From Transformers to Weighted Automata: Towards the Verification of Large Language Models

Smayan Agarwal[1], Aslah Ahmad Faizi[2], Shobhit Singh[1], and Aalok Thakkar[1]

[1] Ashoka University, India
{smayan.agarwal_ug25, shobhit.singh, aalok.thakkar}@ashoka.edu.in
[2] Independent Researcher
aslahahmadfaizi@gmail.com

Abstract. Large language models (LLMs) are increasingly deployed in safety-critical settings, yet their black-box nature makes it difficult to provide formal guaranties about their behavior. Existing verification approaches rely primarily on empirical probing and testing, leaving open the question of how to reason rigorously about general-purpose transformer architectures.

In this work, we establish a principled bridge between transformers and weighted automata, a classical model from formal language theory. This connection enables us to transfer verification tools from automata theory to the analysis of LLMs. Our contributions are twofold: First, we develop a formal correspondence between transformer architectures and weighted automata over reals, showing how distributional properties of LLMs can be captured within this framework. Second, we introduce an identity testing algorithm for weighted automata that provides a statistical method for distinguishing whether two stochastic models define the same distribution up to a tolerance threshold.

This work provides the first formal bridge between modern neural sequence models and classical automata theory, clarifying both the potential and the computational challenges for rigorous LLM verification.

Keywords: Large Language Models, Transformers, Weighted Automata, Verification, Identity Testing, Formal Methods

1 Introduction

The success of transformer-based language models has created an urgent need for principled methods to understand and verify their behaviour. As these models are deployed in safety-critical applications, ranging from medical diagnosis to legal reasoning, the black-box nature of their decision-making processes poses fundamental challenges for certification and reliability guarantees.

Theoretical results from formal language theory offer an alternative approach. By establishing systematic correspondences between transformer variants and

well-characterized computational models, we can leverage automata theory, circuit complexity, and descriptive logic to obtain predictive accounts of capabilities and limitations for specific architectural choices.

1.1 The Gap Between Benchmarks and True Capabilities

Large Language Models (LLMs) perform well on benchmarks, and yet fail on general tasks. Recent work demonstrates that even state-of-the-art models collapse when faced with simple inductive reasoning problems [5, 15]. Furthermore, these models often rely on superficial neighbour-based heuristics rather than generalization [18].

Current evaluation practices systematically inflate performance estimates while obscuring true model capabilities. A critical issue is *benchmark data contamination*, where test items appear in training corpora. Studies show that GPT-4 achieves over 50% accuracy on contaminated MMLU items, fundamentally violating the train–test split principle [10, 28]. This contamination operates alongside preference leakage, where evaluator LLMs exhibit systematic bias toward models sharing training data or architectural relationships, creating circular evaluation systems that measure memorization rather than understanding.

Recent frameworks expose these limitations. Cohen-Inger et al.'s C-BOD analysis demonstrates that high-performing models rely on "superficial memorization of benchmark cues" [7]. Similarly, Shojaee et al.'s examination of Large Reasoning Models reveals "complete accuracy collapse beyond certain complexities" and "counterintuitive scaling limits" where reasoning effort paradoxically decreases on challenging problems despite adequate computational resources [24]. These highlight a need for formal guarantees.

1.2 The Verification Challenge

Formal verification of neural models faces a fundamental computational barrier. For a model $M : \Sigma^* \to \mathcal{Y}$, and a property R, verification requires establishing the universal claim $\forall w \in D : R(M(w)) = 1$. This is only tractable when the input domain D admits symbolic specification (e.g., as a regular language or logical constraint). When D is defined extensionally by length bounds $D = \Sigma^{\leq n}$, the domain size becomes $\Theta(|\Sigma|^n)$. Any sound black-box verifier then faces a worst-case $\Omega(|D|)$ query lower bound, making universal verification computationally infeasible for even modest n.

Practical verification therefore requires structured domains D that admit finite abstraction through techniques like product constructions or model checking, rather than exhaustive enumeration of all token sequences up to length n.

1.3 Why Global Constraints Are Insufficient

While enforcing global constraints, such as K-Lipschitz bounds, appears attractive for certification, this approach fundamentally mismatches capability

requirements. Lipschitz constraints depend on continuous embedding metrics rather than discrete string functions, compound pessimistically across layers, and force low-contrast attention that prevents the sharp routing needed for algorithmic behaviors. Empirically, global Lipschitz training trades accuracy for minimal certified radii, over-regularizing the selective attention mechanisms that give Transformers their power while providing certificates too weak to justify the capability loss [16, 29, 11, 27, 9, 17, 22, 20].

1.4 Automata-Theoretic Foundations

Recent theoretical advances have begun mapping transformer expressivity to classical computational models. Notably, Angluin et al. [1] proved that rightmost hard-attention encoders with strict causal masking recognize exactly the star-free languages (first-order logic over strings). Adding periodic positional predicates upgrades this class to FO[MOD], while arbitrary monadic predicates yield FO[Mon]. These precise equivalences reveal that transformers align with well-studied fragments of logic and automata theory. Similarly, architectural variants map naturally to circuit complexity classes: unique and averaging hard-attention encoders correspond to AC^0 and TC^0 respectively [12, 25]. Such characterizations suggest that transformer expressivity analysis can be grounded in established frameworks rather than relying solely on empirical exploration.

1.5 Weighted Finite Automata as an Abstraction Layer

While the above results apply to restricted transformer variants, modern LLMs use soft attention mechanisms that compute distributions over sequences. This motivates our focus on Weighted Finite Automata (WFA) over reals as a natural model for stochastic sequence generation that generalizes both classical finite automata and hidden Markov models. WFA provide several advantages for LLM analysis:

1. Compositional structure: Unlike neural networks, automata admit modular analysis through product constructions and closure properties.
2. Algorithmic foundation: Classical algorithms exist for equivalence testing, minimization, and property verification.
3. Statistical learning theory: Spectral methods enable provable recovery of automata from sample data [2, 14].

1.6 Technical Approach and Contributions

The key two insights of our work are reflected in the two-step reduction:

Step 1: Transformer to WFA. We prove that the self-attention mechanism in decoder-only transformers can be exactly computed by a structured RNN that maintains running accumulators for the attention normalization. We then apply spectral learning to the RNN's induced function, using Hankel matrix analysis to extract a finite-state weighted automaton approximation. We discuss this in Section 3.

