arXiv:submit/8216730 [cs.LG] 12 Oct 2026

TRAIN SHORT, ACT LONG: MULTISTABILITY AND THE HORIZON INDEPENDENCE OF RECURRENT POLICIES

Asad Bakija, Florent De Geeter, Julien Brandoit, Pierre Sacré & Guillaume Drion
Montefiore Institute, University of Liège
Liège, Belgium
{Asad.Bakija,gdrion}@uliege.be

ABSTRACT

In partially observable reinforcement learning (RL), a decision often depends on a cue seen many steps earlier, and training at such horizons is costly. Ideally, a policy trained on short horizons would remain optimal at arbitrarily long ones, making the training horizon a matter of convenience. We formalize this property as *delay-horizon independence* (DHI): optimality at every horizon, at a memory, computation and precision cost that is bounded in the horizon. We study agents that carry the cue in a recurrent memory of fixed size, whose cost does not grow with the delay. This paper characterizes DHI through the dynamics of that memory, in theory and on benchmarks. Although quantified over all horizons, DHI reduces to a property of a single map: a recurrent policy is DHI if and only if it admits a continuous readout of the memory, invariant under the idle dynamics and separating the cues. Monostable memory then precludes DHI at any parameter values, whereas multistable memory enables it. Monostability covers all contracting affine recurrences, hence state space models and gated linear recurrent neural networks (RNNs). Under arbitrarily small perturbations of the idle dynamics, multistability is the only mechanism supporting DHI at bounded memory. On the T-maze, the effective horizon of monostable models resists hyperparameter search: selected horizons fall short by up to $745\times$ when retrained. Multistable models trained on short horizons stay optimal up to 10^6, on the T-maze and on MiniGrid. Current parallelizable RNNs thus trade horizon independence for scalability: parallelizable multistable architectures are the direction this points to.

1 INTRODUCTION

In long-horizon partially observable Markov decision processes (POMDPs), an informative cue and the decision that depends on it are separated by many uninformative steps. The agent must carry the cue across that interval, and both the exploration cost and the sample inefficiency grow with its length. One way around that cost is to *train short and act long*: to train on a short version of the task and deploy on a long one. This only helps if the policy keeps working at the deployment horizon. Generalization of this kind is difficult in reinforcement learning (RL): policies overfit to their training environment (Korkmaz, 2024) and become unpredictable in unusual situations, even under full observability (Ghosh et al., 2021). Myers et al. (2025) introduced *horizon generalization*, the ability to learn from short horizons and generalize to long ones, for fully observed goal-conditioned MDPs. Their goal is visible at every step, so their policies need no memory and cannot forget.

Our setting, however, differs in two ways. First, the cue is presented once and must be carried over steps, so the agent can forget it. Forgetting is the entire difficulty here. When the goal is visible at every step, the agent has nothing to forget. Second, a guarantee that covers a finite range of horizons is usable only when its boundary is known. For a memory that fades, that boundary can be found only by testing at the deployment horizon. Fading-memory models do generalize over a finite range of horizons, but cannot be independent of the horizon in both the decision and the cost (Section 3.2). The question is therefore whether a policy validated at short horizons can be trusted at *every* horizon, at a cost that does not grow with the delay, without ever being tested at the deployment horizon.

To answer it, we formalize *delay-horizon independence* (DHI) for a class of delayed-cue POMDPs: optimality at every horizon, at a memory, computation and precision cost bounded in the horizon, which excludes solutions that buffer the history. Several mechanisms can carry the cue across the delay. Attention over a stored history does it at a cost that grows with the delay. A recurrent memory of fixed size does it at a cost that stays constant. We focus on the second approach because the property requires a bounded cost. This restriction to a fixed-size memory enables us to frame the process as a dynamical system.

This paper characterizes DHI through the dynamics of that memory, classifies the memory of common recurrent architectures accordingly, and tests the classification on standard POMDP benchmarks. Related work is discussed in Section A. Our main contributions are as follows.

DHI reduces to a property of one map. We prove that DHI holds if and only if the policy admits a continuous readout of the memory that is invariant under the idle dynamics and separates the cues.

Monostability precludes DHI, multistability enables it. A monostable memory, with a single global attractor, erases the separation between cues and admits no horizon-independent policy for any parameters or training procedure. This is checkable in closed form for every affine recurrence, and thus excludes state space models (SSMs) (Gu et al., 2021; 2022; Gu & Dao, 2024) and gated linear recurrent neural networks (RNNs) (Martin & Cundy, 2018; Qin et al., 2023; Beck et al., 2024; Feng et al., 2024; Zhu et al., 2025): selectivity and gating only slow the decay. A multistable memory supports DHI as long as its state never crosses a basin boundary. Under arbitrarily small perturbations of the idle dynamics, it is the only mechanism that supports DHI at bounded memory.

The effective horizon of monostable memory is not a design parameter. On a T-maze benchmark, every monostable model, nonlinear or parallelizable, is indeed optimal only up to a finite effective horizon $T^\star$. This horizon spans more than five orders of magnitude across a hyperparameter search, but a configuration selected for a long horizon does not reproduce it when retrained from different seeds, falling short by up to $745\times$. Multistable models trained on short horizons remain optimal up to 10^6 steps across cell types, and the bistable memory recurrent unit (BMRU) (De Geeter et al., 2026), parallelizable and free of transient dynamics, points to multistability as the cause: every BMRU configuration that solves the task stays optimal up to the evaluation ceiling. The same contrast holds on MiniGrid-Memory and MiniGrid-GoToDoor, evaluated at grid sizes up to 10^6.

2 DELAY-HORIZON INDEPENDENCE

We first formalize the setting and class of decision processes under study (Section 2.1), then define delay-horizon independence (Section 2.2).

2.1 DELAYED-CUE POMDPS

We consider a POMDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{O}, p_0, P, \Omega, R, \gamma)$ with discount factor $\gamma \in (0, 1]$, in the notation of Section B. The observation $o_t \in \mathcal{O}$ only partially reveals the state $s_t \in \mathcal{S}$, so acting optimally generally requires the entire past history of observations and actions.

We study agents that compress that history into a memory of fixed size. Formally, the memory state $h_t \in \mathcal{H}$, equipped with a norm $\|\cdot\|$, is updated by a recurrent equation:

$$h_t = \Phi(h_{t-1}, x_t), \qquad x_t \coloneqq (o_t, a_{t-1}), \tag{1}$$

where $o_t \in \mathcal{O}$ is the current observation and $a_{t-1} \in \mathcal{A}$ is the previous action. The agent then selects $a_t = \pi(h_t)$, where π is a policy. We call a policy that uses a memory state updated by Equation (1) to map past observations and actions to a current action a *recurrent policy*. The map Φ, with $\Phi_x \coloneqq \Phi(\cdot, x)$, is realized by an RNN with learnable parameters θ, so that recurrent policies in this sense cover the standard architectures for POMDPs, from classical nonlinear RNNs to more recent parallelizable linear RNNs and SSMs. Update equations, steady-state derivations and architectural details for every model we discuss are collected in Section E. A policy is *optimal* for $\mathcal{M}$ if it maximizes the expected return over all history-dependent policies (Section B).

We analyze POMDPs in which an informative cue is separated from the decision that depends on it by an uninformative phase, which we call the idle phase.

Definition 2.1 (Delayed-cue POMDP)**.** *A family $\{\mathcal{M}_T\}_{T \geq T_{\min}}$ of POMDPs, with $T_{\min} \geq 1$ and sharing the observation and action spaces, is a* delayed-cue family with minimum horizon $T_{\min} \geq 1$ *if, for every horizon $T \geq T_{\min}$:*

(A1) Cue: *At $t = 0$, the agent receives a key observation σ from a finite set of cues $\Sigma \subseteq \mathcal{O}$.*
(A2) Uninformative observations: *For every $1 \leq t \leq T$, the conditional law of o_t given the past actions is independent of σ.*
(A3) Decision step: *At $t = T$, the optimal action is $a^*(\sigma, o_T)$, and it does not depend on T.*
(A4) Reachability: *From every state reachable during the idle phase, some sequence of actions reaches the decision step.*

Taken together, these conditions say that the cue must be retained, and nothing more. (A1) is a normalization: a cue arriving later is handled by re-anchoring $t = 0$ at its arrival. (A2) and (A3) say that σ carries *information* that is necessary at the decision step and not recoverable during the idle phase, so a policy that has not retained it cannot act optimally. They do not say that it is *sufficient*, only that its influence on the optimal action is the same at every horizon. (A4) is a property of the environment, not of a policy. Setting $T_{\min} \geq 1$ excludes trivial instantaneous decisions at $T = 0$. Cues calling for the same action at every decision step never have to be told apart: we write $[\sigma]$ for the class of σ under that equivalence and require separation only between classes. This partition is the π^*-irrelevance abstraction of Li et al. (2006), discussed in Section D.4.

Definition 2.1 is the elementary unit out of which realistic long-horizon partially observable tasks are built (Section C). Its idle phase models any domain where informative observations are sparse and bursty relative to their downstream decisions, from a robot executing an initial instruction after a long traversal to an assistant holding an early user preference across unrelated turns.

