# Simulating Weighted Automata with Transformers

## (Over both sequences and trees!)

Michael Rizvi [13] Maude Lizaire [13] Clara Lacroce [23] Guillaume Rabusseau [13]

[1]Université de Montréal [2]McGill University [3]Mila

TAUDos, June 2024

# Table of Contents

# Table of Contents

# Introduction

- **Objective:** show sequential reasoning capacities of Transformer architecture

# Introduction

- **Objective:** show sequential reasoning capacities of Transformer architecture
- Result on the **expressivity** of the architecture not **learnability**!

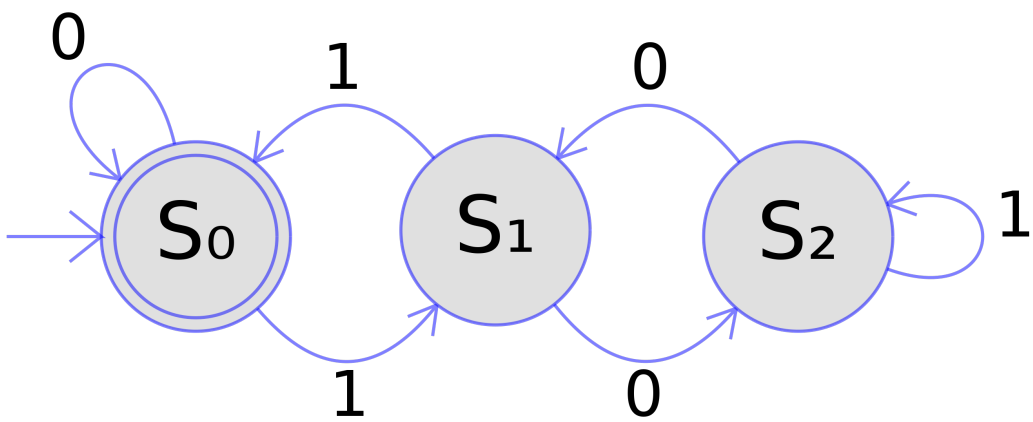
# Introduction

- **Objective:** show sequential reasoning capacities of Transformer architecture
- Result on the **expressivity** of the architecture not **learnability**!
- Take the lense of **simulation**

# What do we mean by simulation?

- Simulation = showing steps
- Previous results by Liu et al. introduce this idea for DFA
- For some DFA $\mathcal{A}$ over $\Sigma$:
  - Input: $w \in \Sigma^*$
  - Output: sequence of visited states

Example of a DFA for multiples of 3 on $\Sigma = \{0, 1\}$



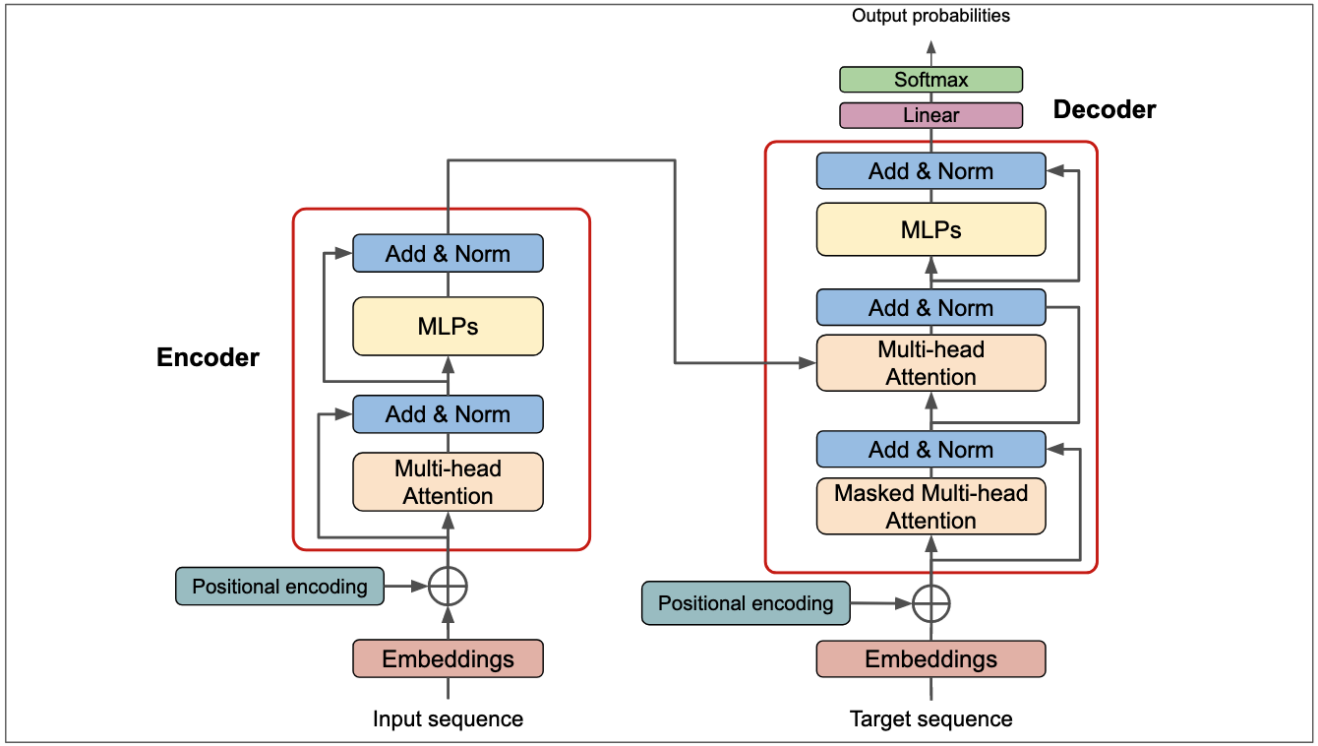
# What do we mean by simulation?

