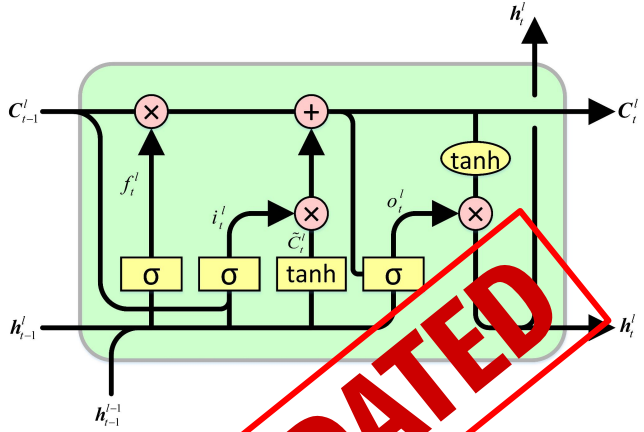
The Expressive Capacity of State Space Models: A Formal Language Perspective

Yash Sarrof, Yana Veitsman, and Michael Hahn
Saarland University, Germany
NeurIPS ‘24 (Poster)

Transformers are great!

Transformers are great!

But sometimes they are not.



OUTDATED

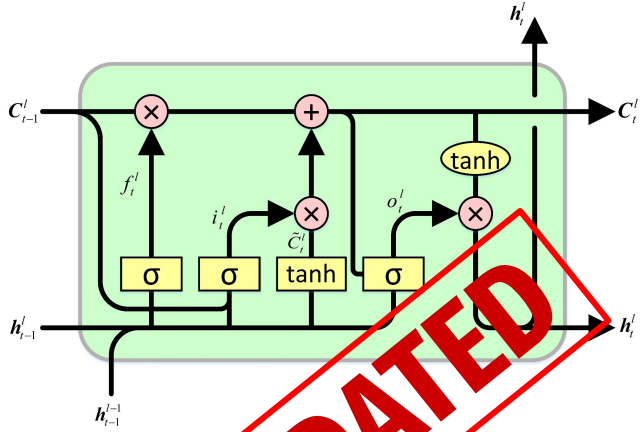
LONG SHORT-TERM MEMORY

NEURAL COMPUTATION 9(8):1735–1780, 1997

Sepp Hochreiter
Fakultät für Informatik
Technische Universität München
80290 München, Germany
hochreit@informatik.tu-muenchen.de
http://www7.informatik.tu-muenchen.de/~hochreit

Jürgen Schmidhuber
IDSIA
Corso Elvezia 36
6900 Lugano, Switzerland
juergen@idsia.ch
http://www.idsia.ch/~juergen

And LSTMs are impractical and outdated



LONG SHORT-TERM MEMORY

NEURAL COMPUTATION 9(8):1735–1780, 1997

OUTDATED

Sepp Hochreiter
Fakultät für Informatik
Technische Universität München
80290 München, Germany
hochreit@informatik.tu-muenchen.de
http://www7.informatik.tu-muenchen.de/~hochreit

Jürgen Schmidhuber
IDSIA
Corso Elvezia 36
6900 Lugano, Switzerland
juergen@idsia.ch
http://www.idsia.ch/~juergen

And LSTMs are impractical and outdated

But on some tasks they are better than Transformers!

Meanwhile State Space Models are shiny and promising

Meanwhile State Space Models are shiny and promising

But they desperately fail some tasks that are easy for Transformers!

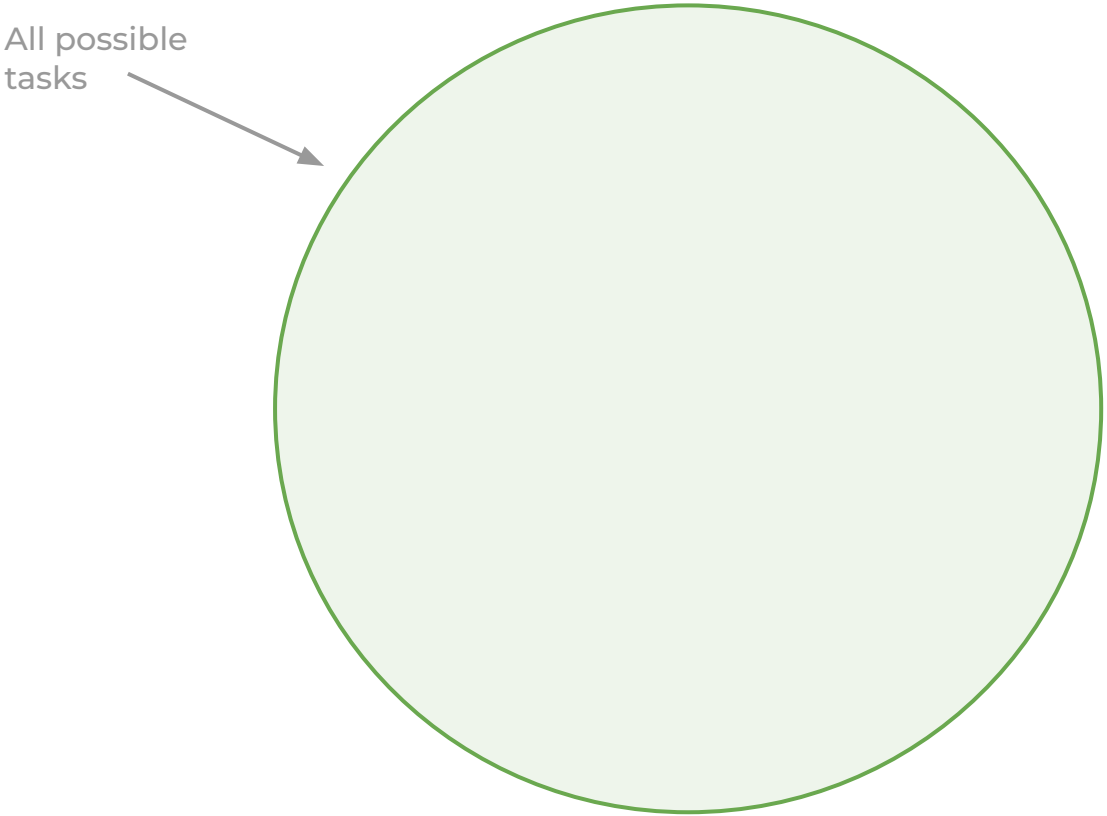
Then how can we even say that one architecture is better than the others?

Then how can we even say that one architecture is better than the others?

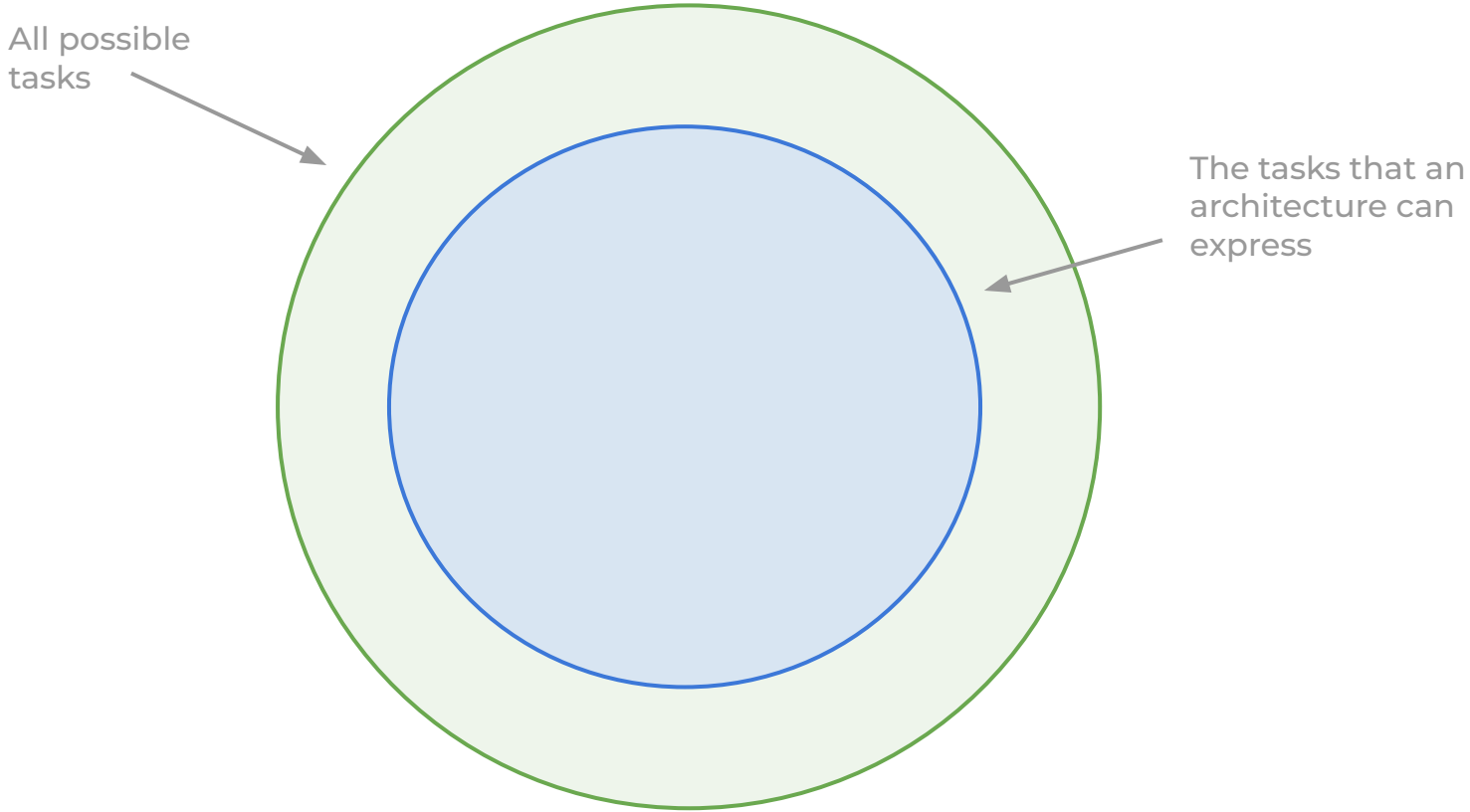
We can’t! But different architectures can solve different tasks.

