LENGTH-INDEPENDENT STATE TRACKING UNDER A PARALLEL SCAN

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning robust and scalable finite-state tracking is fundamental to sequence processing. While linear recurrent neural networks (RNNs), linear attention, and state space models enable scalable parallel training through affine recurrences, their theoretical expressivity guarantees assume idealized arithmetic and do not extend to finite precision, where the parallel scan that makes them fast is itself a source of perturbation. We formalize finite-state tracking at finite precision and characterize *length independence*: tracking that stays correct at every sequence length, at a precision cost that does not grow with the length. We show that length-independent state tracking requires two competing dynamics within a single map: *contraction* to suppress numerical perturbations and *separation* to keep distinct states apart. We prove that affine recurrences, which offer a single rate at each step to serve both roles, realize at most *definite* automata at finite precision. Instead of treating scan compatibility as a restriction on the update map, we reinterpret it as a computational budget and introduce the *Neural Finite-State Machine* (NFSM): a nonaffine, scan-compatible RNN built for length-independent finite-state tracking. On synthetic benchmarks spanning abelian and nonabelian groups, noninvertible monoids, and textual state-tracking tasks, affine baselines fail on every nondefinite task, most of them within a few hundred steps. A single NFSM layer instead learns the exact transition tables of every algebraic task, which certifies correctness beyond the tested lengths, and a stack of NFSMs keeps perfect accuracy on the textual tasks at every tested length.

1 INTRODUCTION

State tracking is the maintenance of an internal memory variable that is sequentially updated as inputs arrive, capturing the cumulative effect of the past inputs. It enables models to resolve pronouns across texts (Kim & Schuster, 2023), trace variable states during code execution (Zaremba & Sutskever, 2015), and maintain beliefs about partially observable environments (Kaelbling et al., 1998). Synthetic tasks such as flip-flop retrieval, permutation composition, and modular counting isolate this capability directly (Liu et al., 2023a;b; Delétang et al., 2023). Learning state tracking from data is a central challenge in sequence processing.

Recurrent neural networks (RNNs) naturally possess the structure required for state tracking (Casey, 1996; Omlin & Giles, 1996a), but traditional training requires unrolling through time, leading to $O(L)$ sequential depth for sequence length L. To address this bottleneck, recent architectures leverage affine recurrences, which can be parallelized via scan algorithms to achieve $O(\log L)$ depth (Blelloch, 1990; Martin & Cundy, 2018). This approach, which underlies modern State Space Models (SSMs) (Smith et al., 2023; Gu & Dao, 2024), gated linear RNNs (Martin & Cundy, 2018; Qin et al., 2023; Feng et al., 2024), and linear attention (Katharopoulos et al., 2020; Yang et al., 2024b), introduces a fundamental tradeoff between computational efficiency and the expressivity of the state dynamics. An extensive body of work has analyzed this tradeoff in exact or log-precision arithmetic (Merrill et al., 2024; Sarrof et al., 2024; Grazzi et al., 2025; Siems et al., 2025), yielding architectures that theoretically attain full finite-state tracking capabilities (Terzić et al., 2025).

However, these expressivity guarantees rely on idealized arithmetic and often do not hold for trained models. Practical models operate at finite precision, learn parameters that only approximate optimal target solutions, and are trained on finite context windows. In practice, state tracking succeeds

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

up to the training horizon but collapses toward chance not far beyond it (Shakerinava et al., 2026; Karuvally et al., 2025; Grazzi et al., 2025; Terzić et al., 2025). This collapse shows that theoretical expressivity is necessary but not sufficient: it does not survive the gap between idealized constructions and finite-precision, gradient-trained models. Most of this gap is a learning problem, but not all of it (Appendix E). We address the part that no amount of training removes, finite precision, and focus on *finite-state* tracking, where the state takes values in a finite set and evolves as in a finite automaton. We ask three questions: (i) What conditions must an RNN satisfy to achieve state tracking at finite precision over arbitrary sequence lengths? (ii) Can affine recurrences satisfy these conditions? (iii) If not, can we design a scan-compatible RNN that guarantees robust state tracking for arbitrarily long sequences at finite precision?

We call a realization *length-independent* when it tracks the state correctly at every sequence length at a fixed precision. Unlike empirical extrapolation, length independence requires an RNN to execute exact target transitions at every step *at a precision cost that does not grow with* L. Correctness for any sequence length then follows by composition rather than empirical evaluation. We model finite-precision arithmetic as a perturbation of the hidden state bounded by $\eta > 0$ at every step. The bound η is set by the arithmetic resolution and the conditioning of the recurrence, and it includes the perturbation injected by the scan itself. Our analysis answers questions (i)–(iii) in turn.

(i) Length-independent state tracking at finite precision reduces to three conditions on a single step, which together demand two competing dynamics within one map: *contraction* to suppress numerical errors, and *separation* to maintain distinguishable states (Section 3).

(ii) Affine recurrences offer a single rate, which over repeated steps cannot serve both roles; consequently, at any precision $\eta > 0$, they realize at most *definite* automata, whose state depends only on a bounded suffix of the input (Section 4).

(iii) Scan compatibility is a budget on the representation of the recurrence update maps, not a restriction to affine maps (Section 5). Within this budget, the *Neural Finite-State Machine* (NFSM), a nonaffine recurrence, realizes every finite automaton with length independence and runs under a parallel scan (Section 6).

To validate our theoretical results, we evaluate NFSM (Section 7) on synthetic state-tracking tasks spanning abelian and nonabelian groups, noninvertible monoids, and textual *Tracking Shuffled Objects* tasks (Srivastava et al., 2023). Affine baselines fail beyond a finite length on every nondefinite task, whereas a single NFSM layer keeps perfect sequence accuracy at every tested length on the algebraic tasks, and a stack of two layers keeps perfect question accuracy on the textual tasks. On the algebraic tasks, the transition tables extracted from the trained models match the target exactly, which certifies correctness beyond the tested lengths instead of extrapolating it.

Section 8 draws out the implications. Proofs and additional mathematical derivations are available in Appendix F, and Appendix C discusses related work. All source code is available at `https://dd8wgy467d-hash.github.io/reviewanonymous/`, and readers are invited to follow along with our interactive illustration as they progress through the paper.

2 Formalization

This section formalizes state tracking and connects finite-state automata to RNNs. We describe the transition structure of each system, show how an RNN can realize the dynamics of an automaton, and account for finite-precision perturbations.

State tracking and automata. We formalize state tracking using a semiautomaton (automaton for short), defined as a 3-tuple (Q, Σ, δ), where Q is a set of states, Σ is an input alphabet, and $\delta : Q \times \Sigma \to Q$ is the transition function specifying how inputs change the state. In this work, we focus on state tracking with finite Q of size n. A single input $x \in \Sigma$, called a symbol, acts on the state space through $\delta_x \coloneqq \delta(\cdot, x) : Q \to Q$. We call a finite sequence of symbols a word, and denote the set of all words by Σ^*. For a word $w = x_1 \cdots x_{|w|} \in \Sigma^*$, single-symbol transitions chain to form the map $\delta_w : Q \to Q$, defined by $\delta_w \coloneqq \delta_{x_{|w|}} \circ \cdots \circ \delta_{x_1}$, with $\delta_\varepsilon \coloneqq \mathrm{id}_Q$ for the empty word ε. Thus, $\delta_w(q)$ is the state reached from q after reading the word w. The set of all such maps, $M_\delta = \{\delta_w \mid w \in \Sigma^*\}$, is the *transition monoid* of the automaton, a submonoid of the *full transformation monoid* $T_n \coloneqq \{f : Q \to Q\}$, which contains all n^n possible state

mappings. Since every n-state task induces a submonoid of T_n, an architecture that can implement every element of T_n as a transition covers all n-state tasks; the NFSM of Section 6 is such an architecture (Proposition G.1).

Recurrent neural network. An RNN carries a *hidden state* h_t in a space $\mathcal{H} \coloneqq \mathbb{R}^d$, where d is the hidden state dimension, and updates it through an input-driven update function $\Phi : \mathcal{H} \times \Sigma \to \mathcal{H}$:

$$h_t = \Phi(h_{t-1}, x_t), \qquad t = 1, \dots, L, \tag{1}$$

from an initial hidden state h_0; the hidden state h_t is the memory the RNN keeps of past inputs. Each input symbol x applies a transformation $\Phi_x \coloneqq \Phi(\cdot, x) : \mathcal{H} \to \mathcal{H}$ to the hidden state space, and, as for δ_w above, processing a word $w = x_1 \cdots x_{|w|}$ chains these into a composed map $\Phi_w \coloneqq \Phi_{x_{|w|}} \circ \cdots \circ \Phi_{x_1} : \mathcal{H} \to \mathcal{H}$, with $\Phi_\varepsilon \coloneqq \mathrm{id}_{\mathcal{H}}$. These maps form the *RNN transition monoid*, $M_\Phi = \{\Phi_w \mid w \in \Sigma^*\}$. Throughout, *state* refers to the automaton and *hidden state* to the RNN.

The RNN architectures that have survived the scalability demands of modern deployment, namely SSMs, gated linear RNNs, and linear attention, are affine recurrences

$$h_t = A_{x_t} h_{t-1} + b_{x_t}, \qquad t = 1, \dots, L, \tag{2}$$

where $A_x \in \mathbb{R}^{d \times d}$ and $b_x \in \mathbb{R}^d$ are input-driven, with arbitrary dependence on the symbol; we call such a recurrence *affine in the hidden state*. Their appeal is that affine maps compose in closed form, which lets a parallel scan evaluate them.

Automaton realization. The automaton evolves in a discrete state space and the RNN in a continuous one, so realizing an automaton by an RNN requires an *encoding* $\iota : Q \to \mathcal{H}$ of automaton states as representations and a partial *readout* $\pi : \mathcal{H} \rightharpoonup Q$ decoding them back, such that encoding a state, running the word through the RNN and reading out gives the same state as running the word through the automaton:

$$\pi\big(\Phi_w(\iota(q))\big) = \delta_w(q) \qquad \text{for every } q \in Q, w \in \Sigma^*. \tag{3}$$

Equivalently, M_Φ must contain, for every $\delta_w \in M_\delta$, a map Φ_w reproducing its action on Q through (ι, π), so the class of automata an RNN can express is fixed by the achievable algebraic structure of M_Φ. We say that an RNN *realizes* the automaton (Q, Σ, δ) if such a triple (ι, Φ, π) exists. In this paper, we further ask which state-tracking properties survive computation at finite precision.

Automaton realization at finite precision. At finite precision, each RNN step lands near its intended image rather than exactly on it: $h_t = \tilde{\Phi}_{x_t}(h_{t-1})$, where $\tilde{\Phi}_{x_t}(h_{t-1})$ is the approximation of $\Phi_{x_t}(h_{t-1})$ that accounts for numerical perturbations of x_t and of Φ itself. We write this as $\tilde{\Phi}_{x_t}(h_{t-1}) = \Phi_{x_t}(h_{t-1}) + e_t$, where e_t is a per-step error (Appendix D, Lemma D.1).

Definition 2.1 (η-trajectory and η-reachable set)**.** *Let $\eta > 0$ denote the maximum per-step perturbation level, and let $\bar{B}_\eta \coloneqq \{e \mid \|e\| \le \eta\}$ be the corresponding perturbation ball for a fixed norm $\|\cdot\|$ on $\mathcal{H}$. We call $(h_t)_{t=0}^{|w|}$ an η-trajectory of a word $w = x_1 \cdots x_{|w|}$ from h_0 if $h_t - \Phi_{x_t}(h_{t-1}) \in \bar{B}_\eta$ for every t, and define the η-reachable set of the recurrence as $U^\star \coloneqq \{h_{|w|} \mid q \in Q, w \in \Sigma^*, (h_t)_{t=0}^{|w|}$ is an η-trajectory of w from $\iota(q)\}$.*

An η-trajectory formalizes a realistic, perturbed execution path of an RNN processing an input sequence, where no single update deviates by more than η from its exact representation. The η-reachable set $U^\star$ is the set of all hidden states reachable from an encoded state, over words of any length, under bounded perturbations. For $U^\star$ to admit a representation using a finite number of bits regardless of input length, it must be bounded (Proposition F.1). Automaton realization at finite precision thus requires correct decoding along every η-trajectory and a bounded η-reachable set.

Definition 2.2 (Length-independent realization at finite precision)**.** *A triple (ι, Φ, π) realizes the target automaton (Q, Σ, δ) with length independence at finite precision, modeled by a per-step perturbation level $\eta > 0$, if, for the η-reachable set $U^\star$:*

(T1) Correct at every length. *$\pi(h_{|w|}) = \delta_w(q)$ for every $q \in Q$, $w \in \Sigma^*$, and every η-trajectory $(h_t)_{t=0}^{|w|}$ of w from $\iota(q)$.*

(T2) Fixed precision at every length. *$U^\star$ is bounded.*

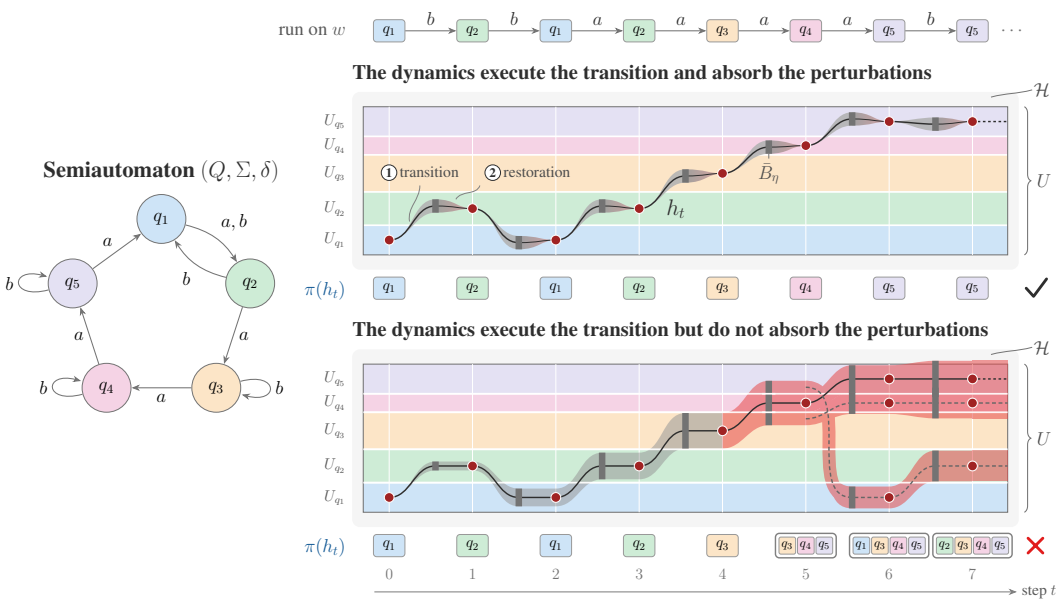

Figure 1: **Length independence requires executing the transition and restoring the state.** The same automaton (left), realized twice and unrolled over time; the top strip is its run on w. The vertical axis sketches $\mathcal{H}$, with the bounded region U split into the cells $U_q = \{u \in U : \pi(u) = q\}$, drawn as bands, and the code points $\iota(q)$ as red dots. **Top**: ① the update moves the hidden state into the next cell, and computing it spreads the state over $\bar{B}_\eta$; ② restoration pulls that ball back onto the code point, so $\pi(h_t) = q_t$ at every length. **Bottom**: nothing pulls the hidden state back, so the ball grows by η per step and at $t = 5$ straddles U_{q_3}, U_{q_4} and U_{q_5}; the single symbol a then drives three runs at once (dashed) and $\pi(h_t)$ is not unique anymore.

3 Length independence requires three conditions on one step

In this section, we first translate the trajectory-level conditions of Definition 2.2 into equivalent step-level conditions on Φ. We then show what these step-level conditions impose on the geometry of the RNN dynamics.

First, (T1) with the empty word requires $\pi \circ \iota = \mathrm{id}_Q$ (Lemma F.2). Second, (T1) requires the perturbed RNN update $\tilde{\Phi}_x$ to map hidden states representing q to hidden states representing the correct automaton successor $\delta_x(q)$ at every step on the η-reachable set $U^\star$ (Lemma F.4). Finally, (T2), the boundedness of $U^\star$, keeps the representation within a fixed region of the hidden state space, so that a fixed, finite number of bits suffices at every sequence length (Proposition F.1). We formalize these conditions in Theorem 3.1, where $\oplus$ denotes the Minkowski sum, $A \oplus B = \{a + b \mid a \in A, b \in B\}$, and maps act setwise, i.e., $\Phi_x(\mathcal{S}) = \{\Phi_x(s) \mid s \in \mathcal{S}\}$.

Theorem 3.1 (Conditions for length-independent realization at finite precision)**.** *A triple (ι, Φ, π) realizes the target automaton (Q, Σ, δ) with length independence at $\eta > 0$* iff *there is a region $U \supseteq \iota(Q)$ on which π is defined and that satisfies*

(R1) Decodable code. $\pi \circ \iota = \mathrm{id}_Q$.
(R2) Restoration. *U is bounded and $\Phi_x(U) \oplus \bar{B}_\eta \subseteq U$ for every $x \in \Sigma$.*
(R3) Correct transition. $\pi(\Phi_x(u) + e) = \delta_x(\pi(u))$ *for every $x \in \Sigma$, $u \in U$ and $\|e\| \le \eta$.*
Whenever such a region U exists, the η-reachable set $U^\star$ itself satisfies (R1)–(R3), and $U^\star \subseteq U$.

(R1) to (R3) reproduce, from a coding argument, von Neumann's division of labor: a chain of unreliable stages is reliable at arbitrary depth only if each carries an *executive organ* performing the computation and a *restoring organ* that "erase[s] the degradation caused by the executive organs" (von Neumann, 1956). (R1) is the code, (R2) the restoring organ, and (R3) the executive organ (Appendix C.1). What is specific to recurrence is that one map must act as both the restoring and the executive organ (Figure 1).

We call the encoding $\iota(q)$ of the states *code points*, and their set $\iota(Q)$ the *code*. The readout partitions U into disjoint cells $U_q \coloneqq \{u \in U \mid \pi(u) = q\}$: (R1) places each code point in its own cell, (R2) keeps the partition U bounded, and (R3) demands that, under an input x, an entire cell U_q land inside $U_{\delta_x(q)}$ with a margin of at least η, so that one-step perturbations cannot push the state out of the target cell.

Theorem 3.2 (Cell partition)**.** *A triple (ι, Φ, π) realizes the target automaton (Q, Σ, δ) with length independence at $\eta > 0$* iff *there exists a bounded region $U \supseteq \iota(Q)$ partitioned into cells $U_q \coloneqq \{u \in U \mid \pi(u) = q\}$ such that $\iota(q) \in U_q$ for all $q \in Q$, and*

$$\Phi_x(U_q) \oplus \bar{B}_\eta \subseteq U_{\delta_x(q)} \qquad \text{for every } q \in Q \text{ and } x \in \Sigma. \tag{4}$$

Theorems 3.1 and 3.2 constrain single steps. Iterated along a word $w \in \Sigma^+$ (with $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$), equation 4 imposes two geometric requirements on every composite map Φ_w, uniformly in $|w|$:

(1) Local contraction: The image $\Phi_w(U_q)$ must be 2η narrower than the cell $U_{\delta_w(q)}$ it lands in, so that a perturbation cannot push it out. When w returns to its starting state, $\delta_w(q) = q$, the cell shrinks strictly into itself: numerical errors are damped rather than compounded.

(2) Global separation: At the same time, the images of cells U_q and $U_{q'}$ with $\delta_w(q) \neq \delta_w(q')$ must stay bounded away from one another, however long w is, preventing η-perturbations from causing cross-cell misclassifications.

Length-independent state tracking therefore requires *dual-rate dynamics*: local contraction inside cells paired with global separation between them.

Theorem 3.3 (Two rates in one map)**.** *Under equation 4, every composite map Φ_w is at once* locally contracting *and* globally separating*: for every $q, q' \in Q$ and every $w \in \Sigma^+$,*

- ***(i) Local contraction.*** $\operatorname{diam}\big(\Phi_w(U_q)\big) \leq \operatorname{diam}\big(U_{\delta_w(q)}\big) - 2\eta$*; in particular,* $\operatorname{diam}\big(\Phi_w(U_q)\big) \leq \operatorname{diam}(U_q) - 2\eta$ *whenever* $\delta_w(q) = q$*.*
- ***(ii) Global separation.*** $\operatorname{dist}\big(\Phi_w(U_q), \Phi_w(U_{q'})\big) \geq 2\eta \quad$ *whenever* $\quad \delta_w(q) \neq \delta_w(q')$*.*

Figure 1 shows both properties at work, and what fails without restoration (Corollary F.6).

4 Length independence requires a nonaffine recurrence

Length-independent realization imposes a dual constraint on Φ along every word: the image of each cell must be 2η narrower than the cell it lands in (Theorem 3.3(i)), while the images of cells heading to different states must stay 2η apart (Theorem 3.3(ii)), with margins that do not shrink as the word grows. This section shows that affine recurrences cannot satisfy this dual constraint along the words of a nondefinite target, and identifies what a recurrence requires to satisfy it.

4.1 Affine recurrences supply a single rate

In affine recurrences, the displacement between two trajectories depends only on their initial displacement:

$$\Phi_w(h) - \Phi_w(h') = A_w(h - h'), \quad A_w \coloneqq A_{x_{|w|}} \cdots A_{x_1}. \tag{5}$$

An affine recurrence supplies a single rate A_w over the whole hidden state space, hence the same operator determines the dynamics of both the signal and the perturbation at each time step (Definition F.7, Proposition F.8). Along repeated words w^k (w written k times, so $A_{w^k} = A_w^k$), the operator A_w^k that must damp the perturbation also shrinks the gap between the codes of two states that the target keeps apart, until that gap falls below 2η, which Theorem 3.3(ii) forbids. Selective (input-dependent) matrices modulate that single rate *through time*, not across space, so they do not escape this.

Specifically, an affine recurrence loses either (R2) or (R3) depending on the spectral radius $\rho(A_w)$: either $\rho(A_w) \geq 1$ for some word, so perturbations accumulate or amplify, or $\rho(A_w) < 1$ for every word, so older inputs are forgotten along with the perturbation. Forgetting old inputs is harmless only when the target never needs them, that is, when it is a *definite* automaton, whose state depends only on the last k inputs, for some fixed k (Perles et al., 1963). The first regime precludes length independence for every target, the second for every nondefinite one. Both still allow length generalization up to a horizon $L_{\max}$.

Theorem 4.1 (Affine recurrences realize at most definite automata)**.** *Let Φ be affine in the hidden state, $\Phi_x(h) = A_x h + b_x$ for $x \in \Sigma$.*

*(i) If $\rho(A_w) \geq 1$ for some word $w \in \Sigma^+$, then (R2) **fails**: no nonempty bounded set $U \subset \mathcal{H}$ satisfies $\Phi_w(U) \oplus \bar{B}_\eta \subseteq U$ for this w. Consequently, Φ realizes no automaton with length independence at any $\eta > 0$.*

*(ii) If $\rho(A_w) < 1$ for all words $w \in \Sigma^+$, then Φ realizes **at most a definite automaton** with length independence at $\eta > 0$. For any nondefinite target, no region satisfying (R1) and (R2) satisfies (R3).*

Most scan-compatible architectures fall on one of the two branches of Theorem 4.1 (see Table 3 in Appendix C.2), and Section 7 shows both failure modes empirically. Concurrent work reaches the same limits of affine recurrences on error propagation by a different route (Chung et al., 2026).

4.2 Two rates require multistability

Theorem 4.1 shifts the question from the *spectrum* of a transition matrix to the *geometry* of the update map. Specifically, realization demands a cell partition (Theorem 3.2) that absorbs local perturbations while mapping each region cleanly into its prescribed successor, a dynamic reminiscent of the restoring organ of Section 3.

The canonical neural system with this error-correcting behavior is the Hopfield network (Hopfield, 1982), but its restoration is *static*: iterating Φ to convergence returns a corrupted state to *the same* pattern it started near. A state tracker must execute a *dynamic restoration*: a single step of Φ_x must simultaneously remove the perturbation *and* move the hidden state to the correct *successor* cell. Satisfying equation 4 thus requires a single update map to maintain multiple distinct cells, one per automaton state. When the target is not definite, some word w acts on Q as an idempotent δ_w that is not constant (Lemma F.11), so it fixes at least two states and, by Theorem 3.3, Φ_w shrinks each of their cells into itself while keeping them 2η apart. No affine map does this (Proposition F.22): the recurrence must be *multistable*, which requires a nonaffine operation on the previous hidden state h_{t-1} (Lemma F.17). Its *robust capacity*, the number of bits one map can hold under perturbation (Definition F.20), is then $\log_2 n$, n being the number of cells, with one attractor per cell and one cell per automaton state. A definite target needs neither multistability nor nonaffinity (Proposition F.13).

This capacity is set by the number of cells rather than by the dimension: extra width allows for wider margins rather than more states, and extra depth leaves the temporal recurrence within each layer affine, so the robust capacity of an affine recurrence beyond a bounded suffix of the input is zero at any width (Proposition F.22) and any depth, whatever the nonlinear maps between layers (Theorem F.25).

5 Length independence is compatible with a parallel scan

To meet modern standards of scalability, a recurrence has to be *compatible with the parallel scan*. The scan operates on state-update maps Φ_x and their compositions Φ_w. Because function composition is associative, these maps form a monoid; the core requirement for a scan-compatible recurrence is therefore not associativity, but *bounded representation*. The parallel scan combines these bounded composites pairwise across sequence steps, with costs governed by storage per composite (b bits) and computation per composition (c operations).

Definition 5.1 (Scan-representable recurrence)**.** *The recurrence $h_t = \Phi(h_{t-1}, x_t)$ is* scan-representable with budget (b, c) *if there exists an injective representation $\zeta : M_\Phi \to \{0,1\}^b$ of the maps of M_Φ, the identity Φ_ε included, and an algorithm computing $\zeta(\Phi_v \circ \Phi_u)$ from $(\zeta(\Phi_v), \zeta(\Phi_u))$ in c operations, for all $\Phi_u, \Phi_v \in M_\Phi$.*

Given per-symbol representations $\zeta(\Phi_{x_1}), \ldots, \zeta(\Phi_{x_L})$, and assuming that building $\zeta(\Phi_x)$ from a symbol and evaluating a composite at a hidden state also cost $O(c)$, a parallel scan computes $h_1, \ldots, h_L$ in depth $O(\log L)$, work $O(Lc)$, and memory $O(Lb)$ (Blelloch, 1990). The depth, the number of successive rounds of compositions, sets the running time with enough processors, whereas a sequential recurrence needs L rounds.

Proposition 5.2 (The budget forces a finite reachable set)**.** *If Φ is scan-representable with budget (b, c), then M_Φ has at most 2^b elements and, for every initial hidden state h_0, so does the set*

$\{\Phi_w(h_0) \mid w \in \Sigma^*\}$ of reachable hidden states, $h_0 = \Phi_\varepsilon(h_0)$ included. On these $K \le 2^b$ hidden states, M_Φ acts as a submonoid of T_K, the full transformation monoid on K elements.

The scan is itself a perturbation source. Affine maps form a monoid only in exact arithmetic: floating-point operations are not associative, so the composite a scan returns differs from the sequential one, and an affine recurrence, lacking restoration, lets this error accumulate (Appendix D). The perturbation floor η is thus as much a product of parallelizing an affine recurrence as an assumption imposed on it.