- Liu et al. showed transformers can **simulate DFA** up to length $T$ with $\mathcal{O} \log T$ layers (even $\mathcal{O}(1)$ in some cases!)
- Notion of shortcuts: **shallow** transformers w.r.t. $T$
- General idea of the theorem
  - Input: a DFA and some sequence length $T$
  - Output: a transformer which can simulate the inner working of DFA for *any word* of length $T$
- Can we do this for **more complex models**?

# Table of Contents

# Transformers

Transformer architecture in our construction is similar to the **encoder in the original transformer architecture**.



# Transformers

The model is defined as follows

- Input: $X \in \mathbb{R}^{T \times d}$ where $T$ is sequence length and $d$ is embedding dimension
- Self-attention block:

$$f(\mathbf{X}) = \text{softmax}(\mathbf{X}\mathbf{W}_Q\mathbf{W}_K^\top\mathbf{X}^\top)\mathbf{X}\mathbf{W}_V,$$

- Attention layer $f_{\text{attn}}$: $h$ copies of $f$, concatenate the outputs
- Feedforward layer $f_{\text{mlp}}$: Simple feedforward MLP

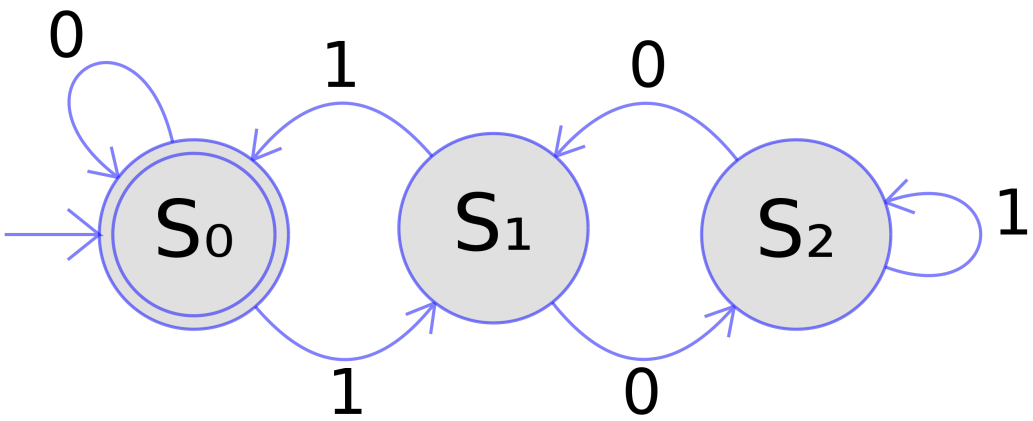
Full $L$-layer model, with $f_{\text{tf}} : \mathbb{R}^{T \times d} \to \mathbb{R}^{T \times d}$ :

$$f_{\text{tf}} = f_{\text{mlp}}^{(L)} \circ f_{\text{attn}}^{(L)} \circ f_{\text{mlp}}^{(L-1)} \circ f_{\text{attn}}^{(L-1)} \circ \ldots \circ f_{\text{mlp}}^{(1)} \circ f_{\text{attn}}^{(1)}.$$

# Table of Contents

# Weighted Finite Automata

- Weighted Finite Automata (WFA) generalize DFAs by **computing a function** over some word $w$ (instead of simply accepting/rejecting)



# Weighted Finite Automata

A *weighted finite automaton* (WFA) of $n$ states over $\Sigma$ is a tuple $\mathcal{A} = \langle \boldsymbol{\alpha}, \{\mathbf{A}^\sigma\}_{\sigma \in \Sigma}, \boldsymbol{\beta} \rangle$, where

- $\boldsymbol{\alpha}$, $\boldsymbol{\beta} \in \mathbb{R}^n$: initial/final weights
- $\mathbf{A}^\sigma \in \mathbb{R}^{n \times n}$: transition matrix for each $\sigma \in \Sigma$