And theoretical analysis is here to help!

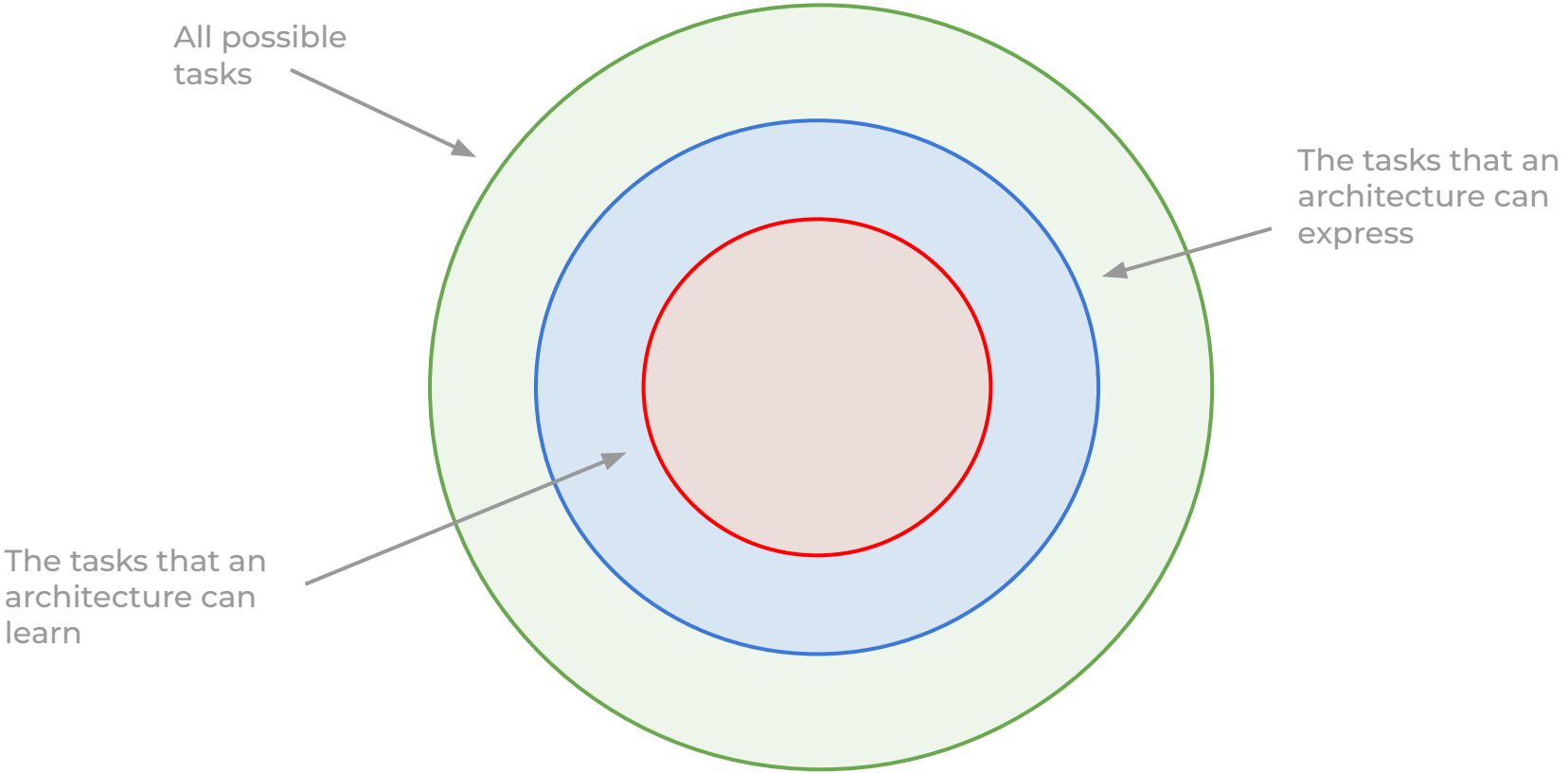
The research questions



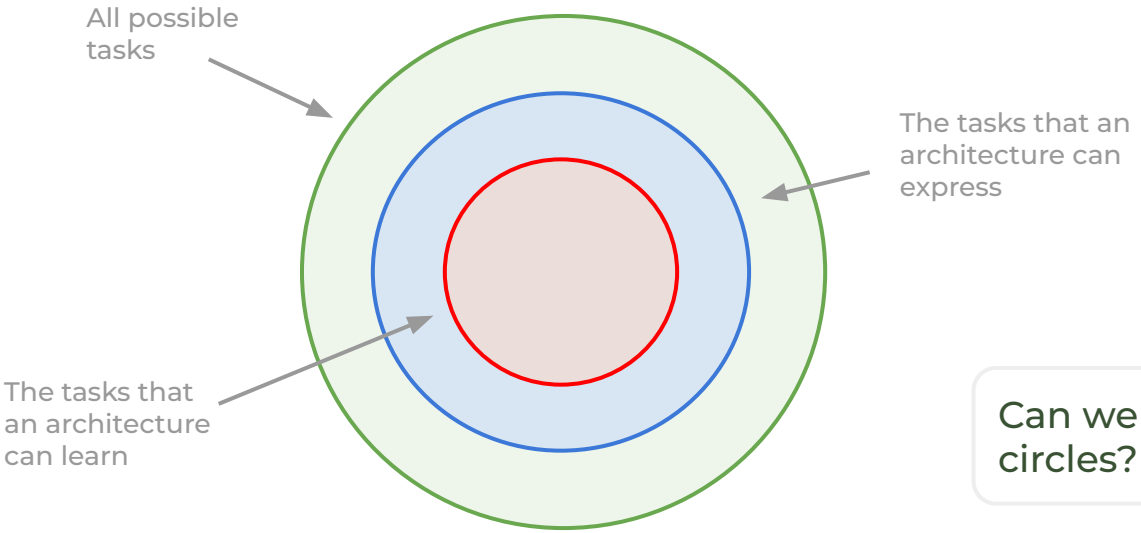
The research questions



The research questions



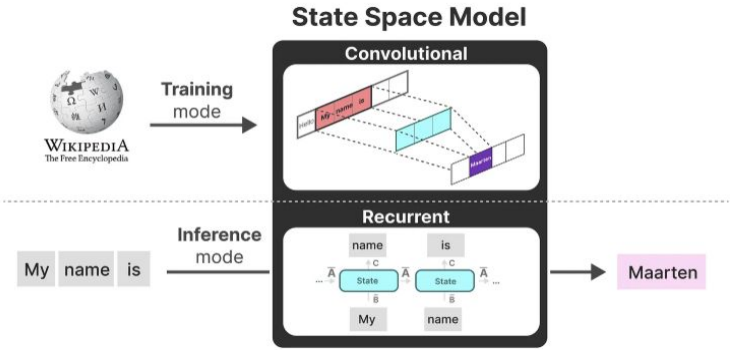
The research questions



Can we find the exact borders of those circles?

And understand why they are like that?

Architecture	Training	Inference
RNN	Not Parallelizable	Linear
Transformer	Parallelizable	Quadratic
SSM	**Parallelizable**	**Linear**



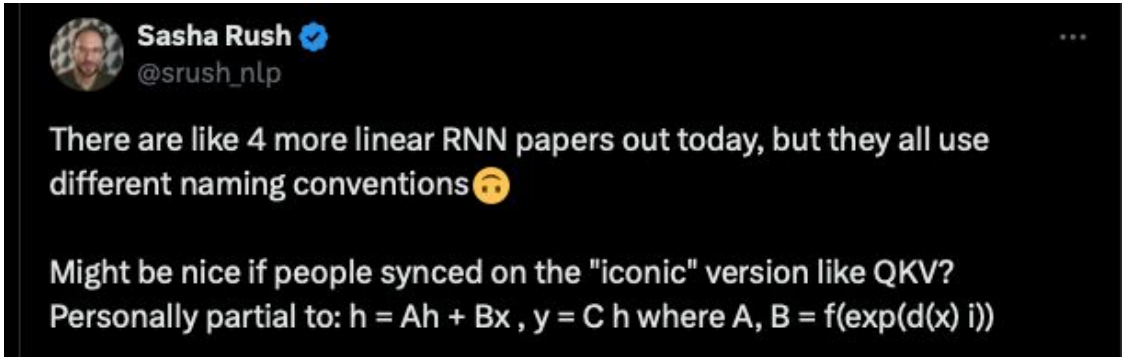
$$h_t = A(x_t) \circ h_{t-1} + B(x_t)$$

- Recurrent design,
- Unlike RNNs ;
  - No non-linearity, while updating the hidden state
- Allows faster training through
  - Convolution view (A is a common factor)
  - Prefix scan (Associative operations)

Source: https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-mamba-and-state

Motivation for studying Expressivity of SSMs

*Merrill, W., Petty, J., & Sabharwal, A. The Illusion of State in State-Space Models. In *Forty-first International Conference on Machine Learning*.


