Words and Temporal Graphs: Comparing the Expressive Power of State Space Models and Recurrent Neural Networks

Eric Alsmann, Lowejatan Noori and Martin Lange

Theoretical Computer Science / Formal Methods, University of Kassel, Germany

Abstract

We compare the expressive power of state space models (SSMs) and recurrent neural networks (RNNs) in two domains: over words and over temporal graphs. Building on recent logical characterisations of SSMs via fragments of pure-past linear temporal logic (pLTL_f), we lift the analysis to the graph domain. Our main results show that graph SSMs (GSSMs) are at least as expressive as the product logic $\mathrm{pLTL}_f \times \mathrm{K}^{\#}$, combining pure-past LTL with graded modal logic over the neighbourhood, while recursive temporal graph neural networks (recTGNNs) capture the strictly stronger logic $\mu\mathrm{TL}_f \times \mathrm{K}^{\#}$, a mu-calculus-style product logic. This mirrors the situation for words: SSMs recognise pLTL_f-definable, i.e. star-free properties, whereas RNNs can simulate arbitrary finite automata and therefore recognise all regular properties. The structural parallel reveals a fundamental architectural boundary: the ability to compute fixpoints over sequences of arbitrary length lies beyond SSMs in both domains.

Keywords

state space models, recurrent neural networks, temporal graph neural networks, expressive power, temporal logic

1. Introduction

State space models (SSMs) such as S4 [1] and Mamba [2] have emerged as a competitive alternative to transformer architectures for sequence modelling. A natural question asks: does their architectural similarity to recurrent neural networks (RNNs) translate into comparable expressive power? Recent theoretical work has shown that this is not the case for word sequences: Merrill et al. [3] established that common SSM variants lie within the circuit complexity class TC^0, and hence cannot simulate finite automata in general (unless $\mathrm{TC}^0 = \mathrm{NC}^1$). Further results by Sarrof et al. [4] and Alsmann et al. [5] complemented this with lower bounds: diagonal SSMs with fixed precision recognise exactly the star-free regular languages, captured by first-order logic or linear-time temporal logic, while RNNs are known to capture all regular languages [6].

The same question arises for *graph sequences* (temporal graphs). Recursive temporal graph neural networks (recTGNNs) generalise RNNs by adding a message-passing step at each time point. Sälzer et al. [7] started the study of the expressive power of temporal graph neural networks by using product logics [8] which can make assertions about the temporal evolution of nodes in a TGNN, as well as their spatial neighbourhood. We complement these recent results on the expressiveness of temporal graph neural network architectures, by investigating a recently proposed extension of SSMs to temporal graphs (GSSMs) [9]. We establish that diagonal-gated GSSMs are at least as expressive as $\mathrm{pLTL}_f \times \mathrm{K}^{\#}$, the natural graph analogue of pLTL_f adding graded counting of neighbours. We introduce the product logic $\mu\mathrm{TL}_f \times \mathrm{K}^{\#}$, inspired by the linear-time mu-calculus over finite traces [10] for temporal graphs, and prove that recTGNNs are at least as expressive as $\mu\mathrm{TL}_f \times \mathrm{K}^{\#}$, which strictly subsumes $\mathrm{pLTL}_f \times \mathrm{K}^{\#}$. This improves the previously known best lower bound by Sälzer et al. [7]. We exhibit the structural parallel: the expressiveness gap between GSSMs and recTGNNs mirrors exactly the gap between SSMs and RNNs at the word level.

LOGICNN 2026: Workshop on Logical Methods for Neural Network Analysis, July 25, 2026, Lisbon, Portugal
eric.alsmann@uni-kassel.de (E. Alsmann); l.noori@uni-kassel.de (L. Noori); martin.lange@uni-kassel.de (M. Lange)
0000-0002-2603-7827 (E. Alsmann); 0000-0002-1621-0972 (M. Lange)

2. Models

Words. An *SSM layer* processes a sequence $\boldsymbol{x}_1 \cdots \boldsymbol{x}_n \in (\mathbb{R}^d)^*$ via a linear recurrence $\boldsymbol{h}_t = A(\boldsymbol{x}_t) \cdot \boldsymbol{h}_{t-1} + \boldsymbol{B}\boldsymbol{x}_t$ and an output map, in general represented by a feed-forward neural network (FNN), $\boldsymbol{z}_t = \phi(\boldsymbol{h}_t, \boldsymbol{x}_t)$ [4]. A *diagonal* SSM restricts $A(\boldsymbol{x}_t)$ to a diagonal matrix, a *time-invariant* SSM fixes $A(\boldsymbol{x}_t) = A$ making it input-independent. An SSM stacks several such layers together with an embedding and an output FNN.

