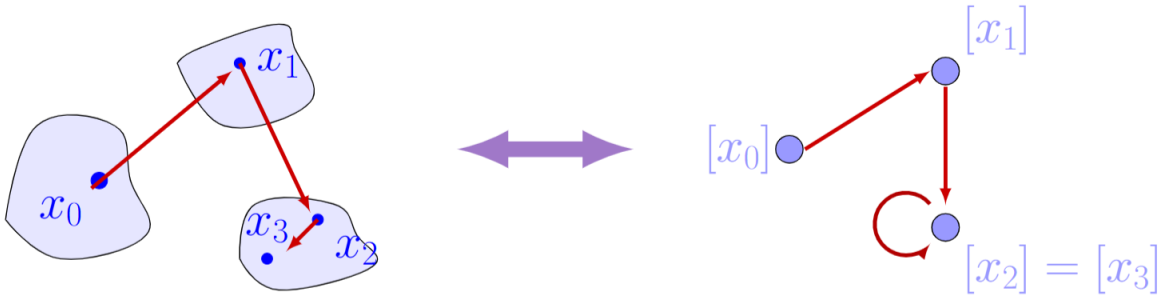
Metric Automata Theory: A Unifying Theory of RNNs

Adam Dankowiakowski, Alessandro Ronca



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

Yash Sarrof, Yana Veitsman, Michael Hahn
Saarland Informatics Campus
Saarland University, Germany
{ysarrof, yanav, mhahn}@lst.uni-saarland.de

Abstract

Recently, recurrent models based on linear state space models (SSMs) have shown promising performance in language modeling (LM), competitive with transformers. However, there is little understanding of the in-principle abilities of such models, which could provide useful guidance to the search for better LM architectures. We present a comprehensive theoretical study of the capacity of such SSMs as it compares to that of transformers and traditional RNNs. We find that SSMs and transformers have overlapping but distinct strengths. In star-free state tracking, SSMs implement length-generalizing solutions to problems that transformers struggle to represent exactly. They can also model bounded hierarchical structure with optimal memory even without simulating a stack. On the other hand, we identify a design choice in current SSMs that limits their expressive power. We discuss implications for SSM and LM research, and verify results empirically [1] on a recent SSM, Mamba.

Theorem 2. *No SSM satisfying* NONNEGATIVE *can recognize PARITY at arbitrary input lengths with finite precision. In particular, this applies to Mamba.*

recurrent achitectures [RNNs Elman, 1990, Hochreiter and Schmidhuber, 1997]. However, building on a long history of continuous dynamical models [e.g. Kalman, 1960, 1963] and work on faster RNNs [Bradbury et al., 2016, Lei et al., 2018], a recent line of work has developed *state space models* (SSMs) rivaling the performance of transformers [e.g. Gu et al., 2021, Gu and Dao, 2023, Sun et al., 2023, De et al., 2024, Yang et al., 2024, Qin et al., 2024a]. These SSMs are recurrent models, formulated in terms of iterative state updates, while still allowing efficient parallelization.

The impressive empirical performance of such SSMs raises the question of whether they might have capabilities that the transformer architecture might lack in principle. Simultaneously, to understand whether SSMs may plausibly overtake the dominant role of transformers, it is an important question whether SSMs may lack abilities present in transformers. A better understanding of these questions may also point the way to future architectures that unite the strengths of both architectures.

One common approach to understanding the capabilities of computational architectures is through their expressive capacity in simulating automata and modeling language classes; indeed, a sizeable literature has studied transformers [e.g. Pérez et al., 2019, Hahn, 2020, Bhattamishra et al., 2020, Yao et al., 2021a, Liu et al., 2023b,a, Deletang et al., 2022, Strobl et al., 2024, Chiang et al., 2023, Sanford et al., 2024, Peng et al., 2024] and RNNs [e.g. Siegelman and Sontag, 1995, Horne and Hush, 1993, Indyk, 1995, Weiss et al., 2018, Hewitt et al., 2020] through this lens. As the difficulty

[1]Code is available at: https://github.com/lacoco-lab/ssm_expressivity

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

few months later…

UNLOCKING STATE-TRACKING IN LINEAR RNNS THROUGH NEGATIVE EIGENVALUES

Riccardo Grazzi[*▽], **Julien Siems**[*◇], **Arber Zela**[◇],
Jörg K.H. Franke[◇], **Frank Hutter**[◇♣], **Massimiliano Pontil**[▽♠]
Equal contribution[*], CSML, Istituto Italiano di Tecnologia[▽], University of Freiburg[◇],
ELLIS Institute Tübingen[♣], AI Centre, University College London[♠]
riccardograzzi4@gmail.com juliensiems@gmail.com