Multistable recurrences can resolve this mismatch by replacing continuous drift with discrete basins. Consider K attractors $a_1, \dots, a_K \in V$, each in its own basin $a_k \in V_k$, whose basins $V_1, \dots, V_K$ partition V. Under **exact restoration**, any hidden state in a basin contracts in one step onto an attractor: Φ is *exactly restoring* on V if every Φ_x maps each basin V_k to a single attractor $a_{\sigma_x(k)}$. The attractors and their basins are those of the restoring organ of Section 3, one and the same map for every symbol; the executive organ then carries a_k into the basin of $a_{\sigma_x(k)}$ and the restoring organ finishes the step, so a_k is a fixed point of Φ_x itself exactly when $\sigma_x(k) = k$ (Remark F.26). Consequently, the hidden states reachable from an attractor are attractors, and each symbol acts as a transition table σ_x on the K attractor indices, which meets the budget of Definition 5.1: a table costs $b = K\lceil \log_2 K \rceil$ bits, and composing two of them is a lookup per index, $c = O(K)$ operations.

This table-composition view explains architectures that achieve scan-compatible multistable dynamics. Memory Recurrent Units (De Geeter et al., 2026; Brandoit et al., 2026), which assume that the hidden state converges onto an attractor between two steps, implicitly build such tables (Appendix C.3).

Scan compatibility and length independence can coexist. Mapping attractor indices onto automaton states, with $K = n$, $a_q = \iota(q)$ and cells $U_q \coloneqq a_q \oplus \bar{B}_\eta$ that fit inside the basins V_q, turns exact restoration into the cell inclusions of equation 4 as soon as every table is the target transition, $\sigma_x = \delta_x$: the recurrence then realizes the target with length independence (Theorem 3.2) and fits the budget at $b = n\lceil \log_2 n \rceil$ and $c = O(n)$. Since composites of index maps are index maps, the merge is exact whatever the shape of the tree and the scanned trajectory is bit-identical to the sequential one (Proposition G.1). Section 6 builds such a recurrence for every finite automaton.

6 The neural finite-state machine

The *Neural Finite-State Machine* (NFSM) is a multistable recurrence that learns the transition tables of Section 5 directly. We first describe one head as an RNN, then show how it realizes a finite automaton, and finally assemble heads into a block.

One head. A head has a state $h_t \in \mathbb{R}^d$ and, for each input x, a logit matrix $\theta_x \in \mathbb{R}^{d\times d}$ given by a learned map of the input. Writing $\mathrm{rd}(h) \coloneqq \mathbf{1}_{\arg\max_j \mathbf{1}_j^\top h}$ for the basis vector of the largest coordinate of h, where $\mathbf{1}_j^\top h$ selects the j-th coordinate of h, the head updates its state as

$$\Phi_x(h) = \mathrm{rd}\big(\theta_x\, \mathrm{rd}(h)\big). \tag{6}$$

The outer rd is the restoring organ of Section 3: it maps the output of the product with θ_x back to the code, absorbing every perturbation that leaves its largest entry in place. The inner rd does the same for the stored state; on a trajectory it acts as the identity, since h_t is already a basis vector. The product with θ_x is the executive organ: since $\theta_x \mathbf{1}_k$ is column k of θ_x, composing it with the outer rd sends $\mathbf{1}_k$ to $\mathbf{1}_{\sigma_x(k)}$, with $\sigma_x(k)$ the row of the largest entry of that column:

$$\sigma_x(k) \coloneqq \arg\max_j\ \mathbf{1}_j^\top \theta_x \mathbf{1}_k, \qquad \mathrm{rd}(\theta_x \mathbf{1}_k) = \mathbf{1}_{\sigma_x(k)}.$$

The head is thus multistable, with d attractors $a_k \coloneqq \mathbf{1}_k$ and basins $V_k \coloneqq \{h \in \mathbb{R}^d \mid \arg\max_j \mathbf{1}_j^\top h = k\}$, the cones of states whose largest coordinate is the k-th. It is exactly restoring in the sense of Section 5: in one step, each basin V_k collapses onto the single attractor $a_{\sigma_x(k)}$. The head outputs a learned vector attached to its current attractor. The map rd jumps wherever the largest coordinate changes, so the recurrence is not affine, as multistability requires (Lemma F.17).

Realization. As at the end of Section 5, we now map attractor indices onto automaton states. Assume $d \ge n$, number the states $Q = \{1, \dots, n\}$, and take

$$\iota(q) \coloneqq \mathbf{1}_q, \qquad \pi(h) \coloneqq \arg\max_{q\in Q} \mathbf{1}_q^\top h,$$

that is, π reads the index of the basin V_k that contains h. (R1) holds, since $\pi(\iota(q)) = \arg\max_{q' \in Q} \mathbf{1}_{q'}^{\mathsf{T}} \mathbf{1}_q = q$, so $\pi \circ \iota = \mathrm{id}_Q$. For (R2), take as cells the η-balls around the code points, $U_q \coloneqq \iota(q) \oplus \bar{B}_\eta$. The basins do not overlap around the attractors, so for η small enough, independently of the length of the word and of the target, each cell U_q lies inside its basin V_q, and the cells are pairwise disjoint. Their disjoint union $U \coloneqq \bigsqcup_{q \in Q} U_q$ is bounded. One step collapses each cell onto an attractor, so $\Phi_x(U) \oplus \bar{B}_\eta \subseteq U$ as soon as σ_x maps Q into itself. One map thus carries the two rates of Theorem 3.3: every cell shrinks to a point, while distinct attractors stay more than 2η apart. (R3) holds if and only if $\sigma_x = \delta_x$ on Q, that is, if the largest entry of column q of θ_x sits in row $\delta_x(q)$ for every $q \in Q$. One head realizes any automaton on at most d states with length independence (Propositions G.1 and G.2). Finite precision can only perturb the logits, and a perturbation below half the smallest column margin leaves every table unchanged at every length (Proposition G.3).

Scan compatibility. Each table σ_x depends on its symbol alone, through the logits θ_x, so a head builds the tables of a sequence in parallel and composes them by a parallel scan of depth $O(\log L)$ whose merges are gathers, $(\sigma_y \circ \sigma_x)(k) = \sigma_y(\sigma_x(k))$. This is the table budget of Section 5, $d\lceil \log_2 d \rceil$ bits and $O(d)$ operations, and every merge is exact (Proposition G.1). The NFSM is one block within a potentially larger model. Training through the $\arg\max$ uses a straight-through Gumbel-softmax estimator (Jang et al., 2017) (Appendix G.6). Appendix H.7 reports wall-clock measurements of the scan implementation.

Heads as memory channels. A target often tracks several quantities at once, and giving each its own memory channel keeps them apart. The NFSM is therefore a block of r heads that read the same input and update independently: head i has d_i attractor indices and holds one component of the joint state $h_t = \left(h_t^{(1)}, \ldots, h_t^{(r)}\right) \in \mathcal{H} = \mathbb{R}^{d_1} \times \cdots \times \mathbb{R}^{d_r}$. The quantities need not be independent: a running product of permutations, for instance, can be carried point by point by several heads (Example G.7). Targets whose factors depend on one another are realized by stacking layers, each reading the attractor indices of the layer below (Proposition G.6). Heads also keep the cost low: k independent quantities of v values cost k heads, kv^2 logits per step, rather than one head of v^k attractor indices, v^{2k} logits. The heads are scanned side by side, so their budgets add, and their outputs are concatenated and projected back to the model width.

7 Experiments

We compare NFSM with three affine recurrences that cover both branches of Theorem 4.1: Mamba$^-$, Mamba with negative eigenvalues (Grazzi et al., 2025), with $\rho < 1$; AUSSM (Karuvally et al., 2025), with $\rho = 1$; and PD-SSM (Terzić et al., 2025), with $\rho < 1$ and a hardmax that selects the transition matrix but does not act on the hidden state.

The algebraic tasks present a sequence of generators of a target monoid, and the target at each position is the composite of the generators seen so far. They cover abelian groups ($\mathbb{Z}_2$, $\mathbb{Z}_{16}$), nonabelian groups (S_3, S_4, and the nonsolvable A_5 and M_{11}, the first Mathieu group), and the noninvertible flip-flop monoid, drawn in two ways: DFF_5 never draws more than four consecutive identities, so that the state is a function of the last five inputs, whereas FF draws identity runs of unbounded length. The textual task TSO_N (Srivastava et al., 2023) gives the same kind of algebra through text: swaps between $N \in \{3, 4, 5\}$ people holding N items are described in natural language, for example "*alice has the key. alice swaps with bob. ... who has the key?*", and the answer must be read off the tracked state.

All models share one backbone (Appendix H.3): an encoder embeds the input tokens into a latent space, the recurrent layers act within this latent space, and a decoder maps it to the predicted states. Only the recurrent layers differ. The baselines use four layers of width 256. The NFSM uses a single layer on the algebraic tasks, with one head on $\mathbb{Z}_2$, $\mathbb{Z}_{16}$ and the flip-flops, and several heads on the permutation groups, for instance 4 heads of 11 attractor indices on M_{11}, each tracking where one point is sent. On TSO_N it stacks two multi-head layers. Models are trained on sequences of a fixed length, with the state supervised at every position, and tested on longer ones. The failing length $L_{\max}$ is the smallest tested length at which accuracy falls below 90% for the baselines and below 100% for NFSM, where accuracy is the fraction of sequences whose predicted states are all

correct on the algebraic tasks, and of correctly answered questions on TSO_N. Appendix H details the baselines, tasks, model sizes, and training.

Table 1: **Failing length on algebraic and textual state-tracking tasks.** Each cell reports the median failing length $L_{\max}$ over the seeds that learned the task, and below it how many of the five seeds did (Appendix H.4). $\geq L$: no failure up to L, the longest sequence that fits on one GPU. †: the extracted tables match the target exactly, which certifies correctness at every length (Appendix H.6.2). $\times$: no seed learned the task. Brackets below the task names give the number of reachable states.

Failing length $L_{\max}$ (NFSM at 100%, baselines at 90%; trained at L=64)									trained at L=256		
	$\mathbb{Z}_2$ [2]	$\mathbb{Z}_{16}$ [16]	S_3 [6]	S_4 [24]	A_5 [60]	M_{11} [7920]	DFF_5 [3]	FF [3]	$\mathrm{TSO}_{N=3}$ [44]	$\mathrm{TSO}_{N=4}$ [202]	$\mathrm{TSO}_{N=5}$ [1132]
NFSM 1 layer, 2 on TSO	$\geq$1M† 5/5	$\geq$1M† 5/5	$\geq$1M† 5/5	$\geq$1M† 5/5	$\geq$1M† 5/5	$\geq$32k† 5/5	$\geq$1M† 5/5	$\geq$1M† 5/5	$\geq$1M 5/5	$\geq$1M 5/5	$\geq$128k 5/5
Mamba$^-$ 4 × 256	4352 5/5	85 4/5	186 5/5	105 4/5	$\times$ 0/5	$\times$ 0/5	$\geq$1M 5/5	178 5/5	1501 2/5	$\times$ 0/5	$\times$ 0/5
AUSSM 4 × 256	144 5/5	135 5/5	128 5/5	89 5/5	82 5/5	$\times$ 0/5	96 5/5	119 5/5	$\times$ 0/5	$\times$ 0/5	$\times$ 0/5
PD-SSM 4 × 256	110 5/5	98 5/5	133 5/5	131 5/5	187 5/5	$\times$ 0/5	$\geq$256k 5/5	133 5/5	363 5/5	1922 1/5	321 1/5

NFSM tracks every task; affine baselines do not. With a single layer, NFSM keeps perfect accuracy up to the longest tested length on every algebraic task and for all five seeds, including the nonsolvable groups and the noninvertible flip-flops, and a stack of two layers does the same on TSO_N (Table 1). The baselines, with four layers and larger states, fail within a few hundred steps on most tasks or do not learn them at all.

Affine recurrences fail as Theorem 4.1 predicts. The flip-flop pair isolates the definite boundary: DFF_5 draws only definite words, which a contracting recurrence can hold, whereas FF does not. The two contracting baselines, Mamba$^-$ and PD-SSM, hold DFF_5 up to the longest tested length but fail on FF, while AUSSM, which does not contract, fails on both. The learned trajectories show the two branches (Appendix H.5). Where the baselines learn, the horizon they reach can vary widely across seeds (Appendix H.4). A horizon measured on one trained model therefore does not transfer to the next, which is why we ask for length independence rather than a long horizon.

Extracted tables certify length independence. Each NFSM step commits to a table entry, so the tables a trained model implements can be read off its logits (Appendix H.6). For every algebraic task and every seed, they match the target exactly. By Propositions G.2 and G.4, this implies correct tracking at every length, provided the logit margin of Proposition G.3 exceeds the error of the logit computation. On TSO_N, Appendix H.6.3 shows how the heads of the two layers track the registers of the task.

8 Discussion

Scan compatibility, expressivity, and robustness to finite precision are often viewed as competing demands; we have shown that a single mechanism meets all three. A bounded representation budget forces the recurrence onto a finite set of reachable hidden states, and a well-separated set of them lets one map both erase the step-wise error and execute the next transition. An affine recurrence has a single rate, so at any finite precision it either lets perturbations grow or realizes at most a definite automaton, at any width and depth. For models built on contracting recurrences, this is an *undeclared context window*: older inputs reach the output only through a term that decays geometrically, so beyond a length set by learned gates and the arithmetic floor they are indistinguishable from noise, and that length is exceeded without any signal. The NFSM removes this barrier and leaves learning, the main practical obstacle, checkable rather than solved: the automaton a trained model implements can be read off its weights, so long-horizon correctness is verified rather than extrapolated (Appendix E). Behaviors that need a state set growing with the input, such as unbounded counting, remain open (Appendix B).

AI USE STATEMENT

In this work, we used generative AI tools in a supporting role for for formulating mathematical claims, assisting in the writing of proofs, proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, implementing methods, and assisting with translation.

We did not use generative AI tools for interpreting results or providing critical ingredients for proving mathematical claims. Generating synthetic datasets, developing theoretical models or conceptual frameworks, cleaning or reformatting datasets, and supporting qualitative or thematic data analysis are not applicable to this work.

Additionally, we partially used generative AI tools for identifying relevant literature, creating or editing software code, and editing the research paper to improve readability.

We have reviewed all AI-assisted work. Specifically, all mathematical claims, proofs, and methodological designs were manually checked for rigor and technical accuracy by the authors; AI-generated and edited code was verified and unit-tested; translated text was checked for semantic accuracy; and all referenced literature and information retrieved via AI were independently verified against original source publications. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

The research presented in this work focuses on general and foundational aspects of artificial intelligence. It does not involve human subjects, sensitive personal data, or direct deployment in high-risk domains. Consequently, we do not believe that our work raises any direct or specific ethical concerns, privacy violations, or immediate risks of misuse.

However, as a contribution to the broader field of artificial intelligence, we are fully conscious of the far-reaching impact that AI tools and methodologies exert on society. AI developments carry significant implications across social, economic, legal, and environmental dimensions, including algorithmic fairness, economic shifts, and the resource demands of computational infrastructure. We strongly believe that these broader socio-technical and environmental impacts must be taken into account in future technical developments, policy formulation, and governance decision-making.

REPRODUCIBILITY STATEMENT

We are committed to ensuring the reproducibility of all theoretical claims and experimental findings presented in this paper. To facilitate full reproducibility, the entire source code of this project is open source. All empirical evaluation datasets used in this work are completely synthetic and can be directly generated, modified, and evaluated using the provided codebase. During the review period, an anonymized repository containing the full implementation, configuration files, and instructions for replicating all results and datasets is accessible at the following link: `https://dd8wgy467d-hash.github.io/reviewanonymous/`. Furthermore, complete proofs of all theoretical claims and additional experimental details are given in the Appendix.

AUTHOR CONTRIBUTIONS

This section will be added once the double-blind review period ends.

ACKNOWLEDGMENTS

This section will be added once the double-blind review period ends.

REFERENCES

Maximilian Beck, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael K. Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. xLSTM: Extended Long Short-Term Memory. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Y. Bengio and P. Frasconi. Input-output HMMs for sequence processing. *IEEE Transactions on Neural Networks*, 7(5):1231–1249, 1996. doi: 10.1109/72.536317.

Guy E. Blelloch. Prefix Sums and Their Applications. Technical Report CMU-CS-90-190, School of Computer Science, Carnegie Mellon University, 1990.

Vincent D. Blondel and John N. Tsitsiklis. The boundedness of all products of a pair of matrices is undecidable. *Systems & Control Letters*, 41(2):135–140, 2000. doi: 10.1016/S0167-6911(00)00049-9.

James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. Quasi-Recurrent Neural Networks. In *International Conference on Learning Representations*, 2017.

Julien Brandoit, Arthur Fyon, Damien Ernst, and Guillaume Drion. Improving the Performance and Learning Stability of Parallelizable RNNs Designed for Ultra-Low Power Applications. In *Forty-Third International Conference on Machine Learning*, 2026.

Mike Casey. The Dynamics of Discrete-Time Computation, with Application to Recurrent Neural Networks and Finite State Machine Extraction. *Neural Computation*, 8(6):1135–1178, 1996. doi: 10.1162/neco.1996.8.6.1135.

Mike Casey. Correction to Proof That Recurrent Neural Networks Can Robustly Recognize Only Regular Languages. *Neural Computation*, 10(5):1067–1069, 1998. doi: 10.1162/089976698300017340.

Jiwan Chung, Heechan Choi, and Seon Joo Kim. Rethinking State Tracking in Recurrent Models Through Error Control Dynamics. arXiv preprint arXiv:2605.07755, 2026.

Federico Danieli, Pau Rodriguez, Miguel Sarabia, Xavier Suau, and Luca Zappella. ParaRNN: Unlocking Parallel Training of Nonlinear RNNs for Large Language Models. In *The Fourteenth International Conference on Learning Representations*, 2026.

Tri Dao and Albert Gu. Transformers are SSMs: Generalized Models and Efficient Algorithms Through Structured State Space Duality. In *Forty-First International Conference on Machine Learning*, 2024.

Soham De, Samuel L. Smith, Anushan Fernando, Aleksandar Botev, George Cristian-Muraru, Albert Gu, Ruba Haroun, Leonard Berrada, Yutian Chen, Srivatsan Srinivasan, Guillaume Desjardins, Arnaud Doucet, David Budden, Yee Whye Teh, Razvan Pascanu, Nando De Freitas, and Çağlar Gülçehre. Griffin: Mixing Gated Linear Recurrences with Local Attention for Efficient Language Models. arXiv:2402.19427, 2024.

Florent De Geeter, Gaspard Lambrechts, Damien Ernst, and Guillaume Drion. Parallelizable memory recurrent units. arXiv preprint arXiv:2601.09495, 2026.

Grégoire Delétang, Anian Ruoss, Jordi Grau-Moya, Tim Genewein, Li Kevin Wenliang, Elliot Catt, Chris Cundy, Marcus Hutter, Shane Legg, Joel Veness, and Pedro A. Ortega. Neural Networks and the Chomsky Hierarchy. In *The Eleventh International Conference on Learning Representations*, 2023.

Samuel Eilenberg. *Automata, Languages, and Machines, Volume A*, volume 59 of *Pure and Applied Mathematics*. Academic Press, New York, 1974.

Leo Feng, Frederick Tung, Mohamed Osama Ahmed, Yoshua Bengio, and Hossein Hajimirsadeghi. Were RNNs All We Needed? arXiv preprint arXiv:2410.01201, 2024.

Riccardo Grazzi, Julien Siems, Arber Zela, Jörg Franke, Frank Hutter, and Massimiliano Pontil. Unlocking State-Tracking in Linear RNNs Through Negative Eigenvalues. In *International Conference on Learning Representations*, volume 2025, pp. 36565–36597, 2025.

Albert Gu and Tri Dao. Mamba: Linear-Time Sequence Modeling with Selective State Spaces. In *First Conference on Language Modeling*, 2024.

Albert Gu, Karan Goel, Ankit Gupta, and Christopher Ré. On the Parameterization and Initialization of Diagonal State Space Models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 35971–35983. Curran Associates, Inc., 2022a. doi: 10.52202/068431-2607.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently Modeling Long Sequences with Structured State Spaces. In *International Conference on Learning Representations*, 2022b.

Ankit Gupta, Albert Gu, and Jonathan Berant. Diagonal State Spaces are as Effective as Structured State Spaces. In *Advances in Neural Information Processing Systems*, volume 35, pp. 22982–22994. Curran Associates, Inc., 2022. doi: 10.52202/068431-1670.

J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, 1982. doi: 10.1073/pnas.79.8.2554.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical Reparameterization with Gumbel-Softmax. In *International Conference on Learning Representations*, 2017.

Raphaël Jungers. *The Joint Spectral Radius: Theory and Applications*, volume 385 of *Lecture Notes in Control and Information Sciences*. Springer, Berlin, Heidelberg, 2009. doi: 10.1007/978-3-540-95980-9.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1):99–134, 1998. doi: 10.1016/S0004-3702(98)00023-X.

T. Kaneko. Limit-cycle oscillations in floating-point digital filters. *IEEE Transactions on Audio and Electroacoustics*, 21(2):100–106, 1973. doi: 10.1109/TAU.1973.1162445.

Arjun Karuvally, Franz Nowak, Andy Keller, Carmen Amo Alonso, Terrence Sejnowski, and Hava Siegelmann. Bridging Expressivity and Scalability with Adaptive Unitary SSMs. In *Advances in Neural Information Processing Systems*, volume 38, pp. 22017–22057. Curran Associates, Inc., 2025. doi: 10.52202/085713-0743.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 5156–5165. PMLR, 2020.

Najoung Kim and Sebastian Schuster. Entity Tracking in Language Models. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3835–3855, Toronto, Canada, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.213.

Kimi Team, Yu Zhang, Zongyu Lin, Xingcheng Yao, Jiaxi Hu, Fanqing Meng, Chengyin Liu, Xin Men, Songlin Yang, Zhiyuan Li, Wentao Li, Enzhe Lu, Weizhou Liu, Yanru Chen, Weixin Xu, Longhui Yu, Yejie Wang, Yu Fan, Longguang Zhong, Enming Yuan, Dehao Zhang, Yizhi Zhang, T. Y. Liu, Haiming Wang, Shengjun Fang, Weiran He, Shaowei Liu, Yiwei Li, Jianlin Su, Jiezhong Qiu, Bo Pang, Junjie Yan, Zhejun Jiang, Weixiao Huang, Bohong Yin, Jiacheng You, Chu Wei, Zhengtao Wang, Chao Hong, Yutian Chen, Guanduo Chen, Yucheng Wang, Huabin Zheng, Feng Wang, Yibo Liu, Mengnan Dong, Zheng Zhang, Siyuan Pan, Wenhao Wu, Yuhao Wu, Longyu Guan, Jiawen Tao, Guohong Fu, Xinran Xu, Yuzhi Wang, Guokun Lai, Yuxin Wu, Xinyu Zhou, Zhilin Yang, and Yulun Du. Kimi Linear: An Expressive, Efficient Attention Architecture. arXiv:2510.26692, 2025.

Kenneth Krohn and John Rhodes. Algebraic theory of machines. I. prime decomposition theorem for finite semigroups and machines. *Transactions of the American Mathematical Society*, 116:450–464, 1965. doi: 10.1090/S0002-9947-1965-0188316-1.

Aakash Lahoti, Kevin Li, Berlin Chen, Caitlin Wang, Aviv Bick, J. Zico Kolter, Tri Dao, and Albert Gu. Mamba-3: Improved Sequence Modeling using State Space Principles. In *The Fourteenth International Conference on Learning Representations*, 2026.

Yi Heng Lim, Qi Zhu, Joshua Selfridge, and Muhammad Firmansyah Kasim. Parallelizing non-linear sequential models over the sequence length. In *The Twelfth International Conference on Learning Representations*, 2024.

Bingbin Liu, Jordan T. Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. Transformers Learn Shortcuts to Automata. In *The Eleventh International Conference on Learning Representations*, 2023a.

Bingbin Liu, Jordan T. Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. Exposing Attention Glitches with Flip-Flop Language Modeling. In *Thirty-Seventh Conference on Neural Information Processing Systems*, 2023b.

Bo Liu, Rui Wang, Lemeng Wu, Yihao Feng, Peter Stone, and Qiang Liu. Longhorn: State Space Models are Amortized Online Learners. In *The Thirteenth International Conference on Learning Representations*, 2025.

Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*, 2019.

Wolfgang Maass and Pekka Orponen. On the Effect of Analog Noise in Discrete-Time Analog Computations. *Neural Computation*, 10(5):1071–1095, 1998. doi: 10.1162/089976698300017359.

Wolfgang Maass and Eduardo D. Sontag. Analog Neural Nets with Gaussian or Other Common Noise Distributions Cannot Recognize Arbitrary Regular Languages. *Neural Computation*, 11 (3):771–782, 1999. doi: 10.1162/089976699300016656.

Eric Martin and Chris Cundy. Parallelizing Linear Recurrent Neural Nets Over Sequence Length. In *International Conference on Learning Representations*, 2018.

Robert McNaughton and Seymour Papert. *Counter-Free Automata*, volume 65 of *Research Monograph*. MIT Press, Cambridge, MA, 1971.

William Merrill, Jackson Petty, and Ashish Sabharwal. The Illusion of State in State-Space Models. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 35492–35506. PMLR, 2024.

B.M. Oliver, J.R. Pierce, and C.E. Shannon. The Philosophy of PCM. *Proceedings of the IRE*, 36 (11):1324–1331, 1948. doi: 10.1109/JRPROC.1948.231941.

Christian W. Omlin and C. Lee Giles. Constructing deterministic finite-state automata in recurrent neural networks. *Journal of the ACM (JACM)*, 43(6):937–972, 1996a. doi: 10.1145/235809.235811.

Christian W. Omlin and C. Lee Giles. Stable Encoding of Large Finite-State Automata in Recurrent Neural Networks with Sigmoid Discriminants. *Neural Computation*, 8(4):675–696, 1996b. doi: 10.1162/neco.1996.8.4.675.

Antonio Orvieto, Samuel L. Smith, Albert Gu, Anushan Fernando, Çağlar Gülçehre, Razvan Pascanu, and Soham De. Resurrecting Recurrent Neural Networks for Long Sequences. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 26670–26698. PMLR, 2023.

S. Parker and S. Hess. Limit-cycle oscillations in digital filters. *IEEE Transactions on Circuit Theory*, 18(6):687–697, 1971. doi: 10.1109/TCT.1971.1083367.

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Leon Derczynski, Xingjian Du, Matteo Grella, Kranthi Gv, Xuzheng He, Haowen Hou, Przemysław Kazienko, Jan Kocoń, Jiaming Kong, Bartłomiej Koptyra, Hayden Lau, Jiaju Lin, Krishna Sri Ipsit Mantri, Ferdinand Mom, Atsushi Saito, Guangyu Song, Xiangru Tang, Johan Wind, Stanisław Woźniak, Zhenyuan Zhang, Qinghua Zhou, Jian Zhu, and Rui-Jie Zhu. RWKV: Reinventing RNNs for the Transformer Era. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 14048–14077, Singapore, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.936.

Bo Peng, Ruichong Zhang, Daniel Goldstein, Eric Alcaide, Xingjian Du, Haowen Hou, Jiaju Lin, Jiaxing Liu, Janna Lu, William Merrill, Guangyu Song, Kaifeng Tan, Saiteja Utpala, Nathan Wilce, Johan S. Wind, Tianyi Wu, Daniel Wuttke, and Christian Zhou-Zheng. RWKV-7 "Goose" with Expressive Dynamic State Evolution. In *Second Conference on Language Modeling*, 2025.