Canonical example. The T-maze (Bakker, 2001) is the minimal instance of Definition 2.1 and our working illustration (Figure 1; formal description in Section F). The agent sees the reward side $\sigma \in \{\text{up}, \text{down}\}$ at $t = 0$ (A1), crosses a corridor whose observations carry no information about σ (A2) and whose end is always reachable (A4), and is signaled the junction by an observation $o_T \in \mathcal{O}$, where the rewarded action $a^*(\sigma, o_T)$ is independent of the corridor length (A3). A corridor of length L gives a horizon $T = L - 1$, with $T_{\min} = 1$: it can vary without changing the optimal policy.

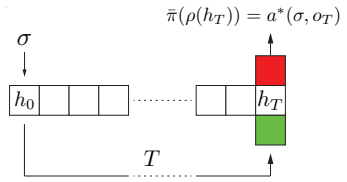

Figure 1: Illustration of the T-maze benchmark.

2.2 Definition of delay-horizon independence

Definition 2.2 (Delay-horizon independence)**.** *Let $\{\mathcal{M}_T\}_{T \geq T_{\min}}$ be a delayed-cue family and let π be a policy optimal at $T_{\min}$. Then π is* delay-horizon independent *(DHI) if:*

(a) Decision independence: *π is optimal for all horizons $T \geq T_{\min}$; and*
(b) Cost independence: *The memory footprint of h_t, the per-step computation of the state update and of π, and the precision at which h_t must be resolved are uniformly bounded in T.*

Decision independence asks for optimality at every $T \geq T_{\min}$, rather than up to some finite horizon. This requirement *decouples the training horizon from the deployment horizon*: a policy validated on short episodes remains optimal at any longer episode length, without ever being tested at that length. To make the contrast explicit, we also measure how far a policy generalizes when decision independence fails.

Definition 2.3 (Effective horizon)**.** *The* effective horizon *of a policy π is*

$$T^\star(\pi) := \sup \{T' \geq T_{\min} \mid \pi \text{ is optimal for every } T_{\min} \leq T \leq T'\}.$$

By construction, $T^\star(\pi) = \infty$ if and only if π satisfies decision independence. A relaxed property, requiring optimality only up to a finite $T^\star$, is a strictly weaker claim that does not provide the same guarantee. It is operationally usable only if $T^\star$ can be predicted in advance and the maximal effective horizon of the environment is known. All claims below are *asymptotic* statements about the limit $T \to \infty$.

Cost independence rules out history buffers, whose state grows with the sequence length. Its requirements on memory and computation are immediate for a recurrent policy. The one on precision needs a quantitative form.

Definition 2.4 (Resolvable separation)**.** *Let $\mathcal{R}_\sigma$ denote the set of memory states the agent can be in during the idle phase, once the key observation σ has been encoded. The* resolvable separation *is*

$$\mathrm{sep} \coloneqq \inf \{\|h - h'\| \mid h \in \mathcal{R}_\sigma, h' \in \mathcal{R}_{\sigma'}, [\sigma] \neq [\sigma']\},$$

the smallest distance between memory states carrying cues that require different actions.

By construction, $\mathrm{sep} > 0$ if and only if π satisfies the precision requirement of cost independence (Definition 2.2(b)). States that come arbitrarily close yet demand different actions force a precision that grows without bound. The infimum runs over all delays, not over matched ones, because the policy reads its memory without knowing the delay: it must resolve the two sets, not two snapshots taken at the same instant.

Delay-horizon independence is the conjunction of the two: $T^\star(\pi) = \infty$ and $\mathrm{sep} > 0$.

3 Memory dynamics for delay-horizon independence

The dynamics of the memory decide whether a recurrent policy is delay-horizon independent. We first hold the idle observation and action constant, which reduces delay-horizon independence from a property quantified over all delays to a property of a single map (Section 3.1). We then classify memory dynamics by that map: monostable dynamics preclude delay-horizon independence (Section 3.2), multistable dynamics enable it (Section 3.3). We finally let the observation and action vary again, and show that the classification survives (Section 3.4). Section D.2 delineates the scope of each result, and all proofs are given in Section D.1.

3.1 Conditions for delay-horizon independence

Simplified dynamics during the idle phase. We first consider a simplified idle phase ($t = 1, \dots, T-1$) with constant observation $o_t = o_c$ and constant action $a_{t-1} = a_c$. Equation (1) then reduces to a fixed map on $\mathcal{H}$,

$$\Phi_{\bar{x}} \colon \mathcal{H} \to \mathcal{H}, \qquad \Phi_{\bar{x}}(h) \coloneqq \Phi(h, \bar{x}), \qquad \bar{x} \coloneqq (o_c, a_c), \tag{2}$$

which we call the *idle map*, and we write $\Phi_{\bar{x}}^n$ for its n-fold composition, so that the memory state at the decision step is $h_T = \Phi_{\bar{x}}^T(h_0(\sigma))$, where $h_0(\sigma) = \Phi(h_{\mathrm{init}}, (\sigma, a_{\mathrm{init}}))$ denotes the memory state just after the key observation has been encoded. The last update, at $t = T$, is absorbed into the readout introduced below (Section D.2). We write

$$\mathcal{R}_\sigma \coloneqq \{\Phi_{\bar{x}}^n(h_0(\sigma)) \mid n \geq 0\}, \qquad \mathcal{R} \coloneqq \bigcup_{\sigma \in \Sigma} \mathcal{R}_\sigma,$$

for the states the memory can occupy after a cue. Holding the idle input constant makes fixed points, attractors and basins properties of a single map.

Readout. We introduce a *readout function* $\psi \colon \mathcal{H} \to \mathcal{D}$, where $\mathcal{D}$ is the space of decision-relevant representations. The readout extracts from h_t the information needed to act, and the policy factorizes as $\pi = \bar{\pi} \circ \psi$ with $\bar{\pi} \colon \mathcal{D} \to \mathcal{A}$.

Definition 3.1 (Invariant and separating readout)**.** *A function $\psi \colon \mathcal{H} \to \mathcal{D}$ is* invariant *under the idle map $\Phi_{\bar{x}}$ if*

$$\psi(\Phi_{\bar{x}}(h)) = \psi(h), \qquad \textit{for all } h \in \mathcal{R}. \tag{3}$$

It is separating *if it distinguishes cues in different classes,*

$$[\sigma] \neq [\sigma'] \implies \psi(h_0(\sigma)) \neq \psi(h_0(\sigma')).$$

Invariance is imposed on $\mathcal{R}$ only, and is not required for every $h \in \mathcal{H}$.

Lemma 3.2 (Readout characterization of delay-horizon independence)**.** *Let $\{\mathcal{M}_T\}_{T \geq T_{\min}}$ be a delayed-cue family and let π be a recurrent policy, optimal at $T_{\min}$. Then π is delay-horizon independent if and only if it admits a factorization $\pi = \bar{\pi} \circ \psi$ whose readout is uniformly continuous, invariant under the idle map $\Phi_{\bar{x}}$, and separating.*

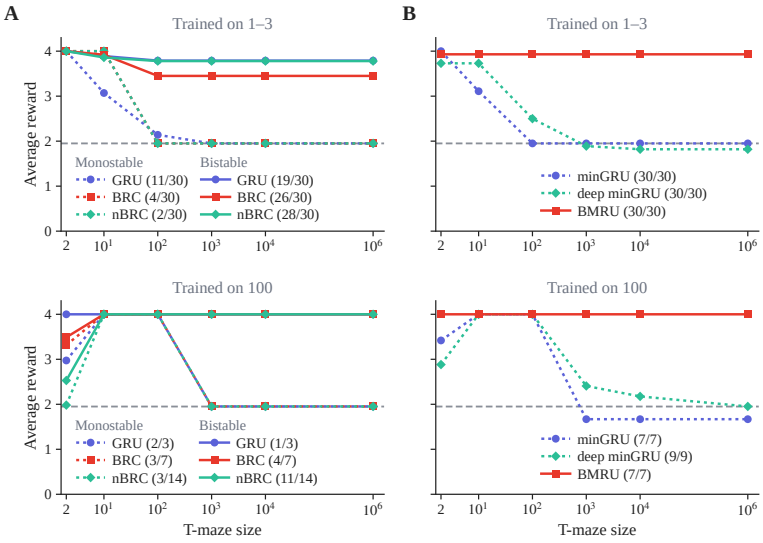

Figure 2: **Generalization capabilities on the T-maze. A.** Average reward obtained by each nonlinear RNN as a function of T-maze size, for models trained at $T \in [1, 3]$ (top) and at $T = 100$ (bottom). **B.** The same for the parallelizable RNNs. In both panels, monostable models are dashed and bistable models solid, and the gray dashed line marks random choice.

The content of Lemma 3.2 is the change of quantifier. Optimality at every horizon cannot be established by testing, since the delays are unbounded. The lemma replaces it by a one-step invariance on $\mathcal{R}$, which is a property of a single map. The next two subsections exploit this, since monostability and multistability are properties of $\Phi_{\bar{x}}$ and not of its iterates.

