethical concerns. The research conforms with the NeurIPS Code of Ethics.

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This is foundational theoretical and empirical research studying internalization as a learning phenomenon in neural networks. The work uses synthetic tasks (semiautomata, parities) and does not introduce new models, tools, or techniques that have a direct path to societal harm. The Discussion (Section 6) notes that internalization could reduce AI inference costs but also cautions about OOD degradation.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release any pre-trained models, datasets, or other artifacts that pose risks for misuse. All experiments involve small-scale synthetic tasks (semiautomata simulation and parity learning).

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [N/A]

Justification: The paper does not use any existing code packages, pre-trained models, or external datasets. All experiments are based on synthetically generated data from well-defined mathematical structures (semiautomata, parity functions). Prior work is properly cited throughout.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new datasets, models, or code as assets.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve research with human subjects.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs are not used as a component of the core methods in this research. The paper studies internalization in transformers trained from scratch on synthetic tasks.