# Weighted Finite Automata

A *weighted finite automaton* (WFA) of $n$ states over $\Sigma$ is a tuple $\mathcal{A} = \langle \boldsymbol{\alpha}, \{\mathbf{A}^\sigma\}_{\sigma\in\Sigma}, \boldsymbol{\beta}\rangle$, where

- $\boldsymbol{\alpha}$, $\boldsymbol{\beta} \in \mathbb{R}^n$: initial/final weights
- $\mathbf{A}^\sigma \in \mathbb{R}^{n\times n}$: transition matrix for each $\sigma \in \Sigma$

WFA $\mathcal{A}$ computes a function $f_A : \Sigma^* \to \mathbb{R}$:

$$f_{\mathcal{A}}(x) = f_{\mathcal{A}}(x_1 \cdots x_t) = \boldsymbol{\alpha}^\top \mathbf{A}^{x_1} \cdots \mathbf{A}^{x_t} \boldsymbol{\beta} = \boldsymbol{\alpha}^\top \mathbf{A}^{x} \boldsymbol{\beta}$$

# Weighted Finite Automata

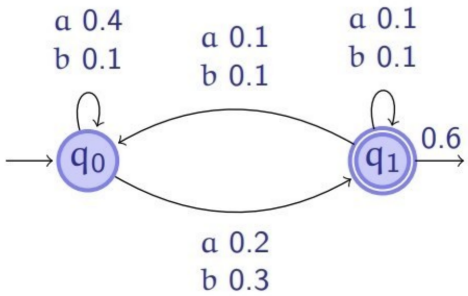
A *weighted finite automaton* (WFA) of $n$ states over $\Sigma$ is a tuple $\mathcal{A} = \langle \boldsymbol{\alpha}, \{\mathbf{A}^\sigma\}_{\sigma\in\Sigma}, \boldsymbol{\beta}\rangle$, where

- $\boldsymbol{\alpha}$, $\boldsymbol{\beta} \in \mathbb{R}^n$: initial/final weights
- $\mathbf{A}^\sigma \in \mathbb{R}^{n\times n}$: transition matrix for each $\sigma \in \Sigma$

WFA $\mathcal{A}$ computes a function $f_A : \Sigma^* \to \mathbb{R}$:

$$f_{\mathcal{A}}(x) = f_{\mathcal{A}}(x_1 \cdots x_t) = \boldsymbol{\alpha}^\top \mathbf{A}^{x_1} \cdots \mathbf{A}^{x_t} \boldsymbol{\beta} = \boldsymbol{\alpha}^\top \mathbf{A}^{x} \boldsymbol{\beta}$$

**Example** Consider the following WFA with **2** states on $\Sigma = \{a, b\}$




Operator Representation

$$\boldsymbol{\alpha} = \begin{bmatrix}1.0\\0.0\end{bmatrix} \quad \mathbf{A}^{a} = \begin{bmatrix}0.4 & 0.2\\0.1 & 0.1\end{bmatrix}$$

$$\boldsymbol{\omega} = \begin{bmatrix}0.0\\0.6\end{bmatrix} \quad \mathbf{A}^{b} = \begin{bmatrix}0.1 & 0.3\\0.1 & 0.1\end{bmatrix}$$

$$f(ab) = 0.4 \times 0.3 \times 0.6 + 0.2 \times 0.1 \times 0.6 = 0.084$$

$$= \boldsymbol{\alpha}^\top \mathbf{A}^{a}\, \mathbf{A}^{b}\, \boldsymbol{\omega}$$