An *RNN* replaces the linear recurrence with a non-linear one: $\boldsymbol{h}_t = \sigma(\boldsymbol{A}\boldsymbol{h}_{t-1} + \bar{B}\boldsymbol{x}_t)$ for some activation σ. A classical result of Minsky's [6] shows that a single-layer RNN with threshold activation can simulate any finite automaton.

Temporal graphs. A *temporal graph* $TG = G_1 \cdots G_n$ is a sequence of graphs sharing the same node set V, where $G_t = (V, E_t, c_t)$ and $c_t : V \to \{0,1\}^k$ is a binary node colouring.

A *message-passing GNN (MPNN)* $M = (l^1, \ldots, l^k)$ is a sequence of layers $l^i = (\mathrm{comb}^i, \mathrm{agg}^i)$. Applied to a static graph $G = (V, E, c)$, it computes node embeddings $\boldsymbol{h}_v^{(0)} = c(v)$ and

$$\boldsymbol{h}_v^{(i+1)} = \mathrm{comb}^{i+1}\big(\boldsymbol{h}_v^{(i)},\ \mathrm{agg}^{i+1}(\{\{\boldsymbol{h}_u^{(i)} \mid \{u,v\} \in E\}\})\big),$$

where comb is an FNN and agg is the element-wise sum. The final embedding of v is $M(V, E, c, v) := \boldsymbol{h}_v^{(k)}$. The expressiveness of MPNNs was extensively studied using fragments of *graded modal logic* (the modal logic with counting modalities) [11, 12].

A *graph SSM (GSSM) layer* (A, B, M, ϕ) generalises an SSM layer by replacing the scalar input $\boldsymbol{x}_t$ with the output of an MPNN M applied to the current graph G_t:

$$(\boldsymbol{h}_v)_t = A(\boldsymbol{h}_v)_{t-1} + B \cdot M(V_t, E_t, [u \mapsto (\boldsymbol{x}_u)_t], v).$$

Here $A \in \mathbb{R}^{d \times d}$ is the gate matrix (diagonal for *diagonal-gated* GSSMs, arbitrary but fixed for *time-invariant* GSSMs), and M aggregates neighbourhood information. A GSSM stacks L such layers with an initialisation and an output FNN.

A *recursive temporal graph neural network (recTGNN)* (M, out) applies a fixed MPNN M repeatedly: at time step t, it runs M on G_t with the node feature augmented by the output of the previous step,

$$\boldsymbol{z}_v(t) = M\left(V_t, E_t, \left[u \mapsto \begin{pmatrix} c_t(u) \\ \boldsymbol{z}_u(t-1) \end{pmatrix}\right], v\right).$$

This is a direct graph generalisation of an RNN.

3. Logics

Over words. *Pure-past linear temporal logic over finite traces* (pLTL_f) [13] uses operators $\mathtt{Y}\varphi$ (yesterday), $\hat{\mathtt{Y}}\varphi$ (weak yesterday; also satisfied at the first position), $\mathtt{P}\varphi$ (at some past position), and $\varphi\,\mathtt{S}\,\psi$ (since). The semantics is given on word positions: $\sigma, i \models \varphi\,\mathtt{S}\,\psi$ iff there exists $k \le i$ with $\sigma, k \models \psi$ and $\sigma, j \models \varphi$ for all $k < j \le i$. A word σ satisfies φ iff $\sigma, |\sigma| \models \varphi$. The *unary fragment* un-pLTL_f omits $\mathtt{S}$.

pLTL_f is expressively equivalent to first-order logic $\mathrm{FO}[<]$ and to the class of star-free regular languages [14]. The extension $\mathrm{pLTL}_f[\mathrm{MOD}]$ adds modular predicates MOD_r^m ($i \equiv r \pmod m$) extending the expressiveness to all regular languages contained in AC^0 [15].