Sasha Rush
@srush_nlp
There are like 4 more linear RNN papers out today, but they all use different naming conventions🙃
Might be nice if people synced on the "iconic" version like QKV?
Personally partial to: h = Ah + Bx , y = C h where A, B = f(exp(d(x) i))

Griffin

$$
\begin{aligned}
r_t &= \sigma(W_a x_t + b_a), \quad \textit{recurrence gate} \\
i_t &= \sigma(W_x x_t + b_x), \quad \textit{input gate} \\
a_t &= a^{cr_t}, \\
h_t &= a_t \odot h_{t-1} + \sqrt{1 - a_t^2} \odot (i_t \odot x_t).
\end{aligned}
$$

HGRN2

$$
\begin{aligned}
\mathbf{g}_t &= \sigma\left(\mathbf{U}\mathbf{x}_t + \mathbf{b}_u\right), \\
\mathbf{i}_t &= \mathbf{V}\mathbf{x}_t + \mathbf{b}_v, \\
\mathbf{o}_t &= \tau\left(\mathbf{W}\mathbf{x}_t + \mathbf{b}_w\right), \\
\mathbf{h}_t &= \mathbf{g}_t \odot \mathbf{h}_{t-1} + (1 - \mathbf{g}_t) \odot \mathbf{i}_t, \\
\mathbf{y}_t &= \mathbf{h}_t \odot \mathbf{o}_t,
\end{aligned}
$$

RetNet

$$
\begin{aligned}
\boldsymbol{s}_n &= A\boldsymbol{s}_{n-1} + K_n^\intercal v_n, \\
o_n &= Q_n \boldsymbol{s}_n = \sum_{m=1}^{n} Q_n A^{n-m} K_m^\intercal v_m,
\end{aligned}
$$

Megalodon

$$
\begin{aligned}
\mathbf{h}_t^{(j)} &= \boldsymbol{\alpha}_j(\cos\theta_j + i\sin\theta_j) \odot \mathbf{u}_t^{(j)} + (1 - \boldsymbol{\alpha}_j \odot \boldsymbol{\delta}_j)(\cos\theta_j + i\sin\theta_j) \odot \mathbf{h}_{t-1}^{(j)} \\
\mathbf{y}_{t,j} &= \mathrm{Re}(\boldsymbol{\eta}_j^T \mathbf{h}_t^{(j)})
\end{aligned}
$$

Mamba

$$
\begin{aligned}
h'(t) &= \boldsymbol{A}h(t) + \boldsymbol{B}x(t) \quad (1a) & h_t &= \overline{\boldsymbol{A}}h_{t-1} + \overline{\boldsymbol{B}}x_t \\
y(t) &= \boldsymbol{C}h(t) \quad (1b) & y_t &= \boldsymbol{C}h_t
\end{aligned}
$$

GLA

$$
\mathbf{S}_t = \mathbf{G}_t \odot \mathbf{S}_{t-1} + \boldsymbol{k}_t^\top \boldsymbol{v}_t,
$$

where we use an outer product to obtain $\mathbf{G}_t = \boldsymbol{\alpha}_t^\top \boldsymbol{\beta}_t$ for parameter-efficiency (Mao, 2022; Pramanik et al., 2023), where $\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t \in (0,1)^{1\times d}$. In preliminary experiments we found that simply setting $\boldsymbol{\beta}_t = \mathbf{1}$ was sufficient, and thus we adopt the following simplified recurrent form of GLA,

$$
\mathbf{S}_t = (\boldsymbol{\alpha}_t^\top \mathbf{1}) \odot \mathbf{S}_{t-1} + \boldsymbol{k}_t^\top \boldsymbol{v}_t, \quad (3)
$$

Single Layer SSM

Map at input length T : $(x_t)_{t=1,..t} \rightarrow (z_t)_{t=1,..t}$

$$h_t = A(x_t) \circ h_{t-1} + B(x_t)$$

$$z_t = \varphi(h_t, x_t) = Mix_1(Norm(Mix_2(h_t, x_t)), x_t)$$

- $GLU(x) = (xW + b) \otimes \sigma(xV + c)$
- $SwiGLU(x) = Swish(xW_1 + b_1) \otimes (xW_2 + b_2)$
  where $Swish(y) = y.\sigma(y)$

Time-invariant SSMs

$A(x_t)$ does not dependent on x_t

- Examples: Hungry Hippos, **Retnet****

$$s_n = As_{n-1} + K_n^\intercal v_n,$$
$$o_n = Q_n s_n = \sum_{m=1}^{n} Q_n A^{n-m} K_m^\intercal v_m,$$

Non-negative SSMs

All entries in $A(x_t)$ >= 0

- Examples : **Mamba***, GLA, HGRN2

$$h_t = \overline{A}h_{t-1} + \overline{B}x_t \qquad \overline{A} = \exp(\Delta A)$$
$$y_t = Ch_t$$

$$h_t = A(x_t) \circ h_{t-1} + B(x_t)$$

$$z_t = \varphi(h_t, x_t) = Mix_1(Norm(Mix_2(h_t, x_t)), x_t)$$

* Gu, A., & Dao, T. (2023). Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*.
** Sun, Y., Dong, L., Huang, S., Ma, S., Xia, Y., Xue, J., ... & Wei, F. (2023). Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621*.

Star-Free Languages

Regular language class that is closed under finite union, product and complement

but not

Kleene-star, aka *

Example :: (ab)* = { ε, ab, abab, ...}

$$(ab)^* = (b\emptyset^c + \emptyset^c a + \emptyset^c aa\emptyset^c + \emptyset^c bb\emptyset^c)^c$$

Non Star-Free Language

All Regular language that are not Star-Free :)

Along with Union, Product and Complement

REQUIRE THE INCLUSION OF
Kleene-star *

Example :: (aa)* = { ε, aa, aaaa, ...}

Non Star-Free Language

Example: **PARITY** function

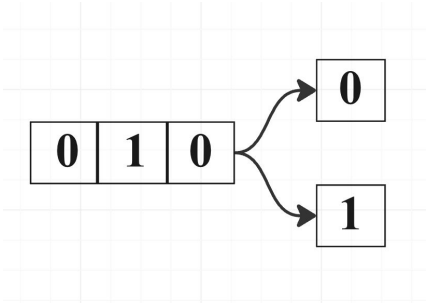
Is the number of “1” in a bitstring even or odd?

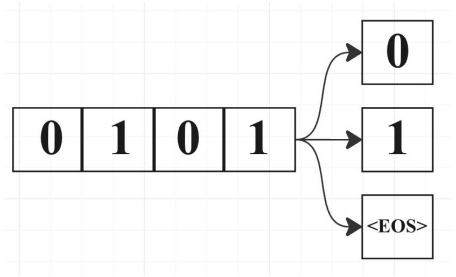
110010000	1
110000000	0
110001110	1
000000000	0

Parity is just (aa)* with an additional neutral symbol

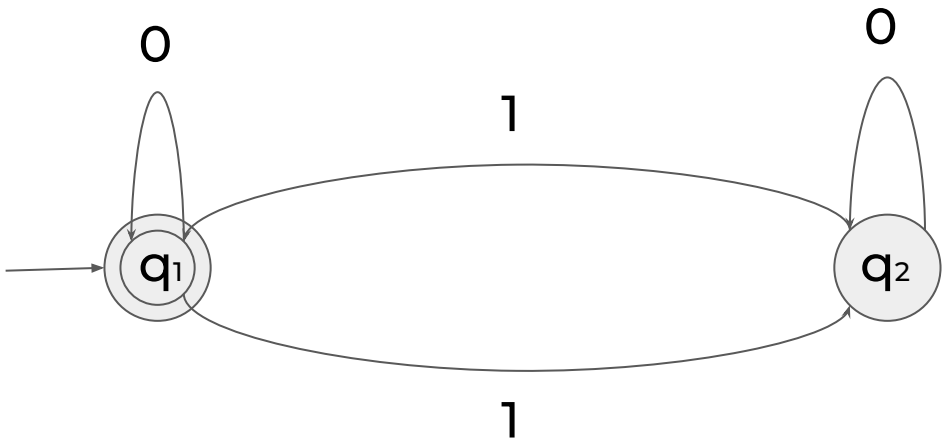
Recognition

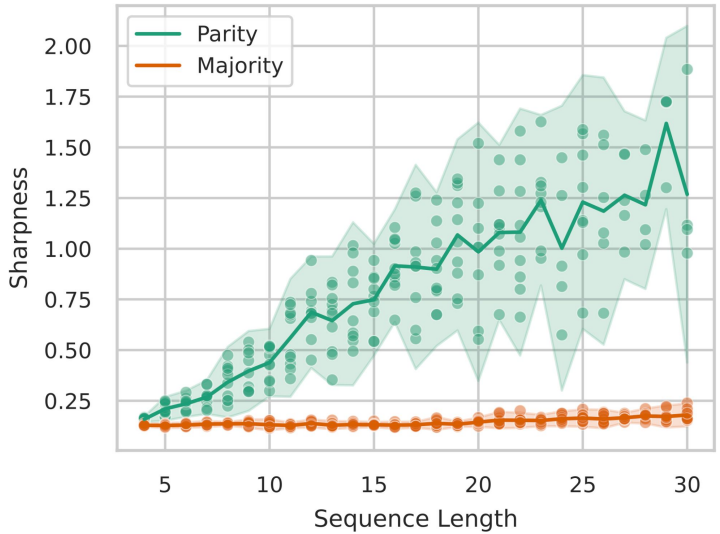
Predictive Modelling





RNNs on Parity

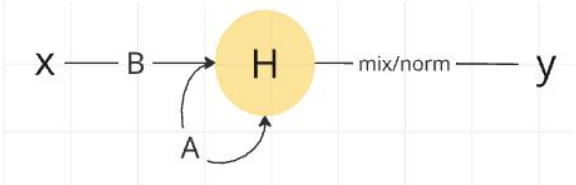




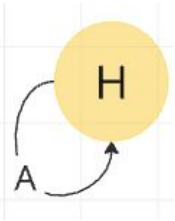
Minima of Transformers on sensitive functions is brittle*

Hahn, Michael, and Mark Rofin. "Why are Sensitive Functions Hard for Transformers?" *ACL 2024*

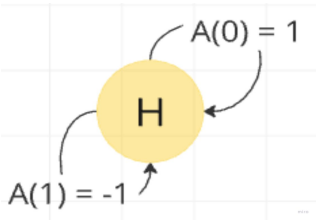
How do SSMs perform?