Step 2: Identity Testing of WFA. We develop the first identity testing procedure for rational stochastic languages on infinite domains. This technique can be applied to the WFA obtained in Step 1 to provide formal guarantees about a restricted transformer architecture. We discuss this in Section 4.

Through these, our work makes two primary contributions toward verification and explainability of LLMs:

1. **Formal reduction theorem:** We prove that decoder-only transformers are equivalent to structured RNNs, which can be approximated by WFA with controlled error bounds.
2. **A black-box algorithm for identity testing.** We design an algorithm that, given only sample access to an unknown distribution over strings (e.g., LLM outputs), tests whether this distribution is ε-close to some WA with confidence $1 - \delta$. This test serves as a *certificate of abstraction*: whenever it accepts, one can meaningfully reason about the black-box model via the corresponding WA. This enables (a) verification of safety and correctness properties, (b) improved interpretability through automaton-based explanations of sequential behavior, and (c) applicability to proprietary black-box models where internal parameters are inaccessible.

2 Notation and Preliminaries

This section establishes the notation and foundational concepts used throughout the paper. We assume familiarity with basic formal language theory, linear algebra, and probability theory.

2.1 Sets, Alphabets, and Strings

Let Σ denote a finite alphabet. In the context of LLMs, the alphabet Σ typically represents the model's *vocabulary* or *tokenizer*, where each symbol $\sigma \in \Sigma$ corresponds to a unique token. The set of all finite strings over Σ is denoted Σ^*, which includes the empty string ϵ. The length of a string x is denoted $|x|$. The set of all strings of length at most θ is denoted $\Sigma^{\leq\theta}$.

2.2 Distributions and Distances

A distribution P over a countable set $\mathcal{X}$ (in our case, $\mathcal{X} = \Sigma^*$) is a function $P : \mathcal{X} \to [0, 1]$ such that $\sum_{x\in\mathcal{X}} P(x) = 1$. We denote the set of all distributions over Σ^* by $\mathcal{S}(\Sigma^*)$, and call them stochastic languages.

The primary distance measure between two distributions P and Q is the ℓ_1 distance (which is twice the total variation distance):

$$\|P - Q\|_1 = \sum_{x\in\Sigma^*} |P(x) - Q(x)|.$$

For a distribution P and a set $S \subseteq \Sigma^*$, we write $P(S) = \sum_{x\in S} P(x)$.

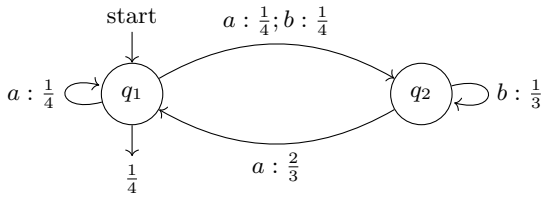

Fig. 1. A weighted finite automaton over the alphabet $\{a, b\}$. State q_1 is both initial and accepting with weight 1. The transition labels indicate the symbol and the weight (probability) associated with each move.

2.3 Weighted Finite Automata and Rational Stochastic Languages

Definition 1 (Weighted Finite Automaton (WFA)). *Let Σ be a finite alphabet. For a given semiring $(K, +, \cdot, 0, 1)$, a weighted finite automaton is a tuple $\mathcal{A} = (\Sigma, Q, \lambda, \mu, \rho)$, where Q is a finite set of states, $\lambda, \mu \in K^Q$ are the initial and final weight functions, and $\rho\colon Q \times \Sigma \times Q \to K$ is the transition weight function. The semantics of $\mathcal{A}$ is the function $f_{\mathcal{A}}\colon \Sigma^* \to K$ given by*

$$f_{\mathcal{A}}(w) \;=\; \sum_{q_0,\ldots,q_n \in Q} \lambda(q_0) \cdot \rho(q_0, w_1, q_1) \cdots \rho(q_{n-1}, w_n, q_n) \cdot \mu(q_n),$$

for $w = w_1 \cdots w_n \in \Sigma^*$.

Definition 2 (Rational Stochastic Language). *The class of rational stochastic languages over alphabet Σ and field $\mathbb{R}$, denoted $\mathcal{S}^{rat}_{\mathbb{R}}(\Sigma)$, consists of all functions $f\colon \Sigma^+ \to \mathbb{R}$ such that:*

1. *$f = [\![\mathcal{A}]\!]$ for some WFA $\mathcal{A}$ over the semiring $(\mathbb{R}, +, \cdot, 0, 1)$*
2. *f is a stochastic language*

The restricted class $\mathcal{S}^{rat}_{\mathbb{R}^+}(\Sigma)$ considers only automata with non-negative weights. When the semiring is clear from context, we write simply $\mathcal{S}^{rat}(\Sigma)$.

Figure 1 shows a WFA over the alphabet $\{a, b\}$. It induces a rational stochastic language over $\{a, b\}^*$. For a detailed discussion on WFA and stochastic languages, see [4].

2.4 Recurrent Neural Networks

Definition 3 (Recurrent Neural Network (RNN)). *A* recurrent neural network (RNN) *is a triple $R = (\alpha, g, \beta)$ where: (i) $\alpha \in \mathbb{R}^d$ is the initial state vector, (ii) $g : \mathbb{R}^d \times \Sigma \to \mathbb{R}^d$ is the recurrent transition function, (iii) $\beta : \mathbb{R}^d \to \mathbb{R}$ is the output function. The RNN processes a sequence $w = w_1 \ldots w_n \in \Sigma^*$ by producing a sequence of hidden states:*

$$\begin{aligned} h_0 &= \alpha, \\ h_i &= g(h_{i-1}, w_i) \quad \text{for } i = 1, \ldots, n. \end{aligned}$$

The output of the RNN for the sequence is $f_R(w) = \beta(h_n)$. *[19]*

Intuitively, an RNN maintains a hidden state h_i that serves as a compressed memory of all past inputs up to position i. At each step, the transition function g updates this state based on the previous hidden state and the current input symbol. The initial vector α represents the network's starting memory, and the output function β maps the final hidden state to a scalar value or prediction. In standard neural implementations, g and β are parameterized by weight matrices and nonlinearities such as tanh or ReLU, allowing the RNN to approximate complex sequence-to-value mappings.

