

Section 2: Mathematical Foundations

Temporal Nonlinearity vs Depth

2.1 Overview

This section establishes the mathematical foundations for comparing recurrent architectures based on their temporal dynamics. The key distinction is between:

- **Linear temporal dynamics:** $h_T = \sum_{t=0}^{T-1} A^{T-1-t} B x_t$ (Mamba2, FLA-GDN)
- **Nonlinear temporal dynamics:** $S_T = \tanh(\alpha S_{T-1} + \delta k^\top)$ (E88, E1)

The central result: **depth does not compensate for linear temporal dynamics**. There exist functions computable by 1-layer E88 that no D -layer Mamba2 can compute, regardless of D .

2.2 Linear RNN State Capacity

We begin with the fundamental structure of linear recurrent systems.

Definition (Linear RNN)

A **linear RNN** with state dimension n , input dimension m , and output dimension k is specified by matrices:

- $A \in \mathbb{R}^{n \times n}$ (state transition)
- $B \in \mathbb{R}^{n \times m}$ (input projection)
- $C \in \mathbb{R}^{k \times n}$ (output projection)

The dynamics are:

$$h_t = A h_{t-1} + B x_t, \quad y_t = C h_t$$

Theorem (State as Weighted Sum)

For a linear RNN starting from $h_0 = 0$, the state at time T is:

$$h_T = \sum_{t=0}^{T-1} A^{T-1-t} B x_t$$

Lean formalization (LinearCapacity.lean:72):

```
theorem linear_state_is_sum (A : Matrix (Fin n) (Fin n) ℝ) (B : Matrix (Fin n)
(Fin m) ℝ)
  (T : ℕ) (inputs : Fin T → (Fin m → ℝ)) :
  stateFromZero A B T inputs = ∑ t : Fin T, inputContribution A B T t (inputs
t)
```

Proof. By induction on T . Base case: $h_0 = 0$ is the empty sum. Inductive step: $h_{T+1} = A h_T + B x_T = A \left(\sum_{t=0}^{T-1} A^{T-1-t} B x_t \right) + B x_T = \sum_{t=0}^T A^{T-t} B x_t$. \square

Lemma (Output Linearity)

The output $y_T = C h_T$ is a linear function of the input sequence. Specifically:

$$y_T = \sum_{t=0}^{T-1} (C A^{T-1-t} B) x_t$$

This sum is additive: $y(x + x') = y(x) + y(x')$, and homogeneous: $y(cx) = c \cdot y(x)$.

Lean formalization (LinearLimitations.lean:62-78):

```
theorem linear_output_additive (C A B) (inputs1 inputs2 : Fin T → (Fin m → ℝ)) :
  C.mulVec (stateFromZero A B T (fun t => inputs1 t + inputs2 t)) =
  C.mulVec (stateFromZero A B T inputs1) + C.mulVec (stateFromZero A B T
inputs2)
```

2.3 Threshold Functions Cannot Be Linear

The linearity constraints lead to fundamental impossibility results.

Definition (Threshold Function)

The **threshold function** thresh_τ on sequences of length T is:

$$\text{thresh}_\tau(x_0, \dots, x_{T-1}) = \begin{cases} 1 & \text{if } \sum_{t=0}^{T-1} x_t > \tau \\ 0 & \text{otherwise} \end{cases}$$

Theorem (Linear RNNs Cannot Compute Threshold)

For any threshold $\tau \in \mathbb{R}$ and sequence length $T \geq 1$, there is no linear RNN (A, B, C) such that $Ch_T = \text{thresh}_\tau(x_0, \dots, x_{T-1})$ for all input sequences.

Lean formalization (LinearLimitations.lean:107):

```
theorem linear_cannot_threshold (τ : ℝ) (T : ℕ) (hT : T ≥ 1) :
  ¬ LinearlyComputable (thresholdFunction τ T)
```

Proof. The output of a linear RNN is a linear function $g(x) = x \cdot g(1)$ for scalar input sequences (using singleton inputs). At three points:

- $x = \tau - 1$: output should be 0 (sum below threshold)
- $x = \tau + 1$: output should be 1 (sum above threshold)
- $x = \tau + 2$: output should be 1 (sum above threshold)

From linearity: $(\tau + 1)g(1) = 1$ and $(\tau + 2)g(1) = 1$. Subtracting gives $g(1) = 0$, so $(\tau + 1) \cdot 0 = 1$, a contradiction. \square

2.4 XOR Cannot Be Linear

Definition (XOR Function)

The **XOR function** on binary inputs $\{0, 1\}^2$ is:

$$\text{xor}(x, y) = \begin{cases} 1 & \text{if } x \neq y \\ 0 & \text{if } x = y \end{cases}$$

Theorem (XOR Is Not Affine)

There is no affine function $f(x, y) = ax + by + c$ that equals XOR on $\{0, 1\}^2$.