ABSTRACT

Linear Recurrent Neural Networks (LRNNs) such as Mamba, RWKV, GLA, mLSTM, and DeltaNet have emerged as efficient alternatives to Transformers for long sequences. However, both Transformers and LRNNs struggle to perform state-tracking, which may impair performance in tasks such as code evaluation. In one forward pass, current architectures are unable to solve even parity, the simplest state-tracking task, which non-linear RNNs can handle effectively. Recently, Sarrof et al. (2024) demonstrated that the failure of LRNNs like Mamba to solve parity stems from restricting the value range of their diagonal state-transition matrices to $[0, 1]$ and that incorporating negative values can resolve this issue. We extend this result to non-diagonal LRNNs such as DeltaNet. We prove that finite precision LRNNs with state-transition matrices having only positive eigenvalues cannot solve parity, while non-triangular matrices are needed to count modulo 3. Notably, we also prove that LRNNs can learn any regular language when their state-transition matrices are products of identity minus vector outer product matrices, each with eigenvalues in the range $[-1, 1]$. Our experiments confirm that extending the eigenvalue range of Mamba and DeltaNet to include negative values not only enables them to solve parity but consistently improves their performance on state-tracking tasks. We also show that state-tracking enabled LRNNs can be pretrained stably and efficiently at scale (1.3B parameters), achieving competitive performance on language modeling and showing promise on code and math tasks.

1 INTRODUCTION

Transformer architectures (Vaswani et al., 2017) have revolutionized NLP but scale quadratically in sequence length, posing computational challenges for long sequences. To address this, Linear Recurrent Neural Networks (LRNNs) have emerged as promising alternatives that offer linear scaling while maintaining competitive performance (Gu & Dao, 2024; Dao & Gu, 2024; Yang et al., 2024a; Peng et al., 2023; Deletang et al., 2023; Sun et al., 2024; Beck et al., 2024). LRNNs update their state via matrix-vector products with structured and often input-dependent state-transition matrices. The structure of the state-transition matrices largely determines the expressivity of LRNNs. While successful models like Mamba (Gu & Dao, 2024) and GLA (Yang et al., 2024a) use diagonal matrices (diagonal LRNN) which only mix tokens along the sequence dimension, recent work explores more complex forms. Notably, non-diagonal matrices using generalized Householder (GH) transformations, defined as $\boldsymbol{I} - \boldsymbol{u}\boldsymbol{u}^\top$ where $\boldsymbol{u}$ is a learnable vector and $\boldsymbol{I}$ is the identity, enable models like DeltaNet (Schlag et al., 2021; Yang et al., 2024b) and TTT-Linear (Sun et al., 2024) to achieve richer expressiveness through simultaneous token-channel mixing while maintaining efficiency.

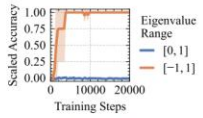


Figure 1: Extending the eigenvalue range of the state transition matrices of diagonal LRNNs improves performance from random guessing (range $[0, 1]$) to perfect score (range $[-1, 1]$) on learning parity. Trained on sequences up to length 40; Tested on lengths 40–256 (3 seeds).

Sarrof et al., 2024

Grazzi et al., 2025

Sarrof et al. vs Grazzi et al.

Finite Precision*:

"Linear RNNs with non-negative diagonal gates are star-free"

Finite Precision**:

"Linear RNNs with non-negative eigenvalue gates are star-free"

Similar theorems and arguments, however…

Finite Precision* ≠ Finite Precision**

Simplifying assumptions on finite-precision arithmetic
Not "real life"

Key Issues

No principled, commonly accepted framework.

Analysis of finite precision datatypes is messy!