M. Perles, M. O. Rabin, and E. Shamir. The Theory of Definite Automata. *IEEE Transactions on Electronic Computers*, EC-12(3):233–243, 1963. doi: 10.1109/PGEC.1963.263534.

Zhen Qin, Songlin Yang, and Yiran Zhong. Hierarchically Gated Recurrent Neural Network for Sequence Modeling. In *Thirty-Seventh Conference on Neural Information Processing Systems*, 2023.

Zhen Qin, Songlin Yang, Weixuan Sun, Xuyang Shen, Dong Li, Weigao Sun, and Yiran Zhong. HGRN2: Gated Linear RNNs with State Expansion. In *First Conference on Language Modeling*, 2024.

S.V. Raković, E.C. Kerrigan, K.I. Kouramas, and D.Q. Mayne. Invariant approximations of the minimal robust positively Invariant set. *IEEE Transactions on Automatic Control*, 50(3):406–410, 2005. doi: 10.1109/TAC.2005.843854.

Hubert Ramsauer, Bernhard Schäfl, Johannes Lehner, Philipp Seidl, Michael Widrich, Thomas Adler, Lukas Gruber, Markus Holzleitner, Milena Pavlović, Geir Kjetil Sandve, Victor Greiff, David Kreil, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. Hopfield Networks is All You Need. arXiv preprint arXiv:2008.02217, 2021.

T. Konstantin Rusch and Daniela Rus. Oscillatory State-Space Models. In *The Thirteenth International Conference on Learning Representations*, 2025.

Yash Sarrof, Yana Veitsman, and Michael Hahn. The Expressive Capacity of State Space Models: A Formal Language Perspective. In *Advances in Neural Information Processing Systems*, volume 37, pp. 41202–41241. Curran Associates, Inc., 2024. doi: 10.52202/079017-1304.

M. P. Schützenberger. On finite monoids having only trivial subgroups. *Information and Control*, 8 (2):190–194, 1965. doi: 10.1016/S0019-9958(65)90108-7.

Mehran Shakerinava, Behnoush Khavari, Siamak Ravanbakhsh, and Sarath Chandar. The Expressive Limits of Diagonal SSMs for State-Tracking. In *The Fourteenth International Conference on Learning Representations*, 2026.

C. E. Shannon. A Mathematical Theory of Communication. *The Bell System Technical Journal*, 27 (3):379–423, 1948. doi: 10.1002/j.1538-7305.1948.tb01338.x.

C.E. Shannon. Communication in the Presence of Noise. *Proceedings of the IRE*, 37(1):10–21, 1949. doi: 10.1109/JRPROC.1949.232969.

Julien Siems, Timur Carstensen, Arber Zela, Frank Hutter, Massimiliano Pontil, and Riccardo Grazzi. DeltaProduct: Improving State-Tracking in Linear RNNs via Householder Products. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Jimmy T. H. Smith, Andrew Warrington, and Scott Linderman. Simplified State Space Layers for Sequence Modeling. In *The Eleventh International Conference on Learning Representations*, 2023.

Aarohi Srivastava et al. Beyond the Imitation Game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*, 2023.

Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive Network: A Successor to Transformer for Large Language Models. arXiv:2307.08621, 2023.

Aleksandar Terzić, Nicolas Menet, Michael Hersche, Thomas Hofmann, and Abbas Rahimi. Structured Sparse Transition Matrices to Enable State Tracking in State-Space Models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Lloyd N. Trefethen and Mark Embree. *Spectra and Pseudospectra: The Behavior of Nonnormal Matrices and Operators*. Princeton University Press, 2005. doi: 10.1515/9780691213101.

John von Neumann. Probabilistic logics and the synthesis of reliable organisms from unreliable components. In Claude E. Shannon and John McCarthy (eds.), *Automata Studies*, number 34 in Annals of Mathematics Studies, pp. 43–98. Princeton University Press, Princeton, NJ, 1956.

S. Winograd and Jack Cowan. *Reliable Computation in the Presence of Noise*. MIT Press, Cambridge, MA, USA, 1963.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated Linear Attention Transformers with Hardware-Efficient Training. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 56501–56523. PMLR, 2024a.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing Linear Transformers with the Delta Rule over Sequence Length. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024b.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated Delta Networks: Improving Mamba2 with Delta Rule. In *The Thirteenth International Conference on Learning Representations*, 2025a.

Songlin Yang, Yikang Shen, Kaiyue Wen, Shawn Tan, Mayank Mishra, Liliang Ren, Rameswar Panda, and Yoon Kim. PaTH Attention: Position Encoding via Accumulating Householder Transformations. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025b.

Wojciech Zaremba and Ilya Sutskever. Learning to Execute. arXiv preprint arXiv:1410.4615, 2015.

Rui-Jie Zhu, Yu Zhang, Steven Abreu, Ethan Sifferman, Tyler Sheaves, Yiqiao Wang, Dustin Richmond, Sumit Bam Shrestha, Peng Zhou, and Jason K. Eshraghian. Scalable MatMul-free Language Modeling. arXiv:2406.02528, 2025.

CONTENTS OF THIS APPENDIX

- Appendix A: Notation
- Appendix B: Extended discussion
- Appendix C: Related work
- Appendix D: From finite precision to the perturbation model
- Appendix E: Learning versus finite precision
- Appendix F: Proofs of theorems and lemmas
- Appendix G: The NFSM construction in detail
- Appendix H: Experiments

A NOTATION

Table 2: Notation.

Target		*Model*	
Σ, Σ^*	alphabet; free monoid of words, with $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$.	$\mathcal{H} = \mathbb{R}^d$	hidden state space
Q, n	automaton state set; $n \coloneqq \lvert Q\rvert \geq 2$	Φ_x, Φ_w	per-symbol map; composite $\Phi_{wx} \coloneqq \Phi_x \circ \Phi_w$
δ_x, δ_w	symbol action $\delta_x \coloneqq \delta(\cdot, x) : Q \to Q$; extension to words, $\delta_w \coloneqq \delta_{x_{\lvert w\rvert}} \circ \cdots \circ \delta_{x_1}$	A_x, b_x	affine data, $\Phi_x(h) = A_x h + b_x$
M_δ	transition monoid, $M_\delta = \{\delta_w \mid w \in \Sigma^*\} \leq Q^Q$, under composition	A_w	$A_{x_{\lvert w\rvert}} \cdots A_{x_1}$ for $w = x_1 \cdots x_{\lvert w\rvert}$
Q^Q	all maps $Q \to Q$ under composition; $Q^Q \cong T_n$, $M_\delta \leq Q^Q$	ρ, κ	spectral radius; $\kappa \coloneqq \max_x \lVert A_x\rVert_{\mathrm{op}}$, with $\lVert A\rVert_{\mathrm{op}} \coloneqq \sup_{v\neq 0} \lVert Av\rVert / \lVert v\rVert$
T_n, S_n	full transformation monoid; symmetric group	M_Φ	RNN transition monoid, $M_\Phi \coloneqq \{\Phi_w \mid w \in \Sigma^*\}$
Arithmetic and error		*Code*	
$[\cdot]_{\mathcal{Q}}$	representable object standing for a given one (Appendix D)	ι, U, π	encoding; decodable region; readout
$\mathcal{Q}$	representable points $[\mathcal{H}]_{\mathcal{Q}}$	U_q	the cell $U \cap \pi^{-1}(q)$
η, $\bar{B}_\eta$	per-step perturbation bound; ball $\{e \mid \lVert e\rVert \leq \eta\}$	$U^\star$	η-reachable set of Definition 2.1
$\oplus$	Minkowski sum	diam, dist	diameter of a set; distance between two
η-traj.	any (h_t) with $\lVert h_t - \Phi_{x_t}(h_{t-1})\rVert \leq \eta$	a_k, V_k, σ_x	attractors; their basins; table on attractor indices (Sections 5 and 6)
		P_x, rd	column one-hot matrix of σ_x (Appendix G.3); restoring map to the basis vector of the largest coordinate
c_a, S	constant map of value a; transition semigroup $\{\delta_w \mid w \in \Sigma^+\}$	$\mathbf{1}_j, \theta_x$	basis vector j, attractor of index j; logit matrix of equation 6
ζ, b, c	scan representation and its budget (Definition 5.1)	r, d_i, γ	number of heads and their numbers of attractor indices (Section 6); logit margin (Appendix G.4)
$\tilde{\Phi}_x, e_t$	executed step; per-step error $\tilde{\Phi}_{x_t}(h_{t-1}) - \Phi_{x_t}(h_{t-1})$	K	number of reachable states (Proposition 5.2)
		$L_{\max}$	failing length (Section 7)
		χ	injective map of Q into the joint attractor indices of the heads (Appendix G.5)

B EXTENDED DISCUSSION

Robust state tracking asks one map to both erase the perturbation the previous step left behind, and move the state to its successor. An affine recurrence has a single rate, so it buys one at the expense of the other and its horizon is finite at any precision. NFSM does both, because a step is a rounding followed by a table lookup.

The horizon is not an artifact of precision. Our argument turns on the existence of a perturbation, not its amplitude. For any $\eta > 0$ a contracting affine recurrence has a finite horizon, and η only sets where that horizon falls; more precise formats moves it, but does not remove it. Nor is it an artifact of the hardware: the scan that makes an affine recurrence fast reassociates its products and so supplies η by itself, whatever the format. NFSM escapes the horizon not by being more precise but by returning the state to a code point at every step, so that perturbations are annihilated instead of accumulated and, under a scan, are not created at all.

An undeclared context window. For language models built on contracting recurrences the consequence is practical. The realized automaton is definite up to precision, so inputs older than k steps reach the output only through a term of size $O(\kappa^k)$: the model has a context window, but an undeclared one. Its length is set by learned gates and the arithmetic floor rather than by specification, it varies with the input as selectivity varies the decay rate, and, unlike overflow in attention, it is exceeded without any signal.

Forgetting as computation, not decay. Krohn–Rhodes decomposes every finite-state behavior into simple groups, which remember reversibly, and flip-flops, which remember until reset (Krohn & Rhodes, 1965). Realizing both makes forgetting a chosen property of the transitions rather than an unwanted residual of the dynamics: what is discarded is chosen by the arriving symbol, a reset erases exactly what it collapses, and what no reset targets survives indefinitely. A contracting recurrence offers erasure through contraction, which is graded and unconditional. Persistence is therefore not the price of forgetting; it is what remains once forgetting is expressed as computation.

Beyond gradient descent. Because the executive organ followed by the restoring organ is a table read off the logits by an $\arg\max$, the automaton a trained NFSM implements can be extracted, inspected, and compared to a target. The same object can in principle be inferred in context rather than learned by gradient descent: a table is small enough to be written from a handful of demonstrations, which makes automaton induction at inference time, rather than in the weights, a natural next question.

Limitations and outlook. Our guarantees cover finite-state tracking. Behaviors requiring unbounded memory, such as counting without a modulus or stack-like nesting, fall outside the transformation-monoid setting and need a state set that grows with the input; how to keep restoration meaningful in that regime is open. The number of states is also a design choice, and while heads and their sizes can be over-provisioned, the trade-off between capacity and optimization is not yet characterized. Finally, durable state is exactly what partially observable control requires, so NFSM is a natural candidate for reinforcement learning agents whose belief must survive long episodes, which we leave to future work.

C RELATED WORK

This appendix situates our results among five lines of work: parallel scans and the affine recurrences built on them, the expressivity of these recurrences, their length generalization and finite-precision limits, scan-compatible recurrences that are not affine, and recurrent networks as automata. Three subsections then go further: the classical roots of the restoration requirement (Appendix C.1), a survey placing scan-compatible affine architectures on the branches of Theorem 4.1 (Appendix C.2), and Memory Recurrent Units read as transition tables (Appendix C.3).

Parallel scans and affine recurrences. A parallel scan computes all prefixes of a sequence under an associative operation in logarithmic depth (Blelloch, 1990). Martin & Cundy (2018) used it to train linear recurrences over the sequence length. Linearity in the hidden state has since become the design principle of scalable recurrent models: structured and diagonal state space models (Gu et al., 2022b;a; Gupta et al., 2022; Smith et al., 2023), selective ones (Gu & Dao, 2024; Dao & Gu, 2024; Lahoti et al., 2026), gated linear RNNs (Orvieto et al., 2023; Qin et al., 2023; 2024; De et al.,

2024; Feng et al., 2024), and linear attention with its gated and delta-rule variants (Katharopoulos et al., 2020; Yang et al., 2024a;b; 2025a). They parallelize over the sequence in different ways, through a parallel scan (Smith et al., 2023; Gu & Dao, 2024; Orvieto et al., 2023; Feng et al., 2024), a convolution for the time-invariant ones (Gu et al., 2022b;a; Gupta et al., 2022), or a chunkwise algorithm (Yang et al., 2024a;b; Dao & Gu, 2024), but all of them rely on the update being affine in the hidden state, so that the composite of two steps has a closed form. Appendix C.2 surveys them from this angle. We take the opposite reading of the same constraint: what a scan requires is a bounded representation of the composites (Definition 5.1), not an affine update, and Proposition 5.2 shows what this budget implies.

Expressivity of affine recurrences. A large body of work asks which automata these recurrences can express. At logarithmic precision and under standard complexity assumptions, state space models such as S4 and Mamba cannot track nonsolvable groups such as S_5 (Merrill et al., 2024). At finite precision, state space models with nonnegative gates model exactly the star-free regular languages (Sarrof et al., 2024), and a single layer of input-dependent complex diagonal transitions tracks no nonabelian group, while k such layers track exactly the groups with a subnormal series of length k with abelian factors (Shakerinava et al., 2026). A line of architectures then enlarges the reachable transition monoids: negative eigenvalues bring parity, and products of generalized Householder matrices bring every regular language (Grazzi et al., 2025), with the number of factors trading expressivity for cost (Siems et al., 2025), and an NC^1-complete problem within attention (Yang et al., 2025b); adaptive unitary transitions (Karuvally et al., 2025) and generalized delta rules (Peng et al., 2025) target state tracking directly, and the column one-hot transitions of PD-SSM reach every finite automaton with as many hidden dimensions as states (Terzić et al., 2025). These results hold at logarithmic precision (Merrill et al., 2024), at a precision that grows logarithmically with the length (Karuvally et al., 2025), or at finite precision (Sarrof et al., 2024; Grazzi et al., 2025; Siems et al., 2025; Shakerinava et al., 2026), but in none of them is the computation perturbed. In the finite-precision model of Sarrof et al. (2024), numbers carry a fixed number of fractional bits and an unbounded number of integer bits, and arithmetic on them is exact: a gate equal to one then holds an exactly representable state forever, which is how a state space model with nonnegative gates models the flip-flop and, through the Krohn–Rhodes decomposition, every star-free language. Our model instead adds a perturbation of size up to η at every step and asks for correctness under every such perturbation (Definition 2.1). A gate equal to one then accumulates the perturbations, which is branch (i) of Theorem 4.1, and a nondefinite target such as the flip-flop FF is out of reach of every affine recurrence. The two results are consistent: theirs describe what an affine recurrence expresses when its arithmetic does not drift, ours what it retains when it does, as under a floating-point scan (Section 5 and Appendix D). A second difference is the target: they ask for the recognition of a language, we ask for the realization of the automaton at every step. Expressivity and robustness to finite precision are separate axes.

Length generalization and finite precision. Empirically, state tracking learned by affine recurrences holds up to around the training length and collapses not far beyond it (Shakerinava et al., 2026; Karuvally et al., 2025; Grazzi et al., 2025; Terzić et al., 2025), and synthetic benchmarks isolate the failure: automata and group word problems (Liu et al., 2023a; Delétang et al., 2023), flip-flops with long gaps (Liu et al., 2023b), and tracking shuffled objects (Srivastava et al., 2023). The η-trajectory of Definition 2.1 is the pseudo-orbit used by Casey (1996) to analyze recurrent networks under bounded noise, while Maass & Orponen (1998) show that arbitrarily small stochastic analog noise reduces discrete-time analog computation to the power of finite automata; and that rounding changes the dynamics of a recursion, rather than only blurring them, has long been known from the limit cycles of recursive digital filters (Parker & Hess, 1971; Kaneko, 1973); under a scan, the nonassociativity of floating-point arithmetic makes the scan itself a perturbation source (Section 5). Concurrent work reaches the same limits of affine recurrences through the dynamics of error propagation (Chung et al., 2026). Our contributions are the step-level characterization of Theorem 3.1, the two-branch classification of Theorem 4.1, its extension to any depth (Theorem F.25), and a scan-compatible recurrence that escapes it.

Scan-compatible recurrences that are not affine. Two routes lead beyond affine updates without giving up parallel training. The first keeps an arbitrary nonlinear recurrence and solves for its whole trajectory at once: DEER (Lim et al., 2024) and ParaRNN (Danieli et al., 2026) cast the sequence of updates as a system of equations and solve it by Newton iterations, each of which scans the affine linearization of the recurrence. They make nonlinear recurrences such as GRUs and LSTMs trainable

at scale, but they return the sequential trajectory only up to the tolerance of the solver, which adds to η instead of removing it, and they place no constraint on the dynamics of the recurrence itself. The second route, which we take, chooses a nonlinear recurrence whose composites have a bounded exact representation. Memory Recurrent Units (De Geeter et al., 2026; Brandoit et al., 2026) are the closest prior instance: they assume that the hidden state converges onto an attractor between two steps, and on the attractors each symbol acts as a transition table (Appendix C.3). The NFSM makes this table explicit and learns it for every element of T_d (Proposition G.1). PD-SSM (Terzić et al., 2025) shares its column one-hot structure but applies its hardmax to the input, not to the hidden state: its step stays affine in h_{t-1}, and its diagonal factor makes it contract (Appendix H.2). Restoration requires the nonlinearity to act on the hidden state (Lemma F.17).

Recurrent networks as automata. That recurrent networks can implement finite automata, and must organize their hidden states into attractors to do so robustly, is classical (Omlin & Giles, 1996a;b; Casey, 1996); Appendix C.1 discusses this lineage and what changes here. The extraction of finite-state machines from trained networks (Casey, 1996) infers the automaton after training, from the organization of the hidden state space. In the NFSM the automaton is the parameterization itself: each step commits to a table entry, so the extracted tables are exactly the ones the model executes (Appendix H.6). Finally, the algebraic view of targets as transition monoids, with groups and flip-flops as the building blocks of the Krohn–Rhodes decomposition (Krohn & Rhodes, 1965), motivates both our benchmarks and the stacking of layers (Remark G.5).

C.1 Provenance of the restoration requirement and the target

This subsection places (R1) to (R3) and the target of Definition 2.2 in the lineage of classical work, and states what changes here.

Roots. The restoration requirements have a strong connection to classical work. Separation is linked to Shannon (1948; 1949): signals must be distinguishable to be told apart at all and repeated restoration relates to Oliver et al. (1948) and von Neumann (1956): error must be erased at every stage, not once at the end. (R1) to (R3) inherit both. Likewise, the η-trajectory of Definition 2.1 is the η-pseudo-orbit of dynamical systems, used in this form for recurrent networks by Casey (1996); and that a bounded set closed under one perturbed step forces a strictly stable map is standard in set-invariance control (Raković et al., 2005).

Reliable computation from unreliable parts. von Neumann (1956) shows that reliable computation from unreliable components requires executive and restoring organs in series at every stage, described in 1952 for a setting containing no recurrent networks. Winograd & Cowan (1963) shows that replication without restoration admits no computation capacity: for a redundant automaton the probability of malfunction falls as $P \approx d\exp(-c/R)$ and so vanishes only as the redundancy grows without bound, where for a coded channel $P \approx 2^{-n(C-R)}$ vanishes with block length at any fixed rate $R < C$. Their conclusion is that such automata "do not exhibit noise-free behavior of the type that would permit computation capacities to be defined," and that what buys capacity is complexity rather than copies.

Recurrent networks as automata. Hopfield (1982) gives the canonical multistable recurrent system: an energy descent whose minima are the stored patterns, with basins that absorb perturbation, a restoring organ with the identity as index map. Omlin & Giles (1996a) exhibit the state-tracking version, attractor-based encodings of arbitrary deterministic finite automata, stable at arbitrary word length with finite weights, with the sigmoid-discriminant analysis in Omlin & Giles (1996b). Casey (1996) supplies the converse, that a recurrent network performing a finite-state computation must organize its state space into attractors mirroring the states of the minimal automaton.[1] Studying finite-state machine extraction from trained networks, with no architecture class in view, their Theorem 3.1 shows that a network robustly performing an automaton computation carries mutually disjoint closed sets with nonempty interior, one per state of the minimal DFA, on which its behavior agrees with the automaton; their Corollary 3.1 then recovers finiteness from compactness of the phase space. Maass & Orponen (1998) show that arbitrarily small analog noise reduces discrete-time analog computation to finite automata, and to the definite languages in particular together with Maass & Sontag (1999). The modern Hopfield line (Ramsauer et al., 2021) replaces the discrete

[1] The noise corollary of Casey was subsequently found technically incorrect (Casey, 1998); the attractor-structure theorem is unaffected, and Maass & Orponen (1998) give the sharper and correct statement.

update with a softmax and recovers attention, which moves the construction out of the recurrent setting and so does not bear on (R2).

What is new here. In Theorem 3.2, the partition is *characterized* rather than shown necessary, so the conditions can be checked of a candidate architecture and not only extracted from a working one. The target is *realization of the automaton* rather than recognition of a language, which is worst-case over every word and compositional. And the cells carry a *margin* η that does not shrink with t, rather than being merely disjoint. The third change is central: disjointness is topological, preserved by every injective map, and therefore cannot distinguish one architecture from another, where a margin is metric: Theorem 3.3(i) and (ii) convert it into two demands along every word, which a single rate cannot meet along repeated words (Theorem 4.1). (R2) also states restoration *deterministically*, as a set inclusion with a worst-case guarantee, where every source above states it probabilistically or asymptotically, and that is what makes it composable with a budget in Section 5.

C.2 A REVIEW OF SCAN-COMPATIBLE AFFINE RECURRENCES

Most recurrent architectures proposed for scalable sequence modeling since S4 are affine in the state, so that a parallel scan composes them with the closed-form merge $(A_2, b_2) \circ (A_1, b_1) = (A_2A_1, A_2b_1 + b_2)$, up to rounding (Section 5). Existing comparisons organize them by family, by transition structure or by exact-arithmetic expressivity (Merrill et al., 2024; Sarrof et al., 2024; Grazzi et al., 2025); Table 3 places them instead on the branches of Theorem 4.1.

Method and reading. For every architecture, we read off the transition matrix A_t of equation 5 from the parameterization stated in its paper and compute its spectrum. A gate or eigenvalue modulus that *attains* 1 places the architecture on the $\rho = 1$ branch, where (R2) fails; one that only approaches 1 places it on $\rho < 1$, where (R3) fails and the definite automata remain. The survey records what each paper specifies, not what a trained model does, and is not exhaustive.

Caveats. A spectral radius is asymptotic: a nonnormal A with $\rho(A) \leq 1$ can amplify by a polynomial factor before it decays (Trefethen & Embree, 2005). And when the matrix is chosen by the input, long products are governed by the joint spectral radius of the family (Jungers, 2009), whose boundedness is undecidable in general (Blondel & Tsitsiklis, 2000); this is why Theorem 4.1 quantifies over words. Both caveats are vacuous for normal transitions (diagonal, scalar, generalized Householder, and factor by factor for DeltaProduct), for which $\|A_w\| \leq \prod_i \rho(A_{x_i})$.

Table 3: Surveyed scan-compatible affine architectures, grouped by the branch of Theorem 4.1. For each, the recurrence as its paper writes it, the transition matrix $A_t = A_{x_t}$ of equation 5, and its spectrum; keys k_t and w_t have unit norm.