2.5 The Spectral Learning Framework

Spectral learning offers a method to learn a finite state representation from i.i.d. samples [14, 2]. The *Hankel matrix* $H_f \in \mathbb{R}^{\Sigma^* \times \Sigma^*}$ of a function $f : \Sigma^* \to \mathbb{R}$ is a bi-infinite matrix whose entries are defined for any $(p, s) \in \Sigma^* \times \Sigma^*$ by $H_f[p, s] = f(ps)$, where ps denotes the concatenation of p and s.

A fundamental result [23] states that f is rational (i.e., computable by a WFA of d states) if and only if its Hankel matrix H_f has finite rank d. We discuss this in further detail in Section 3.2.

2.6 Transformer Attention

Transformers [26] are the popular LLM design architecture. At a high level, they operate over sequences of discrete symbols from a finite alphabet Σ, which are mapped to sequences of vectors via an embedding layer. Each transformer layer is a length-preserving function on such vector sequences, combining *self-attention*—which retrieves and mixes information from a variable-length history—and *feed-forward networks* applied position-wise. Stacks of these layers, together with input and output projections, yield a powerful sequence model.

For theoretical analysis, it is crucial to treat transformers as operating on inputs of *unbounded length*, rather than fixing a maximum context window. This allows us to compare them with formal language models such as automata, circuits, and logics, and to ask: *what languages can transformers recognize or generate?* Variants of the architecture (encoder, decoder, encoder–decoder), choices of attention mechanism (softmax, hard), and restrictions on precision or positional encodings all impact expressivity [25].

Definition 4 (Decoder-Only Transformer). *Let* Σ *be a finite alphabet and* $d \in \mathbb{N}$. *A one-layer decoder-only Transformer is a tuple*

$$\mathcal{T} = (\alpha, W_Q, W_K, W_V, \beta),$$

where

- $\alpha : \Sigma \to \mathbb{R}^d$ *is the embedding map assigning each symbol an embedding vector;*

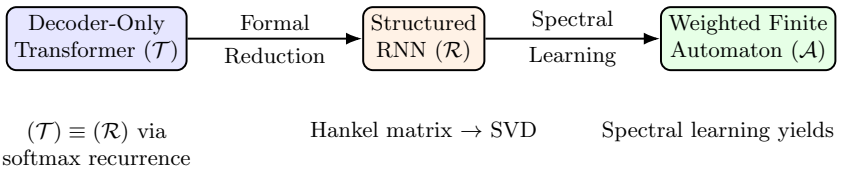

Fig. 2. Pipeline for reducing a decoder-only Transformer $\mathcal{T}$ to a weighted automaton $\mathcal{A}$. Step 1: establish equivalence with a structured RNN $\mathcal{R}$ (Sec. 3.1). Step 2: recover $\mathcal{A}$ from $\mathcal{R}$ via spectral learning on truncated Hankel matrices (Sec. 3.2).

- $W_Q, W_K, W_V \in \mathbb{R}^{d\times d}$ *are the learned query, key, and value projection matrices;*
- $\beta : \mathbb{R}^d \to \mathbb{R}$ *is the output map.*

Given a sequence $w = \sigma_1 \cdots \sigma_n \in \Sigma^*$*, the model first maps it to embeddings* $e_i = \alpha(\sigma_i)$*. At each position* n*, the layer computes a* query *vector* $q_n = W_Q e_n$ *for the current symbol, as well as* key *and* value *vectors* $k_i = W_K e_i$*,* $v_i = W_V e_i$ *for all previous symbols. The hidden state at position* n *is then obtained by attending to the past:*

$$h_n = \frac{\sum_{i=1}^{n} v_i \, e^{q_n^\top k_i}}{\sum_{j=1}^{n} e^{q_n^\top k_j}},$$

that is, a weighted sum of the values v_i*, where the weights depend on how similar the current query* q_n *is to each past key* k_i*. Finally, the model output on* w *is*

$$f_{\mathcal{T}}(w) := \beta(h_n).$$

Intuitively, a decoder-only Transformer processes a sequence one token at a time, much like an RNN, but instead of maintaining a single recurrent hidden state, it computes a new hidden representation h_n for the current token by looking at all previous tokens simultaneously through the *attention mechanism*. This mechanism allows Transformers to model long-range dependencies more effectively than RNNs, because each token can directly attend to any previous token without being bottlenecked by a single hidden state.

3 From Transformers to Weighted Finite Automata

We reduce a one-layer decoder-only transformer to a WFA via an intermediate RNN. Figure 2 illustrates the pipeline: the transformer $\mathcal{T}$ is first unfolded into an equivalent structured RNN $\mathcal{R}$; then, $\mathcal{R}$ is approximated by a WFA $\mathcal{A}$ through spectral learning.

3.1 Decoder-Only Transformer as a Recurrent Model

We now demonstrate an interesting viewpoint: that the self-attention mechanism in transformers can be computed by a carefully constructed RNN [30].

This equivalence provides a fascinating perspective on the computational relationship between these two fundamental architectures.

Construction. We construct a specific RNN that computes the softmax self-attention mechanism of a decoder-only transformer at position n. Let $\sigma(x) = (1 + e^{-x})^{-1}$ denote the sigmoid function.

For the query vector $q_n \in \mathbb{R}^d$ and a sequence of key-value pairs $\{(k_i, v_i)\}_{i=1}^n$, the goal of the RNN is to compute the value accumulator:

$$f_n = \frac{\sum_{j=1}^n v_j e^{q_n^\top k_j}}{\sum_{j=1}^n e^{q_n^\top k_j}}. \tag{1}$$

We define an RNN whose state $h_i \in \mathbb{R}^{d+1}$ is a composite vector containing two components:

1. $g_i \in \mathbb{R}$: A scalar for the running log-partition function ($\log \sum_{j=1}^i e^{q_n^\top k_j}$).
2. $f_i \in \mathbb{R}^d$: A vector for the running weighted sum of values ($\sum_{j=1}^i v_j e^{q_n^\top k_j}$).

We pack these into a single state vector: $h_i = (g_i, f_i)$.

The initial state, transition function, and output function for this RNN are defined as follows:

$$\begin{aligned} \alpha = h_0 &= (q_n^\top k_1, v_1), \\ g(h_{i-1}, w_i) &= \begin{pmatrix} \log(e^{g_{i-1}} + e^{q_n^\top k_i}) \\ v_i \cdot \sigma(q_n^\top k_i - g_{i-1}) + f_{i-1} \cdot \sigma(g_{i-1} - q_n^\top k_i) \end{pmatrix}, \\ \beta(h_n) &= \frac{f_n}{e^{g_n}}. \end{aligned}$$

Here, the input w_i to the RNN at step i is used to retrieve the corresponding key-value pair (k_i, v_i). The output function β performs the final normalization.