# Table of Contents

# Simulating WFA

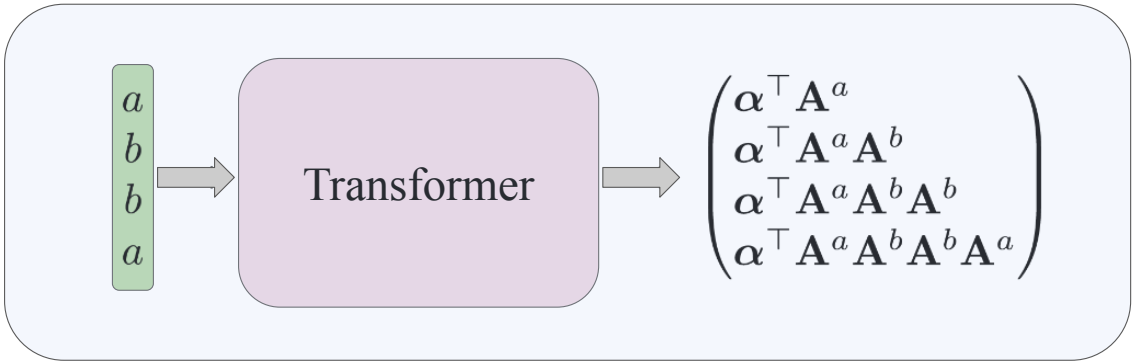
## Exact Simulation

Given a WFA $\mathcal{A}$ over some alphabet $\Sigma$, a function $f : \Sigma^T \to \mathbb{R}^{T \times n}$ *exactly* simulates $\mathcal{A}$ at length $T$ if, for all $x \in \Sigma^T$ as input, we have $f(x) = \mathcal{A}(x)$, where $\mathcal{A}(x) = (\boldsymbol{\alpha}^\top, \boldsymbol{\alpha}^\top \mathbf{A}^{x_1}, \ldots, \boldsymbol{\alpha}^\top \mathbf{A}^{x_{1:T}})^\top$.

# Simulating WFA

## Exact Simulation

Given a WFA $\mathcal{A}$ over some alphabet $\Sigma$, a function $f : \Sigma^T \to \mathbb{R}^{T \times n}$ *exactly* simulates $\mathcal{A}$ at length $T$ if, for all $x \in \Sigma^T$ as input, we have $f(x) = \mathcal{A}(x)$, where $\mathcal{A}(x) = (\boldsymbol{\alpha}^\top, \boldsymbol{\alpha}^\top \mathbf{A}^{x_1}, \ldots, \boldsymbol{\alpha}^\top \mathbf{A}^{x_{1:T}})^\top$.



# Simulating WFA

## Approximate Simulation

Given a WFA $\mathcal{A}$ over some alphabet $\Sigma$, a function $f : \Sigma^T \to \mathbb{R}^{T \times n}$ *approximately* simulates $\mathcal{A}$ at length $T$ with precision $\epsilon > 0$ if for all $x \in \Sigma^T$, we have $\|f(x) - \mathcal{A}(x)\|_F < \epsilon$.

# Main Results for WFAs

First Theorem: **exact** simulation:

## Theorem 1

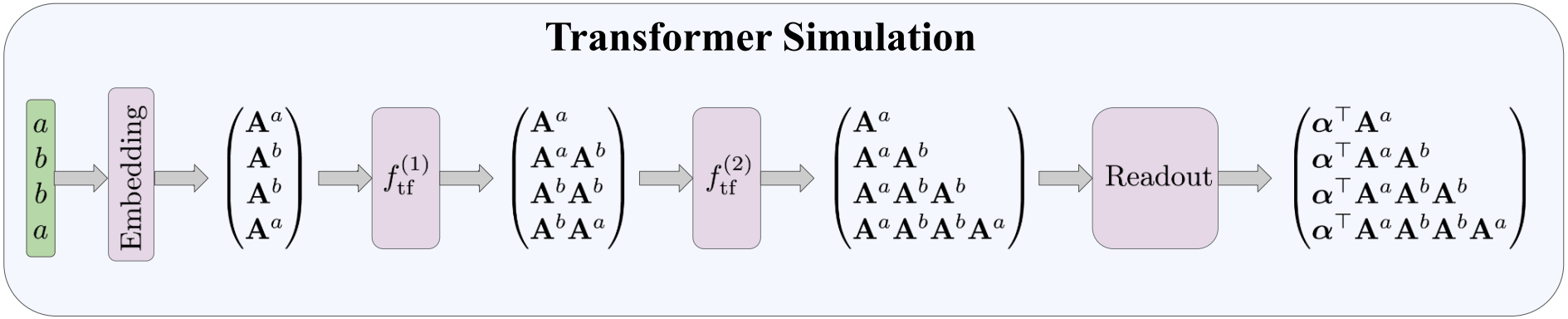
**Theorem 1** Transformers using bilinear layers in place of an MLP and hard attention can *exactly* simulate all WFAs with $n$ states at length $T$, with depth $\mathcal{O}(\log T)$, embedding dimension $\mathcal{O}(n^2)$, attention width $\mathcal{O}(n^2)$, MLP width $\mathcal{O}(n^2)$ and $\mathcal{O}(1)$ attention heads.

# Main Results for WFAs

First Theorem: **exact** simulation:

## Theorem 1

**Theorem 1** Transformers using bilinear layers in place of an MLP and hard attention can *exactly* simulate all WFAs with $n$ states at length $T$, with depth $\mathcal{O}(\log T)$, embedding dimension $\mathcal{O}(n^2)$, attention width $\mathcal{O}(n^2)$, MLP width $\mathcal{O}(n^2)$ and $\mathcal{O}(1)$ attention heads.



# Main Results for WFAs

Second Theorem: **approximate** simulation

## Theorem 2

Transformers can *approximately* simulate all WFAs with $n$ states at length $T$, up to arbitrary precision $\epsilon > 0$, with depth $\mathcal{O}(\log T)$, embedding dimension $\mathcal{O}(n^2)$, attention width $\mathcal{O}(n^2)$, MLP width $\mathcal{O}(n^4)$ and $\mathcal{O}(1)$ attention heads.