Architecture	Recurrence	Transition A_t	Spectrum
$\rho > 1$**: loses contraction, fails (R2); realizes nothing.** *Not targeted in our survey*[a]			
$\rho = 1$**: loses contraction, fails (R2); realizes nothing,** $L_{\max} = O(1/\eta)$ **for isometries**[b]			
linear attention (Katharopoulos et al., 2020)	$S_t = S_{t-1} + \phi(k_t)v_t^\top$	I	$\{1\}$, multiplicity d
DeltaNet (Yang et al., 2024b)	$S_t = S_{t-1}(I - \beta_t k_t k_t^\top) + \beta_t v_t k_t^\top$	$I - \beta_t k_t k_t^\top$, $\beta_t = \sigma(\cdot) \in (0,1)$	$\{1^{(d-1)}, 1-\beta_t\}$, $1-\beta_t \in (0,1)$
+ negative eigenvalues (Grazzi et al., 2025)	$H_t = (I - 2\beta_t k_t k_t^\top)H_{t-1} + \beta_t k_t v_t^\top$	$I - 2\beta_t k_t k_t^\top$, $\beta_t \in (0,1)$	$\{1^{(d-1)}, 1-2\beta_t\}$, $1-2\beta_t \in (-1,1)$
DeltaProduct (Siems et al., 2025)	$H_t = A_t H_{t-1} + B_t$	$\prod_{j=1}^{n_h}(I - \beta_{t,j}k_{t,j}k_{t,j}^\top)$, $\beta_{t,j} \in (0,1)$ or $(0,2)$	1 with multiplicity $\geq d - n_h$
PaTH (Yang et al., 2025b)	$S_t = S_{t-1}H_t + v_t k_t^\top$	$H_t = I - \beta_t w_t w_t^\top$, $\beta_t \in (0,2)$	$\{1^{(d-1)}, 1-\beta_t\}$, $1-\beta_t \in (-1,1)$
AUSSM (Karuvally et al., 2025)	$x_t = e^{\Delta_t A_t}x_{t-1} + \Delta_t B u_t$	$e^{\Delta_t A_t}$, A_t skew-symmetric, input-dependent	$\lvert\lambda\rvert = 1$
LinOSS-IMEX (Rusch & Rus, 2025)	$x_n = Mx_{n-1} + F_n$	$M = \left(\begin{smallmatrix} I & -\Delta t A \\ \Delta t I & I - \Delta t^2 A \end{smallmatrix}\right)$, $A \geq 0$ diagonal	$\lvert\lambda\rvert = 1$ if $\Delta t^2 A_{kk} \leq 4$
LinOSS-IM[g] (Rusch & Rus, 2025)	$x_n = M^{-1}x_{n-1} + M^{-1}F_n$	$M^{-1} = \left(\begin{smallmatrix} S & -\Delta t S \\ \Delta t S & S \end{smallmatrix}\right)$, $S = (I + \Delta t^2 A)^{-1}$, $A = \mathrm{ReLU}(\hat{A})$ diagonal	$\lvert\lambda\rvert^2 = S_{kk} \in (0,1]$

Table 3, continued

Architecture	Recurrence	Transition A_t	Spectrum
S4[f] (Gu et al., 2022b)	$x_k = \bar{A}x_{k-1} + \bar{B}u_k$	$\bar{A} = (I - \frac{\Delta}{2}A)^{-1}(I + \frac{\Delta}{2}A)$, HiPPO-initialized A	$\lvert\bar{\lambda}\rvert < 1$ iff $\operatorname{Re}\lambda < 0$
S5[f] (Smith et al., 2023)	$x_k = \bar{\Lambda}x_{k-1} + \bar{B}u_k$	$\bar{\Lambda} = e^{\Lambda\Delta}$, Λ complex diagonal	$\lvert\bar{\lambda}\rvert = e^{\Delta \operatorname{Re}\lambda} < 1$ iff $\operatorname{Re}\lambda < 0$
Longhorn[h] (Liu et al., 2025)	$S_t = A_t \odot S_{t-1} + B_t$	$A_t = 1 - \varepsilon_t \otimes k_t^{\odot 2}$, $\varepsilon_t = \beta_t/(1 + \beta_t k_t^{\mathsf{T}} k_t)$, $k_t = W_k x_t$ not normalized	$\lambda \in (0, 1]$
Kimi Delta Attention (Kimi Team et al., 2025)	$S_t = (I - \beta_t k_t k_t^{\mathsf{T}})\mathrm{Diag}(\alpha_t)S_{t-1} + \beta_t k_t v_t^{\mathsf{T}}$	$(I - \beta_t k_t k_t^{\mathsf{T}})\mathrm{Diag}(\alpha_t)$, $\alpha_{t,j}, \beta_t \in [0, 1]$	$\rho \le \max_j \alpha_{t,j}$
RWKV-4 (Peng et al., 2023)	$a_t = e^{-w} \odot a_{t-1} + e^{k_t} \odot v_t$	$\mathrm{diag}(e^{-w})$, $w \in \mathbb{R}^d_{\ge 0}$	$\lambda \in (0, 1]$
mLSTM, exp. gate[i] (Beck et al., 2024)	$C_t = f_t C_{t-1} + i_t v_t k_t^{\mathsf{T}}$	$f_t I$, $f_t = \exp(\tilde{f}_t)$	$\lambda = f_t \in (0, \infty)$
$\rho < 1$**: loses separation, fails (R3); realizes the definite automata,** $L_{\max} = O(\log \frac{1}{\eta})$[j]			
S4D (Gu et al., 2022a)	$x_k = \bar{A}x_{k-1} + \bar{B}u_k$	$\exp(\Delta A)$, $A = -\exp(A_{\mathrm{Re}}) + \mathrm{i}A_{\mathrm{Im}}$	$\lvert\bar{\lambda}\rvert = e^{-\Delta e^{A_{\mathrm{Re}}}} < 1$
DSS (DSS-exp) (Gupta et al., 2022)	$x_k = \bar{A}x_{k-1} + \bar{B}u_k$	$\exp(\Delta\Lambda)$, $\Lambda = -\exp(\Lambda_{\mathrm{re}}) + \mathrm{i}\Lambda_{\mathrm{im}}$	$\lvert\bar{\lambda}\rvert = e^{-\Delta e^{\Lambda_{\mathrm{re}}}} < 1$
Mamba (Gu & Dao, 2024)	$h_t = \bar{A}_t h_{t-1} + \bar{B}_t x_t$	$\exp(\Delta_t A)$, A real diagonal, negative	$\lambda \in (0, 1)$
Mamba$^-$ (Grazzi et al., 2025)	$h_t = \mathrm{Diag}(2s_t - 1)h_{t-1} + \bar{B}_t x_t$	$\mathrm{Diag}(2s_t - 1)$, $s_t = \exp(-\Delta_t \odot e^w)$	$\lambda \in (-1, 1)$
Mamba-2 / SSD (Dao & Gu, 2024)	$h_t = a_t h_{t-1} + b_t$	$a_t I$, a_t scalar per head	$\lambda = a_t \in (0, 1)$
Mamba-3 (Lahoti et al., 2026)	$h_t = e^{\Delta_t A_t} R_t h_{t-1} + \Delta_t B_t x_t$	$e^{\Delta_t A_t} R_t$, $A_t < 0$ scalar, R_t block-diagonal 2×2 rotations	$\lvert\lambda\rvert = e^{\Delta_t A_t} \in (0, 1)$
LRU (Orvieto et al., 2023)	$x_k = \Lambda x_{k-1} + \gamma \odot B u_k$	$\Lambda = \mathrm{diag}(\lambda_j)$, $\lambda_j = e^{-e^{\nu_j} + \mathrm{i}e^{\theta_j}}$	$\lvert\lambda_j\rvert = e^{-e^{\nu_j}} < 1$
gated linear attention (Yang et al., 2024a)	$S_t = G_t \odot S_{t-1} + k_t^{\mathsf{T}} v_t$	$\mathrm{diag}(\alpha_t)$, $\alpha_t = \sigma(\cdot)^{1/\tau}$	$\lambda \in (0, 1)$
gated DeltaNet[c] (Yang et al., 2025a)	$S_t = S_{t-1}\alpha_t(I - \beta_t k_t k_t^{\mathsf{T}}) + \beta_t v_t k_t^{\mathsf{T}}$	$\alpha_t(I - \beta_t k_t k_t^{\mathsf{T}})$, $\alpha_t, \beta_t \in (0, 1)$	$\alpha_t\{1^{(d-1)}, 1 - \beta_t\}$, $\rho = \alpha_t$
RetNet (Sun et al., 2023)	$s_n = A s_{n-1} + K_n^{\mathsf{T}} v_n$	$A = \Lambda(\gamma e^{\mathrm{i}\theta})\Lambda^{-1}$, $\gamma = 1 - 2^{-5-\mathrm{arange}(0,h)}$	$\lvert\lambda\rvert = \gamma < 1$
RWKV-7 (Peng et al., 2025)	$S_t = S_{t-1}G_t + v_t^{\mathsf{T}} k_t$	$G_t = \mathrm{diag}(w_t) - \hat{\kappa}_t^{\mathsf{T}}(a_t \odot \hat{\kappa}_t)$, $w_t \in (e^{-e^{-1/2}}, 1)$, $a_t \in (0, 1)$	$\lambda \in (-1, 1)$ (their Theorem 1)
HGRN / HGRN2 (Qin et al., 2023; 2024)	$h_t = \lambda_t \odot e^{\mathrm{i}\theta} \odot h_{t-1} + (1 - \lambda_t) \odot c_t$	$\mathrm{diag}(\lambda_t e^{\mathrm{i}\theta})$, $\lambda_t = \gamma_k + (1 - \gamma_k)\mu_t$	$\lvert\lambda\rvert \in (\gamma_k, 1)$
Hawk / Griffin RG-LRU (De et al., 2024)	$h_t = a_t \odot h_{t-1} + \sqrt{1 - a_t^2} \odot (i_t \odot x_t)$	$\mathrm{diag}(a_t)$, $a_t = a^{cr_t}$, $a = \sigma(\Lambda)$	$\lambda \in (0, 1)$
QRNN (Bradbury et al., 2017)	$h_t = f_t \odot h_{t-1} + (1 - f_t) \odot z_t$	$\mathrm{diag}(f_t)$, $f_t = \sigma(\cdot)$	$\lambda \in (0, 1)$
GILR[d] (Martin & Cundy, 2018)	$h_t = g_t \odot h_{t-1} + (1 - g_t) \odot i_t$	$\mathrm{diag}(g_t)$, $g_t = \sigma(U x_t + b_g)$	$\lambda \in (0, 1)$
minGRU / minLSTM[d] (Feng et al., 2024)	$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$	$\mathrm{diag}(1 - z_t)$ resp. $\mathrm{diag}(f'_t)$, gates read x_t only	$\lambda \in (0, 1)$
MLGRU[d] (Zhu et al., 2025)	$h_t = f_t \odot h_{t-1} + (1 - f_t) \odot c_t$	$\mathrm{diag}(f_t)$, $f_t = \sigma(\cdot)$, ternary weights	$\lambda \in (0, 1)$
PD-SSM[e] (Terzić et al., 2025)	$x_t = P(u_t)D(u_t)x_{t-1} + Bu_t$	P column one-hot, D complex diagonal, $\lvert D\rvert = \sigma(\cdot)$	$\rho \le \max_j \lvert d_j\rvert < 1$

[a] No surveyed architecture targets this branch; unconstrained ones (notes f and i) can enter it. [b] Upper bound of Proposition F.15, proven for transitions that are isometries (unitary, orthogonal, identity). For the other rows of the block, only the failure of (R2) is proven; a nonnormal transition with $\rho = 1$ can amplify perturbations polynomially and shorten the horizon further, e.g. to $O(\eta^{-1/2})$ for a 2×2 Jordan block. $\{\rho = 1\}$ has empty interior. [c] Moves to the $\rho = 1$ branch if α_t attains 1. [d] Gates read the input only, which makes them scannable. [e] P alone has $\rho = 1$; the factor D makes the product contract. [f] The paper does not constrain $\operatorname{Re}\lambda$: the initialization is stable, but training can reach $\rho \ge 1$. [g] $\lvert\lambda\rvert = 1$ where $A_{kk} = 0$, which ReLU allows. [h] $\lambda = 1$ where $k_{t,j} = 0$; the exact update $I - \varepsilon_t k_t k_t^{\mathsf{T}}$ it approximates is on the $\rho = 1$ branch. [i] The exponential forget gate is unconstrained, like S4 and S5 (note f): it attains 1 and can exceed it, which enters the $\rho > 1$ branch. With the sigmoid forget gate, $f_t \in (0, 1)$ and mLSTM belongs to the $\rho < 1$ branch. [j] Upper bound of Corollary F.14, under the stronger hypothesis $\max_x \lVert A_x\rVert_{\mathrm{op}} < 1$; the definite automata are attained by Proposition F.13.

C.3 Memory Recurrent Units as transition tables

Memory Recurrent Units (De Geeter et al., 2026) start from a multistable recurrence and assume that it converges between two time steps, so that the state only ever sits on an attractor. Read on those attractors, each input symbol acts as a transition table, and the parallel scan composes tables exactly.

The BMRU. With state dimension d and all operations coordinatewise, the Bistable Memory Recurrent Unit (BMRU) of De Geeter et al. (2026) computes from the input

$$\hat{h}_t = W_x x_t + b_x, \qquad \beta_t = |W_\beta x_t + b_\beta|, \qquad z_t = \mathrm{H}\big(|\hat{h}_t| - \beta_t\big), \tag{7}$$

with H the Heaviside step, and updates

$$h_t = z_t \odot \operatorname{sign}(\hat{h}_t) \odot \alpha + (1 - z_t) \odot h_{t-1}, \tag{8}$$

with a learnable $\alpha \in \mathbb{R}^d_{>0}$.

As a transition table. On the attractors $\{-\alpha_i, +\alpha_i\}$ of coordinate i, a symbol acts in one of three ways: it sets the coordinate to $+\alpha_i$ or to $-\alpha_i$ when $z_i = 1$, and keeps it when $z_i = 0$. These are the three elements of the flip-flop monoid FF of Section 7, so the transition monoid of a BMRU layer on its attractors is a submonoid of FF^d, one flip-flop per coordinate. Each table costs 2 bits per coordinate, and composition is exact: a later set overrides, a later keep returns the earlier element. The BMRU is therefore scan-representable in the sense of Definition 5.1 with $b = 2d$ and $c = O(d)$, and the scanned trajectory is the sequential one.

The update equation is affine in the state, with $A_x = \operatorname{diag}(1 - z)$, and a keep symbol gives $A_x = I$: read as an update on $\mathbb{R}^d$, it lies on the $\rho = 1$ branch of Theorem 4.1 and does not restore. Its length independence comes from the state being held on the attractors, that is, stored as an index, which is the restoration the approximation assumed and which the implementation must then enforce. The tables act coordinate by coordinate and never read the state: no symbol exchanges $+\alpha_i$ and $-\alpha_i$, so a single layer realizes no nontrivial group, not even $\mathbb{Z}_2$.

The CMRU. Brandoit et al. (2026) add a term $\omega\, h_{t-1}$, with $\omega \in [0, 1]$, to the set branch of equation 8. For $0 < \omega < 1$ the reachable states form an infinite set, and at $\omega = 1$ coordinate i moves on the unbounded lattice $\alpha_i\mathbb{Z}$, so (T2) fails; confining each coordinate to $\{-k\alpha_i, \ldots, +k\alpha_i\}$ with a barrier at each end makes every symbol a table on these $n = 2k + 1$ states again, scan-representable with $b = d\, n\lceil \log_2 n \rceil$ and $c = O(dn)$.

D From finite precision to the perturbation model

Definition 2.1 replaces finite-precision arithmetic by an arbitrary perturbation of size at most η at every step. This appendix explains why this is a correct model: η is set by the number format, every run of a real machine is an η-trajectory, and asking for correctness under every such perturbation is what makes a guarantee robust to the hardware and to the implementation.

Where η comes from. A machine stores states, parameters, and intermediate results on a fixed number of bits. It can therefore represent only a finite set $\mathcal{Q} \subset \mathcal{H}$ of points; write $[\,\cdot\,]_{\mathcal{Q}}$ for the rounding that replaces an object by a representable one, e.g. the nearest point of $\mathcal{Q}$ under round-to-nearest. Since a finite set cannot contain every exact result, rounding moves points in general. It need not move the points that a particular model visits: an affine recurrence whose coefficients and codes are 0 and 1, such as a permutation matrix acting on one-hot vectors, computes every product and sum exactly, in any order, and its rounding error is zero. Such exact structures are the exception for learned models. Gates produced by sigmoids and exponentials of trained weights are generically not exactly representable, the moduli of PD-SSM lie in $(0, 1)$ and those of Mamba$^-$ in $(-1, 1)$, and the rotation of Appendix E is not representable in any format. We model this generic case by a per-step bound $\eta > 0$.

What a step injects. The machine does not run equation 1 but its rounded version

$$h_t = \tilde{\Phi}_{x_t}(h_{t-1}) \coloneqq \Big[\, [\Phi]_{\mathcal{Q}}\big(h_{t-1}, [x_t]_{\mathcal{Q}}\big) \Big]_{\mathcal{Q}}, \tag{9}$$

where $[x_t]_{\mathcal{Q}}$ is the rounded input embedding of the symbol x_t, $[\Phi]_{\mathcal{Q}}$ is the update with rounded parameters, evaluated in finite precision, and the outer bracket rounds the result. The per-step perturbation

$$e_t \coloneqq \tilde{\Phi}_{x_t}(h_{t-1}) - \Phi_{x_t}(h_{t-1})$$

is the error made by this step alone. It has three sources: the rounding of the parameters and of the input, the rounding of the intermediate operations, and the rounding of the output. It does not include the error already carried by h_{t-1}, since both maps are evaluated at the same h_{t-1}. Each step therefore adds a fresh, bounded error. Whatever the order of the operations, the machine stores $h_t \in \mathcal{Q}$, and we write $e_t \coloneqq h_t - \Phi_{x_t}(h_{t-1})$ for the error of step t.

Lemma D.1 (Every machine run is an η-trajectory)**.** *Assume that Σ is finite, that $\iota(Q) \subseteq \mathcal{Q}$, and that no overflow occurs. Then $\eta_{\mathcal{Q}} \coloneqq \max \|e_t\|$, over all steps, words, and initial states, exists and does not depend on t, on $|w|$, or on the input, and every run of the machine is an η-trajectory for every $\eta \geq \eta_{\mathcal{Q}}$. Hence, for every $\eta > 0$ with $\eta \geq \eta_{\mathcal{Q}}$, a guarantee over all η-trajectories, (T1) in particular, holds for every run.*

Proof. Since $h_{t-1}, h_t \in \mathcal{Q}$ and $x_t \in \Sigma$, the error e_t lies in the finite set $\{q - \Phi_x(q') \mid q, q' \in \mathcal{Q},\, x \in \Sigma\}$, so the maximum exists. A sequence whose steps deviate by at most $\eta_{\mathcal{Q}}$ deviates by at most any $\eta \geq \eta_{\mathcal{Q}}$. □

The bound $\eta_{\mathcal{Q}}$ can be zero, as in the exact example above; the results of this paper then apply at every $\eta > 0$, and in particular a guarantee still holds for the actual runs. In floating point, rounding errors are relative, so $\eta_{\mathcal{Q}}$ scales with the size of the states visited.

How small η is depends on the format and on the evaluation: for the sequential evaluation equation 9, e_t is the rounding error of a single step, and a different order of the operations, such as a parallel scan, changes e_t but not the conclusion.

Why every perturbation, and not the actual rounding. The rounding errors of a run are not a property of the model. They depend on the hardware, through the number format, and on the implementation, through the order of the operations: floating-point addition and multiplication are not associative, so two runs of the same model on the same input, one sequential and one with a parallel scan, are not bit-identical. A particular rounding pattern can even help: rounding turns the recurrence into a map on the finite set $\mathcal{Q}$, which can hold a few states at the scale of the rounding error, as the dead bands and limit cycles of recursive digital filters do, in fixed point and in floating point (Parker & Hess, 1971; Kaneko, 1973). Such behavior is not robust, since it can disappear when the number format or the order of the operations changes. We therefore ask for correctness under every perturbation of size at most η, which is what quantifying over all η-trajectories does (Definition 2.2). This makes the guarantee a worst-case one, and the impossibility results should be read accordingly: Theorem 4.1 states that no affine recurrence is *guaranteed* correct at every length under perturbations of size η, not that every run of every affine model fails. A model whose rounding happens to be harmless, or zero, can still be correct on its actual runs, but that correctness is not robust: it can be lost when the number format, the order of the operations or the parameters change (Appendix E).

Sequential versus parallel evaluation. The order of the operations alone is enough to change the trajectory. We evaluate each affine baseline of Section 4 twice on the same word: once with the sequential recurrence $h_t = A_{x_t} h_{t-1} + b_{x_t}$, which rounds the products and sums from left to right, and once with the parallel scan used for training, which combines the pairs (A_{x_t}, b_{x_t}) along a tree of depth $\log_2 |w|$ and therefore rounds the same operations in a different order. The two runs share the parameters, the input and the coefficients mapping (A_{x_t}, b_{x_t}), so they differ only in this order. By Lemma D.1, both are η-trajectories of the same word from the same h_0, and the gap $\|h_t^{\mathrm{par}} - h_t^{\mathrm{seq}}\|_\infty$ shows what the recurrence does with the errors injected along the way (Figure 2).

Three observations follow. First, the gap is nonzero from the first steps, in every model and format. Second, its evolution follows the spectral radius, as Theorem 4.1 predicts. With $\rho < 1$ (Mamba$^-$, PD-SSM), earlier errors are forgotten as fast as new ones arrive, and the gap saturates. With $\rho = 1$ (AUSSM), every error is carried forward, and the gap grows until, in fp16, it reaches the size of the state itself. Third, the number format sets the level of the curves and the recurrence sets their shape: a coarser format shifts every curve up. A guarantee that relied on the rounding pattern of one evaluation would not transfer to the other; a guarantee over all η-trajectories covers both.

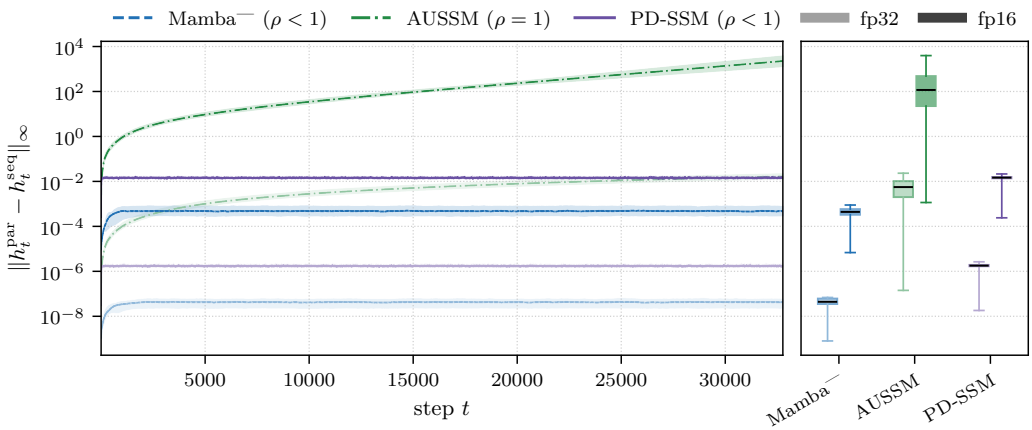

Figure 2: **Sequential and parallel evaluations of the same model may follow different trajectories.** Gap $\|h_t^{\text{par}} - h_t^{\text{seq}}\|_\infty$ between the parallel scan and the sequential recurrence, with the same coefficients (A_{x_t}, b_{x_t}), computed in fp32 and rounded to the format (light: fp32; dark: fp16; complex states stored as two real arrays). Untrained models of width 32, 64 words of length 2^{15} over five letters, 5 seeds. **Left:** gap at each step (mean over words and seeds; band: range of the per-seed means). **Right:** distribution over all steps and seeds (whiskers: minimum and maximum). Absolute values are not comparable across models; the trend in t is.

E Learning versus finite precision

In practice, the main obstacle to length generalization is learning: a model must find, by gradient descent on short sequences, parameters that implement the target transitions. Our results do not address this obstacle directly. They address a different one, which remains even when learning succeeds.

Three requirements. Tracking at every length needs a target that is *expressible* in exact arithmetic (Appendix C), a realization that is *robust* to a perturbation at every step (Theorems 3.1 to 4.1), and parameters that are *found* by training.

A perfectly trained model still drifts. Figure 3 isolates the robustness failure by removing learning altogether. The target is $\mathbb{Z}_7$ under the constant word, and the recurrence is the $\rho = 1$ affine realization that a perfectly trained model would hold: the rotation of $\mathbb{R}^2$ by $2\pi/7$, whose seven code points are the vertices of a regular heptagon. We round each entry of this rotation once to each number format, and then run the recurrence in fp64, so that no rounding occurs after the weights are set. Rounding the cosine and the sine separately leaves a rotation by a slightly wrong angle, scaled by a radius slightly different from one. The phase error therefore grows linearly with t, and the model decodes the wrong state as soon as it exceeds $\pi/7$, half the angle between two code points. The radius grows or decays geometrically, so the code points themselves leave the bounded region that restoration would require.

First, the drift involves no training error: the weights are the exact solution, up to the resolution of the format. Second, it involves no rounding of the hidden state: the run itself is in fp64, and rounding at every step, or reassociating the products under a scan, only adds to it (Figure 2). Third, a finer format moves the failure but does not remove it. A better choice of representable weights can move the failing length, but no representable rotation by $2\pi/7$ exists in any format.

F Proofs of theorems and lemmas

This appendix collects the proofs deferred from the main text, together with the additional definitions, lemmas, and propositions.

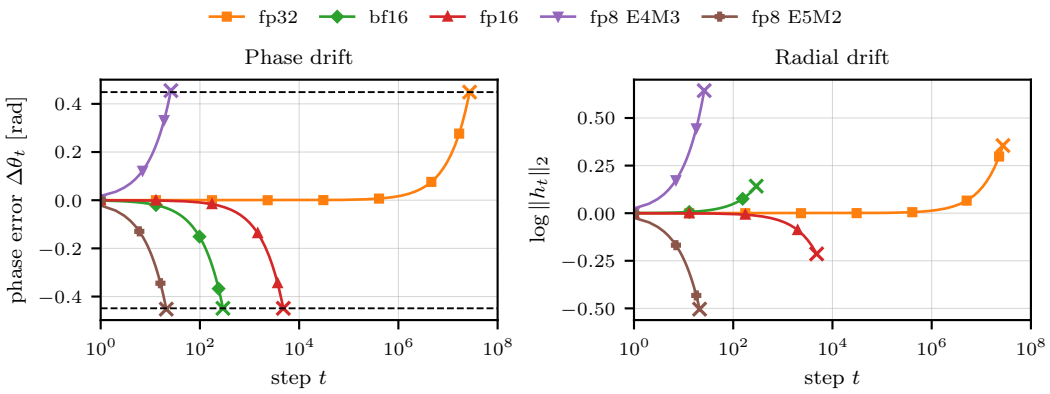

Figure 3: **A perfectly trained $\rho = 1$ recurrence still drifts at finite precision.** The exact rotation by $2\pi/7$, which realizes $\mathbb{Z}_7$ under the constant word, is rounded once to each format and then iterated in fp64 from $h_0 = (1, 0)$. **Left:** phase error $\Delta\theta_t$ with respect to the exact run; dashed lines: $\pm\pi/7$, beyond which the readout returns a neighbouring state. **Right:** log-norm of the hidden state. Crosses mark the first step at which the readout is wrong; every format fails at a finite length, fp32 included.

F.1 Length-independent realization

Condition (T2) links boundedness to a length-independent precision cost, because a machine stores its state on a fixed finite set of points (Appendix D).

Proposition F.1 (Boundedness is length-independent precision)**.** *A single finite set of points represents every state of $U^\star$ to within η if and only if $U^\star$ is bounded. In that case, $\lceil d \log_2(1 + 2\mathrm{diam}(U^\star)/\eta) \rceil$ bits suffice, at every step and every length.*

Proof. If a finite set P represents $U^\star$ to within η, then $U^\star \subseteq P \oplus \bar{B}_\eta$, which is bounded. Conversely, let $U^\star$ be bounded, and let $P \subseteq U^\star$ be a maximal set of points pairwise more than η apart. By maximality, every point of $U^\star$ lies within η of P. The balls of radius $\eta/2$ centered at the points of P are disjoint and lie in a ball of radius $\mathrm{diam}(U^\star) + \eta/2$, so comparing volumes gives $|P| \leq (1 + 2\mathrm{diam}(U^\star)/\eta)^d$ in any norm, and indexing P takes $\lceil \log_2 |P| \rceil$ bits. Neither $U^\star$ nor η depends on t or on $|w|$, so the same set serves at every step and every length. □

For Theorem 3.1, necessity is proved with $U = U^\star$, by applying (T1) and (T2) to a word and to its extension by one symbol (Lemmas F.2 to F.4); sufficiency is an induction on the length of the word.

Lemma F.2 (The encoding must be a code)**.** *Under (T1), $\iota(Q) \subseteq U^\star$ and $\pi \circ \iota = \mathrm{id}_Q$; in particular ι is injective.*

Proof. Take $w = \varepsilon$. The one-point sequence $h_0 = \iota(q)$ is an η-trajectory of ε from $\iota(q)$, so $\iota(q) \in U^\star$ and (T1) gives $\pi(\iota(q)) = \delta_\varepsilon(q) = q$. □

Lemma F.3 (Restoration on the reachable set)**.** (i) *For every Φ and every encoding, $\Phi_x(U^\star) \oplus \bar{B}_\eta \subseteq U^\star$ for every $x \in \Sigma$.* (ii) *Hence, under (T2), $U^\star$ satisfies (R2).*

Proof. (i) Let $u \in U^\star$, $x \in \Sigma$ and $\|e\| \leq \eta$. Choose q, w and an η-trajectory $(h_t)_{t=0}^{|w|}$ of w from $\iota(q)$ with $h_{|w|} = u$, and append $h_{|w|+1} := \Phi_x(u) + e$. Since $\|h_{|w|+1} - \Phi_x(h_{|w|})\| = \|e\| \leq \eta$, the extended sequence is an η-trajectory of wx from $\iota(q)$, so its endpoint lies in $U^\star$. As u, x and e were arbitrary, $\Phi_x(U^\star) \oplus \bar{B}_\eta \subseteq U^\star$ for every x. The inclusion holds for any recurrence. (ii) (R2) asks for a bounded region that absorbs one perturbation step: (i) gives the inclusion, and (T2) gives the boundedness. □

Lemma F.4 (The step must implement the transition)**.** *Under (T1), $\pi(\Phi_x(u) + e) = \delta_x(\pi(u))$ for every $x \in \Sigma$, every $u \in U^\star$ and every $\|e\| \leq \eta$.*

Proof. Let $u \in U^\star$, $x \in \Sigma$, $\|e\| \le \eta$, and take q, w and an η-trajectory of w from $\iota(q)$ with endpoint u, as in Lemma F.3. (T1) applied to w gives $\pi(u) = \delta_w(q)$; the extended trajectory of wx has endpoint $\Phi_x(u)+e$, so (T1) applied to wx gives $\pi(\Phi_x(u)+e) = \delta_{wx}(q) = \delta_x(\delta_w(q)) = \delta_x(\pi(u))$, using $\delta_{wx}(p) = \delta_x(\delta_w(p))$ for every p, which is immediate from the definition of δ_w. □

Theorem 3.1 (Conditions for length-independent realization at finite precision)**.** *A triple (ι, Φ, π) realizes the target automaton (Q, Σ, δ) with length independence at $\eta > 0$* iff *there is a region $U \supseteq \iota(Q)$ on which π is defined and that satisfies*

(R1) Decodable code. $\pi \circ \iota = \mathrm{id}_Q$.
(R2) Restoration. *U is bounded and $\Phi_x(U) \oplus \bar{B}_\eta \subseteq U$ for every $x \in \Sigma$.*
(R3) Correct transition. *$\pi(\Phi_x(u) + e) = \delta_x(\pi(u))$ for every $x \in \Sigma$, $u \in U$ and $\|e\| \le \eta$.*

Whenever such a region U exists, the η-reachable set $U^\star$ itself satisfies (R1)–(R3), and $U^\star \subseteq U$.