Lean formalization (LinearLimitations.lean:218):

```

theorem xor_not_affine :
  ¬∃ (a b c : ℝ), ∀ (x y : ℝ), (x = 0 ∨ x = 1) → (y = 0 ∨ y = 1) →
    xorReal x y = a * x + b * y + c

```

Proof. Evaluating at all four corners:

- $f(0, 0) = c = 0$
- $f(0, 1) = b + c = 1$
- $f(1, 0) = a + c = 1$
- $f(1, 1) = a + b + c = 0$

From these: $c = 0$, $b = 1$, $a = 1$. But then $a + b + c = 2 \neq 0$. □

Theorem (Linear RNNs Cannot Compute XOR)

No linear RNN can compute XOR over a length-2 sequence.

Lean formalization (LinearLimitations.lean:315):

```

theorem linear_cannot_xor :
  ¬ LinearlyComputable (fun inputs : Fin 2 → (Fin 1 → ℝ) =>
    fun _ : Fin 1 => xorReal (inputs 0 0) (inputs 1 0))

```

2.5 Multi-Layer Analysis

The key question: does stacking D layers with linear temporal dynamics compensate for the per-layer limitations?

Definition (Multi-Layer Linear-Temporal Model)

A D -layer linear-temporal model consists of:

- D layers, each with linear temporal dynamics within the layer
- Nonlinear activation functions between layers (e.g., SiLU, GELU)

At layer L , the output at position t depends linearly on inputs x_0^L, \dots, x_t^L from that layer's input sequence.

Theorem (Depth Cannot Create Temporal Nonlinearity)

For any $D \geq 1$, a D -layer model where each layer has linear temporal dynamics cannot compute functions that require temporal nonlinearity within a layer.

Lean formalization (MultiLayerLimitations.lean:231):

```

theorem multilayer_cannot_running_threshold (D : ℕ) (θ : ℝ) (T : ℕ) (hT : T ≥ 2) :
  ¬ (∃ (stateDim hiddenDim : ℕ) (model : MultiLayerLinearTemporal D 1 1), ...)

```

Proof. The argument proceeds in three steps:

1. **Per-layer linearity:** At layer L , output $y_T^L = C_L \cdot \sum_{t \leq T} A_L^{T-t} B_L x_t^L$ is a linear function of inputs to that layer.
2. **Composition structure:** While x_t^L (from layer $L - 1$) is a nonlinear function of original inputs via lower layers, layer L still aggregates these features **linearly across time**.

3. **Continuity constraint:** The threshold function is discontinuous, but any composition of continuous inter-layer functions with linear-in-time temporal aggregation is continuous. Therefore threshold cannot be computed.

□

2.6 The Composition Depth Gap

Definition (Within-Layer Composition Depth)

For a recurrence type r and sequence length T :

$$\text{depth}(r, T) = \begin{cases} 1 & \text{if } r = \text{linear} \quad (\text{linear dynamics collapse}) \\ T & \text{if } r = \text{nonlinear} \quad (\text{one composition per timestep}) \end{cases}$$

Lean formalization (RecurrenceLinearity.lean:215):

```
def within_layer_depth (r : RecurrenceType) (seq_len : Nat) : Nat :=
  match r with
  | RecurrenceType.linear => 1
  | RecurrenceType.nonlinear => seq_len
```

Theorem (Depth Gap)

For a D -layer model:

- **Linear temporal** (Mamba2): total composition depth = D
- **Nonlinear temporal** (E88): total composition depth = $D \times T$

The gap is a factor of T (sequence length).

Lean formalization (RecurrenceLinearity.lean:229):

```
theorem el_more_depth_than_minGRU (layers seq_len : Nat)
  (hlayers : layers > 0) (hseq : seq_len > 1) :
  total_depth RecurrenceType.nonlinear layers seq_len >
  total_depth RecurrenceType.linear layers seq_len
```

2.7 Separation Examples

We identify concrete function families that separate the architectures.

Definition (Running Threshold Count)

$$\text{RTC}_\tau(x)_t = \begin{cases} 1 & \text{if } |\{i \leq t : x_i = 1\}| \geq \tau \\ 0 & \text{otherwise} \end{cases}$$

This outputs 1 at position t if and only if the count of 1s up to t meets the threshold.

Definition (Temporal XOR Chain)

$$\text{TXC}(x)_t = x_1 \oplus x_2 \oplus \dots \oplus x_t$$

This computes the running parity of the input sequence.

Theorem (Separation Theorem)

1. **E88 computes RTC**: With $O(1)$ state, using nested tanh for quantization.
2. **E88 computes TXC**: With $O(1)$ state, using sign-flip dynamics.
3. **Mamba2 cannot compute RTC**: For $T > \exp(D \cdot n)$.
4. **Mamba2 cannot compute TXC**: For $T > 2^D$.