# Main Results for WFAs

Second Theorem: **approximate** simulation

## Theorem 2

Transformers can *approximately* simulate all WFAs with $n$ states at length $T$, up to arbitrary precision $\epsilon > 0$, with depth $\mathcal{O}(\log T)$, embedding dimension $\mathcal{O}(n^2)$, attention width $\mathcal{O}(n^2)$, MLP width $\mathcal{O}(n^4)$ and $\mathcal{O}(1)$ attention heads.

## Remark

Notice how in Theorem 2, the size of the construction **does not** depend on $\epsilon$!

# Main Results for WFAs

Second Theorem: **approximate** simulation

## Theorem 2

Transformers can *approximately* simulate all WFAs with $n$ states at length $T$, up to arbitrary precision $\epsilon > 0$, with depth $\mathcal{O}(\log T)$, embedding dimension $\mathcal{O}(n^2)$, attention width $\mathcal{O}(n^2)$, MLP width $\mathcal{O}(n^4)$ and $\mathcal{O}(1)$ attention heads.

## Remark

Notice how in Theorem 2, the size of the construction **does not** depend on $\epsilon$!

**Theorem 4.** *(abridged version of Theorem 3.1 of (Chong, 2020)) Let $d \geq 2$ be an integer, let $f \in \mathcal{P}_{\leq d}(\mathbb{R}^{m_1}, \mathbb{R}^{m_2})$ and let $\rho^{\sigma}_{\Theta}$ be a two-layer MLP with activation function $\sigma$ and parameters $\Theta = (\mathbf{W}_1, \mathbf{W}_2)$. If $\sigma \in \mathcal{C}(\mathbb{R}) \setminus \mathcal{P}_{\leq d-1}$, then for every $\epsilon > 0$, there exists some $\Theta \in \{(\mathbf{W}_1, \mathbf{W}_2) \mid \mathbf{W}_1 \in \mathbb{R}^{m_1 \times N}, \mathbf{W}_2 \in \mathbb{R}^{N \times m_2}\}$ with $N = \binom{m_1+d}{d}$ such that $\|f - \rho^{\sigma}_{\Theta}\|_{\infty} < \epsilon$.*

# Table of Contents

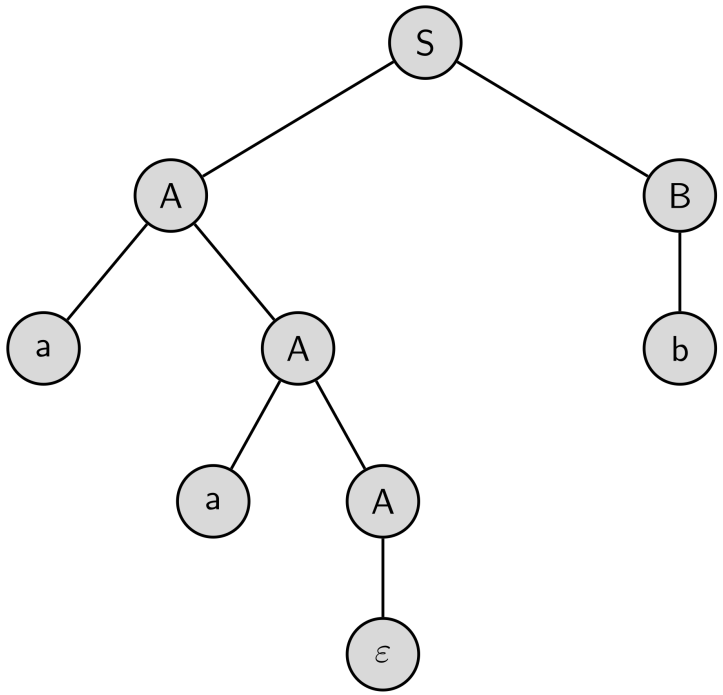
# Weighted Tree Automata

- **Intuition**: Same thing as WFAs but for tree-structured inputs!

# Weighted Tree Automata

## Binary Trees

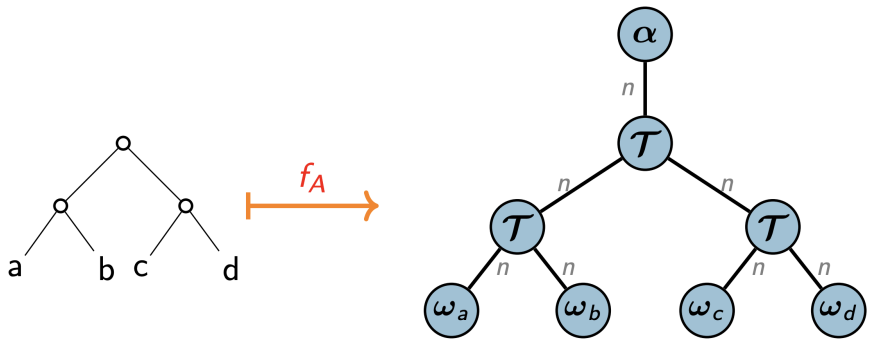
Given a finite alphabet $\Sigma$, the set of binary trees with leafs labeled by symbols in $\Sigma$ is denoted by $\mathscr{T}_\Sigma$. Formally, $\mathscr{T}_\Sigma$ is the smallest set such that $\Sigma \subset \mathscr{T}_\Sigma$ and $(t_1, t_2) \in \mathscr{T}_\Sigma$ for all $t_1, t_2 \in \mathscr{T}_\Sigma$.