Theorem 1 (Equivalence of Attention and RNN). *For any input sequence $(e_1, \ldots, e_n)$ with associated keys and values, the output of the self-attention mechanism at position n (Eq. 1) is equal to the output of the constructed RNN:*

$$h_n^{attention} = \beta(g^*(\alpha, (k_1 v_1, \ldots, k_n v_n))).$$

Proof (Sketch). We prove by induction that after processing the i-th element, the RNN's state contains the correct intermediate values:

$$e^{g_i} = \sum_{j=1}^i e^{q_n^\top k_j}, \qquad f_i = \sum_{j=1}^i v_j e^{q_n^\top k_j}.$$

The base case $i = 1$ holds by the definition of α. For the inductive step:

– The update for g_i is defined as the log-sum-exp of the previous accumulator and the new term, which correctly computes the new log-partition.

– The update for f_i is a weighted sum. Using the identities for the sigmoid function,

$$\sigma(q_n^\top k_i - g_{i-1}) = \frac{e^{q_n^\top k_i}}{e^{g_{i-1}} + e^{q_n^\top k_i}} = \frac{e^{q_n^\top k_i}}{e^{g_i}},$$
$$\sigma(g_{i-1} - q_n^{\top k_i}) = \frac{e^{g_{i-1}}}{e^{g_{i-1}} + e^{q_n^\top k_i}} = \frac{e^{g_{i-1}}}{e^{g_i}}.$$

Substituting these into the transition function shows that:

$$f_i = \frac{e^{q_n^\top k_i}}{e^{g_i}} v_i + \frac{e^{g_{i-1}}}{e^{g_i}} f_{i-1}.$$

By the inductive hypothesis, $f_{i-1} = \sum_{j=1}^{i-1} v_j e^{q_n^\top k_j}$ and $e^{g_{i-1}} = \sum_{j=1}^{i-1} e^{q_n^\top k_j}$. Therefore, this update rule ensures $f_i = \sum_{j=1}^{i} v_j e^{q_n^\top k_j}$.

The output function β then divides f_n by e^{g_n} to produce the final normalized output as in Equation 1, which is precisely the output of the attention mechanism, completing the proof.

Interpretation. This construction shows that the softmax self-attention mechanism in a transformer layer is not inherently non-recurrent. It can be *exactly* computed by an RNN with a carefully designed state and transition function. The RNN's state maintains two accumulators: one for the normalizing denominator and one for the numerator of the attention-weighted sum. This establishes a formal equivalence between the two models for this specific computation.

3.2 Spectral Learning for RNN Approximation

We now reduce the RNN constructed in the previous section, to a WFA using the spectral learning method [2, 3, 21]. The core insight is that the function $f_R : \Sigma^* \to \mathbb{R}$ computed by the RNN defines a Hankel matrix H whose rank reveals the intrinsic state complexity of f_R. If this matrix is low-rank, the function can be compactly represented by a WA [23].

A direct application of the spectral method is infeasible for unbounded sequences. To manage this, we introduce a truncation parameter $\theta \in \mathbb{N}$, restricting our analysis to the set of sequences of length at most θ, denoted $\Sigma^{\leq\theta}$. For any function $f : \Sigma^* \to \mathbb{R}$, we write $f_{\leq\theta}$ for its restriction to this finite domain.

Spectral Learning Procedure. The procedure for converting the RNN R to a WA A is formalized in Algorithm 1 and involves the following steps:

1. **Define Prefix and Suffix Sets:** Fix the truncation length θ. Set the prefix set P and the suffix set S to both be $\Sigma^{\leq\theta}$.
2. **Construct the Hankel Matrix:** Form the Hankel matrix $H \in \mathbb{R}^{P\times S}$, where each entry is defined by the output of the RNN:

$$H[p, s] = f_R(ps), \quad \text{for all } p \in P, s \in S. \tag{2}$$

Constructing H requires evaluating the RNN on all concatenations ps where $|p|, |s| \leq \theta$.

Algorithm 1: CONVERT-RNN-TO-WA(R, θ, r)

Input: RNN R, truncation length θ, target rank r.
Output: WA $A = (\nu, \{M_\sigma\}, \tau)$ of dimension r.

Set $P \leftarrow \Sigma^{\leq\theta}$, $S \leftarrow \Sigma^{\leq\theta}$ and index them arbitrarily.
for $p \in P, s \in S$ **do**
  $H[p, s] \leftarrow f_R(ps)$; `// Construct Hankel matrix (Eq. 2)`
end
Compute rank-r truncated SVD: $H \approx U_r \Sigma_r V_r^\top$
Define vectors $\nu \leftarrow U_r \Sigma_r^{1/2} e_\epsilon$ and $\tau \leftarrow (\Sigma_r^{1/2} V_r^\top)^\top e_\epsilon$; `// (Eq. 3)`
for $\sigma \in \Sigma$ **do**
  for $p \in P, s \in S$ **do**
    $H_\sigma[p, s] \leftarrow f_R(p\sigma s)$; `// Construct shifted matrix (Eq. 4)`
  end
  $M_\sigma \leftarrow \Sigma_r^{-1/2} U_r^\top H_\sigma V_r \Sigma_r^{-1/2}$; `// Project to get transitions (Eq. 5)`
end
return $A = (\nu, \{M_\sigma\}, \tau)$

3. **Compute Low-Rank Approximation:** Compute a rank-r truncated Singular Value Decomposition (SVD) of the Hankel matrix:

$$H \approx U_r \Sigma_r V_r^\top,$$

where r is the target dimension of the WA. The matrices U_r and V_r provide orthonormal bases for the column and row spaces of H, respectively, and Σ_r contains the top r singular values.
4. **Extract WA Parameters:** The WA parameters are derived by projecting the Hankel matrix and its shifts into the low-dimensional space defined by the SVD.
   - **Initial and Final Vectors:** Let e_ϵ be the basis vector corresponding to the empty string. The initial vector ν and final vector τ are defined as:

$$\nu = U_r \Sigma_r^{1/2} e_\epsilon, \quad \tau = (\Sigma_r^{1/2} V_r^\top)^\top e_\epsilon. \tag{3}$$

   - **Transition Matrices:** For each symbol $\sigma \in \Sigma$, form the shifted Hankel matrix H_σ, where:

$$(H_\sigma)[p, s] = f_R(p\sigma s). \tag{4}$$

     The transition matrix M_σ for symbol σ is then obtained by projection:

$$M_\sigma = \Sigma_r^{-1/2} U_r^\top H_\sigma V_r \Sigma_r^{-1/2}. \tag{5}$$

Outcome. The resulting automaton $A = (\nu, \{M_\sigma\}_{\sigma\in\Sigma}, \tau)$ approximates the behavior of the RNN R on sequences up to length θ. If the Hankel matrix H has exact rank r, the recovery by the WA is exact for all $w \in \Sigma^{\leq\theta}$. Otherwise, the approximation error is bounded by the spectral error of the truncated SVD [8].