Over temporal graphs. *$pLTL_f \times K^\#$* (pure-past LTL times graded K) combines the temporal operators of pLTL_f with a modal counting inequality over the neighbourhood:

$$\varphi ::= c \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathtt{Y}\varphi \mid \hat{\mathtt{Y}}\varphi \mid \mathtt{P}\varphi \mid \varphi\,\mathtt{S}\,\varphi \mid \Diamond\varphi \mid \textstyle\sum_j a_j \cdot \mathbf{1}_{\varphi_j} + \sum_k b_k \cdot \#\psi_k \le d$$

Table 1
Expressiveness bounds for SSMs and RNNs over words.

Architecture	Lower bound
Diagonal SSM	$\geq$ pLTL$_f \equiv$ FO[<] [4, 5]
Time-inv. SSM	$\geq$ un-pLTL$_f$[MOD] [5]
Mixed SSM	$\geq$ pLTL$_f$[MOD] $\equiv$ REG $\cap$ AC^0 [5]
All SSM variants	$\subseteq$ TC^0 [3]
RNN	$\geq \mu$TL $\equiv$ REG [3, 6]

where $\Diamond\varphi$ is satisfied if a neighbour of the current node satisfies φ, $\#\psi$ counts the neighbours of the current node at the current time step satisfying ψ, and $a_j, b_k, d \in \mathbb{Z}$. Semantics is given on *pointed temporal graphs* (TG, v, t).

The product structure captures two independent axes of reasoning and their interaction. The *temporal axis* is handled by the pLTL$_f$ operators $\mathtt{Y}$, $\hat{\mathtt{Y}}$, $\mathtt{P}$, $\mathtt{S}$: they look *backwards in time* at the history of a single fixed node. The *spatial axis* is handled by $\Diamond$ and the counting inequalities $\#\psi$: they look *sideways across the graph* at the neighbours of the current node *at the current time step*. The two axes interact because the argument of a spatial operator can itself be a temporal formula, and vice versa. For example, $\#(\mathtt{Y}\, c)$ counts how many neighbours held colour c one step ago, while $\mathtt{P}(\Diamond\, c)$ asserts that some neighbour held c at some past time.

$\mu TL_f{\times}K^{\#}$ replaces the $\mathtt{S}$ operator and $\mathtt{P}$ operator with a fixpoint operator $\mathtt{rec}X.\varphi$: The fixpoint $\mathtt{rec}X.\varphi$ denotes the least set $S \subseteq V \times \{1, \ldots, n\}$ satisfying $S = [\![\varphi]\!]^{TG}[X \mapsto S]$, i.e. the smallest extension of X that makes φ true of every pair it contains; equivalently, it is the limit of the approximants $\mathtt{rec}^0 X.\varphi := \emptyset$ and $\mathtt{rec}^{k+1} X.\varphi := [\![\varphi]\!]^{TG}[X \mapsto \mathtt{rec}^k X.\varphi]$. A formula is (temporally) *guarded* if every bound variable X appears under the scope of a $\mathtt{Y}/\hat{\mathtt{Y}}$ operator within its fixpoint body. As with the modal μ-calculus, non-guarded formulas can be transformed into equivalent guarded ones, incurring an exponential blow-up in general [16, 17]. Since here we are interested in matters of expressive power, formula size plays a minor role and we can simply assume that formulas are guarded. This, together with the interpretation in TGNNs of finite temporal length, also ensures that fixpoints are unique, cf. [18] which is why we only introduce a single recursion operator $\mathtt{rec}$ instead of the dual μ/ν.

Lemma 1 (Convergence). *Let TG be a temporal graph of length n and φ a guarded $\mu TL_f{\times}K^{\#}$ formula. Then $[\![\mathtt{rec}^n X.\varphi]\!]^{TG} = [\![\mathtt{rec}X.\varphi]\!]^{TG}$.*

The key intuition is: guardedness forces every recursive call to "spend" a $\mathtt{Y}$-step, so the depth of unfolding equals the length of the graph. The standard pLTL$\times$K operators are expressible in μTL$_f{\times}$K$^{\#}$ via $\mathtt{P}\varphi \equiv \mathtt{rec}X.(\varphi \vee \mathtt{Y}X)$ and $\varphi\, \mathtt{S}\, \psi \equiv \mathtt{rec}X.(\psi \vee (\varphi \wedge \mathtt{Y}X))$, so pLTL$_f{\times}K^{\#} \subseteq \mu$TL$_f{\times}K^{\#}$.