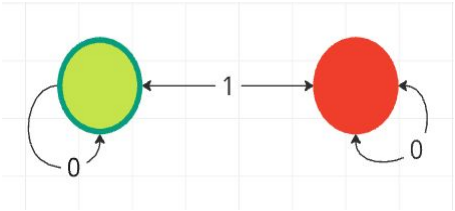
SSM Recurrence



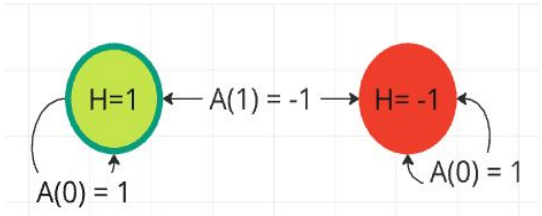
B = 0



Solution



FSA for solving Parity



Our Construction

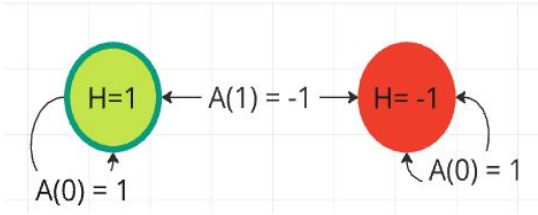
How do SSMs perform?

Our Solution

- A depends on input
  :: Time Variant

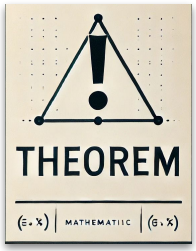
- A(1) = -1
  :: Negative

- Modern SSMs are either
  - Time Invariant
  - Non negative



NONNEGATIVE SSMs cannot recognize PARITY

- at **arbitrary input lengths**
- with **finite precision**.



➔ Inputs of the form 1_N

➔ Activations Z_N converge as $N \to \infty$,

➔ Since x_t is the same, the following are constant $h_t = \alpha^{t-T_0}\left(h_{T_0} + \frac{\beta}{\alpha - 1}\right) + \frac{\beta}{1-\alpha}$

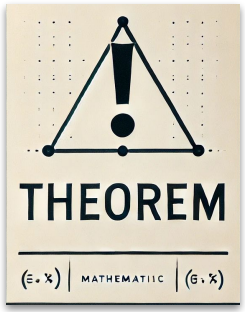
- ◆ $\beta = B(x_t)$
- ◆ $\alpha = A(x_t)$

- **Diverge exponentially ($\alpha > 1$), Converge ($\alpha < 1$), Diverge linearly ($\alpha = 1$)**
- **The Norm after Divergence causes overall convergence.**
- **Mix_1, Norm, Mix_2 also don't change this.**

TIME-INVARIANT SSMs cannot recognize PARITY

even with Complex Valued gates, as long as each entry in each A has a rational angle in the complex plane.

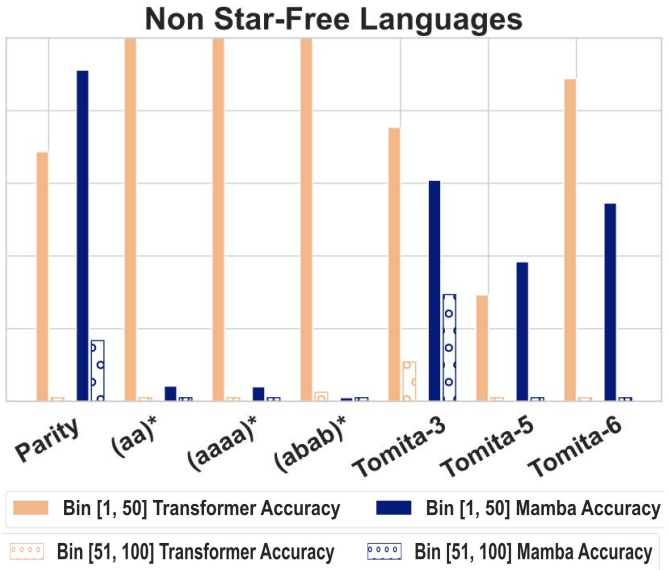
- at **arbitrary input lengths**
- with **finite precision**.

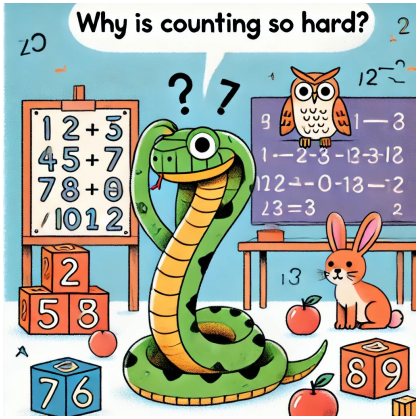


- Expand Recurrence, we will get several components in the equation
- Some will not depend on the sequence length t.
- The ones that will, make the overall value either
  - **diverge exponentially,**
  - **converge ($\alpha < 1$)**
  - **diverge linearly ($\alpha = 1$)**

Takeaway #1

- SSMs will struggle with Modular counting whenever required. (Non Star-Free languages require it)

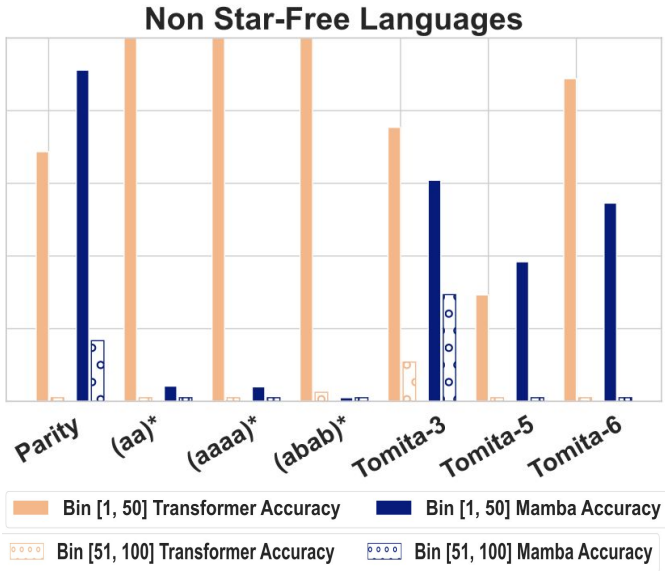


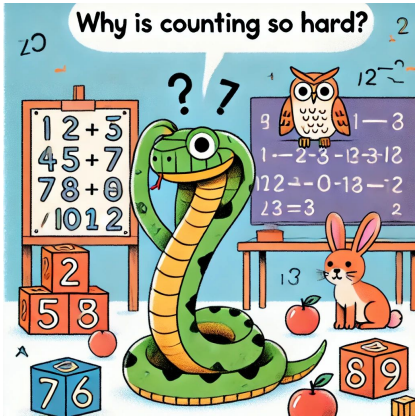


Bhattamishra, Satwik, Kabir Ahuja, and Navin Goyal. "On the Ability and Limitations of Transformers to Recognize Formal Languages." *EMNLP* 2020.

Takeaway #1

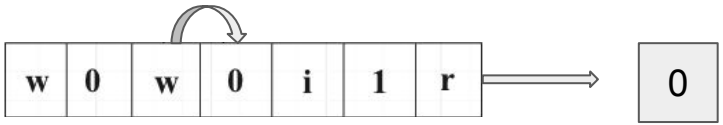
- SSMs will struggle with Modular counting whenever required. (Non Star-Free languages require it)
- Certain design choices cause this limitation





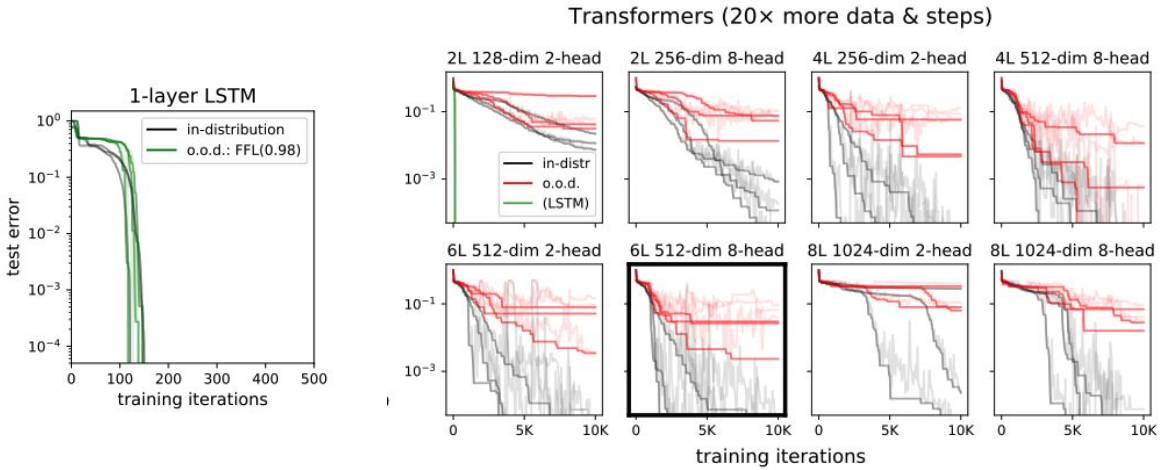
Bhattamishra, Satwik, Kabir Ahuja, and Navin Goyal. "On the Ability and Limitations of Transformers to Recognize Formal Languages." *EMNLP* 2020.

Flip-Flop



Minimalistic long-range dependency benchmark ~ Proxy for closed domain hallucinations.

Liu, Bingbin, et al. "Exposing attention glitches with flip-flop language modeling." NeurIPS, 2023



- Attention Heads: Commutative
- Attending to last write token ~ strong positional dependence in attention score

Liu, Bingbin, et al. "Exposing attention glitches with flip-flop language modeling." NeurIPS, 2023

Jobanputra et al. “Born a Transformer – Always a Transformer? On the Effect of Pretraining on Architectural Abilities.” NeurIPS, 2025.