3.3 Correctness and Analysis of the Reduction

We now establish the correctness of the conversion procedure outlined in Algorithm 1. The following theorem characterizes the approximation guarantees, separating errors arising from truncation and those from spectral approximation.

Theorem 2 (Approximation Guarantees). *Let R be an RNN with induced function $f_R : \Sigma^* \to \mathbb{R}$. Fix a truncation length $\theta > 0$ and let $f_{R,\leq\theta}$ be its restriction to $\Sigma^{\leq\theta}$. Let H be the Hankel matrix indexed by $P = S = \Sigma^{\leq\theta}$ and let A be the WA returned by Algorithm 1 with target rank r. Then:*

1. ***Exact Recovery:*** *If $\operatorname{rank}(H) = r$ and the SVD is computed exactly, then $f_A(w) = f_R(w)$ for all $w \in \Sigma^{\leq\theta}$.*
2. ***Approximate Recovery:*** *If H is approximated by a rank-r matrix $\widehat{H}$ such that $\|H - \widehat{H}\|_2 \leq \eta$, then for every $w \in \Sigma^{\leq\theta}$, the pointwise error is bounded by:*
$$|f_A(w) - f_R(w)| \leq C(\theta, |\Sigma|, r) \cdot \eta, \tag{6}$$
where $C(\theta, |\Sigma|, r)$ is a polynomial factor dependent on the truncation length, alphabet size, and target rank.
3. ***Full-Space Guarantee:*** *For a target distribution Q over Σ^*, if the tail mass beyond length θ is bounded ($\sum_{|w|>\theta} Q(w) \leq \delta$) and the approximation error on $\Sigma^{\leq\theta}$ is bounded ($\|f_A - f_R\|_1 \leq \gamma$), then the WA A satisfies:*
$$\|f_A - Q\|_1 \leq \|f_R - Q\|_1 + \gamma + 2\delta. \tag{7}$$
If the RNN itself is an approximation to Q (i.e., $\|f_R - Q\|_1 \leq \epsilon$), then the total error is $\epsilon + \gamma + 2\delta$.

Proof (Proof Sketch). (1) The exact recovery case is a direct consequence of the spectral learning theory for WFA [2, 3]. When $\operatorname{rank}(H) = r$, the truncated SVD yields a minimal linear representation for the function $f_{R,\leq\theta}$, which is exactly captured by the constructed WA A.

(2) The approximate case follows from perturbation analysis. The error η in the Hankel matrix approximation propagates to the parameters $(\nu, \{M_\sigma\}, \tau)$ of the WA. Standard bounds on the sensitivity of the SVD [8] imply that this propagation is controlled, leading to a pointwise error that is polynomial in the problem parameters and linear in η.

(3) The full-space guarantee is obtained by a decomposition of the ℓ_1 error:

$$\begin{aligned}\|f_A - Q\|_1 &= \sum_{w\in\Sigma^*} |f_A(w) - Q(w)| \\ &\leq \underbrace{\sum_{|w|\leq\theta} |f_A(w) - f_R(w)|}_{\leq\gamma} + \underbrace{\sum_{|w|\leq\theta} |f_R(w) - Q(w)|}_{\leq\epsilon} + \underbrace{2\sum_{|w|>\theta} Q(w)}_{\leq 2\delta}.\end{aligned}$$

The result follows from applying the bounds for each term.

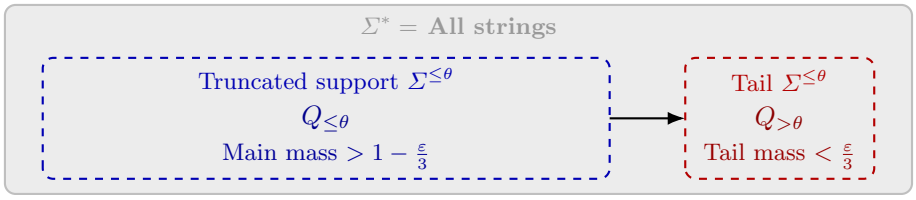

Fig. 3. Schematic of truncation and tail error decomposition. The gray rectangle represents the full domain Σ^*. The blue box is the truncated support $\Sigma^{\leq\theta}$, containing most of the probability mass ($Q_{\leq\theta} > 1 - \varepsilon/3$). Truncation ensures the tail mass $Q_{>\theta}$ is bounded by $\varepsilon/3$, and hence contributes minimally. The total ℓ_1 error is decomposed as γ (estimation) $+C\eta$ (spectral) $+\delta$ (tail mass), highlighting that the bulk of the distribution is concentrated in a manageable region while the tail is small and controlled.

Complexity Analysis. The computational complexity of Algorithm 1 is dominated by the construction of the Hankel matrices and the SVD computation. Let $k = |\Sigma^{\leq\theta}| = O(|\Sigma|^\theta)$. The algorithm requires:

- $O(k^2)$ evaluations of f_R, each requiring $O(\theta \cdot T_{\text{step}})$ time, where T_{step} is the cost of one RNN transition.
- $O(k^3)$ time for the SVD of a $k \times k$ matrix.
- $O(|\Sigma| \cdot k^2)$ memory for storing the Hankel and shifted Hankel matrices.

This exponential dependence on θ limits practical application to small truncation lengths, though sampling techniques can mitigate this cost. [13]

Remarks. This reduction demonstrates that the function computed by the RNN (and hence, by Theorem 1, the self-attention mechanism) can be approximated by a WFA. The quality of approximation is governed by the truncation length θ and the spectral rank r of the finite Hankel matrix. This provides a foundational link between transformers and automata theory, enabling the application of formal methods to analyse their behaviour.