4. Expressiveness over Words

We summarise known results for SSMs and RNNs over words as shown in Table 1. Diagonal SSMs with fixed precision are characterised by pLTL$_f$: the upper bound $\subseteq$ TC^0 comes from Merrill et al. [3], and the non-expressibility of the non-star-free language $(aa)^*$ by diagonal fixed-precision SSMs follows from a monotonicity argument (an SSM processing the same symbol repeatedly must eventually stabilise). The lower bound for diagonal SSMs is established constructively: each subformula of a pLTL$_f$ formula is mapped to one dimension of the hidden state. Propositional operators are computed by the FNN; the operators $\mathtt{P}$ and $\mathtt{Y}$ use time-invariant gates; the $\mathtt{S}$-operator requires an input-dependent diagonal gate to combine the current evaluation of its left argument with the accumulated past, following the unfolding $\varphi\, \mathtt{S}\, \psi \equiv \psi \vee (\varphi \wedge \mathtt{Y}(\varphi\, \mathtt{S}\, \psi))$. Time-invariant SSM layers – despite their inability to express the since operator – have the ability to track positions periodically by using permutation matrices [5].

Combining this together with diagonal SSM layers lifts the expressiveness of such mixed SSMs to all regular languages contained in AC^0.

RNNs, by contrast, can recognise any regular language by encoding the transition function of a finite automaton in a threshold activation layer, immediately placing them at the full class of regular languages.

5. Expressiveness over Temporal Graphs

5.1. Lower Bound for GSSMs

The following result lifts the word-level lower bound for diagonal SSMs to the graph setting.

Theorem 1 (GSSM $\geq$ pLTL$_f$×K$^\#$)**.** *For every pLTL$_f$×K$^\#$ formula φ there exists a diagonal-gated GSSM S_φ that correctly classifies all pointed temporal graphs with respect to φ. The construction uses a number of layers equal to the nesting depth of φ.*

Analogously, every formula in un-pLTL$_f$[MOD]×K$^\#$ is recognised by a diagonal, time-invariant GSSM.

Proof sketch. The structure follows the word-level construction of Alsmann et al. [5]. Fix a pLTL$_f$×K$^\#$ formula φ and an injective index map ι from subformulas to hidden-state dimensions. Each layer evaluates all subformulas of a fixed nesting depth simultaneously, propagating results to deeper layers.

The crucial observation is that the node feature sequence $(\boldsymbol{x}_v)_1 \cdots (\boldsymbol{x}_v)_n$ for a fixed node v behaves exactly like a word: the temporal operators $\mathtt{Y}/\hat{\mathtt{Y}}$ and $\mathtt{P}$ are handled by the same gating constructions as in the word case (leaving M as the identity GNN for those operators), and the $\mathtt{S}$-operator uses an input-dependent diagonal gate.

The new case is the modal inequality $\sum_j a_j \cdot \mathbf{1}_{\varphi_j} + \sum_k b_k \cdot \#\psi_k \leq d$. Here the GSSM gate is set to zero (the formula is purely positional), and the GNN M is a one-layer MPNN with sum aggregation:

$$M(V_t, E_t, [u \mapsto \boldsymbol{x}_u], v) = \boldsymbol{x}_v \oplus \sum_{u \in N_t(v)} \boldsymbol{x}_u,$$

where $\oplus$ denotes concatenation. The second summand collects $\#\psi_k$ counts in the hidden state, and the output FNN applies the linear comparison. All other dimensions are left unchanged by identity-projections. □

5.2. Lower Bound for recTGNNs

The simulation of μTL$_f$×K$^\#$ by recTGNNs relies on a normal form that structures fixpoint formulas so that their evaluation reduces to a single forward pass over the temporal graph.