In what follows, we use Lemma 3.2 to study which memory dynamics support DHI, and verify theoretical statements experimentally on the T-maze environment: we train recurrent policies at short (T drawn in 1–3) and medium ($T = 100$) horizons, then evaluate them, without retraining, at horizons up to 10^6 and extract the effective horizon $T^\star$ when it exists. We study three nonlinear cells, the GRU (Cho et al., 2014), bistable recurrent cell (BRC) and its neuromodulated variant (nBRC) (Vecoven et al., 2021), and two parallelizable ones, minGRU (Feng et al., 2024) and BMRU (De Geeter et al., 2026) (+ Mamba (Gu & Dao, 2024) for the $T^\star$ analysis), with 30 seeds each. Their equations, training procedure, hyperparameters and network sizes are given in Sections E and I. The average reward achieved by each model at all tested lengths is shown in Figure 2 and discussed below.

3.2 Monostability precludes delay-horizon independence

Definition 3.3 (Monostable map)**.** *The idle map $\Phi_{\bar{x}} : \mathcal{H} \to \mathcal{H}$ is* monostable *(or globally attracting) if it has a unique globally attracting fixed point $h^\star \in \mathcal{H}$:*

$$\Phi_{\bar{x}}^n(h) \to h^\star \quad \text{as } n \to \infty, \quad \forall\, h \in \mathcal{H}.$$

Theorem 3.4 (Monostability precludes delay-horizon independence)**.** *Let the idle map $\Phi_{\bar{x}}$ be monostable with attractor $h^\star$, and suppose at least two cues require different actions at the decision step. Then* $\mathrm{sep} = 0$*. Consequently no recurrent policy whose memory is updated by $\Phi_{\bar{x}}$ satisfies cost independence, and none achieves DHI.*

The proof (Section D.1) never uses the readout: monostability defeats horizon independence through the memory itself. Requiring in addition that the policy act at a fixed precision recovers the stronger statement that the decision is lost as well.

Corollary 3.5 (Bounded effective horizon)**.** *Let the idle map $\Phi_{\bar{x}}$ be monostable, and let the policy act at a fixed precision: there is an $\eta > 0$ such that $\pi(h) = \pi(h')$ whenever $h, h' \in \mathcal{R}$ and $\|h - h'\| < \eta$. Then $T^\star(\pi) < \infty$, for every choice of parameters and training procedure.*

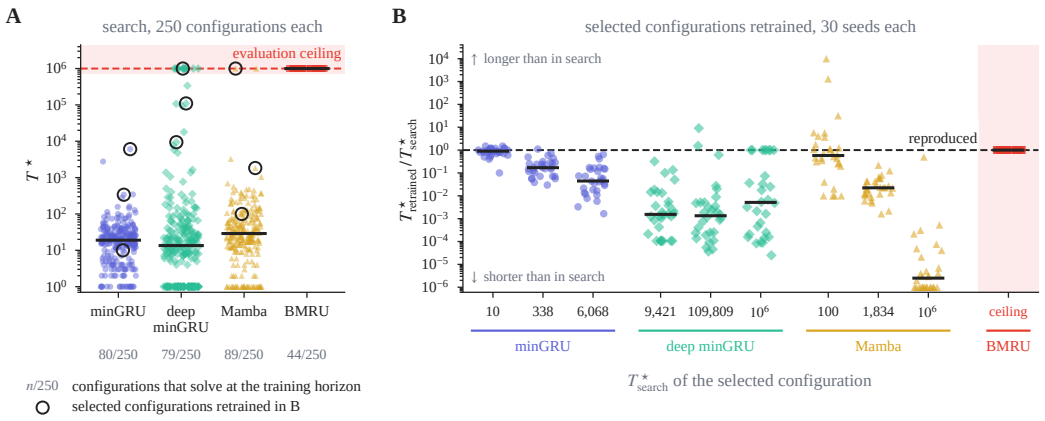

Figure 3: $T^\star$ **is unpredictable for monostable memory. A.** $T^\star$ of every configuration in a search of 250 configurations per architecture on the T-maze (for BMRU, the 44 configurations that solve the task). Fractions below the labels count the configurations that solve the task at the training horizon; black bars are medians; the shaded band is the evaluation ceiling ($T = 10^6$). Circled configurations are retrained on 30 seeds in B. **B.** $T^\star$ after retraining, as a ratio of the $T^\star$ that configuration reached in the search (dashed line: target reproduced).

Theorem 3.4 and Corollary 3.5 are verified experimentally on the T-maze (Figure 2A). Extracting the monostable nonlinear models from our trained population (using the variability among attractors metric, Section L), every one of them has a finite effective horizon: it is optimal between $T_{\min}$ and $T^\star$ and decays to random behavior beyond it, whether trained at the short or at the medium horizon.

For an affine recurrence $h_t = A(x_t)h_{t-1} + B(x_t)x_t$, monostability can be verified in closed form and without training: under a constant idle input $\bar{x}$, the idle map is affine, with a unique and globally attracting fixed point whenever the spectral radius satisfies $\rho(A(\bar{x})) < 1$ (Section E). This class contains SSMs such as Mamba (Gu & Dao, 2024) and gated linear RNNs such as minGRU (Feng et al., 2024), all of which are therefore excluded from DHI at every contracting parameter value. Selectivity and input gating can only change the *rate* of decay: they lengthen $T^\star$ without making it infinite. The noncontracting boundary, where $A(\bar{x})$ has an eigenvalue on the unit circle, is treated in Remark 3.10.

We confirm this experimentally on the T-maze using minGRU (Figure 2B): as for the nonlinear monostable models, the effective horizon of minGRU is bounded, and increased capacity does not repair it: a deep minGRU (four layers of 128 units, against one layer of five) lengthens $T^\star$, but performance still decays to random, only further along the horizon axis, as Theorem 3.4 predicts.

$T^\star$ is not a design parameter. The horizon of a monostable policy is an emergent quantity, which can be established only by testing at the deployment horizon. A finite $T^\star$ would be good enough if tuning could make it large. It cannot, because it cannot be predicted. In a hyperparameter search across 250 configurations per architecture (Figure 3A), minGRU, deep minGRU and Mamba solve the task at the training horizon in 80, 79 and 89 configurations respectively. Across the search, $T^\star$ spans more than five orders of magnitude, up to the evaluation ceiling of 10^6, which 14 deep minGRU and two Mamba configurations reach. A long horizon can therefore be selected for, but retraining the selected configuration on different seeds leads to highly inconsistent horizons (Figure 3B): only the two shortest targets are reproduced within a factor of two, and for finite targets the median over 30 seeds falls short of the value that motivated the selection by up to $745\times$. Reaching the ceiling in the search is no exception: the configurations selected there fall back to a median $T^\star$ of 5,151 (deep minGRU) and 2.5 (Mamba), and at least half of these Mamba seeds no longer solve the task at the training horizon.

3.3 MULTISTABILITY ENABLES DHI

Definition 3.6 (Multistable map)**.** *The idle map $\Phi_{\bar{x}}\colon \mathcal{H} \to \mathcal{H}$ is* multistable *if it has $K \geq 2$ disjoint attractors $A_1, \ldots, A_K \subset \mathcal{H}$, compact invariant sets that each attract a neighborhood of itself in $\mathcal{H}$, with basins of attraction $B_k = \{h \in \mathcal{H} \mid \mathrm{dist}(\Phi^n_{\bar{x}}(h), A_k) \to 0 \text{ as } n \to \infty\}$. Fixed-point attractors ($A_k = \{h^\star_k\}$) are the simplest case, but limit cycles and more general compact invariant sets are included.*

Theorem 3.7 (Multistability enables delay-horizon independence)**.** *Let $\Phi_{\bar{x}}$ be multistable. Suppose:*

(i) For each σ, the encoding places $h_0(\sigma)$ in a specific basin $B_{k(\sigma)}$, i.e., the acquisition phase is well-posed, and $[\sigma] \neq [\sigma']$ implies $k(\sigma) \neq k(\sigma')$.

(ii) There exists a readout $\psi\colon \mathcal{H} \to \mathcal{D}$ that

- *is invariant within each basin: $\psi(\Phi_{\bar{x}}(h)) = \psi(h)$ for all $h \in B_k$, $k = 1, 2, \ldots, K$, and*
- *separates the basins: $\psi(h) \neq \psi(h')$ with $h \in B_i$, $h' \in B_j$ and $i \neq j$.*

If $\pi = \bar{\pi} \circ \psi$ is optimal at $T_{\min}$, then it is delay-horizon independent.

Assumption (i) places the memory state in the basin the cue selected, and it stays there because basins of attraction are forward invariant; (ii) makes the readout constant along that orbit. Cost independence is not automatic and is checked in the proof: distinct attractors are disjoint compact sets, hence at positive distance, so the orbits seeded by two cues in different basins stay apart. Section D.1 gives the details.