4 Identity Testing for Rational Stochastic Languages

A fundamental verification question is whether the distribution P generated by a black-box model (e.g., an LLM) is close to a known, well-understood reference distribution Q. For our purposes, Q is a rational stochastic language, specified by a weighted automaton or a stochastic regular expression (SRE). This motivates the study of *identity testing* (also known as goodness-of-fit testing) for distributions over infinite domains [6].

In this section, we present an identity testing algorithm for rational stochastic languages, which, to our knowledge, is the first to tackle this problem for infinite domains by leveraging the specific structure of rational distributions. The algorithm reduces the infinite-domain problem to a finite one via a tailored truncation strategy (illustrated in Figure 3), enabling the use of sample-optimal finite-domain testers.

Definition 5 (Identity Testing for Stochastic Languages). *Let Q be a known rational stochastic language, and let P be an unknown distribution over Σ^* accessible only via i.i.d. samples. Given parameters $\varepsilon > 0$ and $\delta \in (0, 1)$, an* identity tester *is an algorithm that, with probability at least $1 - \delta$, outputs:*

- ACCEPT *if $\|P - Q\|_1 < \varepsilon$, and*
- REJECT *if $\|P - Q\|_1 > \varepsilon$.*

The sample complexity of the tester is the number of samples required from P to achieve this guarantee.

We employ the ℓ_1 distance as it corresponds to the total variation distance, providing a robust and interpretable measure of discrepancy across the entire support. This is preferable to the ℓ_∞ distance, which is very sensitive to pointwise errors, and the KL divergence, which requires absolute continuity.

4.1 Reduction to Finite Support via Truncation

The core of our approach is to truncate the distributions to strings of bounded length, ensuring the tail mass of the reference distribution Q is negligible. This reduction allows us to apply finite-domain identity testers.

Definition 6 (Truncation Length θ). *For a distribution Q, define $\theta(Q, \xi)$ as:*

$$\theta(Q, \xi) = \min_{[\![\mathcal{A}]\!]=Q} \left(|\mathcal{A}_Q| \cdot \max \left\{ 1, \left\lceil \frac{\log(\xi)}{\log(1 - \rho(\mathcal{A}))} \right\rceil \right\} \right)$$

where the minimum is taken over all equivalent WFAs $\mathcal{A}$ that represent Q, $|\mathcal{A}_Q|$ is the number of states, and $\rho(\mathcal{A})$ is the spectral radius (maximum eigenvalue) of the automaton's transition matrix.

Lemma 1. *Let $\varepsilon > 0$ be the target accuracy. For the reference distribution Q, the truncation length $\theta = \theta(Q, \varepsilon/3)$ satisfies:*

$$Q_{>\theta} := \sum_{x \in \Sigma^{>\theta}} Q(x) < \frac{\varepsilon}{3}.$$

Proof. Let $\mathcal{A}$ be a WFA realizing Q that achieves the minimum in the definition of $\theta(Q, \varepsilon/3)$. Let n be its number of states and ρ its spectral radius.

By the spectral properties of WFAs, the probability that a string generated by $\mathcal{A}$ has length greater than L decays roughly as $O(\rho^L)$ in the sense of total mass. More formally, there exists a constant C such that for all $L \geq n$,

$$Q_{>L} \leq C \cdot \rho^{L-n+1}.$$

Choosing $L = \theta$ as defined, we have $\theta \geq n \cdot \max \left\{ 1, \left\lceil \frac{\log(\varepsilon/3)}{\log(1-\rho)} \right\rceil \right\}$. In particular, $\theta \geq n \cdot \left\lceil \frac{\log(\varepsilon/3)}{\log(1-\rho)} \right\rceil$ when $\rho < 1$.

Algorithm 2: ℓ_1-IdentityTester for Rational Stochastic Languages

Input: Known distribution Q, sample access to P, parameters $\varepsilon > 0$, $\delta \in (0, 1)$
Output: Accept if $\|P - Q\|_1 < \varepsilon$; Reject if $\|P - Q\|_1 > \varepsilon$ (w.p. $\geq 1 - \delta$)

Compute truncation length $\theta(Q, \varepsilon/3)$.
Let $k \leftarrow |\Sigma^{\leq \theta}| = \Theta(|\Sigma|^{\theta+1})$.
Draw $N = \Theta\left(\frac{\sqrt{k}}{\varepsilon^2} + \frac{k}{\log k} \cdot \log(\frac{1}{\delta})\right)$ i.i.d. samples from P.
Let η be the fraction of samples with $|x| > \theta$.
if $\eta > \varepsilon/3$ **then**
  return Reject ; `// Empirical tail mass is too large`
end
Let $\widehat{P}_{\leq \theta}$ be the empirical distribution of the remaining samples over $\Sigma^{\leq \theta}$.
Let $Q_{\leq \theta}$ be the restriction of Q to $\Sigma^{\leq \theta}$.
Run a finite-domain *tolerant* identity tester on $\widehat{P}_{\leq \theta}$ and $Q_{\leq \theta}$ with distance thresholds $(\varepsilon/3, \varepsilon)$ and confidence $1 - \delta$.
return the output of the finite-domain tester.

Using the inequality $\log(1 - \rho) \leq -\rho$, we get

$$\left\lceil \frac{\log(\varepsilon/3)}{\log(1-\rho)} \right\rceil \geq \frac{\log(1/(\varepsilon/3))}{\rho} = \frac{\log(3/\varepsilon)}{\rho}.$$

Thus $\theta \geq \frac{n \log(3/\varepsilon)}{\rho}$, so $\rho^\theta \leq e^{-n \log(3/\varepsilon)} = (\varepsilon/3)^n$.

Hence $Q_{>\theta} \leq C \cdot (\varepsilon/3)^n \cdot \rho^{-n+1}$. For sufficiently small ε, this is less than $\varepsilon/3$ (adjusting constants if needed by the minimization over WFAs).

4.2 Correctness and Sample Complexity Analysis

We now establish the formal guarantees for Algorithm 2.

Theorem 3 (Correctness of Identity Tester). *Let $Q \in \mathcal{S}^{\mathrm{rat}}_{\mathbb{R}^+}(\Sigma^*)$ be given as an SRE, and let P be an unknown distribution over Σ^*. For any $\varepsilon > 0$ and $\delta \in (0, 1)$, Algorithm 2 satisfies the following with probability at least $1 - \delta$:*

- ***(Completeness)** If $\|P - Q\|_1 < \varepsilon$, then output is* Accept.
- ***(Soundness)** If $\|P - Q\|_1 > \varepsilon$, then output is* Reject.