Any guarded μTL$_f$×K$^\#$ formula can be rewritten as a *vectorial fixpoint system* $\mathtt{rec}X_j.\Psi$, where $\Psi = (X_1{=}\psi_1, \ldots, X_m{=}\psi_m)$ is a vector of equations with a designated entry variable X_j. Mutual fixpoints are unfolded into nested single-variable fixpoints via the Bekić Lemma [19]: given equations $X{=}\varphi$ and $Y{=}\psi$, one has

$$\mathtt{rec}X.\begin{bmatrix} X = \varphi \\ Y = \psi \end{bmatrix} \equiv \mathtt{rec}X.\,\varphi\big[\mathtt{rec}Y.\psi \,/\, Y\big].$$

Applying this repeatedly reduces a mutual system to a sequence of single-variable fixpoints. Within the resulting system, every free occurrence of a bound variable X in a right-hand side ψ_i may be nested under several $\mathtt{Y}/\hat{\mathtt{Y}}$ operators. We *normalise to single-Y depth*: for every $\mathtt{Y}\phi$ in ψ_i such that ϕ contains a bound variable, we introduce a fresh equation $Z = \phi$ and replace ϕ by Z. This does not change the semantics (an application of the Bekić Lemma in the other direction), and terminates after at most $|\Psi|$ rounds. The result is a vectorial system in which every bound variable occurrence is under *exactly one* $\mathtt{Y}/\hat{\mathtt{Y}}$.

Lemma 2 (Vectorial normal form)**.** *Every guarded $\mu TL_f{\times}K^{\#}$ formula φ is equivalent to a vectorial formula $\mathtt{rec}X.\Psi$ in which each bound variable occurs under exactly one $\mathtt{Y}/\hat{\mathtt{Y}}$ operator.*

The single-$\mathtt{Y}$ condition has a crucial semantic consequence: evaluating ψ_i at (v,t) can only *query a bound variable at time $t-1$*. Hence, if the $(t{-}1)$-th approximant is correctly known for all earlier positions, the t-th approximant can be computed from it in one formula evaluation. This yields the following.

Lemma 3 (Convergence, restated)**.** *Let $\mathtt{rec}X.\Psi$ be in vectorial normal form and TG a temporal graph of length n. Then $[\![\mathtt{rec}^n X.\Psi]\!]^{TG} = [\![\mathtt{rec}X.\Psi]\!]^{TG}$.*

Proof sketch. The proof is by induction on t: for $t=1$ the approximant is empty (since $\mathtt{Y}$-formulas are false at position 1); for $t>1$ one can show that $T_{\leq t} \subseteq [\![\mathtt{rec}^t X.\Psi]\!]$ by observing that a pair (v,t) in the fixpoint uses only pairs $(v',t-1)$, which are already covered by the induction hypothesis.

For the MPNN construction it is convenient to require that for every $\mathtt{Y}\phi/\hat{\mathtt{Y}}\phi$ occurring in the normal form, the body ϕ is already in a form where $\mathtt{Y}/\hat{\mathtt{Y}}$ cannot be pushed further inward: either $\phi \in \mathcal{V}\cup\mathcal{C}$ (a variable or colour), or $\phi = \Diamond\psi$ where ψ is built from colours, resp. atomic propositions, variables, disjunctions, negations, and modal operators only.

This *simple form* is achieved by pushing $\mathtt{Y}/\hat{\mathtt{Y}}$ through Boolean connectives using distributivity:

$$\mathtt{Y}(\phi_1\vee\phi_2) \equiv \mathtt{Y}\phi_1 \vee \mathtt{Y}\phi_2, \qquad \hat{\mathtt{Y}}(\phi_1\vee\phi_2) \equiv \hat{\mathtt{Y}}\phi_1 \vee \hat{\mathtt{Y}}\phi_2, \qquad \mathtt{Y}(\neg\phi) \equiv \neg\hat{\mathtt{Y}}\phi, \qquad \hat{\mathtt{Y}}(\neg\phi) \equiv \neg\mathtt{Y}\phi.$$

Applying these rules repeatedly in the right-hand sides of Ψ yields an equivalent vectorial formula in simple form. We call the resulting system a *simple* vectorial formula. Given a simple formula $\mathtt{rec}X.\Psi$, the subformulas of Ψ can be split into two disjoint sets:

$$P_1(\Psi) = \{\, \psi \in \mathrm{sub}^+(\Psi) \mid \text{every variable in } \psi \text{ is guarded (under } \mathtt{Y}) \,\},$$
$$P_2(\Psi) = \{\, \psi \in \mathrm{sub}^+(\Psi) \mid \text{every variable in } \psi \text{ is unguarded} \,\}.$$

Simplicity ensures that no subformula contains both guarded and unguarded variable occurrences simultaneously. Intuitively, P_1-formulas read information from the *previous* approximant (via $\mathtt{Y}/\hat{\mathtt{Y}}$), while P_2-formulas assemble the *current* approximant from the P_1-results just computed. This split into two is what allows a single forward pass of the MPNN to advance the fixpoint by exactly one step. □