Theorem 3.7 is likewise verified experimentally on the T-maze environment. The multistable models among the trained nonlinear RNNs are horizon-independent over the whole evaluated range, from $T = 2$ to $T = 10^6$, and this holds across RNN types, for GRU, BRC and nBRC (Figure 2A). Multistability is reached by a fraction of the trained models rather than by all of them, so realizing DHI in the cells in widest use is a question of training procedures that foster multistability, not of architectural capacity.

We further confirm that the mechanism is multistability itself, not the transient dynamics that accompany it, using the BMRU (De Geeter et al., 2026): a parallelizable RNN that is multistable by construction and converges to equilibrium in a single step, and therefore carries persistent memory with no transient component. It is optimal at every maze length and in every training condition (Figure 2B). Contrasting it with minGRU, which has transient dynamics but a single attractor, supports the causal role of multistability: the model with transients and no multistability fails at a finite $T^\star$, the model with multistability and no transients does not. Under the search of in Section 3.2, BMRU solves the task in 44 of 250 configurations, all 44 sit at the evaluation ceiling, and every retrained seed reproduces it (Figure 3). The unpredictability documented earlier is thus a property of monostable memory, not of the search.

3.4 BEYOND THE CONSTANT IDLE PHASE

In practice, the agent keeps acting and observing during the idle phase: observations fluctuate and the agent takes exploratory or habitual actions, or achieves intermediate goals. As long as these perturbations never push the memory state across the boundary between two basins of attraction, the readout ψ remains informative and delay-horizon independence is preserved.

Proposition 3.8 (Non-constant idle phase)**.** *Let the input x_t vary with time. Suppose that assumptions (i) and (ii) of Theorem 3.7 hold with the basins $B_1, \ldots, B_K$ replaced by regions that are pairwise at positive distance and forward invariant under every Φ_x, and with the invariance of ψ required under every Φ_x. Then the conclusions of Theorem 3.7 hold with $\Phi^T_{\bar{x}}$ replaced by the composition $\Phi_{x_T} \circ \cdots \circ \Phi_{x_1}$.*

3.5 MULTISTABILITY IS THE ONLY MECHANISM

Theorems 3.4 and 3.7 do not by themselves make multistability necessary, because a map can be neither monostable nor multistable. An idle map that is the identity on a subspace, a perfect integrator or a line attractor, has no attractor in the sense of Definition 3.6, yet its orbits stay apart and it admits an invariant separating readout. What excludes such maps is that an idle phase is never exactly

one map: observations fluctuate, the agent acts, and arithmetic rounds. Requiring delay-horizon independence to survive arbitrarily small perturbations forces genuine attractors.

Theorem 3.9 (Robust delay-horizon independence requires multistability)**.** *Let $\mathcal{H} \subset \mathbb{R}^d$ be compact, let $\Phi_{\bar{x}}$ be continuous, let $\eta > 0$, and let the idle phase be realized at each step by some map drawn from $\mathcal{W}_\eta := \{\tilde{\Phi}_{\bar{x}} \colon \mathcal{H} \to \mathcal{H} \mid \sup_{h\in\mathcal{H}} \|\tilde{\Phi}_{\bar{x}}(h) - \Phi_{\bar{x}}(h)\| \le \eta\}$. Suppose a policy $\pi = \bar{\pi} \circ \psi$ is delay-horizon independent for* every *realization of the idle phase in $\mathcal{W}_\eta$, with a readout ψ that is uniformly continuous, invariant along every realization and separating, and that at least two cues require different actions. Then $\Phi_{\bar{x}}$ has at least two disjoint attractors: $\Phi_{\bar{x}}$ is multistable in the sense of Definition 3.6.*

The proof, in Section D.1, shows that the states reachable from each cue form a compact trapping region, $\Phi_{\bar{x}}(\mathcal{R}_\sigma)$ lying in its relative interior, and that a trapping region contains an attractor.

Remark 3.10 (The boundary cases)**.** *Theorem 3.9 excludes the maps that Theorem 3.4 does not reach. A contraction has no two disjoint trapping regions. On a connected $\mathcal{H}$, the identity has only one, $\mathcal{H}$ itself, since a closed set with $(\mathcal{R}_\sigma \oplus \bar{B}_\eta) \cap \mathcal{H} \subseteq \mathcal{R}_\sigma$ is also open in $\mathcal{H}$. This disposes of the perfect integrator, of an affine recurrence whose $A(\bar{x})$ has an eigenvalue on the unit circle, and of unnormalized linear attention, whose recurrent form $S_t = S_{t-1} + \phi(k_t)v_t^\mathsf{T}$ is a pure accumulator. The identity survives only on a discrete set of states, where each state is isolated and hence an attractor, as in the BMRU, whose units hold $\pm\alpha_i$ unchanged during the idle phase. Only attracting dynamics absorb the perturbation, and two decisions require two attractors.*

Multistability is therefore not one design option among several: on a memory whose size does not grow with the delay, it is the *only* mechanism that supports delay-horizon independence robustly. The apparent alternative, letting the memory grow, does not escape the conclusion: it moves the failure from one requirement of Definition 2.2 to the other. Section D.2 states the scope of each result explicitly and works through the cases, attention-based policies and external memory in particular.

4 Delay-horizon independence on standard POMDP benchmarks

Both benchmarks reproduce the two patterns established on the T-maze: monostable memory has a finite effective horizon, and that horizon is a property of the individual trained model, not of the architecture. We evaluate the parallelizable models on two partially observable benchmarks adapted from MiniGrid, each chosen to stress a different aspect of Definition 2.1: MiniGrid-Memory extends the T-maze with richer dynamics and egocentric observations, and MiniGrid-GoToDoor extends it to 2D navigation with four possible decisions (Sections G and H). In both, the cue-to-decision delay grows at least linearly with the grid size N, so the horizon axis of Section 3 is the size axis of Figure 4. Models were trained only at short horizons $N \in \{7, 9, 11\}$ and evaluated up to $N = 10^6$, without retraining (Figure 4).

Up to and slightly beyond the training sizes, the three architectures are optimal and indistinguishable, with nearly all seeds on top of each other. Beyond them, the two monostable populations *spread*. At every intermediate grid size, part of the seeds still acts on the cue while the rest have already lost it, and the spread widens with N. On MiniGrid-Memory, the first seeds leave the optimum at $N = 10^2$–10^3, the population straddles the random-behavior line from $N = 10^4$ on, and a fraction of seeds is still optimal at $N = 10^6$, three orders of magnitude past the first failures of the same architecture trained identically. MiniGrid-GoToDoor shows the same picture with an earlier onset. Adding capacity does not change the nature of the failure: the deep minGRU population overlaps with the shallow one throughout, with no systematic gain in the size at which it fails.

The BMRU distribution over seeds at $N = 10^6$ is the one it has at the training sizes, which is the empirical signature of Theorem 3.7. One detail of Figure 4 makes the point sharper: on MiniGrid-GoToDoor a single BMRU seed sits at about half the optimal reward, at every grid size *including the training ones*. That seed is a suboptimal policy, not a policy that forgets. Horizon independence constrains the dependence of the score on the horizon, not its value.

The idle phase is non-constant in both environments: the egocentric observations change with the agent own motion in MiniGrid-Memory, and the agent follows a two-dimensional trajectory of its choosing in MiniGrid-GoToDoor. These results are therefore consistent with Proposition 3.8: the perturbations of the idle phase do not push the memory state out of the region in which the cue placed it, and the cue survives an unbounded number of them.

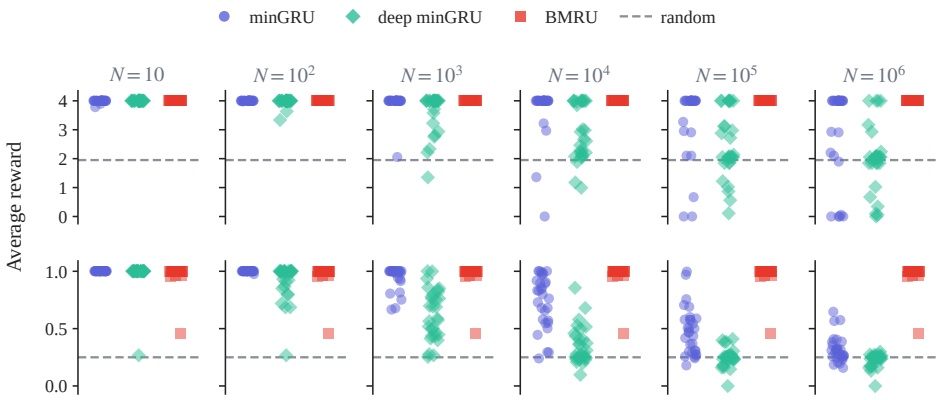

Figure 4: **Horizon independence on standard POMDP benchmarks.** Average reward as a function of the grid size N, for models trained only at short horizons and evaluated up to $N = 10^6$ without retraining. Top row, MiniGrid-Memory; bottom row, MiniGrid-GoToDoor; the two environments have different reward scales, and the gray dashed line marks random choice in each. Every marker is one seed: minGRU (blue circles), deep minGRU (green diamonds) and BMRU (red squares). Both monostable models are optimal at and just beyond the training sizes and then decay to random behavior, deep minGRU only slightly later than minGRU, whereas the BMRU distribution is unchanged from the training sizes up to $N = 10^6$.