Proof of Theorem 3.1. Sufficiency. We prove the stronger statement that correctness holds from *every* $u \in U$ and not only from code points; (T1) is the case $u = \iota(q)$. Induction on $k := |w|$. For $k = 0$ the trajectory is the single point $h_0 = u \in U$ and $\pi(u) = \delta_\varepsilon(\pi(u))$. For the step, let $w \in \Sigma^k$ and $x \in \Sigma$. Any η-trajectory $(h_t)_{t=0}^{k+1}$ of wx from u truncates to an η-trajectory of w from u, so the hypothesis gives $h_k \in U$ and $\pi(h_k) = \delta_w(\pi(u))$. Write $h_{k+1} = \Phi_x(h_k) + e$ with $\|e\| \le \eta$. Then $h_{k+1} \in \Phi_x(U) \oplus \bar{B}_\eta \subseteq U$ by (R2), so $\pi(h_{k+1})$ is defined, and $\pi(h_{k+1}) = \delta_x(\pi(h_k))$ by (R3) applied at $h_k \in U$. Since $\pi(h_k) = \delta_w(\pi(u))$ and $\delta_{wx}(p) = \delta_x(\delta_w(p))$ for every p, $\pi(h_{k+1}) = \delta_{wx}(\pi(u))$. Taking $u = \iota(q)$ and using (R1) gives (T1); and the same induction keeps every η-trajectory from $\iota(Q)$ inside U, so $U^\star \subseteq U$, which is bounded by (R2) and gives (T2). *Necessity, and minimality.* Lemmas F.2, F.3 and F.4 give (R1), (R2) and (R3) on $U^\star$, which is bounded by (T2), so $U = U^\star$ carries the triple. Minimality is the containment just established: any U carrying a triple satisfies $U^\star \subseteq U$, since every η-trajectory from $\iota(Q)$ stays in U. □

Theorem 3.2 (Cell partition)**.** *A triple (ι, Φ, π) realizes the target automaton (Q, Σ, δ) with length independence at $\eta > 0$* iff *there exists a bounded region $U \supseteq \iota(Q)$ partitioned into cells $U_q := \{u \in U \mid \pi(u) = q\}$ such that $\iota(q) \in U_q$ for all $q \in Q$, and*

$$\Phi_x(U_q) \oplus \bar{B}_\eta \subseteq U_{\delta_x(q)} \qquad \text{for every } q \in Q \text{ and } x \in \Sigma. \tag{4}$$

Proof of Theorem 3.2. ($\Rightarrow$) Suppose (ι, Φ, π) realizes M_δ with length independence at η, and take the region U of Theorem 3.1, bounded by (R2). Put $U_q := U \cap \pi^{-1}(q)$; since π is a function, every $u \in U$ lies in exactly one U_q, namely $q = \pi(u)$, so $\{U_q\}_{q \in Q}$ partitions U, and $\iota(q) \in U_q$ by (R1) together with $U \supseteq \iota(Q)$. Let $u \in U_q$, $x \in \Sigma$ and $\|e\| \le \eta$. Since $u \in U$, (R2) gives $\Phi_x(u) + e \in \Phi_x(U) \oplus \bar{B}_\eta \subseteq U$; and $\pi(u) = q$, so (R3) gives $\pi(\Phi_x(u) + e) = \delta_x(q)$, i.e., $\Phi_x(u) + e \in U \cap \pi^{-1}(\delta_x(q)) = U_{\delta_x(q)}$. As u and e were arbitrary, this is equation 4. ($\Leftarrow$) Given such a U and partition, (R1) is immediate: $\iota(q) \in U_q = U \cap \pi^{-1}(q)$ says exactly $\pi(\iota(q)) = q$. (R2) follows by taking the union over q of $\Phi_x(U_q) \oplus \bar{B}_\eta \subseteq U_{\delta_x(q)} \subseteq U$, with U bounded by hypothesis. For (R3), let $u \in U$, $x \in \Sigma$, $\|e\| \le \eta$, and put $q := \pi(u)$, so $u \in U_q$; the same inclusion gives $\Phi_x(u) + e \in U_{\delta_x(q)} = U \cap \pi^{-1}(\delta_x(q))$, so $\pi(\Phi_x(u) + e) = \delta_x(q) = \delta_x(\pi(u))$. Theorem 3.1 then gives realization with length independence. □

Theorem 3.3 (Two rates in one map)**.** *Under equation 4, every composite map Φ_w is at once* locally contracting *and* globally separating*: for every $q, q' \in Q$ and every $w \in \Sigma^+$,*

- ***(i) Local contraction.*** $\mathrm{diam}\big(\Phi_w(U_q)\big) \le \mathrm{diam}\big(U_{\delta_w(q)}\big) - 2\eta$*; in particular,* $\mathrm{diam}\big(\Phi_w(U_q)\big) \le \mathrm{diam}(U_q) - 2\eta$ *whenever* $\delta_w(q) = q$.
- ***(ii) Global separation.*** $\mathrm{dist}\big(\Phi_w(U_q), \Phi_w(U_{q'})\big) \ge 2\eta \quad$ *whenever* $\quad \delta_w(q) \ne \delta_w(q')$.

Proof of Theorem 3.3(i). We first lift equation 4 from letters to words: for $w = w'x$, induction on $|w|$ gives $\Phi_w(U_q) \oplus \bar{B}_\eta = \Phi_x\big(\Phi_{w'}(U_q)\big) \oplus \bar{B}_\eta \subseteq \Phi_x\big(U_{\delta_{w'}(q)}\big) \oplus \bar{B}_\eta \subseteq U_{\delta_w(q)}$, since $\Phi_{w'}(U_q) \subseteq \Phi_{w'}(U_q) \oplus \bar{B}_\eta$. Any nonempty S has $\mathrm{diam}(S \oplus \bar{B}_\eta) = \mathrm{diam}(S) + 2\eta$, since $\|(s+e) - (s'+e')\| \le \|s - s'\| + 2\eta$ with equality approached by taking s, s' nearly realizing $\mathrm{diam}(S)$ and $e, -e'$ along $s - s'$ of norm η. Applying this to the lifted inclusion and using monotonicity of diam under inclusion gives $\mathrm{diam}(\Phi_w(U_q)) + 2\eta \le \mathrm{diam}(U_{\delta_w(q)})$, and taking $\delta_w(q) = q$ gives the loop case. No quantity here depends on $|w|$, so the margin is the same along every word. □

Proof of Theorem 3.3(ii). Let $\delta_w(q) \neq \delta_w(q')$, and suppose $a \in \Phi_w(U_q)$ and $b \in \Phi_w(U_{q'})$ satisfied $\|a - b\| \leq 2\eta$. Their midpoint $z := \frac{1}{2}(a + b)$ has $\|z - a\| = \|z - b\| = \frac{1}{2}\|a - b\| \leq \eta$, so $z \in \Phi_w(U_q) \oplus \bar{B}_\eta$ and $z \in \Phi_w(U_{q'}) \oplus \bar{B}_\eta$. By equation 4 these lie in $U_{\delta_w(q)}$ and $U_{\delta_w(q')}$ respectively, which are distinct cells of a partition and therefore disjoint, so no such z exists. Hence $\|a - b\| > 2\eta$ for every $a \in \Phi_w(U_q)$ and every $b \in \Phi_w(U_{q'})$, which is the pointwise form used below. Passing to the infimum gives the statement, with a strict inequality whenever the two images are compact. Lemma F.5 draws the same conclusion at code points under a weaker hypothesis: correct decoding along the given word only, without a cell partition. □

Every impossibility result below contradicts the following separation.

Lemma F.5 (Separation at code points)**.** *Let $w \in \Sigma^+$, and suppose that (T1) holds for w: $\pi(h_{|w|}) = \delta_w(p)$ for every $p \in Q$ and every η-trajectory $(h_t)_{t=0}^{|w|}$ of w from $\iota(p)$. If $\delta_w(q) \neq \delta_w(q')$, then $\|\Phi_w(\iota(q)) - \Phi_w(\iota(q'))\| > 2\eta$. In particular this holds for every $w \in \Sigma^+$ under (R1) to (R3), by Theorem 3.1.*

Proof. Write $w = w'x$ and set $u := \Phi_{w'}(\iota(q))$, $u' := \Phi_{w'}(\iota(q'))$. Suppose $\|\Phi_x(u) - \Phi_x(u')\| \leq 2\eta$ and let $z := \frac{1}{2}\big(\Phi_x(u) + \Phi_x(u')\big)$ be the midpoint of the two images, at distance at most η from each. The run that follows w' exactly from $\iota(q)$ and then steps to z is an η-trajectory of w from $\iota(q)$, since only its last step is perturbed, by $z - \Phi_x(u)$; likewise from $\iota(q')$, with last perturbation $z - \Phi_x(u')$. (T1) for w then gives $\pi(z) = \delta_w(q)$ and $\pi(z) = \delta_w(q')$, contradicting the hypothesis. □

The next corollary names the shortest failing length $L^\star$ of a triple (ι, Φ, π), over all words and all perturbations. The failing length $L_{\max}$ of Section 7 is measured instead on sampled words, with the perturbations that the machine happens to produce; neither is the worst case, so $L_{\max} \geq L^\star$ for the trained model, read with its own readout and with the η its machine produces (Lemma D.1).

Corollary F.6 (Failure modes)**.** *If Φ does not realize M_δ with length independence at η, then at least one of the following holds.* (T1) fails: *there is a finite $L^\star$ such that every word of length below $L^\star$ decodes correctly while some word of length $L^\star$ carries a mis-decoding η-trajectory, so that correctness on $\Sigma^{\leq L}$ certifies nothing about longer words for any $L < L^\star$.* (T2) fails: *$U^\star$ is unbounded, and no fixed number of bits represents every reachable state to within η at every length.*

Proof. Definition 2.2 is the conjunction of (T1) and (T2), so its failure is the failure of at least one of them. We treat the two cases in turn; they are not exclusive.

(T1) fails. Then some word w carries an η-trajectory from some $\iota(q)$ with $\pi(h_{|w|}) \neq \delta_w(q)$, and $L^\star$ is the least length of such a word.

(T2) fails. Then $U^\star$ is unbounded, and we show that no fixed number of bits serves every length. Suppose m bits sufficed at every length, that is, 2^m representable points, each state being stored as a point within η of it. Since $U^\star$ is unbounded, choose reachable states $u_1, \ldots, u_{2^m+1}$ pairwise more than 2η apart, and let L be the greatest length of a word reaching one of them, so that all of them are reachable at length at most L. Each u_i lies within η of one of the 2^m points, and there are more states than points, so two of them are stored as the same point and are therefore at most 2η apart, contradicting their choice. Hence, for every m there is a length at which m bits no longer suffice: the precision needed grows without bound as L grows. This is the contrapositive of Proposition F.1. □

F.2 Affine recurrence

One affine step does not expose the conflict of Theorem 3.3: a rate below one shrinks the cells and, if the codes start far enough apart, leaves a gap above 2η between them. What an affine map cannot do is meet both demands along a repeated word, because its rate does not depend on where the trajectories sit (Definition F.7, Proposition F.8): along w^k, the rate that damps the perturbation also shrinks the gap between two codes. Input-dependent gates modulate that rate *through time* and leave it uniform in space.

Definition F.7 (Single-rate recurrence)**.** Φ *is* single-rate *if the displacement between two trajectories depends only on their initial displacement: for every $w \in \Sigma^*$ there is a map T_w with $\Phi_w(u) -$*

$\Phi_w(v) = T_w(u - v)$ *for all* $u, v \in \mathcal{H}$*, so that one and the same map governs every pair of points, wherever they sit.*

Proposition F.8 (Single-rate recurrences are exactly the affine ones)**.** *A continuous recurrence is single-rate if and only if it is* affine*, and then* $T_w = A_w$ *is linear. Its local Lipschitz constant is then the operator norm* $\|A_x\|_{\mathrm{op}}$ *at* every *point of* $\mathcal{H}$*, independently of where the pair of points is taken.*

Proof. If Φ is affine then equation 5 gives $\Phi_w(u) - \Phi_w(v) = A_w(u - v)$, so $T_w = A_w$ meets Definition F.7. Conversely, suppose the definition holds and take $w = x$ a single symbol. Setting $v = 0$ gives $\Phi_x(u) = T_x(u) + \Phi_x(0)$, and substituting this back gives $T_x(u) - T_x(v) = T_x(u - v)$ for all u, v, so T_x is additive with $T_x(0) = 0$. An additive map that is continuous is linear, and T_x is continuous because Φ_x is; hence each Φ_x is affine with linear part $A_x \coloneqq T_x$, and every composite is affine with $A_w = A_{x_k} \cdots A_{x_1}$. Finally, $\|\Phi_x(u) - \Phi_x(v)\| = \|A_x(u - v)\|$ for all u, v, so the ratio is bounded by $\|A_x\|_{\mathrm{op}}$ uniformly and attained on the corresponding singular direction, independently of where the pair is taken: the same constant governs pairs inside a cell and pairs in different cells, and along a word w the same operator A_w acts on a perturbation and on the difference between two code points. □

Lemma F.9 (Contraction requires a stable matrix)**.** *If* Φ *is affine and satisfies (R2) on a nonempty* U *at some* $\eta > 0$*, then for every* $w \in \Sigma^+$ *and unit* v*,* $\sum_{j \geq 0} \|(A_w^{\mathsf{T}})^j v\| \leq \mathrm{diam}(U)/2\eta < \infty$*, and consequently* $\rho(A_w) < 1$*; in particular* $\rho(A_x) < 1$ *for every* x*.*

Proof. We take $\|\cdot\|$ Euclidean here, so that an inner product is available; all norms on $\mathbb{R}^d$ are equivalent, and only the constant in the bound depends on that choice, not the conclusion $\rho(A_w) < 1$. Write $A \coloneqq A_w$ and $\Psi \coloneqq \Phi_w$, so $\Psi(u) = Au + b$ for some b. Iterating (R2) along w gives

$$\Psi(U) \oplus \bar{B}_\eta \subseteq U, \tag{10}$$

since $\Phi_{x_2}(\Phi_{x_1}(U) \oplus \bar{B}_\eta) \oplus \bar{B}_\eta \subseteq \Phi_{x_2}(U) \oplus \bar{B}_\eta \subseteq U$, and so on along w.

Fix $u_0 \in U$, fix $k \geq 1$, and let $e_0, \ldots, e_{k-1}$ satisfy $\|e_t\| \leq \eta$. Define $h_0 \coloneqq u_0$ and $h_{t+1} \coloneqq \Psi(h_t) + e_t$. By equation 10 and induction, $h_t \in U$ for all $t \leq k$, and since Ψ is affine,

$$h_k = \Psi^k(u_0) + \sum_{j=0}^{k-1} A^j e_{k-1-j}. \tag{11}$$

Let v be a unit vector. Choose $e_{k-1-j} \coloneqq \eta\,(A^{\mathsf{T}})^j v / \|(A^{\mathsf{T}})^j v\|$ when $(A^{\mathsf{T}})^j v \neq 0$ and $e_{k-1-j} \coloneqq 0$ otherwise, and let h_k^+ and h_k^- be the endpoints equation 11 obtained from (e_t) and from $(-e_t)$. Both lie in U, and using $\langle A^j z, v\rangle = \langle z, (A^{\mathsf{T}})^j v\rangle$,

$$\mathrm{diam}(U) \geq \langle h_k^+ - h_k^-, v\rangle = 2\sum_{j=0}^{k-1} \big\langle e_{k-1-j}, (A^{\mathsf{T}})^j v\big\rangle = 2\eta \sum_{j=0}^{k-1} \big\|(A^{\mathsf{T}})^j v\big\|,$$

the terms with $(A^{\mathsf{T}})^j v = 0$ contributing 0 to both sides. Letting $k \to \infty$ gives $\sum_{j \geq 0} \|(A^{\mathsf{T}})^j v\| \leq \mathrm{diam}(U)/2\eta < \infty$.

A convergent series has vanishing terms, so $\|(A^{\mathsf{T}})^j v\| \to 0$ for every v, hence $\|(A^{\mathsf{T}})^j\| \to 0$ in finite dimension and $\rho(A) = \rho(A^{\mathsf{T}}) < 1$. Taking $w = x$ gives $\rho(A_x) < 1$ for every letter. □

Definition F.10 (Definite automaton)**.** *A semiautomaton* (Q, Σ, δ) *(Perles et al., 1963) is* k-definite *if* δ_w *is a constant map of* Q *for every word* w *with* $|w| \geq k$*, and* definite *if it is* k*-definite for some* k*; we write* $c_a \in Q^Q$ *for the constant map with value* a*. Equivalently, the state reached after reading a word of length at least* k *depends on the last* k *symbols alone: splitting* $w = uv$ *with* $|v| = k$ *gives* $\delta_w(q) = \delta_v(\delta_u(q))$*, the constant value of* δ_v*, for every* q *and every prefix* u*. Definite automata are aperiodic, hence recognize star-free behaviors (Schützenberger, 1965; McNaughton & Papert, 1971; Eilenberg, 1974); the converse fails, so "definite" is strictly the stronger of the two conclusions.*

The next lemma turns a statement made one word at a time, that every repeated word ends up acting as a constant map, into a property of the automaton.

Lemma F.11 (Constant powers force definiteness)**.** *Let $|Q| = n$ and let $S \coloneqq \{\delta_w \mid w \in \Sigma^+\}$ be the transition semigroup of (Q, Σ, δ), carrying the product inherited from concatenation, $\delta_u \cdot \delta_v \coloneqq \delta_{uv} = \delta_v \circ \delta_u$. Suppose that for every $w \in \Sigma^+$ the map δ_{w^k} is constant for all sufficiently large k. Then* (i) *every idempotent of S is a constant map.* (ii) *Whenever every idempotent of S is constant, in particular under this hypothesis, δ_w is constant for every word with $|w| > |S|$, so the automaton is k-definite with $k = |S| + 1 \leq n^n + 1$. Contrapositively, if the automaton is not definite, some word acts as an idempotent of S that is not constant.*

Proof. S is a subsemigroup of Q^Q and hence finite, with $|S| \leq n^n$.

(i) Let $e \in S$ be idempotent and choose $w \in \Sigma^+$ with $e = \delta_w$. Then $\delta_{w^k} = e^k = e$ for every $k \geq 1$, and δ_{w^k} is constant for k large by hypothesis; so e is constant.

(ii) Let $w = x_1 \cdots x_{|w|}$ with $|w| > |S|$ and write $p_i \coloneqq \delta_{x_1 \cdots x_i} \in S$, $1 \leq i \leq |w|$, for its prefixes. Since $|w| > |S|$ two of them coincide, say $p_i = p_j$ with $i < j$. Put $t \coloneqq \delta_{x_{i+1} \cdots x_j} \in S$, so that $p_i = p_j = p_i \cdot t$ and, by induction, $p_i = p_i \cdot t^r$ for every $r \geq 1$. Every element of a finite semigroup has an idempotent power, so t^r is idempotent for some $r \geq 1$, hence $t^r = c_a$ for some $a \in Q$, since every idempotent of S is constant. Then

$$p_i = p_i \cdot t^r = t^r \circ p_i = c_a \circ p_i = c_a,$$

so p_i is itself constant with value a, and $\delta_w = \delta_{x_{i+1} \cdots x_{|w|}} \circ p_i$ is the constant map with value $\delta_{x_{i+1} \cdots x_{|w|}}(a)$. As w was an arbitrary word of length greater than $|S|$, the automaton is $(|S| + 1)$-definite. □

Theorem 4.1 (Affine recurrences realize at most definite automata)**.** *Let Φ be affine in the hidden state, $\Phi_x(h) = A_x h + b_x$ for $x \in \Sigma$.*

- *(i) If $\rho(A_w) \geq 1$ for some word $w \in \Sigma^+$, then (R2) **fails**: no nonempty bounded set $U \subset \mathcal{H}$ satisfies $\Phi_w(U) \oplus \bar{B}_\eta \subseteq U$ for this w. Consequently, Φ realizes no automaton with length independence at any $\eta > 0$.*
- *(ii) If $\rho(A_w) < 1$ for all words $w \in \Sigma^+$, then Φ realizes **at most a definite automaton** with length independence at $\eta > 0$. For any nondefinite target, no region satisfying (R1) and (R2) satisfies (R3).*

Proof of Theorem 4.1. Let Φ be affine in the hidden state, $\Phi_x(h) = A_x h + b_x$ and $A_w = A_{x_k} \cdots A_{x_1}$ as in equation 5, and let $\eta > 0$.

(i) $\rho(A_w) \geq 1$ *for some* $w \in \Sigma^+$. Suppose some bounded U satisfied $\Phi_w(U) \oplus \bar{B}_\eta \subseteq U$. The proof of Lemma F.9 uses no more than that single inclusion: run it with $\Psi \coloneqq \Phi_w$ and $A \coloneqq A_w$ to get $\sum_{j \geq 0} \|(A_w^\mathsf{T})^j v\| \leq \operatorname{diam}(U)/2\eta < \infty$ for every unit v, whence $\|(A_w^\mathsf{T})^j\| \to 0$ and $\rho(A_w) < 1$, contradicting the hypothesis. So no bounded U satisfies the inclusion for Φ_w, and a fortiori none satisfies (R2) for every symbol, since iterating (R2) along w gives $\Phi_w(U) \oplus \bar{B}_\eta \subseteq U$. By Theorem 3.1 no triple (ι, Φ, π) realizes any target with length independence at η, whatever the state dimension, the encoding and the readout.

(ii) $\rho(A_w) < 1$ *for every* $w \in \Sigma^+$. Suppose a bounded U carries (R1) to (R3) at η, and fix $w \in \Sigma^+$ and $q, q' \in Q$. Affineness gives

$$\big\|\Phi_{w^k}(\iota(q)) - \Phi_{w^k}(\iota(q'))\big\| = \big\|A_w^k\big(\iota(q) - \iota(q')\big)\big\| \xrightarrow[k \to \infty]{} 0,$$

since $\rho(A_w) < 1$ forces $A_w^k \to 0$. Lemma F.5 requires that quantity to exceed 2η whenever $\delta_{w^k}(q) \neq \delta_{w^k}(q')$, so there is a threshold $K_{q,q',w}$ with $\delta_{w^k}(q) = \delta_{w^k}(q')$ for every $k \geq K_{q,q',w}$. Taking the maximum over the finitely many pairs (q, q') makes δ_{w^k} a constant map for all k large, and $w \in \Sigma^+$ was arbitrary. Lemma F.11 then makes the target k-definite with $k = |S| + 1 \leq n^n + 1$, where S is its transition semigroup: the state after a word of length at least k is fixed by the last k symbols of that word (Definition F.10).

Contrapositively, if the target is *not* definite then no bounded U carries (R1) to (R3) at any $\eta > 0$: some pair $q \neq q'$ is kept distinguished by δ_{w^k} for infinitely many k, while the images of their code points merge, and (R3) fails at every k beyond the point where the separation of Lemma F.5 is lost. The same operator A_w moves the signal and the perturbation (Proposition F.8), so no choice of d, of ι, of π or of U separates the two rates. □

Corollary F.12 (No affine recurrence realizes a nontrivial group with length independence)**.** *Let M_δ contain a nontrivial group and let Φ be affine in the hidden state. Then, for any state dimension d, any $\eta > 0$, any affine parameters A_x, b_x, and any encoding ι and readout π, no region U satisfies (R1) to (R3).*

Proof. This is a special case of Theorem 4.1: branch (i) excludes every target, and branch (ii) excludes this one because a monoid containing a nontrivial group is not definite. Indeed, let $G \leq M_\delta$ be a nontrivial group with identity e, and pick $g \in G$ with $g \neq e$. The identity map id_Q, if it lies in G, is an idempotent of G and therefore equals e; hence $g \neq \mathrm{id}_Q$, and $g = \delta_w$ for some $w \in \Sigma^+$. If the automaton were k-definite, then, since $g^{|G|} = e$, the element $g = g^{m|G|+1} = \delta_{w^{m|G|+1}}$ would be a constant map c_a as soon as $(m|G| + 1)|w| \geq k$. A constant map is idempotent, so $e = g^{|G|} = c_a^{|G|} = c_a = g$, contradicting $g \neq e$. □

The next proposition shows that every definite automaton (a shift register that remembers the last k symbols) is realized by a contracting affine recurrence, with a precision depending on the order k of the target rather than the word length.

Proposition F.13 (Every definite automaton has a contracting affine realization)**.** *Let (Q, Σ, δ) be k-definite with $k \geq 1$, and let $m := |\Sigma| + |Q|$. There are an affine recurrence $\Phi_x(h) = Ah + b_x$ on $\mathbb{R}^{km}$ with $A^k = 0$, hence $\rho(A_w) = 0$ for every $w \in \Sigma^+$, an encoding ι and a readout π that realize the target with length independence at every $\eta < 1/(2k)$ in the norm $\|\cdot\|_\infty$. In any other norm, the same holds for $\eta < 1/(2k\mu)$, where $\|\cdot\|_\infty \leq \mu\|\cdot\|$.*

Proof. Index the basis of $\mathbb{R}^m$ by $\Sigma \sqcup Q$, write $\mathbf{1}_y$ for the basis vector of $y \in \Sigma \sqcup Q$, and split $h \in \mathbb{R}^{km}$ into k slots $h = (h^{(1)}, \ldots, h^{(k)})$ of $\mathbb{R}^m$, slot 1 being the most recent. Let A shift every slot one position down and drop the last one, $(Ah)^{(1)} = 0$ and $(Ah)^{(j+1)} = h^{(j)}$, and let b_x hold $\mathbf{1}_x$ in slot 1 and 0 elsewhere. Then $A^k = 0$, so $A_w = A^{|w|}$ is nilpotent and $\rho(A_w) = 0$ for every $w \in \Sigma^+$: the recurrence is on branch (ii) of Theorem 4.1. Encode $\iota(q) := (\mathbf{1}_q, 0, \ldots, 0)$.