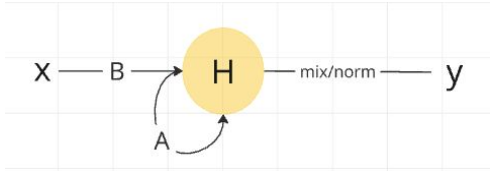
Set-Reset Automaton

FSA where $(Q \setminus q_0)$ also belongs to the Alphabet

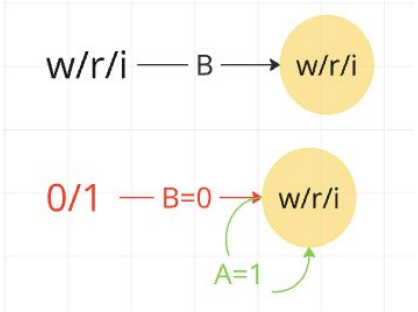
$$\mathbf{u(q, \sigma) = q \text{ if } \sigma \notin Q}$$
$$\mathbf{= \sigma \text{ else}}$$

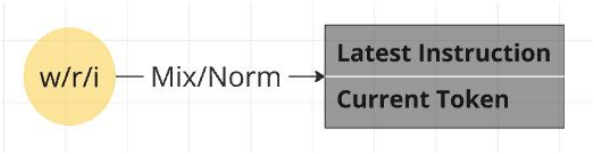
Keeps recording the last seen symbol from the designed set Q

Our Theoretical Construction

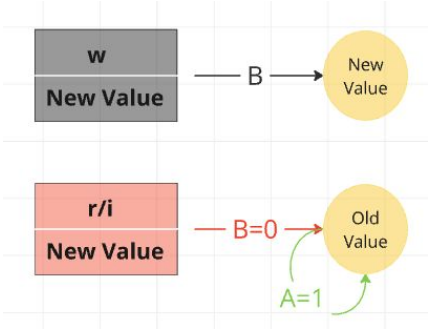


Layer 1

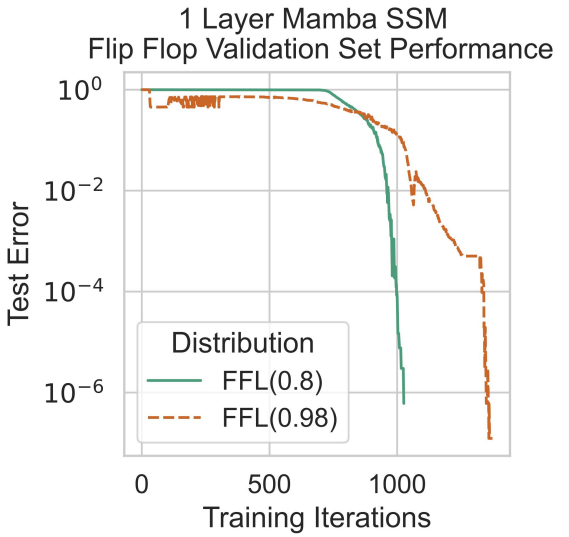


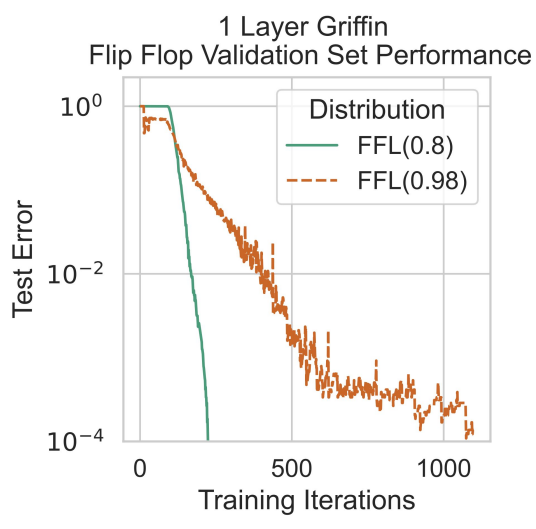


Layer 2



Empirical Results

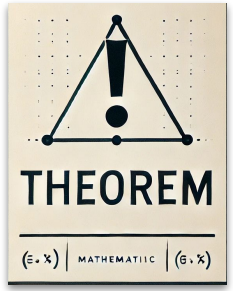




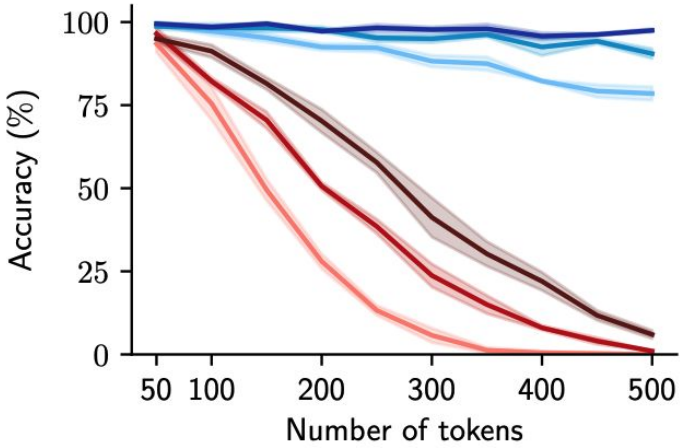
SSMs resolves a critical failure mode of self-attention

A 2 Layer SSM predictively models the Flip flop language

- at **arbitrary input lengths**
- with **finite precision**.

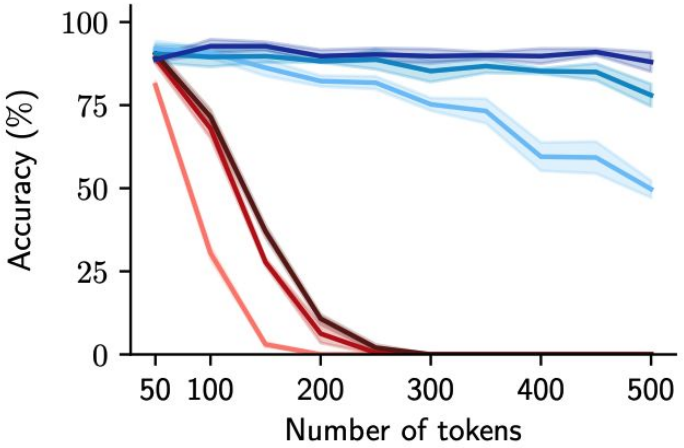


Transformers can copy, SSMs can’t



Pythia: 410M 1.4B 2.8B
Mamba: 360M 1.4B 2.8B

(a) Copy: natural language strings



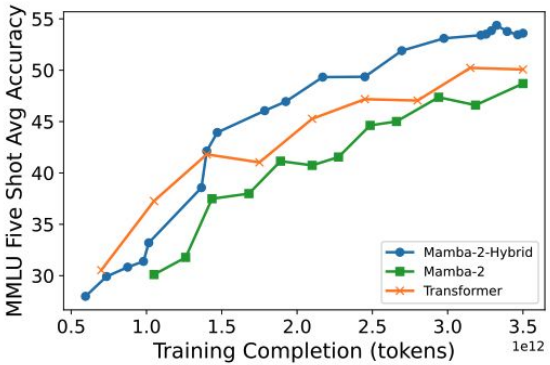
Pythia: 410M 1.4B 2.8B
Mamba: 360M 1.4B 2.8B

(b) Copy: shuffled strings

Jelassi, Samy, et al. "Repeat after me: Transformers are better than state space models at copying." ICML, 2024

Takeaway #2

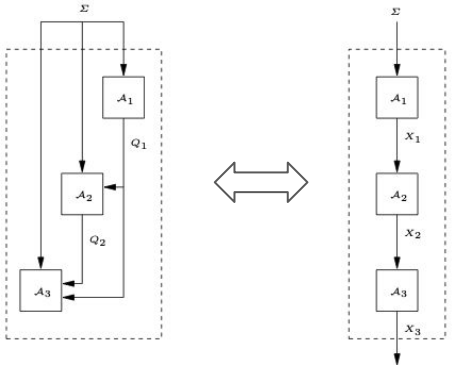
- Complementary Abilities b/w SSMs & Transformers
- Future: Hybrid Architecture



- Lieber, Opher, et al. "Jamba: A hybrid transformer-mamba language model." *arXiv preprint arXiv:2403.19887* (2024).
- Waleffe, Roger, et al. "An Empirical Study of Mamba-based Language Models." *arXiv preprint arXiv:2406.07887* (2024).
- Ren, Liliang, et al. "Samba: Simple Hybrid State Space Models for Efficient Unlimited Context Language Modeling." *arXiv preprint arXiv:2406.07522* (2024).

Star-Free Languages

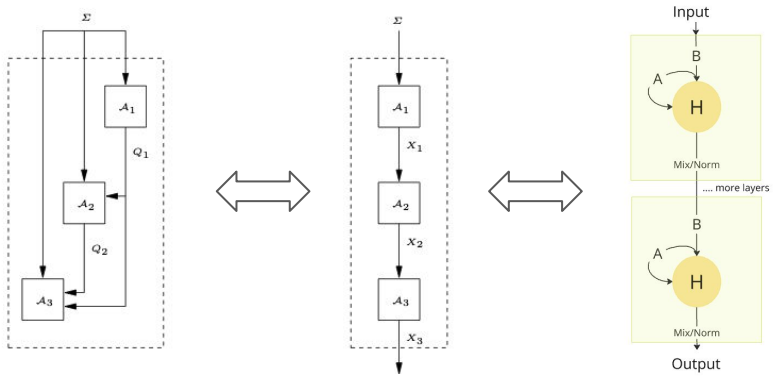
- ALL Star Free languages can be reduced to Flip Flop State Tracking
    - Any counter free FSA => Cascade of simpler Set-Reset Automatas (Flip Flop banks)



*Krohn, Kenneth, and John Rhodes. "Algebraic theory of machines. I. Prime decomposition theorem for finite semigroups and machines." *Transactions of the American Mathematical Society* 116 (1965): 450-464.
*Schützenberger, Marcel Paul. "On finite monoids having only trivial subgroups." *Inf. Control.* 8.2 (1965): 190-194.