$$a(b+c) \neq ab + ac$$

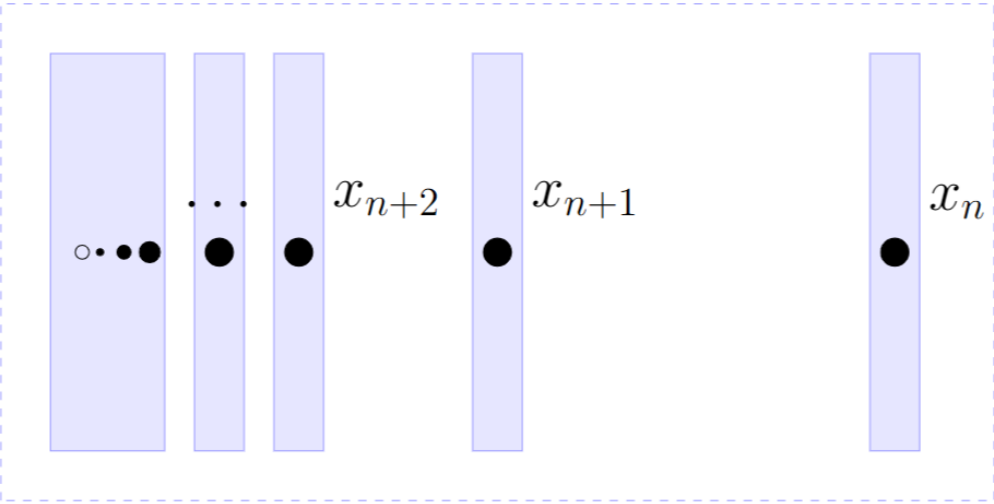
Metric Automata Theory

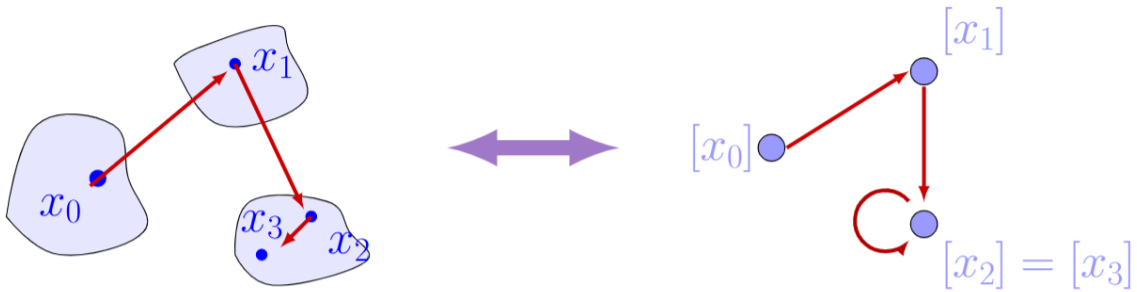
- Continuous Dynamical Systems as a basis.

X	U	$f : X \times U \to X$
state space	input space	*continuous* transition function

η-finite spaces

- Finitely many **compact & path-connected** components.





- **Key property:** convergence $\Longrightarrow$ eventually in the same component.

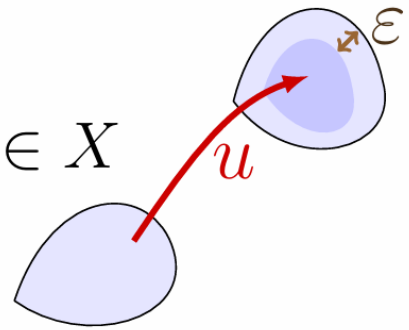
$\approx$ Finite Precision

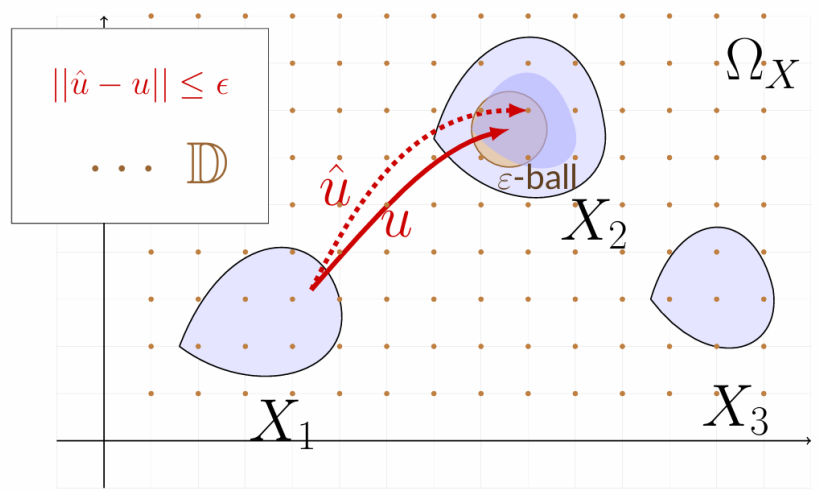
Robustness

- Connects η-finite systems with **real float implementations.**

ε-robust : $B\big(f(x,u),\varepsilon\big) \in X$

"Transitions have ε margin of error"

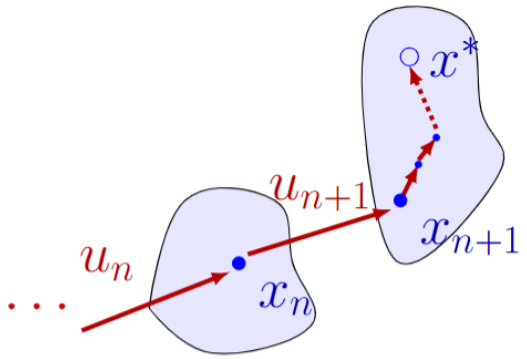




Robust system: approximation = implementation

Results

- η-finiteness formalises the intuitive approach of Sarrof et al.
- The setup in Grazzi et al. restricts the analysis and is still unrealistic.