5 Discussion

In this paper, we introduced the concept of delay-horizon independence in RL, showed how it decouples the training horizon from the deployment horizon, and linked the ability of RNN variants to reach DHI with their dynamical properties. Through a mix of theoretical developments and experimental validations, we have shown that, on a recurrent state whose size does not grow with the delay, multistability is the only mechanism supporting DHI once the idle phase is subject to arbitrarily small perturbations (Theorem 3.9), and that it is absent by construction from most state-of-the-art parallelizable RNN variants.

On the importance of an asymptotic guarantee. Our claims are asymptotic. Theorem 3.4 says that a monostable model cannot be optimal at *every* horizon; it does not say that such a model fails at any given horizon, which is confirmed by our experiments. What we claim is that the boundary of that range, the effective horizon $T^\star$, is finite and, empirically, unpredictable: it varies by orders of magnitude across seeds within a fixed architecture and training setup, so it cannot be relied upon without testing at the deployment horizon, which reintroduces the very cost DHI is meant to remove.

Limitations. *On the sufficiency of multistability.* While our theoretical results establish multistability as sufficient (Theorem 3.7) and, under perturbation, necessary (Theorem 3.9) for the *existence* of delay-horizon independent optimal policies, reaching these policies during training is not guaranteed. Developing generalization-aware training strategies that explicitly foster multistability (Lambrechts et al., 2023), which would also be the route to transferring these results to widely deployed cells such as the GRU and the LSTM, is an essential next step.

The switch toward parallelizable architectures. As the field trends toward parallelizable RNN variants to enable large-scale computation, a critical trade-off has emerged. We showed that current state-of-the-art models such as SSMs and gated linear RNNs are inherently monostable, making them incompatible with DHI. As a solution, we propose a shift toward hybrid architectures that integrate the efficiency of parallelizable transient dynamics with the persistent memory of multistability. Such hybrid designs could lead to scalable, efficient models that are delay-horizon independent.

AI USE STATEMENT

In this work, we partially used generative AI tools to formulate mathematical claims, assist in the writing of proofs, design or provide feedback on research methodology or experiments, implement methods, and assist with translation.

We did not use generative AI tools to interpret results, propose or refine hypotheses, or provide critical ingredients to prove mathematical claims. Generating synthetic datasets, developing theoretical models or conceptual frameworks, cleaning or reformatting datasets, and qualitative or thematic data analysis are not applicable to this work.

Additionally, we partially used generative AI tools to identify relevant literature, create or edit software code, and edit the research paper to improve readability, including figures.

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.

ACKNOWLEDGEMENTS

This work has been the subject of a patent application (Number: EP26151077). This present research was supported by the Belgian Government through the Federal Public Service Policy and Support. It also benefited from computational resources made available on Lucia, the Tier-1 supercomputer of the Walloon Region, infrastructure funded by the Walloon Region under the grant agreement n°1910247. Florent De Geeter gratefully acknowledges the financial support of the Walloon Region for Grant No. 2010235 – ARIAC by DW4AI.

REFERENCES

Gustave Bainier, Antoine Chaillet, Rodolphe Sepulchre, and Alessio Franci. State-space fading memory. *arXiv preprint arXiv:2603.23814*, 2026.

Bram Bakker. Reinforcement learning with long short-term memory. *Advances in neural information processing systems*, 14, 2001.

Maximilian Beck, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. xLSTM: Extended Long Short-Term Memory. *Advances in Neural Information Processing Systems*, 37:107547–107603, December 2024. doi: 10.52202/079017-3417.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48, 2009.

Guy E. Blelloch. Scans as primitive parallel operations. *IEEE Transactions on Computers*, 38(11): 1526–1538, November 1989.

Stephen Boyd and Leon O. Chua. Fading memory and the problem of approximating nonlinear operators with Volterra series. *IEEE Transactions on Circuits and Systems*, 32(11):1150–1161, 1985.

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.

Maxime Chevalier-Boisvert, Bolun Dai, Mark Towers, Rodrigo Perez-Vicente, Lucas Willems, Salem Lahlou, Suman Pal, Pablo Samuel Castro, and Jordan Terry. Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. In *Advances in Neural Information Processing Systems 36, New Orleans, LA, USA*, December 2023.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734, 2014.

Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In *International conference on machine learning*, pp. 1282–1289. PMLR, 2019.

Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *International conference on machine learning*, pp. 2048–2056. PMLR, 2020.

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

Leo Feng, Frederick Tung, Hossein Hajimirsadeghi, Mohamed Osama, Yoshua Bengio, and Roland Memisevic. Were RNNs all we needed? *arXiv preprint arXiv:2410.01201*, 2024.

Dibya Ghosh, Jad Rahme, Aviral Kumar, Amy Zhang, Ryan P Adams, and Sergey Levine. Why generalization in rl is difficult: Epistemic pomdps and implicit partial observability. *Advances in neural information processing systems*, 34:25502–25515, 2021.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626): 471–476, 2016.

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

Albert Gu, Isys Johnson, Karan Goel, Khaled Saab, Tri Dao, Atri Rudra, and Christopher Ré. Combining recurrent, convolutional, and continuous-time models with linear state space layers. *Advances in neural information processing systems*, 34:572–585, 2021.

Albert Gu, Karan Goel, and Christopher Re. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2022.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to Control: Learning Behaviors by Latent Imagination. In *International Conference on Learning Representations*, September 2019.

Danijar Hafner, Timothy P Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. In *International Conference on Learning Representations*, 2020.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640(8059):647–653, 2025.

Matthew J Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. In *AAAI fall symposia*, volume 45, pp. 141, 2015.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, 1997.

Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and João G.M. Araújo. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022.

Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

Robert Kirk, Amy Zhang, Edward Grefenstette, and Tim Rocktäschel. A survey of zero-shot generalisation in deep reinforcement learning. *Journal of Artificial Intelligence Research*, 76: 201–264, 2023.

Ezgi Korkmaz. A survey analyzing generalization in deep reinforcement learning. *https://arxiv.org/pdf/2401.02349v2*, 2024.

Gaspard Lambrechts, Florent De Geeter, Nicolas Vecoven, Damien Ernst, and Guillaume Drion. Warming up recurrent neural networks to maximise reachable multistability greatly improves learning. *Neural Networks*, 166:645–669, 2023.

Lihong Li, Thomas J Walsh, and Michael L Littman. Towards a unified theory of state abstraction for mdps. *AI&M*, 1(2):3, 2006.

Peng Liu, Jun Wang, and Zhigang Zeng. An overview of the stability analysis of recurrent neural networks with multiple equilibria. *IEEE Transactions on Neural Networks and Learning Systems*, 34(3):1098–1111, 2023. doi: 10.1109/TNNLS.2021.3105519. Epub 2023 Feb 28.

Eric Martin and Chris Cundy. Parallelizing linear recurrent neural nets over sequence length. In *International Conference on Learning Representations*, 2018.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937. PmLR, 2016a.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937. PmLR, 2016b.

Vivek Myers, Catherine Ji, and Benjamin Eysenbach. Horizon generalization in reinforcement learning. In *The Thirteenth International Conference on Learning Representations*, 2025.

Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E Taylor, and Peter Stone. Curriculum learning for reinforcement learning domains: A framework and survey. *Journal of Machine Learning Research*, 21(181):1–50, 2020.

Tianwei Ni, Michel Ma, Benjamin Eysenbach, and Pierre-Luc Bacon. When do transformers shine in rl? decoupling memory from credit assignment. *Advances in Neural Information Processing Systems*, 36:50429–50452, 2023.

Emilio Parisotto, Francis Song, Jack Rae, Razvan Pascanu, Caglar Gulcehre, Siddhant Jayakumar, Max Jaderberg, Raphael Lopez Kaufman, Aidan Clark, Seb Noury, et al. Stabilizing transformers for reinforcement learning. In *International conference on machine learning*, pp. 7487–7498. PMLR, 2020.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pp. 1310–1318. PMLR, 2013.

Leonid Peshkin, Nicolas Meuleau, and Leslie Kaelbling. Learning policies with external memory. *Proceedings of the Sixteenth International Conference on Machine Learning, 1999*, pp. 307–314, 1999.

Zhen Qin, Songlin Yang, and Yiran Zhong. Hierarchically Gated Recurrent Neural Network for Sequence Modeling. *Advances in Neural Information Processing Systems*, 36:33202–33221, December 2023.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 23–30. IEEE, 2017.

Jos Van Der Westhuizen and Joan Lasenby. The unreasonable effectiveness of the forget gate. *arXiv preprint arXiv:1804.04849*, 2018.

Nicolas Vecoven, Damien Ernst, and Guillaume Drion. A bio-inspired bistable recurrent cell allows for long-lasting memory. *PLoS ONE*, 16(6):e0252676, 2021.

Amy Zhang, Nicolas Ballas, and Joelle Pineau. A dissection of overfitting and generalization in continuous reinforcement learning. *arXiv preprint arXiv:1806.07937*, 2018.