## Weighted Tree Automata

A weighted tree automaton (WTA) $\mathcal{A}$ with $n$ states on $\mathscr{T}_\Sigma$ is a tuple $\langle \boldsymbol{\alpha} \in \mathbb{R}^n, \boldsymbol{\mathcal{T}} \in \mathbb{R}^{n \times n \times n}, \{\mathbf{v}_\sigma \in \mathbb{R}^n\}_{\sigma \in \Sigma} \rangle$. A WTA $\mathcal{A}$ computes a function $f_\mathcal{A} : \mathscr{T}_\Sigma \to \mathbb{R}$ defined by $f_\mathcal{A}(t) = \langle \boldsymbol{\alpha}, \mu(t) \rangle$ where the mapping $\mu : \mathscr{T}_\Sigma \to \mathbb{R}^n$ is recursively defined by

- $\mu(\sigma) = \mathbf{v}_\sigma$ for all $\sigma \in \Sigma$,
- $\mu((t_1, t_2)) = \boldsymbol{\mathcal{T}} \times_2 \mu(t_1) \times_3 \mu(t_2)$ for all $t_1, t_2 \in \mathscr{T}_\Sigma$.



# Weighted Tree Automata

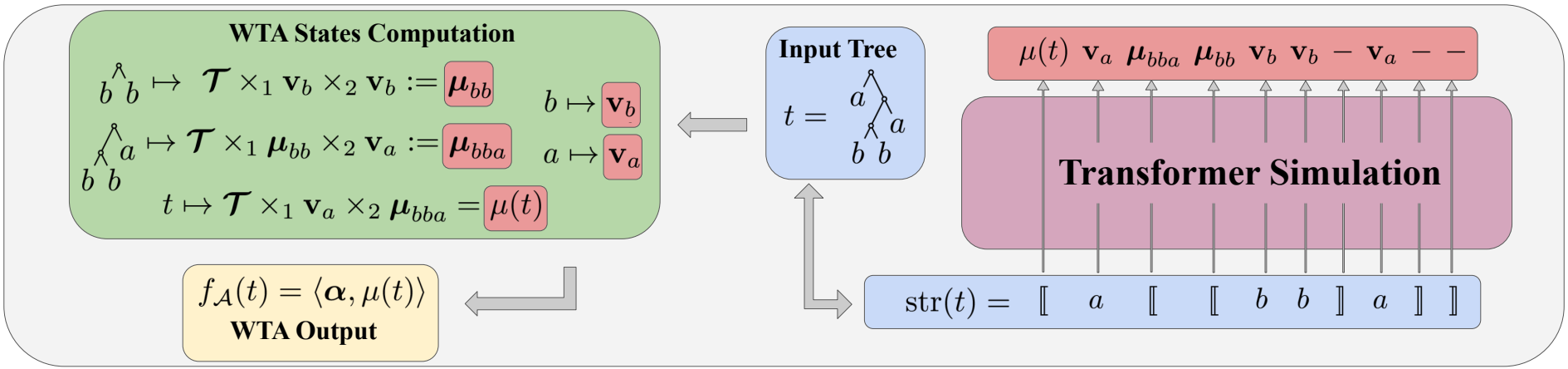
## Simulation by a function

Given a WTA $\mathcal{A} = \langle \boldsymbol{\alpha}, \boldsymbol{\mathcal{T}}, \{\mathbf{v}_\sigma\}_{\sigma \in \Sigma} \rangle$ with $n$ states on $\mathscr{T}_\Sigma$, we say that a function $f : (\Sigma \cup \{[\![, ]\!]\})^T \to (\mathbb{R}^n)^T$ simulates $\mathcal{A}$ at length $T$ if for all trees $t \in \mathscr{T}_\Sigma$ such that $|\mathrm{str}(t)| \leq T$, $f(\mathrm{str}(t))_i = \mu(\tau_i)$ for all $i \in \mathcal{I}_t$.