The sample complexity is:

$$N = \Theta\left(\frac{\sqrt{k}}{\varepsilon^2} + \frac{k}{\log k} \cdot \log\left(\frac{1}{\delta}\right)\right), \quad \text{where } k = |\Sigma^{\leq \theta}|.$$

Proof (Sketch). The proof hinges on the decomposition of the ℓ_1 error and the properties of the finite-domain tolerant tester.

1. **Truncation Guarantee:** By construction, $Q_{>\theta} < \varepsilon/3$.
2. **Tail Mass Check:** If $P_{>\theta} > \varepsilon/3$, then the empirical estimate η will exceed $\varepsilon/3$ with high probability given the sample size N, causing rejection. This handles soundness cases where P diverges from Q in the tail.
3. **Head Analysis:** If the algorithm proceeds to the finite-domain tester, we know $P_{>\theta} \leq \varepsilon/3 + o(1)$ with high probability. The total ℓ_1 distance can be bounded as:
$$\|P - Q\|_1 \leq \underbrace{\|P_{\leq\theta} - Q_{\leq\theta}\|_1}_{\text{Head}} + \underbrace{P_{>\theta} + Q_{>\theta}}_{\text{Tail}}.$$
4. **Completeness ($\|P-Q\|_1 < \varepsilon$):** The tail term is less than $\varepsilon/3+\varepsilon/3 = 2\varepsilon/3$. Thus, $\|P_{\leq\theta} - Q_{\leq\theta}\|_1 < \varepsilon/3$. The tolerant tester, configured with a lower threshold of $\varepsilon/3$, will correctly accept.
5. **Soundness ($\|P - Q\|_1 > \varepsilon$):** Since the tail term is at most $2\varepsilon/3$, the head distance must satisfy $\|P_{\leq\theta} - Q_{\leq\theta}\|_1 > \varepsilon/3$. The tolerant tester, with an upper threshold of ε, will correctly reject.

The sample complexity is dominated by the finite-domain tolerant identity tester [6], which requires $O(\sqrt{k}/\varepsilon^2 + k/\log k)$ samples to distinguish between distances of $\varepsilon/3$ and ε on a domain of size k. The multiplicative $\log(1/\delta)$ factor comes from standard confidence amplification. The number of samples is sufficient to ensure the empirical tail mass check is also correct with probability $1 - \delta$.

Remarks. The algorithm provides a computationally efficient reduction from identity testing on infinite sequences to the finite domain. The sample complexity, while exponential in the truncation length θ, is polynomial in the effective support size $k = |\Sigma^{\leq\theta}|$. For rational stochastic languages with rapidly decaying tails (a common property), θ is manageable, leading to a practical verification procedure for models that are hypothesized to be close to such languages.

5 Conclusion and Future Work

We have established the first formal reduction from decoder-only transformers to weighted automata, creating a bridge between modern neural sequence models and classical automata theory. Our main technical contributions are: (1) a constructive proof that transformer self-attention can be computed by structured RNNs, (2) a spectral learning algorithm that converts such RNNs to weighted automata with provable approximation guarantees, and (3) an identity testing procedure for rational stochastic languages with sample complexity $O(\sqrt{k}/\varepsilon^2 + k\log k)$ where $k = |\Sigma^{\leq\theta}|$.

The Computational Barrier. Our analysis reveals a fundamental bottleneck: spectral learning requires Hankel matrices of size $O(|\Sigma|^{2\theta})$, leading to exponential complexity in the truncation parameter θ. This barrier underscores why automata-based verification of practical language models has remained elusive.

Importantly, this limitation is not merely an artifact of our specific construction. The exponential dependence on θ reflects a deeper tension between the expressivity required to approximate transformer behaviour and the computational tractability of automata-theoretic methods. Any approach that seeks to capture the distributional properties of transformers over realistic vocabularies must confront this fundamental trade-off.

Theoretical Implications. Our results locate transformers within the established hierarchy of computational models. The reduction provides the first formal characterization of transformer expressivity in terms of weighted automata. This connection enables transfer of results: closure properties, decidability results, and algorithmic techniques from automata theory now apply to transformer analysis. Furthermore, our identity testing algorithm provides a principled statistical framework for model comparison and verification.

Algorithmic Challenges and Directions. The exponential bottleneck suggests several concrete research directions:

1. Develop low-rank approximations or sketching techniques for learning that preserve essential spectral properties while reducing computational cost.
2. Design randomized algorithms that estimate automaton parameters without constructing full Hankel matrices, potentially achieving polynomial dependence on θ.
3. Exploit specific properties of transformer architectures (attention patterns, positional encodings, etc.) to reduce the effective dimension of the learning problem, and make it tractable.

We acknowledge that these challenges have not allowed us to implement and evaluate our approach on real-world transformers, and it remains a direction of future work. Beyond this, Several fundamental questions remain: Can the exponential dependence on θ be avoided while preserving approximation quality? What is the minimal class of weighted automata needed to approximate practical transformers? How do architectural choices (depth, attention patterns, normalization) affect the complexity of automata-based approximation?

In conclusion, our framework has provided new evaluation methodologies for language models. Rather than relying solely on empirical benchmarks, our identity testing approach provides statistical certificates of distributional similarity to reference models—a step toward more rigorous model validation. Through this work, we provide both a foundational connection between transformers and automata theory and a precise characterization of the computational challenges that must be overcome for practical verification. The path forward requires algorithmic innovation at the intersection of spectral methods, randomized algorithms, and formal language theory.