Guo-Bing Zhou, Jianxin Wu, Chen-Lin Zhang, and Zhi-Hua Zhou. Minimal gated unit for recurrent neural networks. *International Journal of Automation and Computing*, 13(3):226–234, 2016.

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, 2025. URL `https://arxiv.org/abs/2406.02528`.

Luisa Zintgraf, Sebastian Schulze, Cong Lu, Leo Feng, Maximilian Igl, Kyriacos Shiarlis, Yarin Gal, Katja Hofmann, and Shimon Whiteson. Varibad: Variational bayes-adaptive deep rl via meta-learning. *Journal of Machine Learning Research*, 22(289):1–39, 2021.

CONTENTS OF THIS APPENDIX

- Section A: Related Work
- Section B: Reinforcement Learning in POMDPs
- Section C: Generality of the Delayed-Cue POMDP
- Section D: Additional theoretical material
- Section E: Recurrent Architectures: Dynamics and Steady States
- Section F: T-maze Environment
- Section G: MiniGrid-Memory Environment
- Section H: MiniGrid-GoToDoor Environment
- Section I: Training Procedure and Hyperparameters
- Section J: Hyperparameter Sweep
- Section K: Measuring the Effective Horizon
- Section L: Computation of the VAA
- Section M: Learning curves and RNN dynamics
- Section N: Equations for the different RNNs used in this work

A RELATED WORK

Since our work sits at the intersection of several active research areas, we briefly survey three relevant topics in this section: generalization in RL, memory for POMDPs, and multistability in RNNs.

Generalization in reinforcement learning. Generalization is a long-standing challenge in reinforcement learning: policies learned in one setting often fail to transfer even to superficially similar environments (Korkmaz, 2024). Early work highlighted that deep RL agents can overfit dramatically to the specifics of their training environments, including random seeds and initial state distributions, even when the underlying task remains the same (Zhang et al., 2018). This prompted the development of dedicated generalization benchmarks such as CoinRun (Cobbe et al., 2019) and Procgen (Cobbe et al., 2020), which expose policies to procedurally generated environments at test time, and systematic evaluations of what dimensions of variation agents can and cannot handle.

A comprehensive survey (Kirk et al., 2023) categorizes generalization along several axes: across environment parameters (reward, dynamics, layout), across tasks, and across levels of difficulty, each presenting distinct failure modes. One practical strategy to improve generalization is curriculum learning, in which the agent is progressively exposed to harder instances of the task, reducing the exploration burden and improving sample efficiency (Bengio et al., 2009; Narvekar et al., 2020). A closely related strategy is domain randomization, which widens the training distribution to encourage policies that are robust to unseen variations (Tobin et al., 2017).

Generalization in fully observable settings is already difficult (Ghosh et al., 2021) and partial observability amplifies the difficulty because the agent cannot distinguish states that appear identical but require different actions.

Our work focuses on a specific, formally defined axis of generalization: the temporal horizon. Myers et al. (2025) introduced *horizon generalization* in goal-conditioned MDPs and connected it to planning invariance, whereby an agent pursues the same policy regardless of whether it is given a distant goal or an intermediate waypoint. We study the partially observable setting, where memory is the central bottleneck, and where the property of interest is stronger: not that behavior transfers from short to long horizons, but that neither the decision nor its cost depends on the horizon at all. We provide a formal necessary and sufficient condition linking that independence to the dynamical properties of the agent's recurrent memory. Unlike curriculum-based or domain-randomization approaches, which seek to generalize by widening the training distribution, delay-horizon independence allows an agent trained exclusively on short horizons to remain optimal at arbitrarily longer ones, with no requirement to expose it to long-horizon episodes during training.

Memory and recurrent architectures for POMDPs. Partial observability is ubiquitous in real-world sequential decision making. A principled solution is to maintain a belief state, a probability distribution over the true environment state conditioned on the observation history (Kaelbling et al., 1998). Exact belief-state computation is generally intractable, and in deep RL it is typically replaced by a learned approximate summary encoded in the hidden state of a recurrent neural network. Hausknecht & Stone (2015) proposed the deep recurrent Q-network (DRQN), which replaces one of the feedforward components of DQN (Mnih et al., 2015) with an LSTM layer, showing that a recurrent hidden state can substitute for full observability in Atari games. Equipping actor-critic algorithms such as A3C (Mnih et al., 2016a) and PPO (Schulman et al., 2017) with recurrent encoders has since become one of the standard approaches to handling partial observability.

Beyond standard RNNs, various approaches have been proposed to enhance the memory capacity accessible to the agent. Memory-augmented networks such as the neural turing machine (Graves et al., 2014) and the differentiable neural computer (Graves et al., 2016) decouple capacity from the hidden-state dimension by equipping networks with an addressable external memory, enabling longer retention at the cost of additional architectural complexity. Transformer-based architectures provide a complementary strategy via self-attention over a growing context window. The gated transformer-XL (GTrXL) (Parisotto et al., 2020) adapts this idea for RL by incorporating gating to stabilize training, achieving strong results on memory-intensive benchmarks. However, self-attention is quadratic in context length, which becomes prohibitive for very long horizons, and attention over a finite window cannot retain the cue beyond that window. Attention therefore illustrates the trade at the heart of Definition 2.2: a growing context keeps the decision correct at a cost that grows with the horizon, and a fixed context keeps the cost bounded at the price of a horizon ceiling.

Structured world models offer a further alternative, learning a latent dynamics model of the environment from which a compact belief state can be inferred. Dreamer and its successors (Hafner et al., 2019; 2020; 2025) use a recurrent state space model to separate deterministic from stochastic latent dynamics, enabling policy learning and value estimation through imagined rollouts. VariBAD (Zintgraf et al., 2021) takes a Bayesian perspective, meta-learning a posterior over task parameters that is updated online to approximate the belief state. These approaches enrich memory through architectural or training innovations, but do not address the dynamical question of whether the underlying recurrent cell can sustain the persistent, history-dependent representations that delay-horizon independence demands. Our analysis shows that this is fundamentally a question of multistability in the cell dynamics, a property that is absent in linear and input-gated architectures regardless of how they are embedded in a larger system.

Multistability in RNNs. Monostability and fading memory are closely related concepts (Boyd & Chua, 1985): a system with a single stable equilibrium forces all trajectories to decay exponentially toward that unique attractor, inevitably erasing information about past inputs, which is the key characteristic of fading memory (Boyd & Chua, 1985; Pascanu et al., 2013; Bainier et al., 2026). In contrast, persistent memory requires multistability, where multiple distinct stable states can coexist, enabling the network to maintain and switch between stored values without asymptotic decay. Despite this fundamental link, *multistability remains largely neglected in the machine learning literature on recurrent architectures* (Liu et al., 2023); notable exceptions include bio-inspired bistable cells (Vecoven et al., 2021; Lambrechts et al., 2023) and the MRU line of works (De Geeter et al., 2026; Brandoit et al., 2026), both of which explicitly harness multistable dynamics to overcome the inherent forgetting of monostable linear systems.

B Reinforcement Learning in POMDPs

Formally, a POMDP $\mathcal{M}$ is defined as an 8-tuple

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{O}, p_0, P, \Omega, R, \gamma),$$

where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, and $\mathcal{O}$ is the observation space. The initial state distribution

$$p_0(s_0) \colon \mathcal{S} \to [0, 1]$$

gives the probability of $s_0 \in \mathcal{S}$ being the initial state of the POMDP. The Markovian system dynamics are defined by the transition probability distribution

$$P(s_{t+1} \mid s_t, a_t) \colon \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]\,,$$

which gives the probability that taking action $a_t \in \mathcal{A}$ in state $s_t \in \mathcal{S}$ leads to state $s_{t+1} \in \mathcal{S}$. The reward function

$$R(s_t, a_t, s_{t+1}) \colon \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$$

gives the immediate reward for transitioning from state $s_t \in \mathcal{S}$ to state $s_{t+1} \in \mathcal{S}$ through action $a_t \in \mathcal{A}$. The observation distribution

$$\Omega(o_t \mid s_t) \colon \mathcal{O} \times \mathcal{S} \to [0, 1]$$

gives the probability of getting observation $o_t \in \mathcal{O}$ in state $s_t \in \mathcal{S}$. Finally, $\gamma \in (0, 1]$ is a discount factor that scales the impact of future rewards.

The transition law is denoted P and the observation law Ω, rather than T and O as is also common, to avoid any collision with the horizon T and the observation space $\mathcal{O}$, both of which are central to this work.

When dealing with a POMDP in reinforcement learning, the agent cannot access the state $s_t \in \mathcal{S}$ directly and must rely on the history $\tau_t = (o_0, a_0, \ldots, o_t) \in (\mathcal{O}, \mathcal{A})^t \times \mathcal{O} \coloneqq \mathcal{T}_t$ since observations in a POMDP do not provide sufficient information as $P(o_{t+1} \mid \tau_t, a_t) \neq P(o_{t+1} \mid o_t, a_t)$.