Key: iterating single input gives state convergence.

$[u]$ $[u]$ $[u]$ $[u]$

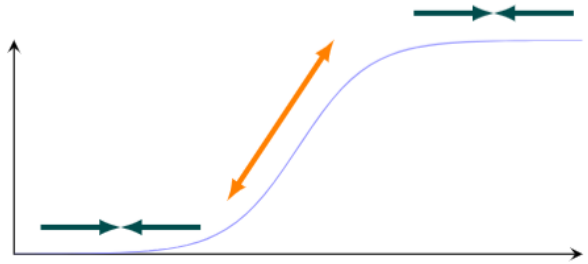
$[x_n]$ $[x_{n+1}] = [x^*]$

Non-negative e-vals $\Longrightarrow$ eventually monotone coordinates
(in Jordan Normal Form basis)

- Significant simplification of proofs afforded by MAT.

Linear vs Non-linear

- Linear recurrence is **incapable of robust state-tracking.**
- xLSTM can **robustly** implement **all star-free languages**.



Non-linear activation gives robustness:

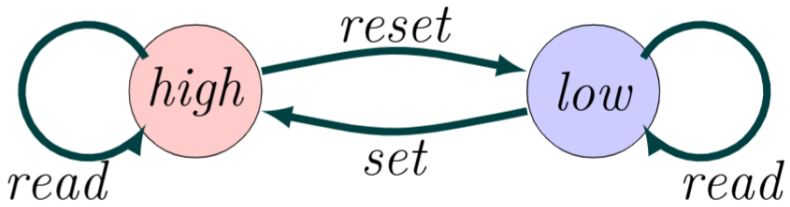
Contracting regions give transition targets margin of error
Expanding regions keep state components apart

- **Consequence:** also can do so with finite-precision implementations.