Exact runs. From $\iota(q)$, after a word w with $|w| < k$, slots 1 to $|w|$ hold $\mathbf{1}_{x_{|w|}}, \ldots, \mathbf{1}_{x_1}$, slot $|w| + 1$ holds the marker $\mathbf{1}_q$, and the other slots are 0; after a word with $|w| \geq k$, the slots hold the last k symbols of w, most recent first. Call these configurations *clean*, and let C be the finite set of clean configurations reached from $\iota(Q)$. A clean configuration c determines a state $D(c)$: if c carries the marker $\mathbf{1}_q$ and the symbols of w, then $D(c) := \delta_w(q)$; if c carries only the last k symbols v, then $D(c)$ is the constant value of δ_v. Then $D(\Phi_x(c)) = \delta_x(D(c))$ for every $c \in C$ and $x \in \Sigma$. If c carries $\mathbf{1}_q$ and w with $|wx| < k$, this is $\delta_{wx}(q) = \delta_x(\delta_w(q))$. If $|wx| = k$, then $\Phi_x(c)$ carries only wx, and the constant value of δ_{wx} is $\delta_{wx}(q) = \delta_x(\delta_w(q))$. If c carries only v, write $vx = yv'$ with $y \in \Sigma$; then $\Phi_x(c)$ carries v', and $\delta_{vx} = \delta_{v'} \circ \delta_y$ is constant with the value of $\delta_{v'}$, while $\delta_{vx} = \delta_x \circ \delta_v$ is constant with value $\delta_x(D(c))$.

Region and margin. Let $U := \{c + e \mid c \in C,\ \|e^{(j)}\|_\infty \leq j\eta \text{ for } j = 1, \ldots, k\}$, which is bounded since C is finite and contains $\iota(Q)$. For $c+e \in U$ and $\|e'\|_\infty \leq \eta$, $\Phi_x(c+e)+e' = \Phi_x(c)+(Ae+e')$ with $\Phi_x(c) \in C$; slot 1 of $Ae+e'$ is $e'^{(1)}$, of norm at most η, and slot $j+1$ is $e^{(j)} + e'^{(j+1)}$, of norm at most $(j + 1)\eta$. Hence $\Phi_x(U) \oplus \bar{B}_\eta \subseteq U$, which is (R2). For $\eta < 1/(2k)$, every coordinate of e is below $\frac{1}{2}$ in absolute value, while two distinct clean configurations differ by 1 in some coordinate, so every point of U has a unique clean configuration c, and $\pi(c + e) := D(c)$ is well defined. (R1) is $\pi(\iota(q)) = D(\iota(q)) = \delta_\varepsilon(q) = q$, and (R3) is $\pi(\Phi_x(c + e) + e') = D(\Phi_x(c)) = \delta_x(D(c)) = \delta_x(\pi(c + e))$. Theorem 3.1 gives realization with length independence. For another norm, $\|e\| \leq \eta$ implies $\|e\|_\infty \leq \mu\eta$, and the same argument applies with $\mu\eta$ in place of η. □

For $k = 1$ the shift drops everything, $A = 0$, and each symbol writes its own code: the realization uses no nonlinearity at all.

Corollary F.14 (Contracting models need exponentially fine precision)**.** *Let $\kappa := \max_x \|A_x\|_{\mathrm{op}} < 1$, which is stronger than the hypothesis $\rho(A_w) < 1$ of branch (ii), let the target be nondefinite, and let $C := \max_{q,q' \in Q} \|\iota(q) - \iota(q')\|$ be the diameter of the code. At perturbation level $\eta > 0$ the model is guaranteed correct on every w with $|w| \leq L$ only if $L \leq L_{\max}$, where $L_{\max} = \lceil \ln(C/2\eta)/\ln(1/\kappa) \rceil - 1$. Equivalently, staying correct on every word of length at most L requires $\eta < C\kappa^L/2$: the perturbation level the model tolerates shrinks exponentially with L.*

Proof. The operator norm is submultiplicative, so $A_w = A_{x_{|w|}} \cdots A_{x_1}$ of equation 5 obeys $\|A_w\|_{\mathrm{op}} \le \prod_i \|A_{x_i}\|_{\mathrm{op}} \le \kappa^{|w|}$, and Φ_w is $\kappa^{|w|}$-Lipschitz by Proposition F.8. Hence for $|w| = L$ and any q, q' with $\delta_w(q) \neq \delta_w(q')$, the two code points, at distance at most C apart, induce states separated by at most $C\kappa^L$. Lemma F.5 forces that separation to exceed 2η, so the bound must itself exceed 2η: correctness at length L requires $C\kappa^L > 2\eta$, which is the stated $L_{\max}$. Non-definiteness is what makes the requirement bite, since it supplies, at every length, a word w and a pair q, q' that the target still keeps apart (a word v with $|v| \ge L$ and δ_v not constant has a suffix of length L whose action is not constant either, since δ_v factors through it); for a definite target every pair is allowed to merge after a bounded number of steps and there is no horizon to speak of. □

For an isometry[2], the gap between two trajectories stays at C while the perturbations pile up, and the horizon is the length at which the pile reaches the gap.

Proposition F.15 (Isometric models need precision of order $1/L$)**.** *Let every A_x be a linear isometry of $\mathbb{R}^d$, let w be a word of length L, and let $q \neq q'$ be states with $\delta_w(q) \neq \delta_w(q')$, encoded $C \coloneqq \|\iota(q) - \iota(q')\|$ apart. Guaranteed correctness on every word of length at most L requires $\eta < C/2L$: the perturbation level the model tolerates shrinks like $1/L$, so no fixed precision serves all lengths, in agreement with Lemma F.9.*

Proof. An η-trajectory deviates from the unperturbed one by $\bar{e}_L = \sum_{t=1}^{L} B_t e_t$, where $B_t \coloneqq A_{x_L} \cdots A_{x_{t+1}}$ and $B_L \coloneqq I$ are products of isometries, hence isometries, hence orthogonal and invertible. The same hypothesis keeps the two unperturbed trajectories from $\iota(q)$ and $\iota(q')$ exactly C apart, so correctness fails as soon as the deviations can close that gap: the trajectories then reach a common point and decode alike, contradicting $\delta_w(q) \neq \delta_w(q')$.

Take $v \coloneqq \big(\Phi_w(\iota(q')) - \Phi_w(\iota(q))\big)/C$ and $e_t \coloneqq \eta B_t^{-1} v$, of norm η; then $\bar{e}_L = \eta L v$. Running this along the trajectory from $\iota(q)$ and its negative along the one from $\iota(q')$ moves the two endpoints $2\eta L$ towards each other, and scaling the perturbations down if $2\eta L > C$ makes them coincide, so $2\eta L < C$ is necessary. □

Corollary F.16 (The horizon bound is not a design parameter)**.** *Write $\Lambda \coloneqq \ln(C/2\eta)$, so that the bound of Corollary F.14 reads $L_{\max} = \Lambda/\ln(1/\kappa)$ up to rounding. Then*

$$\frac{dL_{\max}}{L_{\max}} = \frac{L_{\max}}{\Lambda}\,\frac{d\kappa}{\kappa} \xrightarrow[\kappa \to 1^-]{} \infty.$$

Pinning $L_{\max}$ to a relative tolerance ε therefore requires pinning κ to a relative tolerance $\varepsilon\Lambda/L_{\max}$, which tightens as the horizon lengthens: at $C/2\eta = 10^6$ and $L_{\max} = 10^4$, a horizon specified to within a factor of two requires κ specified to within 7×10^{-4}. Neither κ nor C is set by the designer: the first is set by learned, input-dependent gates, the second is the spread of the code the encoder learns, and the same holds on the other branch, where Proposition F.15 bounds the horizon by $C/2\eta$ with C again learned. The horizon that a trained affine cell can at best reach is therefore a property of that particular trained model, measurable after the fact and not specifiable in advance.

Proof. Differentiate $L_{\max} = -\Lambda/\ln\kappa$: $dL_{\max}/d\kappa = \Lambda/(\kappa \ln^2 \kappa)$, and divide by $L_{\max}$. The limit follows from $\ln(1/\kappa) \to 0$. □

F.3 Multistability

The next lemma reads Theorem 4.1 as a requirement on the architecture. It places the nonlinearity on the previous state, since a gate computed by an arbitrarily nonlinear function of x_t still leaves Φ_x affine in h.

Lemma F.17 (Two rates require the nonlinearity to act on the state)**.** *Let the target automaton be nondefinite, for instance because M_δ contains a nontrivial group (Corollary F.12), and let Φ satisfy equation 4 on a bounded region U as in Theorem 3.2 at some $\eta > 0$. Then some Φ_x is not affine.*

[2] A linear map A is an *isometry* when $\|Av\| = \|v\|$ for every v, equivalently $A^{\mathsf{T}}A = I$ in the Euclidean norm: it neither shrinks nor stretches any distance, so all its eigenvalues have modulus one. It is the distance-preserving representative of the $\rho = 1$ branch of Theorem 4.1, and covers unitary and orthogonal recurrences as well as $A_x = I$.

Proof. Suppose every Φ_x is affine. Theorem 3.2 turns equation 4 on a bounded U into realization of the target with length independence at η, so, by Theorem 3.1, some region satisfies (R1) to (R3). Theorem 4.1 forbids this for a nondefinite target: branch (i) excludes every target, and branch (ii) every nondefinite one. Hence some Φ_x is not affine. □

A restoring step can remove a perturbation outright, as the construction of Section 6 does, or merely shrink it, as a real-analytic recurrence must (Remark F.19). The next definition names the two grades.

Definition F.18 (Grades of restoration)**.** *Let F be one of the maps Φ_x, acting on the region $U = \bigsqcup_{q\in Q} U_q$ of Theorem 3.2. F is* exactly restoring *on U if it sends each cell U_q to a single point, the perturbation being annihilated rather than reduced; for $F = \Phi_x$ we require that point to be $\iota(\delta_x(q))$, as in Section 5 once attractor indices are mapped onto automaton states. F is* asymptotically restoring *on a cell U_q if there is $\lambda < 1$ with $\|F(u)-F(u')\| \le \lambda\|u-u'\|$ for all $u, u' \in U_q$, perturbations being strictly reduced at every step. Exact restoration is the case $\lambda = 0$.*

Remark F.19 (Exact restoration needs gaps or jumps)**.** *Let Φ_x be exactly restoring on U in the sense of Section 5, with the attractors mapped onto the code points, sending each cell U_q to the single point $\iota(\delta_x(q))$. If Φ_x is continuous on U, it is constant on each connected component of U, since a continuous map from a connected set to a finite set is constant. Two cells whose states δ_x sends to different states therefore lie in different components: exact restoration by a continuous map requires gaps between the cells, which the NFSM provides by taking $U = \iota(Q) \oplus \bar{B}_\eta$, a union of disjoint balls. On the whole of $\mathcal{H}$, which is connected, a map with finitely many values cannot be continuous unless it is constant, so the jumps of the* $\arg\max$ *in equation 6 are unavoidable. A real-analytic recurrence, such as a* $\tanh$ *or sigmoid network, cannot be exactly restoring unless δ_x is constant: by equation 4, a cell reached by some symbol contains an η-ball, Φ_x is constant on that cell, and a real-analytic map that is constant on a ball is constant on $\mathcal{H}$.*

To compare the two grades, and later affine maps and stacks, we count what a single map can keep apart under perturbation: the sets it maps into themselves with a margin of η.

Definition F.20 (Held sets and robust capacity)**.** *Let $F : \mathcal{H} \to \mathcal{H}$ and $\eta > 0$.* F holds *a set $V \subseteq \mathcal{H}$ if V is nonempty and bounded and $F(V) \oplus \bar{B}_\eta \subseteq V$. The* robust capacity *of F in a region U is $\log_2 N$ bits, where N is the largest number of pairwise disjoint subsets of U that F holds.*

Under equation 4, a word w with $\delta_w(q) = q$ makes Φ_w hold the cell U_q, by the lifting in the proof of Theorem 3.3. A word whose action fixes m states therefore forces a robust capacity of at least $\log_2 m$ bits on Φ_w; for a nonconstant idempotent δ_w (Lemma F.11), $m \ge 2$.

Proposition F.21 (The two grades have the same capacity)**.** *Let $U = \bigsqcup_{q\in Q'} U_q$ be a union of n pairwise disjoint closed cells, and let F hold every cell and be asymptotically restoring on it, with constant $\lambda \in [0, 1)$. Then* (i) *each U_q contains a unique fixed point α_q of F, which attracts every trajectory of F started in U_q, and the open η-ball around α_q lies in U_q;* (ii) *every subset of U that F holds contains one of these balls, so the robust capacity of F in U is exactly $\log_2 n$ bits. The count is the same for exact restoration ($\lambda = 0$) as for asymptotic restoration ($0 < \lambda < 1$), and it does not mention d.*

Proof. (i) Each cell is closed and bounded, hence complete, and $F(U_q) \subseteq F(U_q) \oplus \bar{B}_\eta \subseteq U_q$. F restricted to U_q is a λ-contraction of U_q into itself, so the Banach fixed point theorem gives a unique $\alpha_q \in U_q$ with $F^k(u) \to \alpha_q$ for every $u \in U_q$. For $k \ge 1$, $F^k(u) \in F(U_q)$, so $\bar{B}_\eta(F^k(u)) \subseteq U_q$; any z with $\|z - \alpha_q\| < \eta$ satisfies $\|z - F^k(u)\| < \eta$ for k large, hence $z \in U_q$. *(ii)* Let $V \subseteq U$ be held by F, pick q with $V \cap U_q \neq \emptyset$ and v in this intersection. Then $F^k(v)$ stays in U_q, since F holds U_q, and converges to α_q. It also stays in V, and for $k \ge 1$ it lies in $F(V)$, so $\bar{B}_\eta(F^k(v)) \subseteq V$; as in (i), the open η-ball around α_q lies in V. Two disjoint held subsets of U therefore contain the balls of two distinct fixed points, so there are at most n of them, and the n cells are n such subsets. □

Proposition F.22 (An affine recurrence has zero capacity at any width)**.** *Let Φ_x be affine and satisfy (R2) on a nonempty bounded U at some $\eta > 0$. Then* (i) *Φ_x has a unique fixed point h^*_x, the open η-ball around it lies in U, and every trajectory of Φ_x in U converges to it; more generally, every set that Φ_x holds contains this ball, so the robust capacity of Φ_x (Definition F.20) is $\log_2 1 = 0$ bits in every region;* (ii) *at most one cell is held, i.e., $\Phi_x(U_q) \oplus \bar{B}_\eta \subseteq U_q$ for at most one $q \in Q$. Neither conclusion mentions the state dimension d.*

Proof. Lemma F.9 gives $\rho(A_x) < 1$, so $I - A_x$ is invertible, Φ_x has a unique fixed point h^*_x, and $\Phi^k_x(u) - h^*_x = A^k_x(u - h^*_x) \to 0$ for every u. Fix $u \in U$: (R2) places $\bar{B}_\eta(\Phi^k_x(u))$ in U for every $k \geq 1$, and any z with $\|z - h^*_x\| < \eta$ satisfies $\|z - \Phi^k_x(u)\| < \eta$ for k large, so $z \in U$; the open η-ball at h^*_x therefore lies in U and its basin is all of U, however U is divided into cells. The argument uses only that Φ_x holds U, so it applies to every set that Φ_x holds, and any two such sets meet. That is (i). For (ii), if two cells were held then the same argument places that ball in each of them, and cells of a partition are disjoint. □

Remark F.23. *Proposition F.22 does not say that an affine recurrence carries no information: on the contracting branch it stays correct up to a horizon (Corollary F.14). The capacity that is zero is the length-independent one.*

A stack of affine recurrences, with arbitrary nonlinear maps between its layers, is not affine as a whole, so Theorem 4.1 does not apply to it directly. Each layer, however, stays affine in its own hidden state.

Definition F.24 (Stack of affine recurrences)**.** *Split the hidden state into layers, $\mathcal{H} = \mathbb{R}^{d_1} \times \cdots \times \mathbb{R}^{d_s}$, each carrying the norm induced by $\|\cdot\|$. A map of $\mathcal{H}$ is* layerwise affine *if the new value of each layer is an affine function of the old value of that same layer, whose coefficients may depend, arbitrarily, on the old values of the layers below it and on nothing else; the coefficients of the bottom layer are constant. A* stack of affine recurrences *is a recurrence whose every Φ_x is layerwise affine.*

Every stack of affine layers is of this form, whatever the maps between them (MLPs, normalizations, residual connections, input-dependent gates), because what a layer receives at a step is computed from the input symbol and from the layers below it. A causal convolution between layers fits as well, once its buffer is counted as a layer of its own. No continuity is assumed. The composite of two layerwise affine maps is layerwise affine, so every composite Φ_w of a stack is.

Theorem F.25 (Depth does not create capacity)**.** *Let Ψ be layerwise affine, of any depth and any widths, and let $\eta > 0$. Then Ψ holds (Definition F.20) at most one set in any family of pairwise disjoint sets, so its robust capacity is zero in every region. Consequently, a stack of affine recurrences realizes with length independence at most a definite automaton, at any depth and any width.*

Proof. The proof is by induction on the number of layers. With one layer, Ψ is affine, and the proof of Proposition F.22(i) uses nothing but the fact that Ψ holds the set: Ψ has a unique fixed point, and every set that Ψ holds contains the open η-ball around it. Two held sets therefore meet.

With more layers, suppose that Ψ held two disjoint sets. The bottom layer reads nothing above it, so it evolves on its own as a single affine map. A perturbation confined to the bottom layer is a perturbation of the whole hidden state of the same size, so this map holds the bottom-layer projection of each of the two sets, and by the one-layer case its fixed point lies in both projections. Now cut both sets at this fixed point: keep, from each set, the hidden states whose bottom layer sits exactly at the fixed point, and drop the bottom layer. The two slices are nonempty, since the fixed point lies in both projections; bounded, since the sets are; and disjoint, since a point common to both slices would, with the bottom layer added back, be common to the two sets. A hidden state whose bottom layer sits at the fixed point keeps it there, because the bottom layer does not read the layers above, so on such hidden states Ψ acts on the upper layers only. This action holds each slice: a perturbation of size at most η on the upper layers, with none on the bottom one, keeps the hidden state inside its held set, and leaves the bottom layer at the fixed point. With the bottom layer frozen, finally, the coefficients of the second layer are constant, so the action on the upper layers is layerwise affine with one layer fewer. It holds two disjoint slices, which the induction hypothesis excludes. Nonlinear maps between layers thus change which point each layer settles on, but give no layer a second one.

For the consequence, suppose that a stack realizes a nondefinite target with length independence at η, with the cells of Theorem 3.2. By Lemma F.11, some word w acts on Q as an idempotent that is not constant, so it fixes two distinct states. Lifting equation 4 to words, as in the proof of Theorem 3.3, shows that Φ_w holds the cells of both states, which are disjoint. Since Φ_w is layerwise affine, this contradicts the first part. □

F.4 THE SCAN BUDGET

Proposition 5.2 (The budget forces a finite reachable set)**.** *If Φ is scan-representable with budget (b, c), then M_Φ has at most 2^b elements and, for every initial hidden state h_0, so does the set $\{\Phi_w(h_0) \mid w \in \Sigma^*\}$ of reachable hidden states, $h_0 = \Phi_\varepsilon(h_0)$ included. On these $K \le 2^b$ hidden states, M_Φ acts as a submonoid of T_K, the full transformation monoid on K elements.*

Proof of Proposition 5.2. The representation ζ is injective into $\{0,1\}^b$, so $|M_\Phi| \le 2^b$. The map $\Phi_w \mapsto \Phi_w(h_0)$ sends M_Φ onto the set of reachable states, $h_0 = \Phi_\varepsilon(h_0)$ included, which therefore has $K \le |M_\Phi| \le 2^b$ elements. For the action, M_Φ is closed under composition, since $\Phi_v \circ \Phi_u = \Phi_{uv}$, and contains the identity. The set of reachable states is invariant under every Φ_v, since $\Phi_v(\Phi_w(h_0)) = \Phi_{wv}(h_0)$, so each element of M_Φ restricts to a map of this K-element set into itself. Restriction preserves composition and the identity, so it is a monoid homomorphism from M_Φ to T_K, and M_Φ acts through its image, a submonoid of T_K. This homomorphism need not be injective, since two maps of M_Φ may agree on every reachable state.

The argument uses only that M_Φ is closed under composition and admits an injective b-bit representation, so it applies verbatim to the executed recurrence of Appendix D, with $\tilde{\Phi}_x$ in place of Φ_x, as long as the merge is exact; a floating-point scan does not compose the executed maps, since its rounding depends on the shape of the tree. □

Remark F.26 (Which map the attractors belong to)**.** *Exact restoration splits each step in two. Define R on V by $R(h) \coloneqq a_k$ for $h \in V_k$, and let E_x be any map of the attractors into V with $E_x(a_k) \in V_{\sigma_x(k)}$. Then $\Phi_x = R \circ E_x \circ R$ for every x and every such E_x. The map R does not depend on the symbol, so it is the restoring organ of Section 3, and E_x is the executive organ: it only needs to land in the right basin, and R does the rest. The inner R is what lets an executive defined on the attractors act on a whole basin, which is what (R3) asks for. For R, each a_k is an attractor and V_k is its basin: R is idempotent, its fixed points are $a_1, \dots, a_K$, and every point of V_k reaches a_k in one step and stays there. The map Φ_x itself fixes a_k exactly when $\sigma_x(k) = k$; if σ_x is a cycle, the attractors form one attracting cycle of Φ_x. The case $\sigma_x(k) = k$ is the multistable one of Section 4.2 and the hypothesis of Proposition F.21. In Section 6, $R = \mathrm{rd}$ and $E_x(h) = \theta_x h$, which sends $\mathbf{1}_k$ to column k of θ_x, a point of $V_{\sigma_x(k)}$; the outer* rd *restores this output and the inner one the stored state.*

G THE NFSM CONSTRUCTION IN DETAIL

A head *as it runs* holds its state as an attractor index and moves it by table lookups (Appendix G.1); the *one-hot model* of Section 6 carries each index k by the attractor $a_k = \mathbf{1}_k$, so that the theory, stated on a continuous state space, applies, and the two agree on every run (Appendix G.3). Until Appendix G.5, we describe a single head, drop its index i, and write d for its number of attractor indices, or *indices* for short. For a matrix θ, we write $\theta_{jk} \coloneqq \mathbf{1}_j^\mathsf{T} \theta \, \mathbf{1}_k$ for the entry in row j and column k.

G.1 A HEAD AS IT RUNS

At step t, a learned affine map of the input x_t, reshaped into a $d \times d$ matrix, gives the logits θ_{x_t}, with $(\theta_{x_t})_{jk}$ scoring the move from index k to index j. The head reads off the table σ_{x_t} of Section 6 and applies it to its current index:

$$\sigma_{x_t}(k) \coloneqq \arg\max_{j \in \{1,\dots,d\}} (\theta_{x_t})_{jk}, \qquad k_t = \sigma_{x_t}(k_{t-1}), \tag{12}$$

from a fixed initial index k_0. Ties in an $\arg\max$ are broken by a fixed arbitrary ordering of the indices, so every column has one largest entry, and the basins V_k of Section 6 partition $\mathbb{R}^d$; exact ties between two logits are in any case very unlikely, and Proposition G.3 measures how far a column sits from one. The output of the head at step t is a learned value vector attached to the index k_t. The state is an index of $\lceil \log_2 d \rceil$ bits, and the only real numbers in a step are the logits and the value vectors.

G.2 THE TABLES, AND THEIR BUDGET

Proposition G.1 (Every table, at a linear merge)**.** (i) *For every $\tau \in T_d$ there are logits with $\sigma_x = \tau$, so the tables of a head range over the full transformation monoid T_d, the largest transition monoid on d indices.* (ii) *A head is scan-representable with $b = d\lceil \log_2 d \rceil$ and $c = O(d)$. This is at most a factor $\lceil \log_2 d \rceil / \log_2 d = 1 + o(1)$ above $\lceil \log_2 |T_d| \rceil = \lceil d \log_2 d \rceil$, the fewest bits that can name every element of T_d, against $\Theta(d^2)$ numbers and $\Theta(d^3)$ operations for a dense real transition matrix. Composites of tables are tables, so the scan returns exactly the sequential trajectory, whatever the shape of its tree.*

Proof. (i) Take $(\theta_x)_{jk} \coloneqq 1$ if $j = \tau(k)$ and 0 otherwise.

(ii) An element of T_d is the list $(\tau(1), \dots, \tau(d))$ of d indices of $\lceil \log_2 d \rceil$ bits each, which gives b. The composite $\tau' \circ \tau$ is obtained by the d lookups $k \mapsto \tau'(\tau(k))$, which gives c; composition is associative with the identity table as unit, so Definition 5.1 is met. Since $|T_d| = d^d$, naming every element needs $\lceil d \log_2 d \rceil \geq d \log_2 d$ bits, whence the factor. Every merge is an exact operation on integers, so the composite returned by the scan equals the sequential one, and so does every index. □

G.3 THE ONE-HOT MODEL

The one-hot model sets $\Phi_x(h) \coloneqq \mathrm{rd}(\theta_x \mathrm{rd}(h))$ on $\mathbb{R}^d$ (Section 6). Write P_x for the column one-hot matrix of σ_x, whose column k is $\mathbf{1}_{\sigma_x(k)}$; then $\Phi_x = P_x \circ \mathrm{rd}$, and, since $\mathrm{rd}(h) = \mathbf{1}_k$ on the basin V_k,

$$\Phi_x(h) = \mathbf{1}_{\sigma_x(k)} \qquad \text{for every } h \in V_k. \tag{13}$$

By induction, the trajectory of the model from $\mathbf{1}_{k_0}$ is the run equation 12 carried by its attractors, step for step. Every statement about such trajectories, including correctness at every length, is therefore a statement about the running head. The perturbations of Definition 2.1 act on the model only, and the next proposition shows that a head tolerates them in any norm.

Proposition G.2 (One head realizes any automaton on at most d states)**.** (i) *For every choice of the logits, every $x \in \Sigma$ and every index k, $\Phi_x(V_k) = \{a_{\sigma_x(k)}\}$: the head is exactly restoring on $\mathbb{R}^d$ in the sense of Section 5, with attractors a_k, basins V_k and tables σ_x.* (ii) *There is $\eta_0 > 0$, depending only on d and on the norm, such that for every $0 < \eta < \eta_0$ and every k, the ball $a_k \oplus \bar{B}_\eta$ lies in the set D_k of points whose largest coordinate is unique and equal to the k-th. In particular, these balls lie in their basins and are pairwise disjoint.* (iii) *Let $n \leq d$, number the states $Q = \{1, \dots, n\}$, and take $\iota(q) \coloneqq \mathbf{1}_q$ and $\pi(h) \coloneqq \arg\max_{q \in Q} \mathbf{1}_q^\mathsf{T} h$, as in Section 6. For $0 < \eta < \eta_0$, let $U_q \coloneqq \iota(q) \oplus \bar{B}_\eta$ and $U \coloneqq \bigsqcup_{q \in Q} U_q$. Then (R1) holds, (R2) holds whenever $\sigma_x(Q) \subseteq Q$ for every $x \in \Sigma$, and (R3) holds if and only if $\sigma_x(q) = \delta_x(q)$ for every $x \in \Sigma$ and $q \in Q$, that is, if and only if the largest entry of column q of θ_x sits in row $\delta_x(q)$, for every q and x. In that case the head realizes the target with length independence at η. Such logits exist for every automaton on $n \leq d$ states.*

Proof. (i) This is equation 13.

(ii) The set $D_k \coloneqq \{u \mid \mathbf{1}_k^\mathsf{T} u > \mathbf{1}_j^\mathsf{T} u \text{ for every } j \neq k\}$ is open, contains $\mathbf{1}_k$, and lies in V_k, and $D_k \cap D_{k'} = \emptyset$ for $k \neq k'$. Since there are finitely many indices, there is $\eta_0 > 0$ such that $\mathbf{1}_k \oplus \bar{B}_\eta \subseteq D_k$ for every k and every $\eta < \eta_0$, and the balls are disjoint because the sets D_k are.