Star-Free Languages

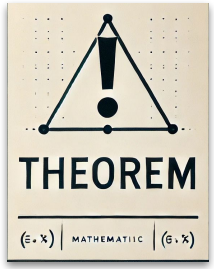
- ALL Star Free languages can be reduced to Flip Flop State Tracking
    - Any counter free FSA => Cascade of simpler Set-Reset Automatas (Flip Flop banks)
- We show, Cascade of simpler Automatas ~ Stacking SSMs



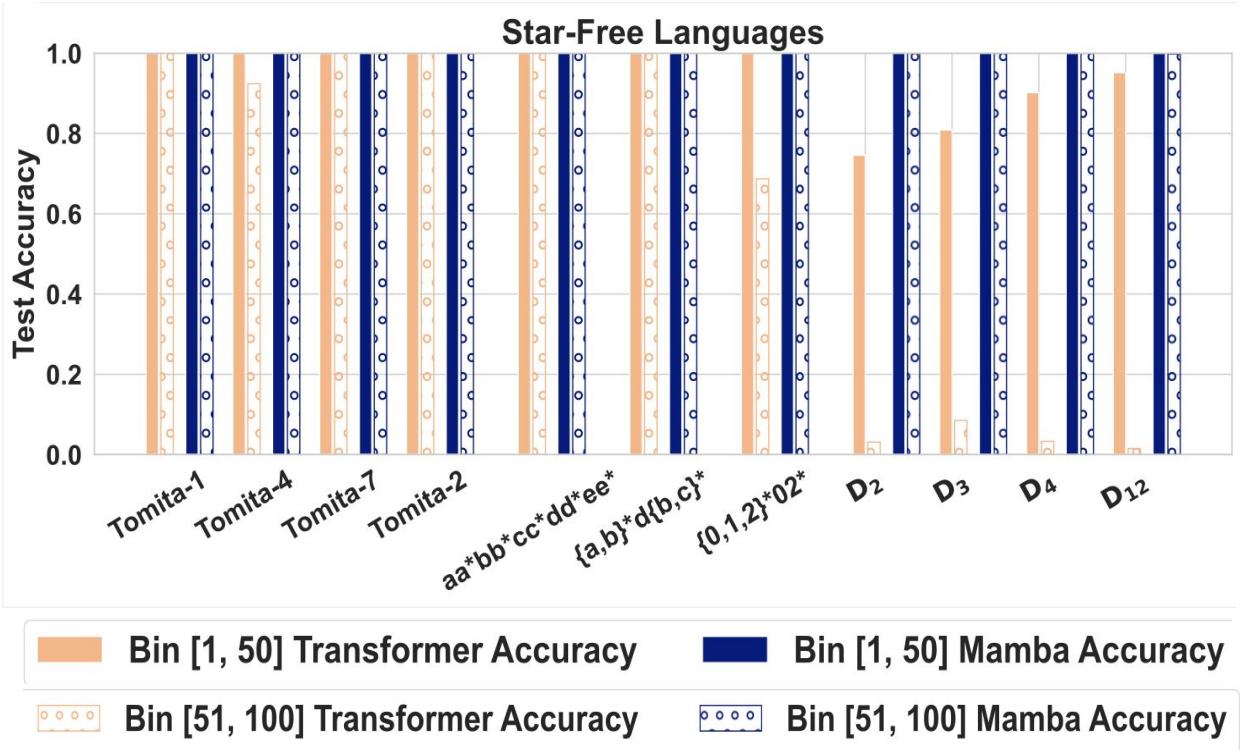
*Krohn, Kenneth, and John Rhodes. "Algebraic theory of machines. I. Prime decomposition theorem for finite semigroups and machines." *Transactions of the American Mathematical Society* 116 (1965): 450-464.
*Schützenberger, Marcel Paul. "On finite monoids having only trivial subgroups." *Inf. Control.* 8.2 (1965): 190-194.

NON-NEGATIVE SSMs can predictively model Regular Languages

- iff **the language is star free**
- with **finite precision**.



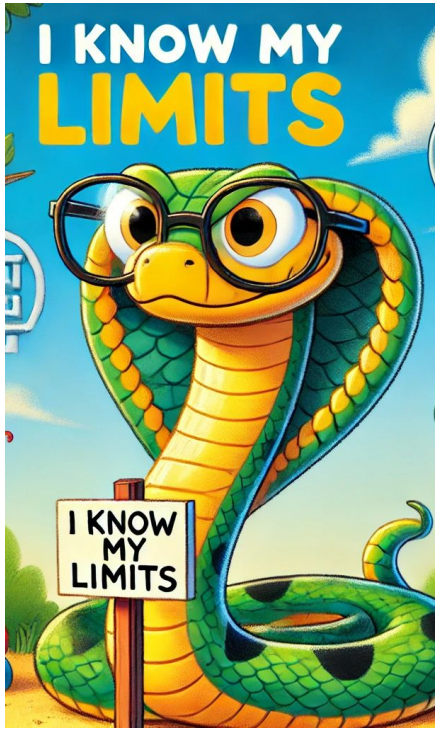
Empirical Results



Bhattamishra, Satwik, Kabir Ahuja, and Navin Goyal. "On the Ability and Limitations of Transformers to Recognize Formal Languages."- *EMNLP* 2020.

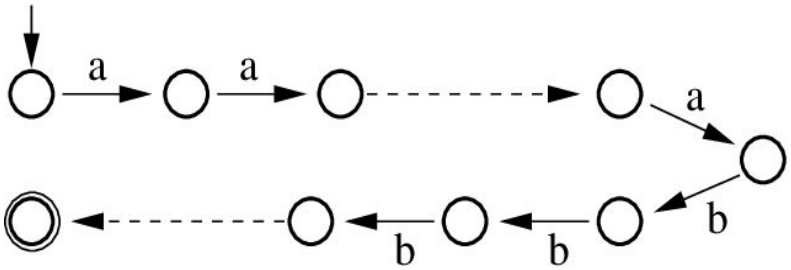
Takeaway #3

- Exact characterisation of Transformers* in Finite state case: Difficult.
- With SSMs, it’s possible!

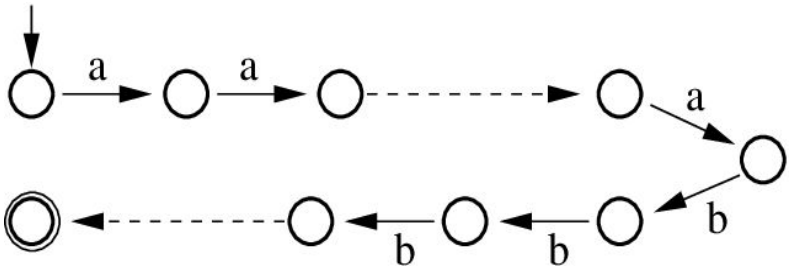


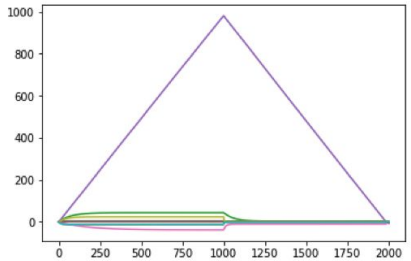
*Angluin, Dana, David Chiang, and Andy Yang. "Masked hard-attention transformers and boolean RASP recognize exactly the star-free languages." *arXiv:2310.13897*(2023).

Unbounded Counting



Unbounded Counting

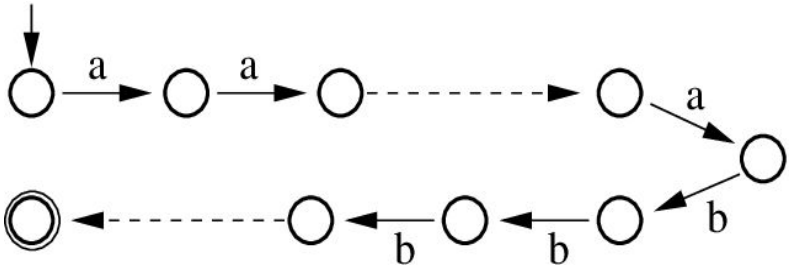


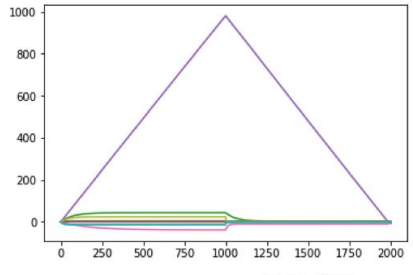


LSTM Activation Pattern

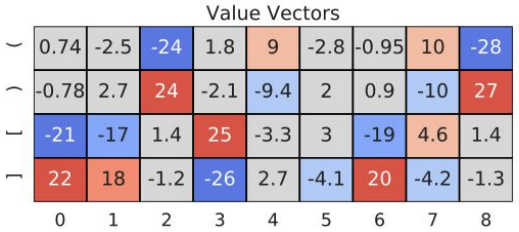
Weiss, Gail, Yoav Goldberg, and Eran Yahav. "On the Practical Computational Power of Finite Precision RNNs for Language Recognition." *(Volume 2: Short Papers)*. *ACL* 2018