Supported by existing empirical evidence…

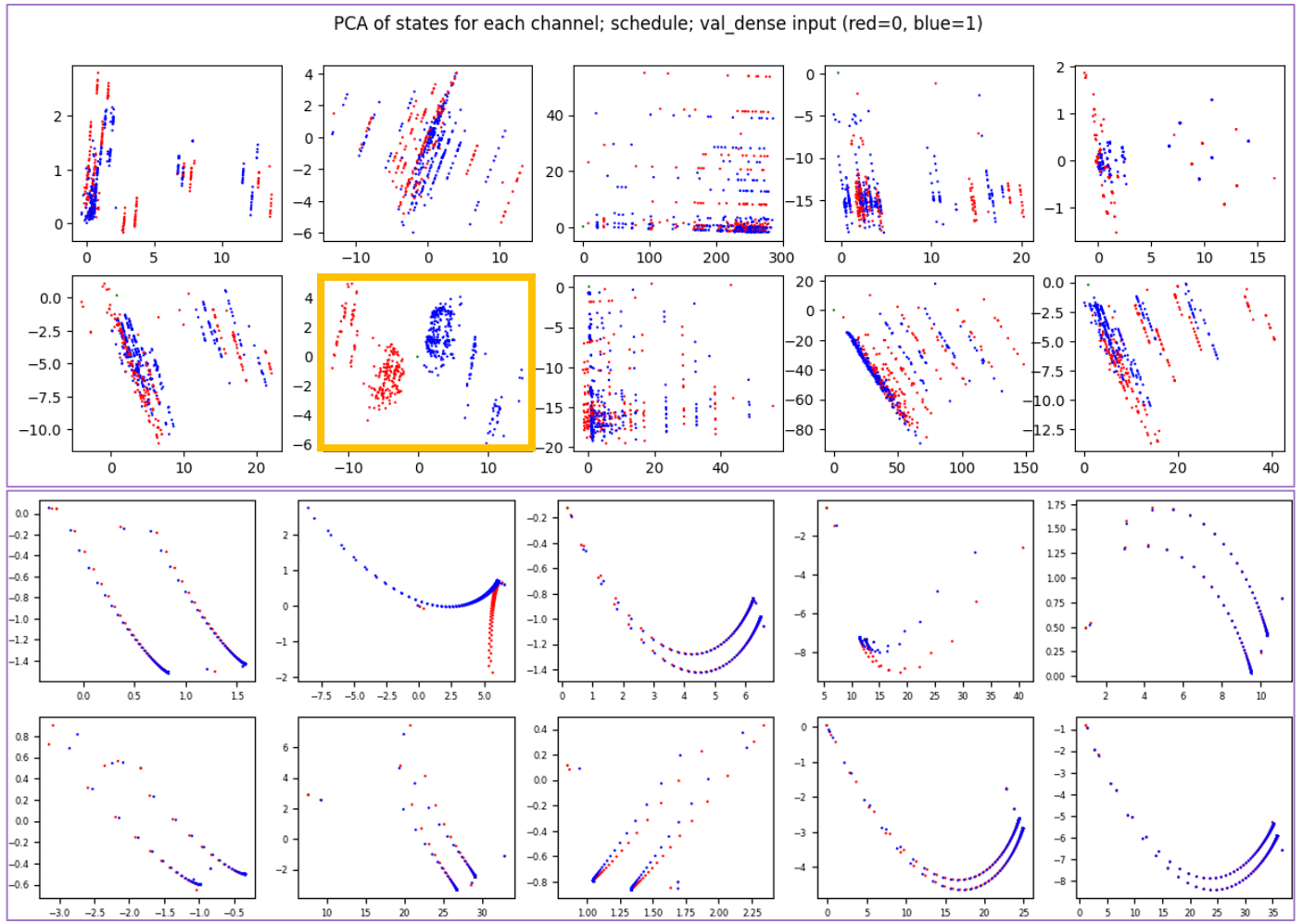
Truth about Mamba…

- Mamba smashes star-free tasks on benchmarks (e.g., Sarrof et al.)
- We show that Mamba cannot recognize **FlipFlop** as an η-finite system



Key: iterating $read$ input collapses
the entire state space to a single point

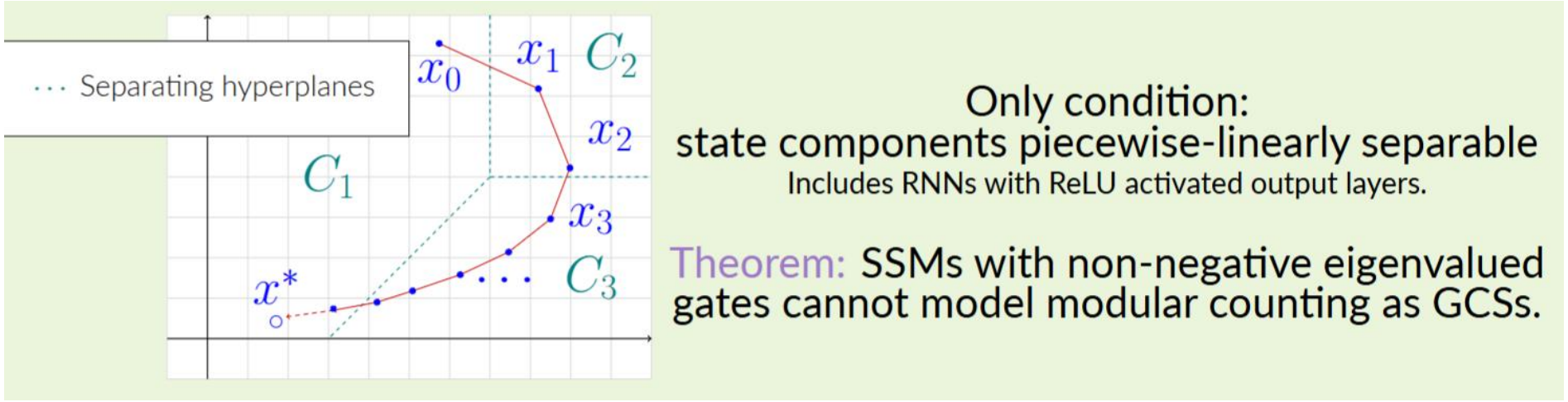
“Works, until it doesn’t”



3. Results, Empirical Validation and Future Work

Geometrically-constrained systems

- Constraint on **output function**



Only condition:
state components piecewise-linearly separable
Includes RNNs with ReLU activated output layers.

Theorem: SSMs with non-negative eigenvalued gates cannot model modular counting as GCSs.

- Explains why Mamba **empirically length-generalises** on star-free tasks and fails completely on modular counting.

Thank you