Theorem 2 (recTGNN $\geq \mu\mathrm{TL}_f{\times}\mathrm{K}^{\#}$)**.** *For every guarded $\mu TL_f{\times}K^{\#}$ formula φ there exists a recTGNN T_φ whose output at time step t encodes the t-th approximant $[\![\mathtt{rec}^t X.\varphi]\!]^{TG}$ for all nodes and all temporal graphs. In particular, by Lemma 3, T_φ recognises $L(\varphi) = \{(TG,v) \mid TG,(v,n) \models \varphi\}$.*

Proof sketch. By Lemmas 2–3 we may assume that φ is a simple vectorial formula $\mathtt{rec}X.\Psi$ with subformula partition $(P_1(\Psi), P_2(\Psi))$.

Encoding. Each node v carries a hidden vector $\boldsymbol{h}_v(t) \in \mathbb{R}^{2d}$ at time step t, where $d = |\mathcal{C}| + |\mathrm{sub}^+(\Psi)|$. The vector is partitioned into an *upper half* (indices via $\iota_\uparrow$) and a *lower half* (indices via $\iota_\downarrow$), each holding one Boolean value per non-variable subformula:

- $(\boldsymbol{h}_v(t))_{\iota_\downarrow(\psi)} = 1$ encodes whether $(v,t{-}1)$ satisfies ψ under the appropriate approximant environment. This is the *read* half, populated from the previous time step via the recTGNN recurrence.
- $(\boldsymbol{h}_v(t))_{\iota_\uparrow(\psi)} = 1$ encodes whether (v,t) satisfies ψ under the current approximant. This is the *write* half, computed layer by layer within the MPNN.

Because the recTGNN initialises $c'_t(v) = (c_t(v), \boldsymbol{z}_v(t{-}1))$, the previous output $\boldsymbol{z}_v(t{-}1) = (\boldsymbol{h}_v(t{-}1))_{\iota_\uparrow(\cdot)}$ is automatically available in the lower half of $\boldsymbol{h}_v(t)$ at the start of each time step.

Two-phase MPNN. The MPNN M consists of $\mathrm{nd}(\Psi) = d_1 + d_2$ layers, split into a P_1-phase (layers $1,\ldots,d_1$) and a P_2-phase (layers $d_1+1,\ldots,d_1+d_2$), where $d_j = \max\{\mathrm{nd}(\psi) \mid \psi \in P_j(\Psi)\}$. Within each phase, layer i simultaneously computes all subformulas of nesting depth i (resp. $i-d_1$) by accumulating the relevant copy matrices without mutual interference (each targets a distinct index):

Table 2
Expressiveness over words and temporal graphs. Each row is an architectural variant; columns show the corresponding logic lower bound in both domains.

Architecture	Words	Temporal graphs
Diag., time-inv.	$\geq$ un-pLTL$_f$[MOD]	$\geq$ un-pLTL$_f$[MOD]$\times$K$^{\#}$
Diagonal	$\geq$ pLTL$_f \equiv$ FO[$<$]	$\geq$ pLTL$_f\times$K$^{\#}$
Mixed	$\geq$ pLTL$_f$[MOD] $\equiv$ REG $\cap$ AC^0	$\geq$ pLTL$_f$[MOD]$\times$K$^{\#}$
Upper bound (all SSM/GSSM)	$\subseteq$ TC^0	$\subseteq$ TC^0 (?)
RNN / recTGNN	$\geq \mu$TL $\equiv$ REG	$\geq \mu$TL$_f\times$K$^{\#}$