(iii) (R1) is $\pi(\iota(q)) = \arg\max_{q' \in Q} \mathbf{1}_{q'}^\mathsf{T} \mathbf{1}_q = q$. Let $u \in U_q$ and $\|e\| \leq \eta$. By (ii), $u \in V_q$, so $\Phi_x(u) = \mathbf{1}_{\sigma_x(q)}$ by (i). If $\sigma_x(q) \in Q$, then $\Phi_x(u) + e \in U_{\sigma_x(q)} \subseteq U$; hence $\Phi_x(U) \oplus \bar{B}_\eta \subseteq U$ whenever $\sigma_x(Q) \subseteq Q$, and U is bounded: this is (R2). In that case, $\Phi_x(u) + e$ also lies in $D_{\sigma_x(q)}$ by (ii), so $\pi(\Phi_x(u) + e) = \sigma_x(q)$, and (R3) at (x, q) holds if and only if $\sigma_x(q) = \delta_x(q)$. If instead $\sigma_x(q) = k \notin Q$, every coordinate of $\mathbf{1}_k$ indexed by Q vanishes, so for any $q' \in Q$ the perturbation $e \coloneqq t\,\mathbf{1}_{q'}$, with $t > 0$ small enough that $\|e\| \leq \eta$, gives $\pi(\Phi_x(u) + e) = q'$; since $n \geq 2$, some $q' \neq \delta_x(q)$ violates (R3). Hence (R3) holds if and only if $\sigma_x = \delta_x$ on Q, which also gives $\sigma_x(Q) \subseteq Q$, hence (R2).

If $\sigma_x = \delta_x$ on Q for every x, Theorem 3.1 turns (R1) to (R3) into realization with length independence. The logits of Proposition G.1(i), with τ any table of T_d that agrees with δ_x on Q, give $\sigma_x = \delta_x$ on Q, and neither η_0 nor this choice depends on the length of the word, on Σ, or on the target. □

G.4 THE PERTURBATIONS CAN AFFECT THE LOGITS

For a running head, the per-step error $e_t = h_t - \Phi_{x_t}(h_{t-1})$ of Appendix D, read through the one-hot model, takes only two kinds of values. If the table of step t is computed correctly, $e_t = 0$ exactly, since the state is an index and a step is a gather. Otherwise e_t is the difference of two distinct attractors, a wrong table entry rather than a small deviation. What finite precision, quantized weights, or an imperfectly learned map can perturb is the logit matrix θ_{x_t}, which is computed from the input alone, and that perturbation changes the table only if it reorders a column.

Proposition G.3 (Logit margin)**.** *Let $\theta \in \mathbb{R}^{d\times d}$ have a unique largest entry in every column, with table σ, and write*

$$\gamma(\theta) \coloneqq \min_k \Big(\theta_{\sigma(k)k} - \max_{j\neq\sigma(k)} \theta_{jk}\Big) > 0$$

for its smallest column margin. If $\max_{j,k} |\Delta_{jk}| < \gamma(\theta)/2$, then $\theta + \Delta$ has the same table σ. Consequently, if at every step every entry of θ_{x_t} is perturbed by less than $\gamma(\theta_{x_t})/2$, the executed tables and indices equal the exact ones at every step, for every word.

Proof. Fix a column k and $j \neq \sigma(k)$, and write $\gamma \coloneqq \gamma(\theta)$. Then

$$(\theta+\Delta)_{\sigma(k)k} > \theta_{\sigma(k)k} - \tfrac{\gamma}{2} \geq \theta_{jk} + \tfrac{\gamma}{2} > (\theta+\Delta)_{jk},$$

so the largest entry of column k is still in row $\sigma(k)$. For the second claim, argue by induction on t: both runs start from k_0, and if they share k_{t-1} they share the table at step t, hence k_t by equation 12. Nothing is carried from one step to the next, because the logits read x_t and not the state. □

G.5 MANY HEADS

The block holds r heads: head i has its own attractor indices $\{1,\dots,d_i\}$, its own logits $\theta^{(i)}_{x_t} \in \mathbb{R}^{d_i\times d_i}$ computed from the same input, and its own $\arg\max$. The heads update independently, so the joint index, the tuple of the indices of the heads, lies in $\prod_i\{1,\dots,d_i\}$, and the budgets add:

$$b = \sum_{i=1}^{r} d_i\lceil\log_2 d_i\rceil, \qquad c = O\Big(\sum_{i=1}^{r} d_i\Big), \qquad \text{state} = \sum_{i=1}^{r}\lceil\log_2 d_i\rceil \text{ bits}, \qquad \text{logits} = \sum_{i=1}^{r} d_i^2. \tag{14}$$

Each index of head i owns a value vector, head i emits the vector of the index it occupies, the r vectors are concatenated, and one shared projection maps them back to the model width.

The block step applies the heads side by side, $\Phi_x(h) = \big(\Phi^{(1)}_x(h^{(1)}),\dots,\Phi^{(r)}_x(h^{(r)})\big)$ on $\mathcal{H} = \prod_i \mathbb{R}^{d_i}$. By Proposition G.2(i) applied to each head, it is exactly restoring in the sense of Section 5 for every value of the logits: its attractors are the tuples of attractors of the heads, indexed by the joint indices $(k^{(1)},\dots,k^{(r)})$, and the tables of the heads act on the joint indices component by component. Realization maps the automaton states injectively into the joint indices by $\chi = (\chi_1,\dots,\chi_r)$, with $\iota(q) = \big(\mathbf{1}_{\chi_1(q)},\dots,\mathbf{1}_{\chi_r(q)}\big)$, and π reads in every head the index of the basin that contains $h^{(i)}$, and inverts χ. Heads that update independently can only carry a target whose transitions act on each head separately.

Proposition G.4 (Heads combine as a direct product)**.** (i) *The joint tables of r heads form a submonoid of the direct product $\prod_i T_{d_i}$, within the budget equation 14.* (ii) *Let $\chi = (\chi_1,\dots,\chi_r) : Q \hookrightarrow \prod_i\{1,\dots,d_i\}$ be injective, with ι and π as above. If $\sigma^{(i)}_x \circ \chi_i = \chi_i \circ \delta_x$ for every $x \in \Sigma$ and every i, the block realizes the target with length independence at every η below some $\eta_0 > 0$. Such tables exist if and only if every symbol acts head by head: $\chi_i(q) = \chi_i(q')$ implies $\chi_i(\delta_x(q)) = \chi_i(\delta_x(q'))$, for every x, i, q and q'.*

Proof. (i) The table of each head depends on the input alone, so the joint step, and every composite of joint steps, acts head by head; the budget is Proposition G.1(ii) summed over heads.

(ii) The set of points whose $\arg\max$ in each head is unique and spells $\chi(q)$ is open, contains $\iota(q)$, and does not meet the corresponding set of any $q' \neq q$, because χ is injective. As in the proof of Proposition G.2, some $\eta_0 > 0$ keeps every ball of radius $\eta < \eta_0$ around a code point inside its set, and head i maps that ball onto $\mathbf{1}_{\sigma^{(i)}_x(\chi_i(q))} = \mathbf{1}_{\chi_i(\delta_x(q))}$, so a step maps the cell of q onto

the code point of $\delta_x(q)$. Hence (R1) to (R3) hold on the union of these balls, which is bounded, and Theorem 3.1 gives realization with length independence. For the last claim, the condition is necessary, since $\chi_i(\delta_x(q)) = \sigma_x^{(i)}(\chi_i(q))$ depends on q through $\chi_i(q)$ only. It is sufficient, since it defines $\sigma_x^{(i)}$ on $\chi_i(Q)$ by $\sigma_x^{(i)}(\chi_i(q)) \coloneqq \chi_i(\delta_x(q))$; any extension to the other indices is a table, attainable by Proposition G.1(i). □

Remark G.5 (Direct products within a block, cascades across layers)**.** *Proposition G.4 covers heads that run side by side, none reading the index of another. In a cascade, as in the wreath products of the Krohn–Rhodes decomposition (Krohn & Rhodes, 1965), the table of one component depends on the current index of another. Inside a single block, a cascade is still a table on the joint indices, and hence still scannable, but only at the budget of that joint space of $\prod_i d_i$ indices, not at equation 14. Stacking layers builds a cascade without that cost. The logits of a layer are computed from the output of the layer below, which is the readout of its current index, so the tables of the upper layer depend on the index of the lower one, while each layer is still scanned at its own budget, one layer after the other. One difference with the cascades of Krohn–Rhodes matters. There, a component is updated from the* previous *state of the components below it and the symbol; here, a layer reads the* updated *index of the layer below at the same step. The two coincide when the lower transition is injective, as for groups, since the previous lower index is then determined by the updated one and the symbol. They differ when it is not: after a reset, the updated index no longer tells which index the lower layer came from. A causal convolution of width 2 before each layer restores this information, since it lets a layer read the output of the layer below at the previous step as well. This is how the stack of NFSM layers operates on TSO (Section 7).*

Proposition G.6 (Stacked layers realize cascades)**.** *Consider a stack of s NFSM layers, layer ℓ having joint indices $S_\ell = \prod_i \{1, \dots, d_i^{(\ell)}\}$ over its heads, and suppose that the logits of layer ℓ at step t are computed from the symbol x_t and from the indices of the layers below it at the same step.* (i) *The joint index $(k_t^{(1)}, \dots, k_t^{(s)})$ follows a table on $\prod_\ell S_\ell$ that depends on the symbol alone, so the stack runs on indices, and every composite of its steps is again such a table.* (ii) *Suppose that the logits of each layer can be chosen as any function of the symbol and of the indices of the layers below. Let $\chi = (\chi_1, \dots, \chi_s) : Q \hookrightarrow \prod_\ell S_\ell$ be injective, write $\chi_\ell = (\chi_{\ell,1}, \dots, \chi_{\ell,r_\ell})$ for its components over the r_ℓ heads of layer ℓ, and $\chi_{<\ell} \coloneqq (\chi_1, \dots, \chi_{\ell-1})$. Suppose that every symbol acts on each head from that head and the updated layers below it: for every x, ℓ and i there is $\tau_x^{(\ell,i)}$ with*

$$\chi_{\ell,i}(\delta_x(q)) = \tau_x^{(\ell,i)}\big(\chi_{<\ell}(\delta_x(q)),\, \chi_{\ell,i}(q)\big) \qquad \text{for every } q \in Q.$$

Then there are logits for which the stack realizes the target with length independence at every η below some $\eta_0 > 0$. (iii) *Layer ℓ is scanned once the indices of the layers below are known, so the layers are scanned one after the other, each at its own budget equation 14, for a total depth $O(s \log L)$.*

Proof. *(i)* Fix x. The table of layer 1 depends on x alone, so $k_t^{(1)}$ is a function of $(x, k_{t-1}^{(1)})$; inductively, the logits of layer ℓ are a function of x and of $k_t^{(1)}, \dots, k_t^{(\ell-1)}$, themselves functions of x and of the previous joint index, so $k_t^{(\ell)}$ is a function of x and of the previous joint index. The joint step is therefore an element of $T_{|\prod_\ell S_\ell|}$ indexed by x, and composites of tables are tables (Proposition G.1). *(ii)* For every symbol x, layer ℓ, head i and tuple κ of indices of the layers below, choose logits whose table $\sigma_{x,\kappa}^{(\ell,i)}$ agrees with $\tau_x^{(\ell,i)}(\kappa, \cdot)$ wherever the latter is defined and is arbitrary elsewhere; such logits exist by Proposition G.1(i), and the choice is allowed because the logits may depend arbitrarily on (x, κ). In the one-hot model on $\prod_\ell \prod_i \mathbb{R}^{d_i^{(\ell)}}$, encode q by the tuple of attractors of $\chi(q)$ and decode a point whose $\arg\max$ in each head spells $\chi(q)$ as q. As in the proof of Proposition G.4, these sets are open, pairwise disjoint because χ is injective, and some $\eta_0 > 0$ keeps every ball of radius $\eta < \eta_0$ around a code point inside its set. Let u lie in the ball around $\iota(q)$. Every head of layer 1 reads its index $\chi_{1,i}(q)$ and moves it to $\tau_x^{(1,i)}(\chi_{1,i}(q)) = \chi_{1,i}(\delta_x(q))$. Inductively, once the layers below ℓ have moved to $\chi_{<\ell}(\delta_x(q))$, the heads of layer ℓ read this tuple, hence apply the tables $\tau_x^{(\ell,i)}(\chi_{<\ell}(\delta_x(q)), \cdot)$, and move from $\chi_{\ell,i}(q)$ to $\chi_{\ell,i}(\delta_x(q))$. The joint step therefore maps the cell of q onto the code point of $\delta_x(q)$. Hence (R1) to (R3) hold on the union of these balls, which is bounded, and Theorem 3.1 gives realization with length independence. *(iii)* The tables of layer ℓ are determined once its inputs are, which are the symbols and the indices of the layers below; scanning

the layers in order therefore costs s scans, each of depth $O(\log L)$ and at the budget equation 14 of its layer. □

Example G.7 (A running product in S_n)**.** *Let the state be the running product $g_1 \cdots g_t$ of permutations of $\{1, \ldots, n\}$, so $|Q| = n!$, and one head on Q costs $b = n!\lceil \log_2 n! \rceil$ bits and $(n!)^2$ logits. Encode the product instead by the images of the points $1, \ldots, n-1$, $\chi_i(g) \coloneqq i \cdot g$, with one head of n indices per point: each head is a channel that carries where its point has been sent, and the $n-1$ channels together carry the product. A permutation is fixed by its values on $n-1$ points, so χ is injective, and reading g_t sends every head from index j to $j \cdot g_t$, the same table for every head. Proposition G.4(ii) applies with $r = n-1$ heads that may share one logit matrix: $b = (n-1)\, n\lceil \log_2 n \rceil$ bits and n^2 logits. For $n = 5$ this is 60 bits and 25 logits, against 840 bits and 14400 logits for one head of 120 indices.*

G.6 The gradient estimator

The $\arg\max$ in equation 12 has zero derivative wherever it is defined, so the logits receive no gradient as written. We train with the straight-through Gumbel-softmax estimator (Jang et al., 2017), and the forward pass stays on indices throughout. We describe one head.

Forward pass. During training, the logits are scaled by a sampling temperature ν, $u_{x_t} \coloneqq \theta_{x_t}/\nu$, one independent standard Gumbel variable $(G_t)_{jk}$ is drawn per entry, and the table is $\sigma_t(k) \coloneqq \arg\max_j (u_{x_t} + G_t)_{jk}$. At test time the Gumbel variables are removed and the table is σ_{x_t} of equation 12. A sampled table is still an element of T_d, so parts (i) and (ii) of Proposition G.2 hold at every training step.

Backward pass. Let $\bar{g}_t(k)$ be the first-order effect on the loss of reading out index k at step t, obtained from the value vectors. The effect of being at index k at step t, counting all later steps, is the adjoint

$$\lambda_t(k) = \bar{g}_t(k) + \lambda_{t+1}\big(\sigma_{t+1}(k)\big),$$

which is backpropagation through $h_t = P_t h_{t-1}$, the one-hot model on its attractors, with P_t the column one-hot matrix of σ_t; it is computed with gathers, itself by a reverse parallel scan. At step t, only the column $k = k_{t-1}$ that the run actually used receives a gradient: with λ_t read as the vector $(\lambda_t(1), \ldots, \lambda_t(d))$ and $p \coloneqq \mathrm{softmax}\big((u_{x_t} + G_t)\mathbf{1}_k\big)$, the surrogate gradient with respect to that column of u_{x_t} is $p \odot (\lambda_t - \langle p, \lambda_t \rangle)$, the softmax Jacobian applied to λ_t, and dividing by ν gives the gradient with respect to θ_{x_t}. The one-hot model is thus what the backward pass differentiates, although no one-hot vector is ever formed.

We think that a custom kernel that implements the forward and backward passes of this estimator, with the parallel scan for the adjoint, could drastically reduce the memory and time cost of training. One observation is that the block itself is not compute-bound but rather bandwidth-bound. We left such optimization for future work. Another direction could draw on the dictionary of PD-SSM (Appendix H.2) to reduce the memory footprint. Instead of computing all d^2 logits of θ_x from the input, a head would learn n_Θ logit matrices $\Theta_1, \ldots, \Theta_{n_\Theta}$ and compute only the mixture weights $\omega_x \in \mathbb{R}^{n_\Theta}$, with $\theta_x = \sum_\ell (\omega_x)_\ell \Theta_\ell$. The projection would then have $O(mn_\Theta + n_\Theta d^2)$ parameters instead of $O(md^2)$, and the backward pass would store n_Θ weights per step rather than d^2 logits, recomputing only the column it uses. The table remains the column-wise $\arg\max$ of θ_x, so Proposition G.2 still applies. The current implementation is available following the interactive notebook whose URL is in the introduction.

H Experiments

H.1 Tasks

Every task is a semiautomaton (Q, Σ, δ) with a start state q_0. A sequence is a random word $x_1 \cdots x_L$, and the target at every position t is the state $\delta_{x_1 \cdots x_t}(q_0)$. Targets are computed exactly, by a scan over the tables of δ.

Algebraic tasks. Each algebraic task is the word problem of a finite monoid M with generators Σ. The states are the elements of M, q_0 is the identity, and a letter x maps q to $x \circ q$, so the target is the

running composite of the letters. Letters are drawn uniformly unless stated otherwise. Table 4 lists the tasks.

Table 4: Algebraic tasks. $|Q|$ is the number of reachable states, the order of the monoid.

Task	Monoid	Letters	$\lvert\Sigma\rvert$	$\lvert Q\rvert$
$\mathbb{Z}_2$	cyclic group	residues $0, 1$	2	2
$\mathbb{Z}_{16}$	cyclic group	residues $0, \ldots, 15$	16	16
S_3	symmetric group	adjacent transpositions $(i\ i{+}1)$	2	6
S_4	symmetric group	adjacent transpositions $(i\ i{+}1)$	3	24
A_5	alternating group	adjacent 3-cycles $(i\ i{+}1\ i{+}2)$	3	60
M_{11}	Mathieu group	a, b, a^{-1}, b^{-1}	4	7920
DFF_5	flip-flop monoid	identity, reset, set	3	3
FF	flip-flop monoid	identity, reset, set	3	3

The Mathieu group. M_{11} is generated by two permutations of $\{0, \ldots, 10\}$,

$$a = (0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10), \qquad b = (2\ 6\ 10\ 7)(3\ 9\ 4\ 5),$$

and has order 7920. Like A_5, it is simple and nonsolvable.

The flip-flops. The flip-flop monoid acts on $\{0, 1\}$ through three maps: the identity, reset (constant 0) and set (constant 1). The target is the identity until the first reset or set, and then the last of them. The two tasks differ only in how letters are drawn. In DFF_5, letters are uniform, except that four identities in a row are always followed by a reset or a set; the target is then a function of the last five letters, and the drawn words are definite. In FF, a letter is a reset or a set with probability 0.1, and the identity otherwise. Runs of identities have unbounded length, with mean 9, so the target depends on arbitrarily old letters and the drawn words are not. The length sweep draws FF with an identity probability of 0.99 instead, so that runs of identities have mean length 99. The test sequences are then long and their information sparse: the target must be held across long runs of identities, which strengthens the dependence on old letters.

Tracking Shuffled Objects. TSO_N is adapted from the task of Srivastava et al. (2023). It involves $N \in \{3, 4, 5\}$ people (alice, bob, charlie, dana, eve) and as many items (ring, key, coin, stamp, card). Every word and punctuation mark is one token, for a vocabulary of $2N + 7$ tokens. A sequence is made of sentences of five tokens, in three parts:

1. a preamble of N sentences, one per person in order, that assigns the items at random: *alice has the key .*;
2. swap sentences, each between a uniformly drawn pair of distinct people: *bob swaps with alice .*;
3. a question about a uniformly drawn item, *who has the key ?*, whose answer, the current holder of that item, is predicted at the last token.

When the length is not a multiple of 5, extra "." tokens are inserted after randomly chosen sentences, at most one per sentence, so that every length is built from the same local patterns.

The underlying automaton reads one token at a time. Its state is a pair (p, g): p is either idle or the first name of the sentence in progress, and g maps every item to its holder, or to "unset" before the preamble names it. A name read in the idle state is stored in p. A second name swaps the items of the two people and makes p idle again. An item closes a sentence p *has the item*, assigning it to p. Every other token leaves the state unchanged, so the question does not modify it. From the start state, with p idle and every item unset, the grammar reaches 44, 202 and 1132 states for $N = 3, 4, 5$. The target at every token is this state, and the answer at the last token is a function of the state and of the asked item.

Answering the question is not state tracking in itself: the answer depends on the asked item, which is not part of the state. The last token is "?" and the item is the token before it, so the causal convolution before a higher layer lets its update at "?" depend on the item, and the final MLP reads the answer off the states of all layers. The question thus tests whether the tracked state can be used, not only whether it is kept.

H.2 BASELINES

All three baselines are affine in the state: $h_t = A_t h_{t-1} + b_t$, where A_t and b_t depend only on the input u_t of the layer. A parallel scan evaluates them with the merge $(A_2, b_2) \circ (A_1, b_1) = (A_2 A_1,\ A_2 b_1 + b_2)$, and each baseline keeps A_t in a class closed under products, so that a merge stays cheap. Each takes the place of the NFSM in the same backbone (Appendix H.3). The class of A_t is what sets them apart, and it places them on different branches of Theorem 4.1.

Mamba$^-$. This is the Mamba block (Gu & Dao, 2024), with the eigenvalues extended to negative values (Grazzi et al., 2025). The inputs $u_t \in \mathbb{R}^d$ and $z_t \in \mathbb{R}^d$ are two linear projections of the layer input, u_t followed by a SiLU. The state has d channels of d_{state} entries, and each entry evolves on its own:

$$h_t[i,j] = a_t[i,j]\, h_{t-1}[i,j] + \Delta_t[i]\, B_t[j]\, u_t[i], \qquad a_t[i,j] = 2\exp\big(-\Delta_t[i]\, e^{\alpha[i,j]}\big) - 1,$$

with step sizes $\Delta_t = \text{softplus}(W_\Delta u_t + b_\Delta)$, vectors $B_t, C_t \in \mathbb{R}^{d_{\text{state}}}$ linear in u_t, and learned $\alpha \in \mathbb{R}^{d \times d_{\text{state}}}$. The output is $y_t[i] = \big(\sum_j C_t[j]\, h_t[i,j] + D[i]\, u_t[i]\big)\, \text{SiLU}(z_t[i])$, followed by a linear projection. Since $\Delta_t > 0$, every $a_t[i,j]$ lies in $(-1, 1)$, so Mamba$^-$ is on the contracting branch, $\rho < 1$. We remove the causal convolution of the original block, so that all models receive the same input.

AUSSM. The adaptive unitary SSM (Karuvally et al., 2025) uses the same block with a complex state and eigenvalues of unit modulus:

$$h_t[i,j] = e^{\mathrm{i}\,\phi_t[i,j]}\, h_{t-1}[i,j] + \Delta_t[i]\, B[j]\, u_t[i],$$

where the phases $\phi_t \in \mathbb{R}^{d \times d_{\text{state}}}$ are linear in u_t, and $B, C \in \mathbb{C}^{d_{\text{state}}}$ are learned but do not depend on the input. The readout takes the real part, $\text{Re} \sum_j C[j]\, h_t[i,j]$, and the rest of the block is as for Mamba$^-$. Every eigenvalue has modulus 1, so AUSSM is on the isometric branch, $\rho = 1$.

PD-SSM. PD-SSM (Terzić et al., 2025) has a complex state $h_t \in \mathbb{C}^N$ and the transition

$$h_t = P_t D_t\, h_{t-1} + b_t, \qquad M_t = \sum_{k=1}^{n_M} \text{softmax}(W_\pi u_t)[k]\, M_k.$$

Here P_t is the column-wise hardmax of M_t, a mixture of n_M learned matrices $M_k \in \mathbb{R}^{N \times N}$, so each column of P_t is one-hot. D_t is diagonal with entries $r_t[j]\, e^{\mathrm{i}\,\psi_t[j]}$, whose moduli $r_t \in (0,1)^N$ and phases $\psi_t \in (0, 2\pi)^N$ are the sigmoid outputs of two small MLPs of u_t, the second scaled by 2π. The drive b_t is linear in u_t, and the output is linear in $(\text{Re}\, h_t, \text{Im}\, h_t)$ with a skip connection from u_t. A product of two such matrices PD is again of this form, and a merge costs $O(N)$ operations. The hardmax is nonlinear, but it acts on the input and not on the state: it selects the matrix $A_t = P_t D_t$, and the step remains affine in h_{t-1}. It does not restore the state. Each column of $P_t D_t$ has a single nonzero entry, of modulus less than 1, so $\rho(P_t D_t) < 1$ and PD-SSM is on the contracting branch. The dictionary is trained through the softmax surrogate gradient of Terzić et al. (2025).

H.3 BACKBONE, TRAINING AND EVALUATION

Backbone. All models share one backbone of width $m = 128$ and differ only in the recurrent block. Tokens are embedded, passed through an MLP and a layer norm. Each layer then applies $h \leftarrow h + \text{block}(\text{conv}(\text{LN}(h)))$ followed by $h \leftarrow h + \text{MLP}(\text{LN}(h))$, and a final layer norm, MLP and linear map give the logits. Every MLP is a gated linear unit with SiLU gate, hidden width $2m$ and dropout 0.1. On TSO, conv is a causal depthwise convolution of width 2 followed by a SiLU. On the algebraic tasks there is no convolution, so each block reads the current token alone. The baselines have four layers with a recurrent width of 256: $d = 256$ channels with $d_{\text{state}} = 16$ for Mamba$^-$ and AUSSM, and $N = 256$ with $n_M = 16$ dictionary matrices for PD-SSM. The NFSM has one layer on the algebraic tasks and two layers on TSO. On the algebraic tasks, the NFSM has a single head with one attractor index per element of the target for $\mathbb{Z}_2$, $\mathbb{Z}_{16}$, DFF_5 and FF, and otherwise several heads that each track a quotient of the target and together separate its elements, that is, a map χ as in Appendix G.5: $(3, 2)$ attractor indices for S_3, $(4, 3, 2)$ for S_4, $(5, 12)$ for A_5, and 4 heads of 11 attractor indices for M_{11}, which track the images of four points. On TSO_N, the first layer has $N + 1$ heads and the second layer N heads, each of $N + 1$ attractor indices.