Unbounded Counting





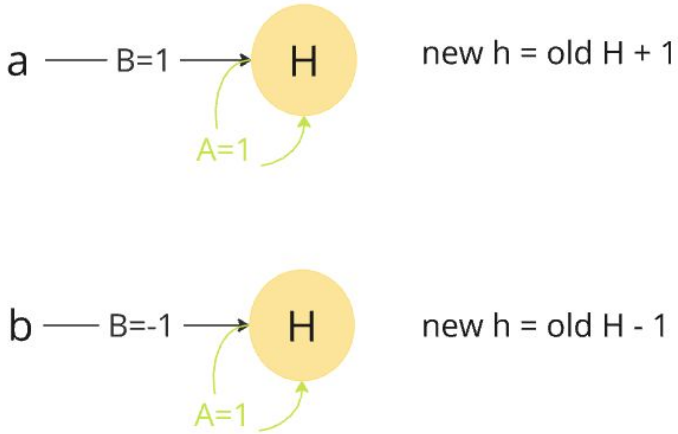
LSTM Activation Pattern



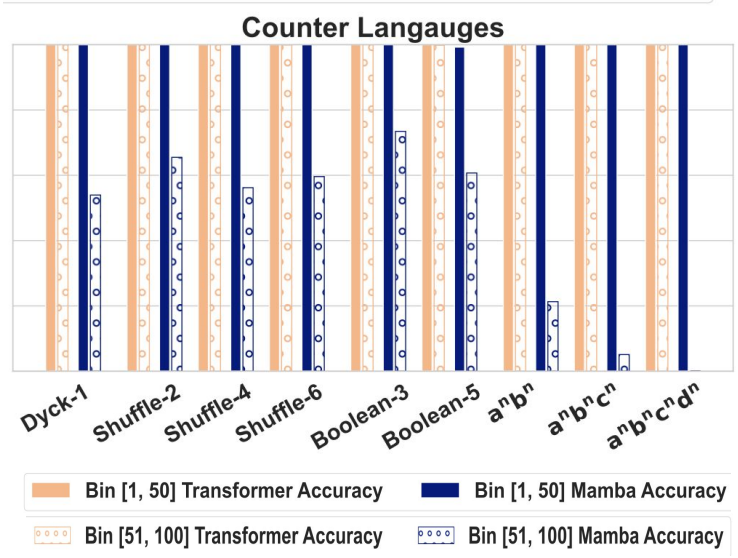
Attention Values in Transformers

Weiss, Gail, Yoav Goldberg, and Eran Yahav. "On the Practical Computational Power of Finite Precision RNNs for Language Recognition." *(Volume 2: Short Papers)*. *ACL* 2018
Bhattamishra, Satwik, Kabir Ahuja, and Navin Goyal. "On the Ability and Limitations of Transformers to Recognize Formal Languages."- *EMNLP* 2020.

Our Construction: $a^n b^n$



Empirical Results

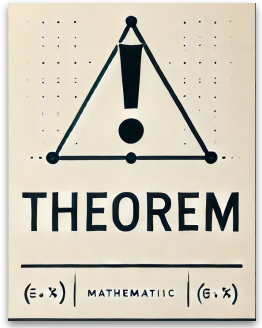


Bhattamishra, Satwik, Kabir Ahuja, and Navin Goyal. "On the Ability and Limitations of Transformers to Recognize Formal Languages." *EMNLP* 2020.

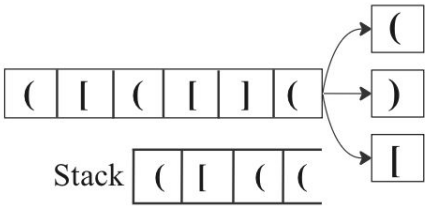
SSMs can predictively model

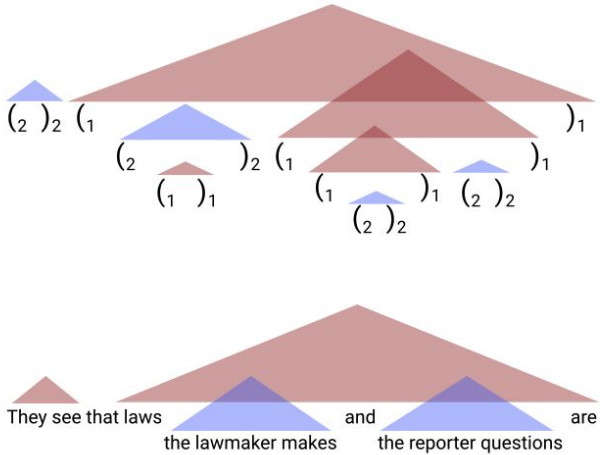
- **Dyck-1**
- **Shuffle-Dyck-k**
- **n-ary Boolean Expressions**
- **$a^n b^n$, $a^n b^n c^n$, $a^n b^n c^n d^n$**

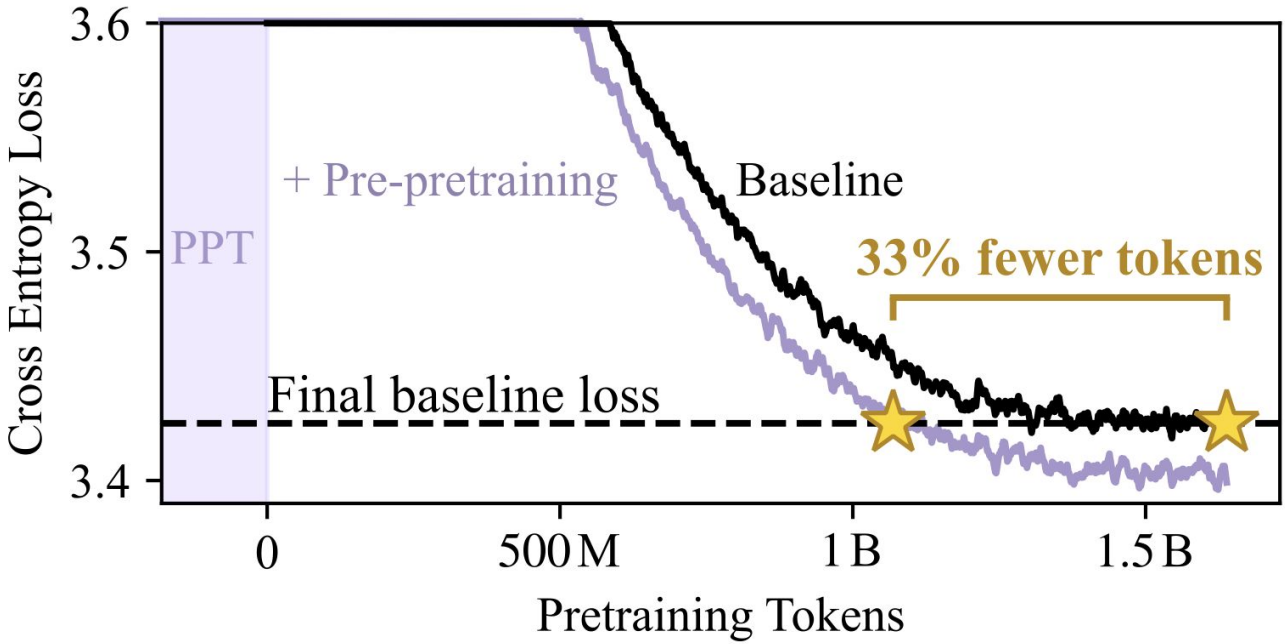
with finite fractional #bits, but unbounded integer #bits



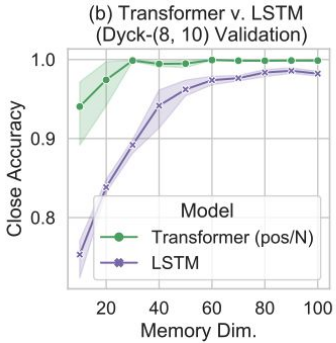
Bounded Dyck : Dyck(K, m)





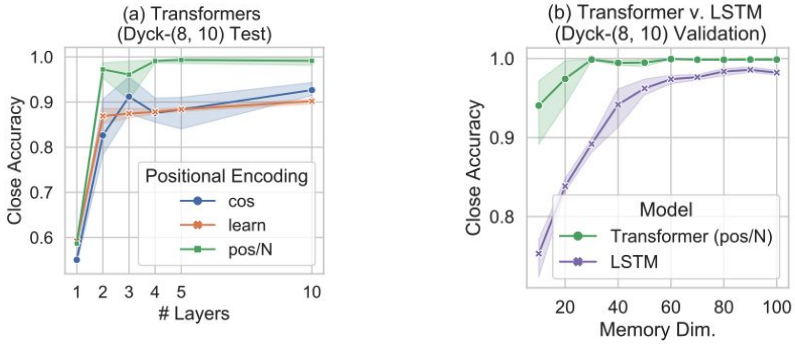


Hu et al. “Between Circuits and Chomsky: Pre-pretraining on Formal Languages Imparts Linguistic Biases.” ACL, 2025.



- LSTMs require O (m log K)
- Transformers - O (log K)

Hewitt, John, et al. "RNNs can generate bounded hierarchical languages with optimal memory." *EMNLP* 2020.
Yao, Shunyu, et al. "Self-Attention Networks Can Process Bounded Hierarchical Languages." ACL 2021.

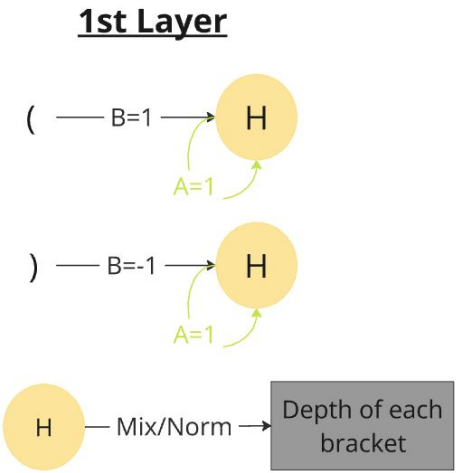


- LSTMs require O (m log K)
- Transformers - O (log K)
- Transformers require specific PE

Hewitt, John, et al. "RNNs can generate bounded hierarchical languages with optimal memory." *EMNLP* 2020.
Yao, Shunyu, et al. "Self-Attention Networks Can Process Bounded Hierarchical Languages." ACL 2021.