- P_1**-phase** ($\psi \in P_1(\Psi)$): variables appear only under $\mathtt{Y}$, so they are read from $\iota_\downarrow$ (the lower half, i.e., the previous time step). The four connectives are handled as in the word case: $\neg$ via bias, $\vee$ via summation and trReLU, $\Diamond$ and counting formulas via the GNN sum-aggregation weight B, and $\mathtt{Y}$ via a copy from $\iota_\downarrow$. After the P_1-phase, $\iota_\uparrow(\psi)$ correctly encodes $(v,t) \in [\![\psi]\!]_{\rho_t}$ for all $\psi \in P_1(\Psi)$, where ρ_t maps each X_i to the $(t-1)$-th approximant $[\![\mathtt{rec}^{t-1}X_i.\Psi]\!]$.
- P_2**-phase** ($\psi \in P_2(\Psi)$): variables appear *unguarded*, so they refer to the *current* fixpoint value. After the P_1-phase, the entries $\iota_\uparrow(\chi(X_i))$ for each variable X_i already hold the t-th approximant value (just computed). The P_2-phase reads these entries directly from $\iota_\uparrow$ and applies the same four connective rules. After this phase, $\iota_\uparrow(\psi)$ encodes $(v,t) \in [\![\psi]\!]_{\rho_{t+1}}$ for all $\psi \in P_2(\Psi)$, where ρ_{t+1} maps X_i to the t-th approximant, exactly the environment needed so that P_2-results serve as the correct lower half input for time step $t+1$.

A final projection layer copies the upper half to the output $\boldsymbol{z}_v(t)$, which becomes the lower half of $\boldsymbol{h}_v(t+1)$ in the next recTGNN step. The correctness of the whole construction is proved by nested induction on the time step t and the layer index i, using the two-phase invariant above. For $t = n$, Lemma 3 gives $[\![\mathtt{rec}^n X.\Psi]\!] = [\![\mathtt{rec}X.\Psi]\!]$, so the output of T_φ at the last step is correct. □

5.3. Separation

The separation pLTL$_f\times$K$^{\#} \subsetneq \mu$TL$_f\times$K$^{\#}$ follows because μTL$_f\times$K$^{\#}$ can express properties requiring counting of temporal patterns, such as "node v has been connected to a blue node at every even time step", by exploiting fixpoints over the $\mathtt{Y}$-operator in combination with $\Diamond$. GSSMs with a fixed gate matrix cannot track such periodic patterns beyond the capacity of their finite-dimensional hidden state, by an argument analogous to the $(aa)^*$ impossibility at the word level. The inclusion pLTL$_f$[MOD]$\times$K$^{\#} \subseteq \mu$TL$_f\times$K$^{\#}$ is strict unless TC$^0 =$ NC^1, because GSSMs cannot express NC^1-complete problems like permutation tracking (S_5), even when combining both types of gates [3].

6. The Structural Parallels

Table 2 displays the expressiveness landscape side by side for words and temporal graphs.

The parallel is striking. In both settings:

1. **SSMs / GSSMs**: Their expressive power in the diagonal case corresponds exactly to the first-order (star-free) fragment of the relevant temporal logic: pLTL$_f$ on words, pLTL$_f\times$K$^{\#}$ on temporal graphs. The modal counting in pLTL$_f\times$K$^{\#}$ corresponds to the FNN-based sum aggregation of the GNN. While SSMs are bounded from above by TC^0 it remains to see whether the same holds for GSSMs.
2. **RNNs / recTGNNs** capture fixpoint properties: all regular languages on words, and the mu-calculus product logic μTL$_f\times$K$^{\#}$ on temporal graphs allowing arbitrary regular properties to be

expressed in the temporal domain. The common mechanism is the ability to accumulate *arbitrary* states across time steps via a non-linear gate, effectively implementing the Kleene iteration of a monotone operator.

3. **The gap** in both cases corresponds exactly to the inability of SSMs/GSSMs to compute fixpoints of non-trivial recursive operators.

7. Conclusion

We have shown that the expressiveness gap between SSMs and RNNs carries over from the word to the temporal-graph domain. Diagonal GSSMs are at least as expressive as $\mathrm{pLTL}_f \times \mathrm{K}^{\#}$, the product of pure-past LTL and graded modal logic, while recTGNNs capture the strictly stronger mu-calculus product logic $\mu\mathrm{TL}_f \times \mathrm{K}^{\#}$. This parallel suggests a unifying principle: the architectural boundary separating SSMs from RNNs is not specific to the sequential word domain; instead the linearised recursive mechanism reflects a general inability to compute fixpoints under bounded depth.

Future work includes establishing matching upper bounds for recTGNN and GSSMs.

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] A. Gu, K. Goel, C. Re, Efficiently modeling long sequences with structured state spaces, in: International Conference on Learning Representations, 2022. URL: https://openreview.net/forum?id=uYLFoz1vlAC.

[2] A. Gu, T. Dao, Mamba: Linear-time sequence modeling with selective state spaces, in: First Conference on Language Modeling, 2024. URL: https://openreview.net/forum?id=tEYskw1VY2.