# Weighted Tree Automata

## Simulation by a function

Given a WTA $\mathcal{A} = \langle \boldsymbol{\alpha}, \boldsymbol{\mathcal{T}}, \{\mathbf{v}_\sigma\}_{\sigma \in \Sigma} \rangle$ with $n$ states on $\mathscr{T}_\Sigma$, we say that a function $f : (\Sigma \cup \{[\![, ]\!]\})^T \to (\mathbb{R}^n)^T$ simulates $\mathcal{A}$ at length $T$ if for all trees $t \in \mathscr{T}_\Sigma$ such that $|\mathrm{str}(t)| \leq T$, $f(\mathrm{str}(t))_i = \mu(\tau_i)$ for all $i \in \mathcal{I}_t$.



# Table of Contents

# Main Results for WTAs

## Theorem 3

Transformers can *approximately* simulate all WTAs $\mathcal{A}$ with $n$ states at length $T$, up to arbitrary precision $\epsilon > 0$, with embedding dimension $\mathcal{O}(n)$, attention width $\mathcal{O}(n)$, MLP width $\mathcal{O}(n^3)$ and $\mathcal{O}(1)$ attention heads. Moreover:

- Simulation over arbitrary trees can be done with depth $\mathcal{O}(T)$
- Simulation over balanced trees (trees whose depth is of order $\log(T)$) with depth $\mathcal{O}(\log(T))$.

# Main Results for WTAs

## Theorem 3

Transformers can *approximately* simulate all WTAs $\mathcal{A}$ with $n$ states at length $T$, up to arbitrary precision $\epsilon > 0$, with embedding dimension $\mathcal{O}(n)$, attention width $\mathcal{O}(n)$, MLP width $\mathcal{O}(n^3)$ and $\mathcal{O}(1)$ attention heads. Moreover:

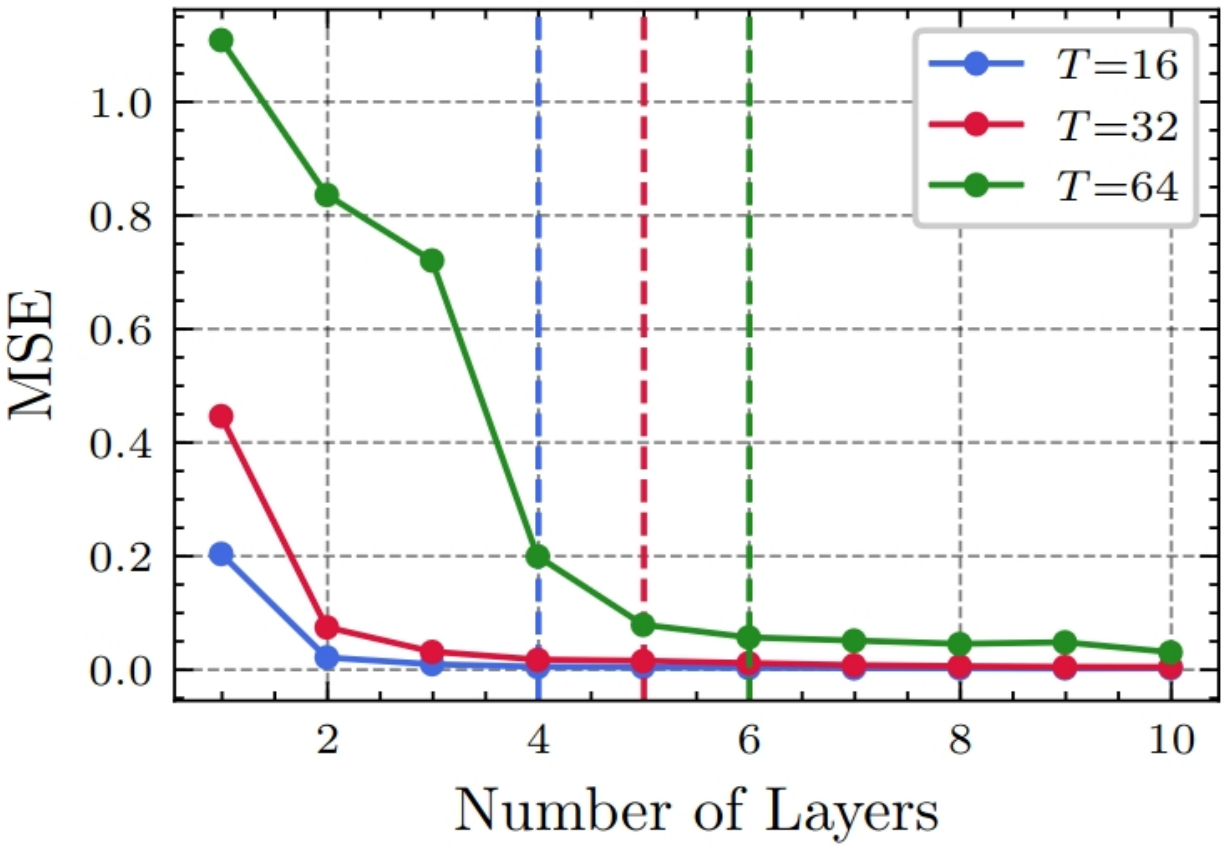
- Simulation over arbitrary trees can be done with depth $\mathcal{O}(T)$
- Simulation over balanced trees (trees whose depth is of order $\log(T)$) with depth $\mathcal{O}(\log(T))$.