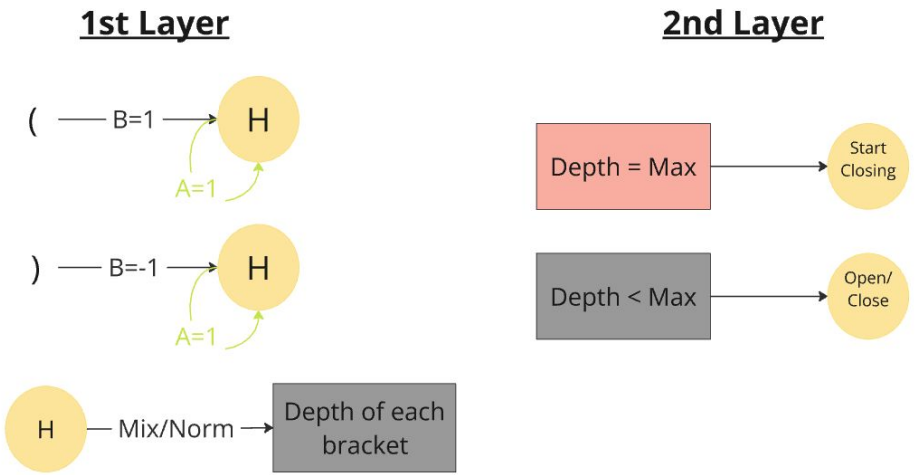
Our Construction

- Dyck(K, m) : Regular language
  - Solution Guaranteed
- EXPLICIT STACK
  NOT REQUIRED
  (shortcut through counting)
- Hence, EFFICIENT

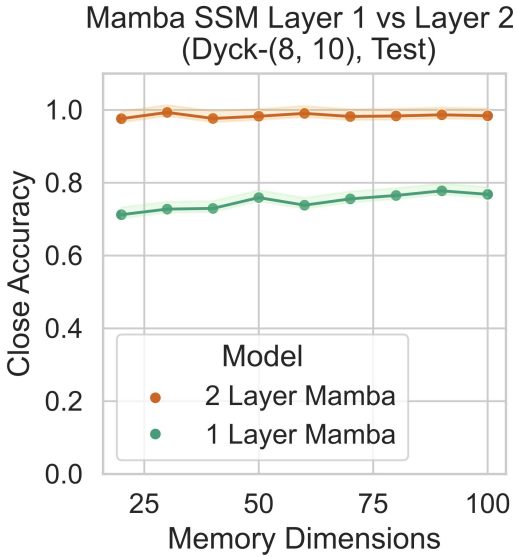


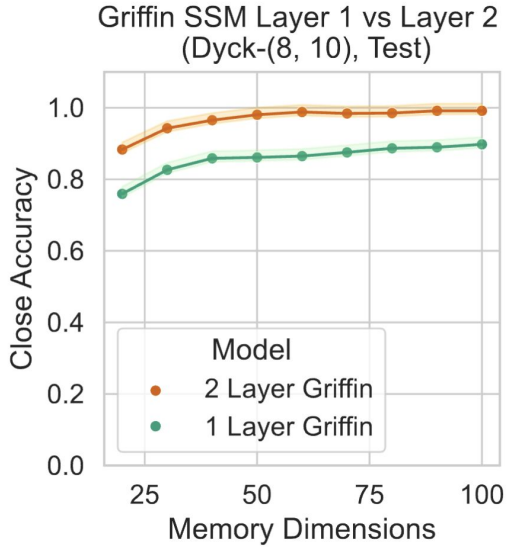
Our Construction

- Dyck(K, m) : Regular language
  - Solution Guaranteed
    (not necessarily efficient)
- EXPLICIT STACK
  NOT REQUIRED
  (shortcut through counting)
- Hence, EFFICIENT



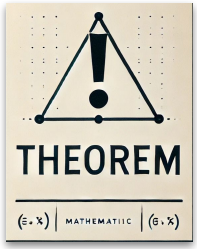
Empirical Results





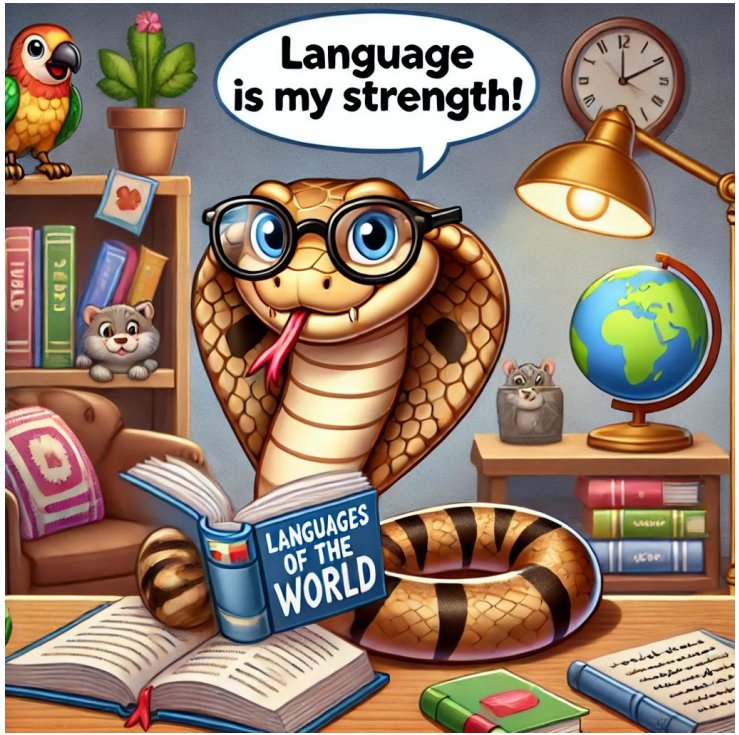
A 2 Layer SSM predictively models Bounded Dyck (K, m)

- with **d = O(mlog K)**
- with **finite precision**.



Takeaway #4

- SSMs can keep track of bounded hierarchical structures EFFICIENTLY!
- SSMs can model hierarchical structure of language



Limitations

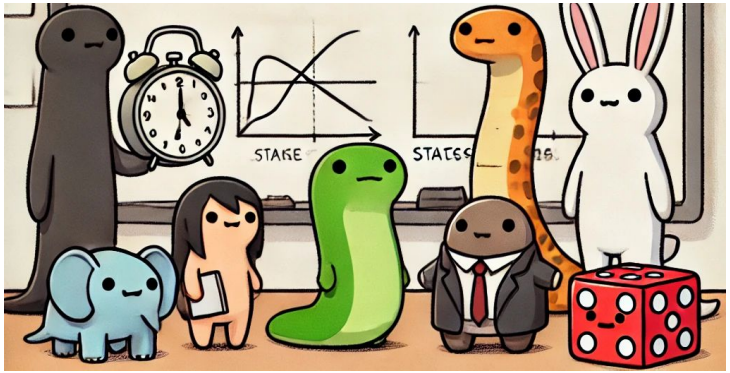
- No comment on Learnability / Generalisability

Limitations

- No comment on Learnability / Generalisability
- Positive claims needs more empirical evidence

Limitations

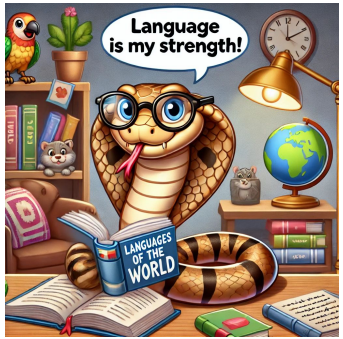
- No comment on Learnability / Generalisability
- Positive claims needs more empirical evidence
- More careful study required on the exact implementational differences



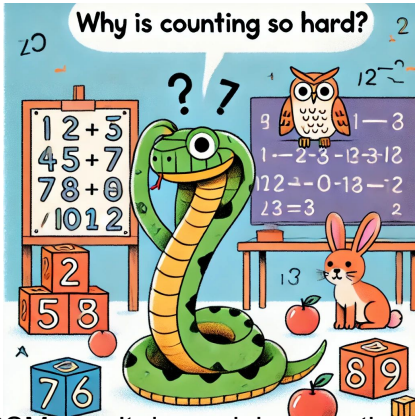
Recapping our main Theorems

- **Finite precision**
  - **NONNEGATIVE SSMs cannot recognize PARITY at arbitrary input lengths.**
  - **NONNEGATIVE SSMs can predictively model a regular language iff it's star-free.**
    - **2 - Layer SSM predictively models the Flip flop language at arbitrary lengths.** *(Solutions were guaranteed with Krohn Rhodes, but not a 2 layer construction)*
    - **2 - Layer SSM predictively models Dyck(K, m) with d = O(mlogK)**
- **Unbounded integer values; Finite precision for fractional components**
  - **Dyck-1, Shuffle-Dyck-k, n-ary Boolean Expressions, a^nb^n, $a^nb^nc^n$, $a^nb^nc^nd^n$ are predictively modeled by an SSM**

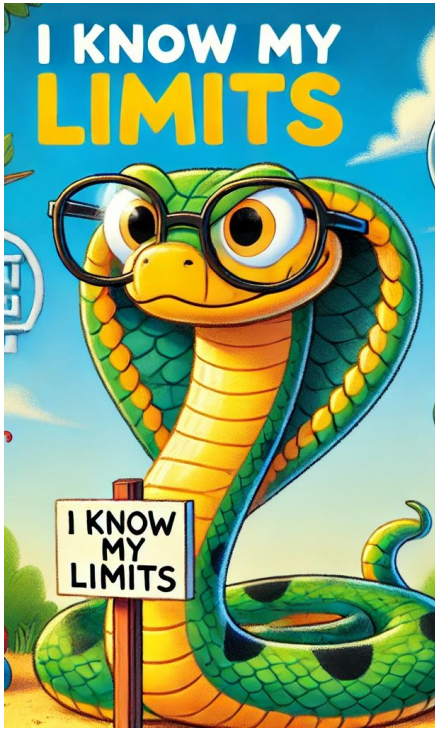
Recapping our takeaways



SSMs can model hierarchical structure of language



SSMs can’t do modular counting



It would be easier to theoretically predict failures & abilities LLMs based on SSMs

LLMs will have Hybrid Architectures