Definition B.1 (Optimal policy)**.** *A policy π is* optimal *for $\mathcal{M}$ if it maximizes the expected discounted return $\mathbb{E}\left[\sum_{t \geq 0} \gamma^t R(s_t, a_t, s_{t+1})\right]$ over all history-dependent policies.*

Due to the unbounded growth of the history over time, a common approach is to encode the history into a fixed-size latent state using RNNs. This latent representation can then be used to learn a history-dependent policy with standard reinforcement learning algorithms.

C GENERALITY OF THE DELAYED-CUE POMDP

Definition 2.1 isolates a single instance of the cue/idle/decision pattern for analytical tractability. This appendix argues that the pattern is not a narrow special case but the elementary unit out of which realistic long-horizon partially observable tasks are built: such tasks rarely present as one clean instance of Definition 2.1, but they contain it as a recurring sub-structure. We give four examples spanning distinct domains (Section C.1), then discuss how single-cue results compose when several such segments interact (Section C.2).

C.1 DELAYED-CUE ACROSS DOMAINS

Manipulation under transient occlusion. A manipulator observes an object identity or target pose only during an initial approach; the object then becomes occluded (by the gripper, by another object, by a viewpoint change) for the remainder of the reach, and the manipulation action must be correct once contact is made. The occluded interval is uninformative about identity **(A2)** and its length is set by the approach trajectory, not by the object **(A4)**; the action at contact is the decision step **(A3)**, and the observation at the start is the cue **(A1)**. This is the T-maze corridor played out as a reach instead of a walk, with the horizon set by geometry rather than by design.

Instruction-conditioned navigation. An agent is told its destination once, at the start of an episode, then must traverse a long, task-irrelevant environment (corridors, junctions, other rooms) before the destination becomes actionable at a door or fork. The intervening steps are exploratory and carry no information about the destination; only the terminal choice depends on it, and the number of intervening steps is set by the building layout, not by the task designer.

Assistive agents with a stated early preference. A user states a preference once, early in an interaction (a dietary restriction, a seating preference), and the agent then handles several unrelated requests before that preference becomes relevant to a downstream choice (an order, a booking). The intervening turns carry information about the unrelated requests but nothing about the stated preference, and their number is set by the conversation, not fixed in advance.

Delayed alarms in monitoring and control. A supervisory controller receives a diagnostic reading once (an anomaly class flagged by a sensor) and must act on it correctly whenever the corresponding fault later manifests, after an arbitrary number of nominal control steps that reveal nothing about which anomaly class was flagged. The delay between flag and fault is not known at design time.

In each case the cue is cheap to observe but costly to forget, and the interval separating it from the decision is not a fixed constant: it is set by an approach trajectory, a building layout, a conversation length, or a time-to-failure that the agent does not control. This is exactly why Definition 2.2 quantifies over every horizon $T \geq T_{\min}$ rather than fixing one: none of the examples above come with a guaranteed bound on how long the idle phase will last, so a policy that is only validated at one horizon offers no guarantee at deployment.

C.2 COMPOSING SINGLE-CUE RESULTS

Realistic tasks rarely stop at one cue. Two extensions matter: cues that arrive and are resolved sequentially, and cues that must be held concurrently.

Sequential cues. Suppose the idle phase does not end with a single decision but with a fresh cue σ_2 at time τ, followed by a second idle phase and a decision at $T > \tau$ that depends on σ_2 (possibly together with σ_1). This is exactly the setting covered by Proposition 3.8: re-anchoring $t = 0$ at τ and treating the readout of σ_1 as part of the (now time-varying) input to the second segment recovers the same forward-invariance argument, applied once per segment. A multistable memory that resolves the first decision and then supports a (possibly different) multistable regime for the second segment inherits delay-horizon independence on each segment separately, and hence on their concatenation, provided the transition at τ does not itself cross a basin boundary before σ_2 is encoded: the same non-crossing condition as assumption *(i)* of Proposition 3.8, applied at every cue arrival rather than only at $t = 0$.

Concurrent cues. The harder case is a decision at T that depends jointly on two cues σ_1 (at $t = 0$) and σ_2 (at $t = \tau$) that must both be retained simultaneously. Theorem 3.7 already covers an arbitrary

finite number of attractors $A_1, \ldots, A_K$ for a single cue drawn from a set of size K; concurrency requires instead that the attractor landscape represent the Cartesian product of the two cue alphabets, i.e., at least $|\Sigma_1| \times |\Sigma_2|$ distinct basins rather than $|\Sigma_1| + |\Sigma_2|$. This is the same combinatorial structure as the register construction of Section D.2 (2^b attractors for a b-bit register): a memory holding k simultaneously outstanding cues needs an attractor count that grows multiplicatively in k.

The failure of monostable maps generalizes. Theorem 3.4 is a negative result about monostable memory, which gets stronger under composition: if a monostable idle map cannot retain even one cue across one uninformative interval, it cannot retain several, whether they arrive sequentially or concurrently. The single-cue analysis therefore identifies the failure mode that recurs in every composed setting above.

D ADDITIONAL THEORETICAL MATERIAL

This appendix collects the proofs of all results stated in the main text (Section D.1), delimits their scope, and relates them to two neighboring frameworks.

D.1 DEFERRED PROOFS

D.1.1 PROOF OF LEMMA 3.2

Sufficiency is an induction on T for the decision, together with the observation that a constant gap between readout values forces, through uniform continuity, a uniform gap between memory states. Necessity constructs the readout: optimality at every horizon makes the action constant along each orbit, cost independence keeps the orbits a positive distance apart, and a truncated distance to each orbit is then a uniformly continuous readout that is invariant and separating.

Proof of Lemma 3.2. **(⇒) Necessity.** Suppose π is delay-horizon independent. By (A3) the optimal action does not depend on T, so for every σ and every $n \geq T_{\min}$,

$$\pi(\Phi_{\bar{x}}^n(h_0(\sigma))) = a^*(\sigma, o_T) :$$

the action is constant along the orbit $\mathcal{R}_\sigma$, and orbits seeded by cues with different optimal actions are disjoint. Cost independence gives $\operatorname{sep} \geq \mu > 0$, so those orbits are at distance at least μ. Enumerate the optimal-action classes $1, \ldots, m$, let O_k be the union of the orbits in class k, and set

$$\psi(h)_k \coloneqq \max(0, 1 - \tfrac{3}{\mu} \operatorname{dist}(h, O_k)), \qquad k = 1, \ldots, m.$$

Each coordinate is a truncated distance function, so ψ is Lipschitz with constant $3/\mu$ and in particular uniformly continuous. On O_k it equals the basis vector e_k, since every other class lies at distance at least $\mu > \mu/3$. It is therefore constant on each orbit, hence invariant on $\mathcal{R}$, and separating because $e_i \neq e_j$ for $i \neq j$. Taking $\bar{\pi}(e_k)$ to be the optimal action of class k gives $\pi = \bar{\pi} \circ \psi$ on $\mathcal{R}$.

The construction is deliberate: necessity cannot be established for an *arbitrary* factorization, because $\bar{\pi}$ is in general not injective, so equality of actions along an orbit constrains $\psi(\Phi_{\bar{x}}(h))$ and $\psi(h)$ only to lie in the same fibre of $\bar{\pi}$, not to coincide. This is why Lemma 3.2 asserts the existence of a factorization rather than a property of every factorization.

(⇐) Sufficiency. If ψ is invariant on $\mathcal{R}$, then by induction $\psi(\Phi_{\bar{x}}^n(h)) = \psi(h)$ for every $n \geq 0$ and every $h \in \mathcal{R}$. Hence $\psi(\Phi_{\bar{x}}^T(h_0(\sigma))) = \psi(h_0(\sigma))$ for all $T \geq T_{\min}$: the representation reaching $\bar{\pi}$ at the decision step is independent of the horizon. Since ψ is separating and π is optimal at $T_{\min}$, $\bar{\pi}$ maps that representation to $a^*(\sigma, o_T)$ at every horizon, which is decision independence.