[3] W. Merrill, J. Petty, A. Sabharwal, The illusion of state in state-space models, in: R. Salakhutdinov, Z. Kolter, K. Heller, A. Weller, N. Oliver, J. Scarlett, F. Berkenkamp (Eds.), Proceedings of the 41st International Conference on Machine Learning, volume 235 of *Proceedings of Machine Learning Research*, PMLR, 2024, pp. 35492–35506. URL: https://proceedings.mlr.press/v235/merrill24a.html.

[4] Y. Sarrof, Y. Veitsman, M. Hahn, The expressive capacity of state space models: A formal language perspective, in: The Thirty-eighth Annual Conference on Neural Information Processing Systems, 2024. URL: https://openreview.net/forum?id=eV5YIrJPdy.

[5] E. Alsmann, L. Noori, M. Lange, On the expressiveness of state space models via temporal logics, in: The Fourteenth International Conference on Learning Representations, 2026. URL: https://openreview.net/forum?id=Vg511oJScS.

[6] M. Minsky, Theory of Neural-Analog Reinforcement Systems and Its Application to the Brain Model Problem, Princeton University, 1954.

[7] M. Sälzer, P. A. Wałęga, M. Lange, The logical expressiveness of temporal GNNs via two-dimensional product logics, in: The Thirty-ninth Annual Conference on Neural Information Processing Systems, 2026. URL: https://openreview.net/forum?id=v13yQBxhut.

[8] D. Gabbay, A. Kurucz, F. Wolter, M. Zakharyaschev, Many-Dimensional Modal Logics: Theory and Applications, volume 148 of *Studies in Logic and the Foundations of Mathematics*, North Holland, 2003. URL: http://www.loc.gov/catdir/enhancements/fy0620/2004272978-d.html.

[9] J. Li, R. Wu, X. Jin, B. Ma, L. Chen, Z. Zheng, State space models on temporal graphs: A first-principles study, in: The Thirty-eighth Annual Conference on Neural Information Processing Systems, 2024. URL: https://openreview.net/forum?id=UaJErAOssN.

[10] Y. Liu, Z. Duan, C. Tian, B. Cui, Satisfiability of Linear Time Mu-Calculus on Finite Traces, in: T. N. Dinh, M. T. Thai (Eds.), Computing and Combinatorics, Springer International Publishing, Cham, 2016, pp. 611–622. doi:10.1007/978-3-319-42634-1_49.

[11] P. Barceló, E. V. Kostylev, M. Monet, J. Pérez, J. Reutter, J. P. Silva, The logical expressiveness of graph neural networks, in: International Conference on Learning Representations, 2020. URL: https://openreview.net/forum?id=r1lZ7AEKvB.

[12] M. Grohe, The logic of graph neural networks, in: Proceedings of the 36th Annual ACM/IEEE Symposium on Logic in Computer Science, IEEE Press, 2021, pp. 1–17. URL: https://doi.org/10.1109/LICS52264.2021.9470677. doi:10.1109/LICS52264.2021.9470677.

[13] G. De Giacomo, A. Di Stasio, F. Fuggitti, S. Rubin, Pure-past linear temporal and dynamic logic on finite traces, in: Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI'20, Yokohama, Yokohama, Japan, 2021, pp. 4959–4965.

[14] J. A. W. Kamp, Tense Logic and the Theory of Linear Order, University of California, 1968.

[15] D. A. Mix Barrington, K. Compton, H. Straubing, D. Thérien, Regular languages in NC1, Journal of Computer and System Sciences 44 (1992) 478–499. doi:10.1016/0022-0000(92)90014-A.

[16] I. Walukiewicz, Completeness of Kozen's axiomatisation of the propositional μ-calculus, Inf. and Comput. 157 (2000) 142–182.

[17] F. Bruse, O. Friedmann, M. Lange, On guarded transformation in the μ-calculus, Logic Journal of the IGPL 23 (2015) 194–216.

[18] R. Mateescu, Local model-checking of modal mu-calculus on acyclic labeled transition systems, in: Proc. 8th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems, TACAS'02, volume 2280 of *LNCS*, Springer, 2002, pp. 281–295.

[19] H. Bekić, Programming Languages and Their Definition, Selected Papers, volume 177 of *LNCS*, Springer, 1984.