Loss. The loss is a cross-entropy on the state at every position, and on the answer at the last position for TSO. Two weights rescale the per-position terms. The first weight is a mask after the first error. For each training sequence, let $t^\star$ be the first position where the predicted state is wrong. Positions up to $t^\star$ have weight 1, and later positions have weight $e^{-\lambda(t-t^\star)}$ with $\lambda = 3$. Once the state is wrong, the later predictions start from a wrong state, and their errors are consequences of the error at $t^\star$ rather than errors of their own. Supervising them at full weight would ask the model to reach correct states from a wrong one, which pulls the transitions away from the target. The mask concentrates the gradient on the first error, where a transition is actually wrong, and it moves along the sequence as training fixes earlier errors. The second weight is class balancing (especially for the DFF/FF and TSO). Each sequence contributes the weighted mean of its terms, and the loss is the mean over the batch. For the NFSM, a third term keeps the sampling exploratory at the first error. The Gumbel noise of Appendix G.6 tries other entries of a column only while the logits of that column are not much more spread than the noise. Once they are, the sampled entry is always the $\arg\max$, and a wrong entry is no longer revisited. At the first error of each sequence, we therefore penalize the spread of the columns used there, beyond the standard deviation $\sigma_G = \pi/\sqrt{6}$ of the Gumbel noise:

$$\mathcal{E}^{(b)} = \frac{1}{r}\sum_{i=1}^{r} \mathrm{relu}\left(\frac{s\big(u^{(i)}_{x_{t^\star}}\mathbf{1}_{k^{(i)}_{t^\star-1}}\big)}{\sigma_G} - 1\right)^2,$$

where s is the standard deviation over the entries of a column, $u^{(i)} = \theta^{(i)}/\nu$ are the sampling logits of head i, and $k^{(i)}_{t^\star-1}$ is its attractor index before the step, treated as a constant. $\mathcal{E}^{(b)} = 0$ for a sequence without error, and with several layers the term is averaged over layers. The full loss is

$$\mathcal{L} = \frac{1}{B}\sum_{b=1}^{B}\left[\frac{\sum_t w_t^{(b)}\,\mathrm{CE}\big(q_t^{(b)}\big)}{\sum_t w_t^{(b)}} + \mathrm{CE}\big(a^{(b)}\big) + \beta_s\,\mathcal{E}^{(b)}\right], \qquad w_t^{(b)} = e^{-\lambda(t-t^\star)_+}\,\kappa_{q_t^{(b)}},$$

where $\mathrm{CE}(q_t)$ is the cross-entropy of the state logits at position t, $\mathrm{CE}(a)$ that of the answer logits at the last position, on TSO only, and κ_q the class-balancing weight of state q. The coefficient $\beta_s = 0.03$ decays by a cosine to 0.003 over the training steps s. The baselines use the same loss without $\mathcal{E}$.

Exploring the most effective learning strategy for the NFSM is beyond the scope of this work. Especially, we think that one could explore specific methods based on the finite-state nature of the model, for example by taking inspiratio from Bengio & Frasconi (1996).

Optimization. We use AdamW (Loshchilov & Hutter, 2019) with learning rate 10^{-3}, cosine-decayed to 10^{-4}, weight decay 10^{-4} and gradient clipping at norm 1. The NFSM is trained with the estimator of Appendix G.6 at temperature $\nu = 0.5$. Batches contain 256 sequences of length 64 on the algebraic tasks, and 64 sequences of length 256 on TSO. Every 50 steps we measure the sequence accuracy on 512 fresh sequences (128 for M_{11}). Training stops after 40 consecutive checks at accuracy 1 for the NFSM and 0.99 for the baselines, or after 10^5 steps, and we keep the checkpoint with the best validation accuracy.

Evaluation. Models are evaluated without dropout and, for the NFSM, without Gumbel noise. The predicted state at a position is the $\arg\max$ of the state logits. The *sequence accuracy* is the fraction of sequences whose predicted states are correct at every position:

$$\mathrm{acc}_{\mathrm{seq}} = \frac{1}{B}\sum_{b=1}^{B}\prod_{t=1}^{L}\mathbb{I}\big[\hat{q}_t^{(b)} = q_t^{(b)}\big],$$

where $q_t^{(b)}$ is the target state of sequence b at position t, and $\hat{q}_t^{(b)}$ the prediction. A single wrong position makes the whole sequence count as wrong, so the measure does not dilute a late error over many correct positions. On TSO, the model also predicts, at the last token, which person holds the asked item. The *question accuracy* is the fraction of sequences whose predicted holder is correct:

$$\mathrm{acc}_{\mathrm{qst}} = \frac{1}{B}\sum_{b=1}^{B}\mathbb{I}\big[\hat{a}^{(b)} = a^{(b)}\big],$$

with $a^{(b)}$ the true holder and $\hat{a}^{(b)}$ the $\arg\max$ of the answer logits. It scores the task as a reader would: one answer per sequence, after a stream of arbitrary length. We report the sequence accuracy

on the algebraic tasks and the question accuracy on TSO. The failing length $L_{\max}$ is the smallest swept length at which this accuracy falls below 1 for the NFSM and below 0.9 for the baselines.

H.4 Seed-to-seed variability

Table 5 complements Table 1 with the failing length of each of the five seeds. Where the baselines learn, their failing length varies across seeds.

H.5 Geometry of the RNN states of the affine baselines

Theorem 4.1 is a statement about the state of a trained model, so we look at it. Figure 4 follows the recurrent state on $\mathbb{Z}_5$, a task we train separately for this figure, driven by the constant word $(+1)^T$, on which the five target states must stay apart forever. For each model, the state of the last recurrent layer is projected on its first two principal components, computed on that run alone, one point per step and colored by the target state at that step; the shaded regions are the decision regions of the classifier at the end of the backbone, read in the same plane, so a point is decoded correctly when its color matches its region. The three columns are three windows of the same run, at increasing time.

The NFSM keeps five points, one per state, and the picture does not change from one window to the next: the run visits attractors, and restoration puts the state back on one of them at every step (Proposition G.2). The three affine baselines all lose this. Mamba$^-$ contracts: the five clusters collapse onto a single point, which lands on a cell boundary, so after roughly one hundred steps the readout no longer separates the states. This is the $\rho < 1$ branch, where forgetting the input also forgets the state. AUSSM shows the $\rho = 1$ branch: the clusters do not collapse onto one point, but they stop being separated, and the states that started apart end up interleaved within the same few cells.

H.6 Extraction of the transition tables

Because every NFSM step commits to a table entry, the tables a trained model executes can be read off its logits, and compared with the target entry by entry. This is a strength of the architecture: it turns the evaluation of length independence into a verification. Testing up to a length L only shows that the model is correct up to L, whereas tables that match the target show, by Proposition G.2, that it is correct on every word of every length. An affine recurrence offers nothing comparable, since it has no finite table to read.

H.6.1 Method

The procedure applies when the input alphabet Σ is finite, and uses the model at test time, without Gumbel noise.

One layer, algebraic tasks. For each symbol $x \in \Sigma$, we pass x alone through the backbone up to the NFSM layer and read the logits $\theta_x^{(i)}$ of every head i. The column-wise $\arg\max$ gives the table $\sigma_x^{(i)} \in T_{d_i}$, that is, the state that head i moves to from each of its d_i states when it reads x. Repeating this for all $|\Sigma|$ symbols gives every transition of the layer.

Convolutions and stacks. With a causal convolution of width k before each layer and several layers, as on TSO, the logits of a layer at step t depend on a *window* rather than on the current symbol: the last k tokens for the first layer, and for a higher layer the last k positions of the residual stream, which are functions of the tokens and of the states of the layers below (Remark G.5). Every window takes values in a finite set, so the method extends in principle: the tables of each layer can be extracted for every window that can occur, from the bottom layer upwards. A causal convolution only remembers its last $k-1$ inputs, a finite amount of memory, so the stack remains a finite-state machine and its tables still describe it completely. The windows that can occur grow quickly with the number of layers and of objects, however, and we do not carry out this enumeration on TSO. Appendix H.6.3 instead inspects what the trained stack computes.

Matching with the target. The tables are indexed by the attractor indices of the heads, not by the states of the task. To relate the two, we explore the joint indices that the tables reach from the initial joint index s_0 of the model, and assign a task state to each of them: q_0 to s_0, and, whenever σ_x

sends a joint index assigned q to s', $\delta_x(q)$ to s'. If no joint index is assigned two task states, this defines a map c from the reached joint indices to the task states, which satisfies $c \circ \sigma_x = \delta_x \circ c$ by construction; when c is a bijection, its inverse is the map χ of Appendix G.5. We check that no joint index is assigned two task states, and that for every reached joint index s and every symbol x, the readout of $\sigma_x(s)$, after reading x, predicts $\delta_x(c(s))$. The model is then correct at every length: by induction on the length of the word, the joint index after reading w is $\sigma_w(s_0)$, its task state is $\delta_w(q_0)$, and its readout predicts this state. The exploration covers every reachable joint index and every symbol, so the check is exhaustive rather than sampled.

Beyond finite alphabets. When the inputs do not come from a finite alphabet, as with continuous features, the enumeration above is no longer possible. The state space is still finite, however, so a head can execute at most $d_i^{d_i}$ distinct tables, and the input space splits into finitely many regions on each of which the table is constant. Extracting the model then amounts to describing these regions, for example as conditions on the input such as membership in an extracted interval. We leave this to future work.

H.6.2 Algebraic tasks

We apply the method to the models of Table 1, for every algebraic task and all five seeds (Table 6). Every model passes the check: no joint index is assigned two task states, and the readout predicts the assigned state on every reached transition. By Proposition G.2, every model is therefore correct at every length, and the failing lengths $\geq L$ of Table 1 are certified beyond the tested lengths.

The exploration reaches exactly one joint index per task state, so c is a bijection between the reached joint indices and the reachable states of the task, and its inverse is the map χ of Appendix G.5: the heads store the task state without redundancy. Every attractor index of every head is reached, and the joint indices that are never reached are exactly the combinations that the task cannot produce. On M_{11}, for instance, the 4 heads of 11 attractor indices reach 7920 of the $11^4 = 14641$ joint indices, one for each element of the group. Single-head models of S_3, S_4 and A_5, with one head of 6, 24 and 60 attractor indices, pass the same check on all five seeds.

An example on S_3. The extracted machines are small enough to read. Take the model of S_3 with seed 42, whose alphabet has the two transpositions $x_1 = (1\,2)$ and $x_2 = (2\,3)$ of $\{1, 2, 3\}$. Its first head has 3 states and its second head 2:

$$\sigma^{(1)}_{x_1} = (2, 1, 3), \quad \sigma^{(1)}_{x_2} = (1, 3, 2), \qquad \sigma^{(2)}_{x_1} = \sigma^{(2)}_{x_2} = (2, 1),$$

where each table lists the images of the attractor indices $1, 2, \ldots$ in order. The first head applies each transposition to its index, and the second head flips at every symbol. The map c sends the joint index (j, k) to the permutation g with $g(1) = j$ and sign $(-1)^{k-1}$, which is unique in S_3. The first head therefore tracks the image of the point 1 and the second head the sign of the permutation, and the two together determine it. This is the factorization announced in Appendix H.3: each head tracks a quotient of the target, and together the heads separate its elements.

H.6.3 Tracking Shuffled Objects

On TSO, we do not certify the model by extraction (Appendix H.6.1). We instead show what a trained stack of two layers computes on TSO_4. Figure 5 follows the attractor index of every head, in both layers, as the model reads one stream. To interpret the heads, we map each attractor index of a head, over many streams, to the value it determines of a register of the task: the first name of the sentence in progress, or the holder of an item.

The resulting interpretation is clear for the second layer. Each of its heads tracks the holder of one item exactly. It is unset until the preamble names its item, takes the holder at the item token, and then changes only at the second name of a swap that involves this holder. The question leaves it unchanged, so at the last token the head of the asked item holds the answer. The heads of the first layer have no clear interpretation when taken separately: none of them matches a single register. Together, however, they must build a parser. No head of the second layer tracks the first name of the sentence in progress, yet the second layer updates the holders at the right tokens, and the readout predicts the first name at every position. The first layer must therefore pass this information to the second layer, along with the current token.

H.7 WALL-CLOCK MEASUREMENTS

Sections 5 and 6 claim that the NFSM runs under a parallel scan whose merge is a gather (Proposition G.1). We measure the cost of one layer of the NFSM and of the baselines of Section 7, Mamba$^-$, AUSSM and PD-SSM, against the sequence length L and the state size d (Figure 6).

Setup. Each layer is timed alone: the embedding, the causal convolution of the backbone and the MLP are the same for all models and are left out. The NFSM has one head of d attractor indices, Mamba$^-$ and AUSSM have d channels with $d_{\text{state}} = 16$, and PD-SSM has $N = d$. We vary L from 2^6 to 2^{17} at $d = 16$, and d from 2 to 256 at $L = 2^{12}$. The inputs are batches of 16 sequences of Gaussian vectors of width 256, in float32 (with complex64 states for AUSSM and PD-SSM). For each model, we compare the parallel scan with a sequential loop over time steps that computes the same layer. We time a training step, that is, the forward pass and the gradient of the mean squared output with respect to the parameters, including the Gumbel noise and surrogate gradient of the NFSM and the surrogate of PD-SSM. Every configuration runs on one NVIDIA A100 (80 GB). The step is compiled once, called 5 times to warm up, and timed over 10 synchronized calls. We report the median; compilation time is not included.

Chunked runs. By default, a layer processes the whole sequence at once. When a configuration does not fit in memory this way, we run it *chunked*: the sequence is split into consecutive blocks of C steps, the layer processes one block at a time and passes its last state on to the next block, and the backward pass recomputes the intermediate quantities of each block instead of storing them for the whole sequence. The result is unchanged up to round-off (which is, once again, another source of perturbation introduced by an implementation choice), and memory is bounded by the size of a block instead of by L, but the blocks are processed one after the other, which costs time. They are circled in red in Figure 6. We emphasize that the chunked approach can be used to fit any configuration in memory, at the cost of time, and that the parallel scan is still used within each block. It was in practice used interchangeably with the unchunked approach during the training of the models depending on the available memory.

Memory. The peak memory in Figure 6 (bottom) is approximate and should be read with care. The panel shows orders of magnitude and trends, not exact footprints.

Results. *Against the length.* The parallel scan is faster than the sequential loop at every measured length, and the gap widens with L. At $d = 16$, the NFSM is also the fastest of the four at every length. *Against the state size.* At $L = 2^{12}$, the NFSM is the fastest model from $d = 4$ to $d = 64$. It grows faster with d than the two diagonal models, since it computes d^2 logits at every step. It remains faster than PD-SSM, which also computes a $d \times d$ matrix at every step, at every state size we measure.

In practice, some architecture have already been optimized for speed and memory by the field, and custom kernels exist for them. We did not used them in our measurements, and we agree on the fact that this is not a fair comparison on deployment. The message of this experiment is not that some models are faster than others, but that the NFSM is compatible with a parallel scan and that it is fast and memory-efficient enough to be trained on long sequences; with scalings that are comparable to those of the baselines, *it is a practical architecture.*

Table 5: **Seed-to-seed variability of the failing length.** For each task and model, the failing length $L_{\max}$ of each of the five seeds, and the median and range $[\min; \max]$ over the seeds that learned the task; the median is the value reported in Table 1. $\times$ marks a seed that did not learn the task; other notation as in Table 1.

		Failing length $L_{\max}$ per seed						
Task	Model	42	43	44	45	46	Median	Range
$\mathbb{Z}_2$	NFSM	$\geq 1M$	$\geq 1M$	$\geq 1M$	$\geq 1M$	$\geq 1M$	$\geq 1M$	$[\geq 1M; \geq 1M]$
	Mamba$^-$	4352	5176	326	655	$\geq 512k$	4352	$[326; \geq 512k]$
	AUSSM	274	115	144	101	151	144	[101; 274]
	PD-SSM	102	117	105	119	110	110	[102; 119]
$\mathbb{Z}_{16}$	NFSM	$\geq 1M$	$\geq 1M$	$\geq 1M$	$\geq 1M$	$\geq 1M$	$\geq 1M$	$[\geq 1M; \geq 1M]$
	Mamba$^-$	78	91	83	87	$\times$	85	[78; 91]
	AUSSM	110	108	145	135	142	135	[108; 145]
	PD-SSM	98	117	94	98	97	98	[94; 117]
S_3	NFSM	$\geq 1M$	$\geq 1M$	$\geq 1M$	$\geq 1M$	$\geq 1M$	$\geq 1M$	$[\geq 1M; \geq 1M]$
	Mamba$^-$	155	163	186	197	200	186	[155; 200]
	AUSSM	102	168	128	216	100	128	[100; 216]
	PD-SSM	130	89	133	139	149	133	[89; 149]
S_4	NFSM	$\geq 1M$	$\geq 1M$	$\geq 1M$	$\geq 1M$	$\geq 1M$	$\geq 1M$	$[\geq 1M; \geq 1M]$
	Mamba$^-$	103	$\times$	113	92	107	105	[92; 113]
	AUSSM	103	83	83	89	98	89	[83; 103]
	PD-SSM	113	144	101	131	171	131	[101; 171]
A_5	NFSM	$\geq 1M$	$\geq 1M$	$\geq 1M$	$\geq 1M$	$\geq 1M$	$\geq 1M$	$[\geq 1M; \geq 1M]$
	Mamba$^-$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	
	AUSSM	81	82	81	82	86	82	[81; 86]
	PD-SSM	187	179	288	189	141	187	[141; 288]
M_{11}	NFSM	$\geq 32k$	$\geq 32k$	$\geq 32k$	$\geq 32k$	$\geq 32k$	$\geq 32k$	$[\geq 32k; \geq 32k]$
	Mamba$^-$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	
	AUSSM	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	
	PD-SSM	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	
DFF_5	NFSM	$\geq 1M$	$\geq 1M$	$\geq 1M$	$\geq 1M$	$\geq 1M$	$\geq 1M$	$[\geq 1M; \geq 1M]$
	Mamba$^-$	$\geq 1M$	$\geq 1M$	$\geq 1M$	$\geq 512k$	$\geq 512k$	$\geq 1M$	$[\geq 512k; \geq 1M]$
	AUSSM	102	120	93	96	87	96	[87; 120]
	PD-SSM	$\geq 256k$	$\geq 256k$	$\geq 256k$	$\geq 256k$	$\geq 256k$	$\geq 256k$	$[\geq 256k; \geq 256k]$
FF	NFSM	$\geq 1M$	$\geq 1M$	$\geq 1M$	$\geq 1M$	$\geq 1M$	$\geq 1M$	$[\geq 1M; \geq 1M]$
	Mamba$^-$	178	173	143	181	1738	178	[143; 1738]
	AUSSM	70	119	148	116	146	119	[70; 148]
	PD-SSM	192	105	133	115	169	133	[105; 192]
TSO_3	NFSM	$\geq 1M$	$\geq 1M$	$\geq 1M$	$\geq 1M$	$\geq 1M$	$\geq 1M$	$[\geq 1M; \geq 1M]$
	Mamba$^-$	1583	$\times$	$\times$	1420	$\times$	1501	[1420; 1583]
	AUSSM	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	
	PD-SSM	358	575	363	286	503	363	[286; 575]
TSO_4	NFSM	$\geq 1M$	$\geq 1M$	$\geq 1M$	$\geq 1M$	$\geq 1M$	$\geq 1M$	$[\geq 1M; \geq 1M]$
	Mamba$^-$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	
	AUSSM	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	
	PD-SSM	$\times$	1922	$\times$	$\times$	$\times$	1922	[1922; 1922]
TSO_5	NFSM	$\geq 128k$	$\geq 128k$	$\geq 128k$	$\geq 128k$	$\geq 128k$	$\geq 128k$	$[\geq 128k; \geq 128k]$
	Mamba$^-$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	
	AUSSM	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	
	PD-SSM	$\times$	$\times$	$\times$	321	$\times$	321	[321; 321]

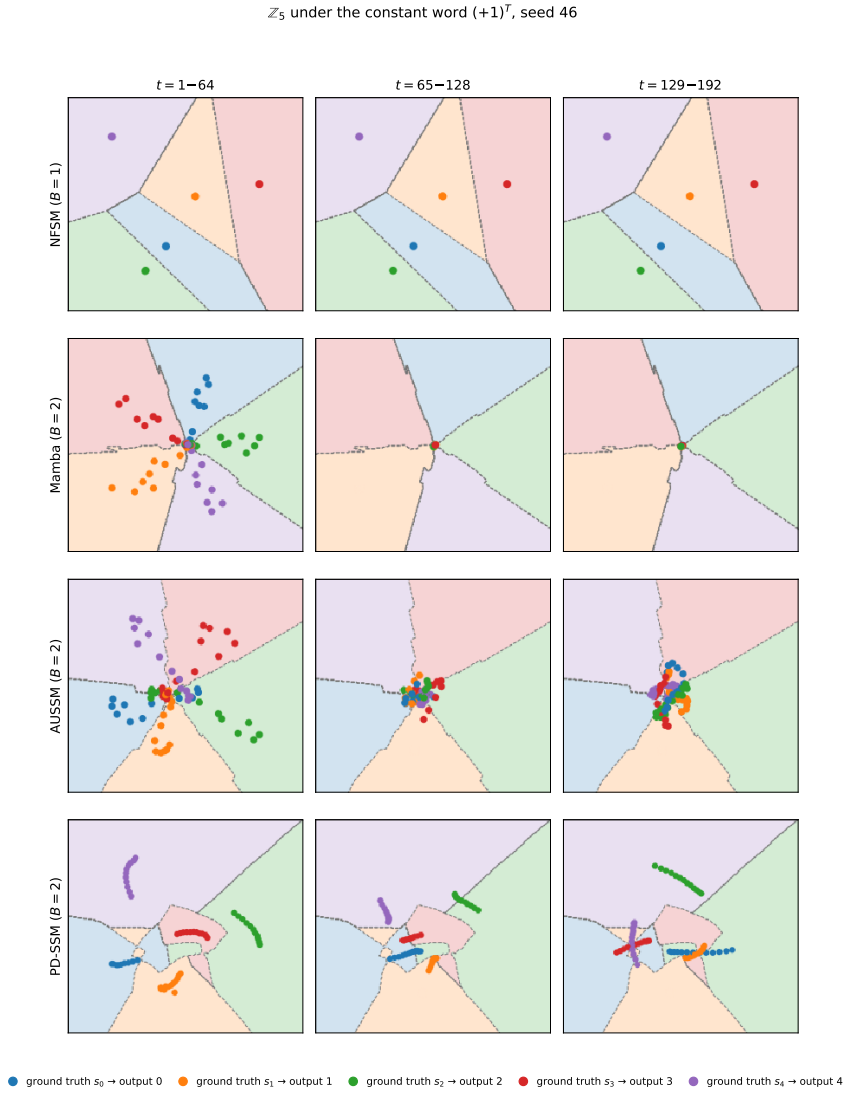

Figure 4: **The state of a trained model on $\mathbb{Z}_5$ under the constant word $(+1)^T$, seed 46.** Each row is a model, each column a window of the same run. Points are the recurrent state projected on its first two principal components, one per step, colored by the target state; the shaded regions are the decision regions of the classifier at the end of the backbone. B is the number of layers of the model.

Table 6: **Extraction on the algebraic tasks.** For each task, the head sizes (d_i), the number of symbols $|\Sigma|$, the number $\prod_i d_i$ of joint indices, the number of joint indices reached from the initial joint index, and the number of reachable task states.

Task	Heads (d_i)	$\lvert\Sigma\rvert$	Joint indices	Reached	Task states
$\mathbb{Z}_2$	(2)	2	2	2	2
$\mathbb{Z}_{16}$	(16)	16	16	16	16
S_3	$(3, 2)$	2	6	6	6
S_4	$(4, 3, 2)$	3	24	24	24
A_5	$(5, 12)$	3	60	60	60
M_{11}	$(11, 11, 11, 11)$	4	14641	7920	7920
DFF_5	(3)	3	3	3	3
FF	(3)	3	3	3	3

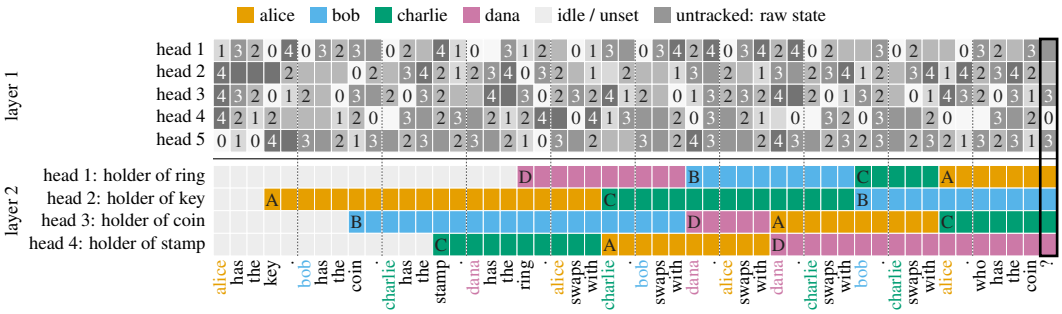

Figure 5: **Attractor indices of the heads of a two-layer NFSM reading a TSO_4 stream.** Each row is a head, each column a token of the stream. A head whose attractor indices determine a register of the task is colored by its value, a person or unset; the other heads show their raw index in gray. The box marks the question token, where the answer is read.

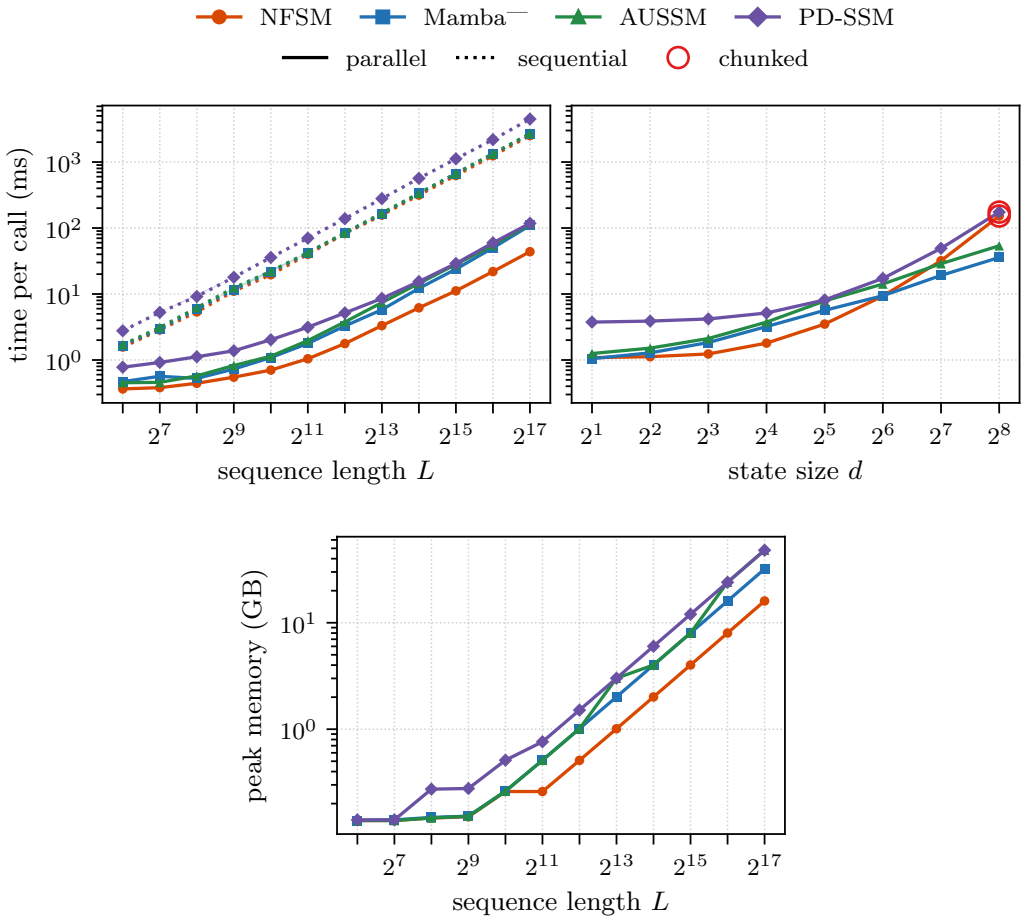

Figure 6: **Cost of one layer in a training step** (forward and backward passes, batch of 16, one A100). **Top left:** time per call against the sequence length L, at $d = 16$; solid lines are the parallel scan, dotted lines the sequential loop. **Top right:** time per call of the parallel scan against the state size d, at $L = 2^{12}$. **Bottom:** peak memory of the parallel scan against L, at $d = 16$; this is an approximate measurement (see text). Points circled in red are *chunked*: the sequence is processed in consecutive blocks, which bounds memory at the cost of time.