References

1. Angluin, D., Chiang, D., Yang, A.: Masked hard-attention transformers and boolean rasp recognize exactly the star-free languages. arXiv preprint arXiv:2310.13897 (2023)
2. Balle, B., Mohri, M.: Spectral learning of general weighted automata via constrained matrix completion. In: Advances in Neural Information Processing Systems (NeurIPS). pp. 2168–2176 (2012)
3. Balle, B., Mohri, M.: Learning weighted automata. arXiv preprint arXiv:2007.08131 (2020)
4. Bollig, B., Zeitoun, M.: Weighted automata. Lecture Notes, ENS Cachan (2015), http://www.lsv.fr/ mz/wa.pdf, version of September 13, 2015
5. Bowen, C., Sætre, R., Miyao, Y.: A comprehensive evaluation of inductive reasoning in llms. In: Findings of the Association for Computational Linguistics: EACL 2024 (2024), https://aclanthology.org/2024.findings-eacl.22
6. Canonne, C.L., Jain, A., Kamath, G., Li, J.: The price of tolerance in distribution testing. In: Loh, P.L., Raginsky, M. (eds.) Proceedings of the 35th Annual Conference on Learning Theory (COLT). Proceedings of Machine Learning Research, vol. 178, pp. 1–52. PMLR (2022), https://proceedings.mlr.press/v178/canonne22a.html
7. Cohen-Inger, N., Elisha, Y., Shapira, B., Rokach, L., Cohen, S.: Forget what you know about llms evaluations - llms are like a chameleon. arXiv preprint arXiv:2502.07445 (2025), https://arxiv.org/abs/2502.07445
8. Davis, C., Kahan, W.M.: The rotation of eigenvectors by a perturbation. iii. SIAM Journal on Numerical Analysis **7**(1), 1–46 (1970)
9. Delattre, B., Araujo, A., Barthélemy, Q., Allauzen, A.: The lipschitz-variance-margin tradeoff for enhanced randomized smoothing (2023), https://arxiv.org/abs/2309.16883
10. Deng, C., Zhao, Y., Tang, X., Gerstein, M., Cohan, A.: Investigating data contamination in modern benchmarks for large language models. arXiv preprint arXiv:2311.09783 (2023), https://arxiv.org/abs/2311.09783
11. Ducotterd, S., Goujon, A., Bohra, P., Perdios, D., Neumayer, S., Unser, M.: Improving lipschitz-constrained neural networks by learning activation functions. Journal of Machine Learning Research **25**(83), 1–30 (2024), https://jmlr.org/papers/volume25/22-1347/22-1347.pdf
12. Hao, Y., Angluin, D., Frank, R.: Formal language recognition by hard attention transformers: Perspectives from circuit complexity. Transactions of the Association for Computational Linguistics **10**, 800–810 (2022). https://doi.org/10.1162/tacl˙a˙00490
13. Hsu, D., Kakade, S.M., Zhang, T.: A spectral algorithm for learning hidden markov models. In: Proceedings of the 22nd Annual Conference on Learning Theory (COLT). pp. 403–414. Omnipress (2009), https://arxiv.org/abs/0811.4413
14. Hsu, D., Kakade, S.M., Zhang, T.: A spectral algorithm for learning hidden markov models. Journal of Computer and System Sciences **78**(5), 1460–1480 (2012)
15. Hua, W., Wong, T., Fei, S., Pan, L., Jardine, A., Wang, W.Y.: Inductionbench: Llms fail in the simplest complexity class. arXiv preprint arXiv:2502.15823 (2025), https://arxiv.org/abs/2502.15823
16. Huster, T., Chiang, C.J., Chadha, R.: Limitations of the lipschitz constant as a defense against adversarial examples. In: ECML PKDD 2018 Workshops. Lecture Notes in Computer Science, vol. 11329, pp. 16–29. Springer (2019). https://doi.org/10.1007/978-3-030-13453-2“˙2

17. Khromov, G., Singh, S.P.: Some fundamental aspects about lipschitz continuity of neural networks. In: International Conference on Learning Representations (ICLR) (2024)
18. Li, J., Cao, P., Jin, Z., Chen, Y., Liu, K., Zhao, J.: Mirage: Evaluating and explaining inductive reasoning in llms. arXiv preprint arXiv:2410.09542 (2024), https://arxiv.org/abs/2410.09542
19. Merrill, W., Weiss, G., Goldberg, Y., Schwartz, R., Smith, N.A., Yahav, E.: A formal hierarchy of rnn architectures. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. pp. 443–459. Association for Computational Linguistics (2020)
20. Newhouse, L., Hess, R.P., Cesista, F., Zahorodnii, A., Bernstein, J., Isola, P.: Training transformers with enforced lipschitz constants (2025). https://doi.org/10.48550/arXiv.2507.13338, https://arxiv.org/abs/2507.13338
21. Okudono, T., Waga, M., Sekiyama, T., Hasuo, I.: Weighted automata extraction from recurrent neural networks via regression on state spaces. ArXiv **abs/1904.02931** (2019), https://api.semanticscholar.org/CorpusID:102350402
22. Qi, X., Wang, J., Chen, Y., Shi, Y., Zhang, L.: Lipsformer: Introducing lipschitz continuity to vision transformers (2023), https://arxiv.org/abs/2304.09856
23. Schützenberger, M.P.: On the definition of a family of automata. Information and Control **4**(2-3), 245–270 (1961)
24. Shojaee, P., Mirzadeh, I., Alizadeh, K., Horton, M., Bengio, S., Farajtabar, M.: The illusion of thinking: Understanding the strengths and limitations of reasoning models via the lens of problem complexity. arXiv preprint arXiv:2506.06941 (2025), https://arxiv.org/abs/2506.06941
25. Strobl, L., Merrill, W., Weiss, G., Chiang, D., Angluin, D.: What formal languages can transformers express? a survey. arXiv preprint arXiv:2311.00208 (2023)
26. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in Neural Information Processing Systems (NeurIPS). vol. 30. Curran Associates, Inc. (2017)
27. Verine, A., Negrevergne, B., Chevaleyre, Y., Rossi, F.: On the expressivity of bi-lipschitz normalizing flows. In: Proceedings of The 14th Asian Conference on Machine Learning (ACML). Proceedings of Machine Learning Research, vol. 189 (2022), https://proceedings.mlr.press/v189/verine23a/verine23a.pdf
28. Xu, C., Guan, S., Greene, D., Kechadi, M.: Benchmark data contamination of large language models: A survey. arXiv preprint arXiv:2406.04244 (2024), https://arxiv.org/abs/2406.04244
29. Zhang, B., Jiang, D., He, D., Wang, L.: Rethinking lipschitz neural networks and certified robustness: A boolean function perspective. In: Advances in Neural Information Processing Systems (NeurIPS) (2022)
30. Zhang, Y., Albarghouthi, A., D'Antoni, L.: A one-layer decoder-only transformer is a two-layer rnn: With an application to certified robustness (2024), https://arxiv.org/abs/2405.17361