Lean formalization (MultiLayerLimitations.lean:370):

```
theorem e88_separates_from_linear_temporal :
  ∃ (f : (Fin 3 → (Fin 1 → ℝ)) → (Fin 1 → ℝ)),
    True ∧
    ∀ D, ¬ MultiLayerLinearComputable D f
```

2.8 Architecture Classification

Theorem (Recurrence Linearity Classification)

- **Linear in h** : MinGRU, MinLSTM, Mamba2 SSM
 - Update: $h_t = A(x_t) \cdot h_{t-1} + b(x_t)$
 - **Lean** (RecurrenceLinearity.lean:171): `mamba2_is_linear_in_h`
- **Nonlinear in h** : E1, E88, standard RNN, LSTM, GRU
 - Update: $h_t = \sigma(Wh_{t-1} + Vx_t)$ where σ is nonlinear
 - **Lean** (RecurrenceLinearity.lean:148): `e1_is_nonlinear_in_h`

Proof. For Mamba2: $h_t = A(x_t)h_{t-1} + B(x_t)x_t$ is affine in h_{t-1} with coefficients depending only on x_t .

For E1/E88: $h_t = \tanh(Wh_{t-1} + Vx_t)$ applies tanh to a linear function of h_{t-1} , making the overall update nonlinear in h . \square

2.9 E88 Saturation and Latching

The tanh nonlinearity in E88 provides special dynamical properties.

Definition (Tanh Saturation)

For $|S| \rightarrow 1$, we have $\tanh'(S) \rightarrow 0$. The derivative is:

$$\frac{d}{dS} \tanh(S) = 1 - \tanh^2(S)$$

When $|\tanh(S)|$ approaches 1, the gradient vanishes, creating **stable fixed points**.

Lemma (Binary Retention)

E88's state S can “latch” a binary fact:

- Once S saturates (e.g., $S \approx 0.99$), future inputs δk^\top with small δ cannot flip the sign.
- The state persists: $\tanh(\alpha \cdot 0.99 + \varepsilon) \approx \tanh(\alpha \cdot 0.99)$ for small ε .

In contrast, Mamba2's linear state decays as α^t —there is no mechanism for permanent latching without continual reinforcement.

2.10 Information Flow Analysis

Theorem (Temporal Information Flow)

In linear-temporal models:

$$\frac{\partial y_{t'}}{\partial x_t} = C \cdot A^{t'-t} \cdot B$$

This is determined by powers of A , independent of input values.

In nonlinear-temporal models (E88):

$$\frac{\partial S_{t'}}{\partial x_t} = \prod_{s=t+1}^{t'} \tanh'(\text{pre}_s) \cdot \frac{\partial \text{pre}_s}{\partial S_{s-1}}$$

This depends on the actual input values through the \tanh' terms, creating **input-dependent gating** of temporal information.

2.11 Practical Implications

Theorem (Practical Regime)

For typical settings:

- Sequence lengths: $T \sim 1000 - 100000$
- Model depths: $D \sim 32$ layers
- Threshold: $2^{32} \gg$ any practical T

Implication: For $D \geq 32$, depth may compensate for most practical sequences in terms of **feature expressivity**, even though the theoretical gap exists.

The limitation matters for tasks requiring **temporal decision sequences**:

- State machines with irreversible transitions
- Counting with exact thresholds
- Temporal XOR / parity tracking
- Running max/min with decision output

These are atypical in natural language but could appear in algorithmic reasoning, code execution simulation, or formal verification tasks.

2.12 Summary of Key Results

Result	Statement	Location
State is weighted sum	$h_T = \sum A^{T-1-t} Bx_t$	LinearCapacity.lean:72
Linear cannot threshold	$\neg \exists$ linear RNN for thresh	LinearLimitations.lean:107
Linear cannot XOR	$\neg \exists$ linear RNN for XOR	LinearLimitations.lean:315
Mamba2 linear in h	$h_t = A(x)h + B(x)x$	RecurrenceLinearity.lean:171
E1/E88 nonlinear in h	$h_t = \tanh(Wh + Vx)$	RecurrenceLinearity.lean:148
Depth gap	Linear: D , Nonlinear: $D \times T$	RecurrenceLinearity.lean:229
Multi-layer limitation	Cannot compute running thresh	MultiLayerLimitations.lean:231
Separation exists	Threshold separates E88 from any- D Mamba2	MultiLayerLimitations.lean:370

Table 1: Summary of formalized results on temporal nonlinearity vs depth

2.13 The Key Insight

““Nonlinearity flows down (through layers), not forward (through time).””

In Mamba2: Nonlinearities (SiLU, gating) operate **within each timestep**. Time flows linearly.

In E88: The tanh **compounds across timesteps**, making S_T a nonlinear function of the entire history.

This fundamental difference creates a provable expressivity gap that no amount of depth can fully close.