## Remark

In the worst case, the tree is completely unbalanced in which case we recover the sequential WFA case!

# Table of Contents

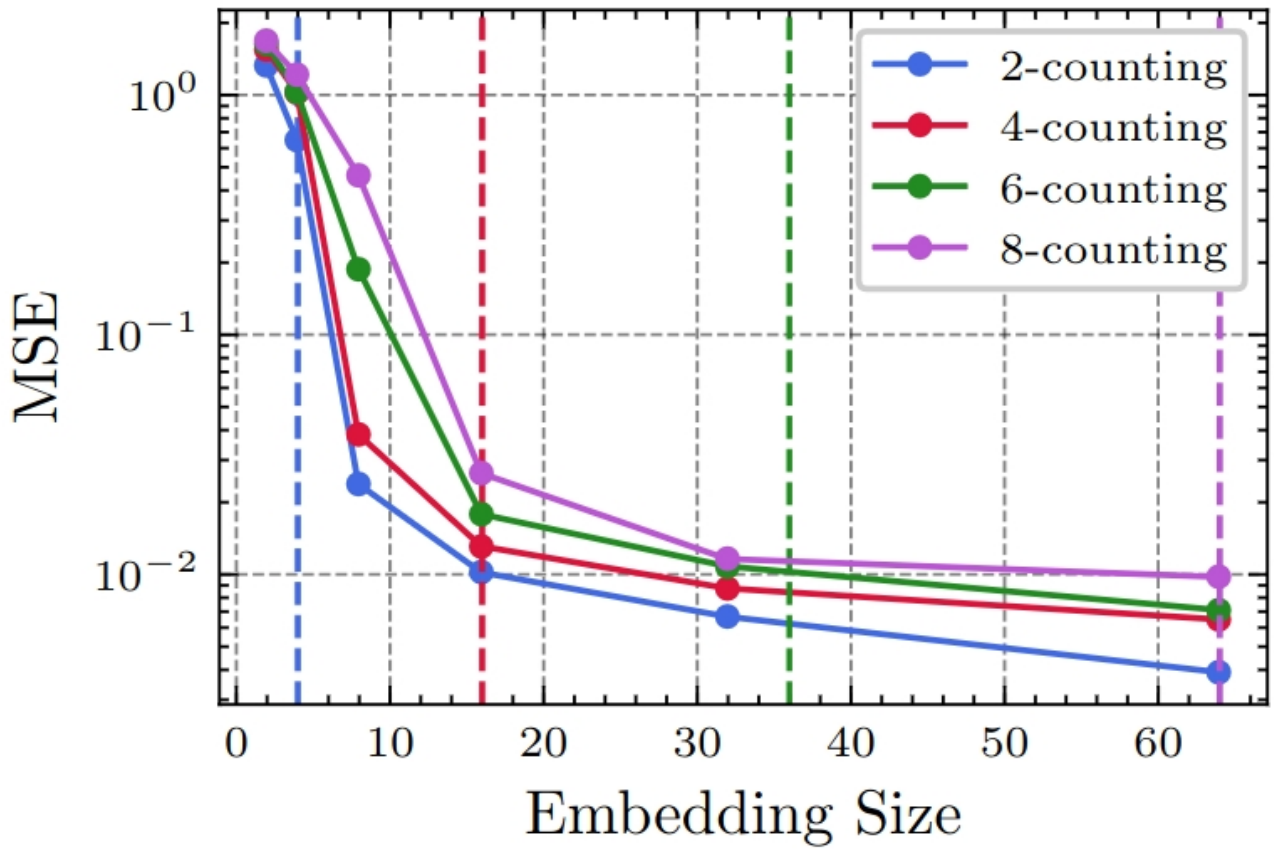
# Depth vs. Length




target: 2 states WFA counting 0's in binary strings

theory: $\log T$ layers for sequences of length $T$

# Width vs. #States




target: $k$ states WFAs counting $k$ symbols in a string

theory: $n^2$ width to simulate WFA with $n$ states

# Table of Contents

# Conclusion

- We define simulation of **weighted automata** for **sequences and trees**
- We derive the notion of **approximate simulation** and how it applies to transformers
- We show that transformers can **simulate WFAs with** $\mathcal{O}(\log T)$ **layers**
- We show transformers can **simulate WTAs with** $\mathcal{O}(\log T)$ **layers**
- Our results extend the ones of Liu et al. for DFAs in **two directions**: from **boolean to real weights** and from **sequences to trees**

# Conclusion

Thank you for listening! :)
Questions?