For cost independence, the memory is $\mathcal{H}$ and one application of Φ is performed per step, neither of which depends on T. For the requirement on precision, let $\Delta \coloneqq \min \|\psi(h_0(\sigma)) - \psi(h_0(\sigma'))\|$ over the pairs with $[\sigma] \neq [\sigma']$. The minimum is attained because Σ is finite, and it is positive by separation. Uniform continuity then yields a $\mu > 0$ such that $\|h - h'\| < \mu$ implies $\|\psi(h) - \psi(h')\| < \Delta$. Since ψ is constant on each orbit with values differing by at least Δ across classes, every pair of such orbits is at distance at least μ, so $\operatorname{sep} \geq \mu > 0$. □

D.1.2 PROOF OF THEOREM 3.4

Proof of Theorem 3.4. Monostability gives $\Phi_{\bar{x}}^n(h) \to h^\star$ for every $h \in \mathcal{H}$, so every orbit $\mathcal{R}_\sigma$ accumulates at $h^\star$. For any pair σ, σ' requiring different actions and any n,

$$\left\|\Phi_{\bar{x}}^n(h_0(\sigma)) - \Phi_{\bar{x}}^n(h_0(\sigma'))\right\| \leq \left\|\Phi_{\bar{x}}^n(h_0(\sigma)) - h^\star\right\| + \left\|h^\star - \Phi_{\bar{x}}^n(h_0(\sigma'))\right\| \xrightarrow[n\to\infty]{} 0,$$

so the infimum defining sep is 0 and cost independence fails. □

D.1.3 PROOF OF COROLLARY 3.5

Proof of Corollary 3.5. Let σ, σ' be cues with $[\sigma] \neq [\sigma']$. Both orbits converge to $h^\star$, so there is an $n \geq T_{\min}$ such that $\|\Phi_{\bar{x}}^n(h_0(\sigma)) - \Phi_{\bar{x}}^n(h_0(\sigma'))\| < \eta$. The policy then takes the same action on both states. One of the two actions is not optimal, so π is not optimal at the horizon n, and $T^\star(\pi) < n$. □

D.1.4 PROOF OF THEOREM 3.7

Proof of Theorem 3.7. Fix σ and let $k = k(\sigma)$. Basins of attraction are forward invariant, so $\Phi_{\bar{x}}^T(h_0(\sigma)) \in B_k$ for all $T \geq 0$. By assumption (ii), ψ is invariant within B_k, so $\psi(\Phi_{\bar{x}}^{T+1}(h_0(\sigma))) = \psi(\Phi_{\bar{x}}^T(h_0(\sigma)))$ for all $T \geq 0$, and the policy therefore produces the same action at every horizon. Since π is optimal at $T_{\min}$, it is then optimal for every σ and every $T \geq T_{\min}$, which is decision independence. For cost independence, the memory is $\mathcal{H}$ and one application of Φ is performed per step, neither of which depends on T. For the requirement on precision, let σ, σ' be cues requiring different actions, so $i = k(\sigma) \neq k(\sigma') = j$. The orbits $\mathcal{R}_\sigma \subseteq B_i$ and $\mathcal{R}_{\sigma'} \subseteq B_j$ have disjoint closures, since basins are disjoint and forward invariant and the orbits converge to A_i and A_j respectively. Disjoint compact sets are at positive distance, and there are finitely many pairs of cues, so $\mathrm{sep} > 0$. □

D.1.5 PROOF OF PROPOSITION 3.8

Proof of Proposition 3.8. By forward invariance, $h_t \in B_{k(\sigma)}$ for all $t \geq 0$: the memory state never leaves the region in which the key observation placed it, whatever the sequence of idle-phase observations and actions. By the invariance of ψ within B_k under every Φ_x,

$$\psi(h_T) = \psi(\Phi_{x_T}(h_{T-1})) = \psi(h_{T-1}) = \cdots = \psi(h_0(\sigma)).$$

The representation reaching $\bar{\pi}$ at the decision step is therefore independent of the horizon and of the idle-phase trajectory. Cost independence follows as in Theorem 3.7: cues with different optimal actions occupy regions at positive distance, so $\mathrm{sep} \geq \min_{i \neq j} \mathrm{dist}(B_i, B_j) > 0$. □

D.1.6 PROOF OF THEOREM 3.9

Proof of Theorem 3.9. Let σ_1, σ_2 be cues with $[\sigma_1] \neq [\sigma_2]$, and let $\mathcal{R}_i$ denote the closure of the set of states reachable from $h_0(\sigma_i)$ under arbitrary compositions of maps in $\mathcal{W}_\eta$.

Step 1. Any particular composition is a realization of the idle phase in $\mathcal{W}_\eta$, along which ψ is invariant by hypothesis. Hence ψ takes the value $\psi(h_0(\sigma_i))$ on the reachable set of every realization, and by continuity the same value c_i on $\mathcal{R}_i$. Let $\Delta = \|c_1 - c_2\|$, positive by separation.

Step 2. Uniform continuity gives a $\mu > 0$ such that $\|h - h'\| < \mu$ implies $\|\psi(h) - \psi(h')\| < \Delta$. Hence $\mathrm{dist}(\mathcal{R}_1, \mathcal{R}_2) \geq \mu > 0$, so $\mathcal{R}_1$ and $\mathcal{R}_2$ are disjoint. Being closed subsets of the compact $\mathcal{H}$, they are compact.

Step 3. Let $h \in \mathcal{R}_i$ and let e satisfy $\|e\| \leq \eta$ and $\Phi_{\bar{x}}(h) + e \in \mathcal{H}$. The map sending h to $\Phi_{\bar{x}}(h) + e$ and agreeing with $\Phi_{\bar{x}}$ elsewhere lies in $\mathcal{W}_\eta$, so $\Phi_{\bar{x}}(h) + e \in \mathcal{R}_i$ by construction of $\mathcal{R}_i$ and continuity of $\Phi_{\bar{x}}$. Thus $\big(\Phi_{\bar{x}}(\mathcal{R}_i) \oplus \bar{B}_\eta\big) \cap \mathcal{H} \subseteq \mathcal{R}_i$: the set $\Phi_{\bar{x}}(\mathcal{R}_i)$ lies in the relative interior of $\mathcal{R}_i$ in $\mathcal{H}$, and $\mathcal{R}_i$ is a trapping region.

Step 4. For a compact N with $\Phi_{\bar{x}}(N)$ in its relative interior and $\Phi_{\bar{x}}$ continuous, the set $A := \bigcap_{n \geq 0} \Phi_{\bar{x}}^n(N)$ is nonempty, compact and invariant, attracts every point of N, and has N as a neighborhood in $\mathcal{H}$. Applying this to $\mathcal{R}_1$ and $\mathcal{R}_2$ yields attractors $A_1 \subseteq \mathcal{R}_1$ and $A_2 \subseteq \mathcal{R}_2$, disjoint by Step 2, with basins containing $\mathcal{R}_1$ and $\mathcal{R}_2$. □

D.2 SCOPE OF THE RESULTS

Indexing of the idle phase. The idle map governs the $T - 1$ steps of the idle phase. At $t = T$, the memory is updated once more, by $D := \Phi_{x_T}$. For each decision observation o_T this is one fixed continuous map, which we absorb into the readout: $\psi \circ D$ is again a readout, continuous if ψ is, and when o_T can take several values, $\bar{\pi}$ also receives o_T. The policy therefore acts on $\Phi_{\bar{x}}^{T-1}(h_0(\sigma))$. We write $h_T = \Phi_{\bar{x}}^T(h_0(\sigma))$ for readability, since a shift of the iterate index by one leaves every asymptotic statement unchanged. Throughout, our analysis concerns the steps $t = 0, \ldots, T$; post-decision dynamics ($t > T$) are not modeled.

Our two results have different scopes. Lemma 3.2, the invariant-and-separating readout condition, is architecture-agnostic: it holds for any policy that factorizes as $\bar{\pi} \circ \psi$. Theorem 3.4, the monostability obstruction, additionally assumes the structure of Equation (1), namely a fixed-dimensional, state-autonomous idle map $\Phi_{\bar{x}} \colon \mathcal{H} \to \mathcal{H}$. This yields a classification of memory mechanisms rather than

a list of exceptions, and the classification is driven by the two requirements of Definition 2.2: a mechanism may fail horizon independence by losing the decision, by paying a cost that grows with the horizon, or by both.

Attention-based policies. A policy attending over its full context escapes Theorem 3.4: the analogue of h_{t-1} is the set of past representations, whose size grows with T, so there is no fixed-point map and the obstruction does not apply. A readout attending to the cue token is invariant and separating, and decision independence is therefore achievable without multistability. It is not, however, horizon independence.

Corollary D.1 (Attention is not delay-horizon independent)**.** *A policy attending over its full context satisfies decision independence, Definition 2.2(a), but violates cost independence, Definition 2.2(b): its memory footprint grows linearly and its per-step attention cost grows at least linearly in T. A policy attending over a context window of fixed length k restores cost independence but loses decision independence: for $T > k$ the key observation has left the window, so no readout can separate cues requiring different actions, and the effective horizon satisfies $T^\star \leq k$. In neither case is the policy delay-horizon independent.*

Attention therefore does not provide a second mechanism for horizon independence; it provides a trade between the two requirements. Unbounded context buys the decision at a cost that grows with the horizon, which is the resource growth the property was defined to exclude; bounded context buys constant cost at a horizon ceiling of a different origin than fading memory but with the same practical consequence, namely that the deployment horizon must be known and bounded at design time. This is consistent with the empirical picture reported by Ni et al. (2023), where transformer policies sustain memory lengths an order of magnitude beyond recurrent baselines, and then stop. Together with Theorems 3.4 and 3.7, this leaves multistability on a bounded-dimension state as the only mechanism satisfying both requirements robustly.

External memory and adaptive memory selection. Architectures that equip the agent with an addressable external register (Peshkin et al., 1999) or with a fixed-size memory stack under learned write and eviction decisions satisfy our assumptions and are instances of Theorem 3.7 rather than counterexamples. When the agent emits memory actions that set and clear the bits of a fixed-size register fed back as input, the update under the null action is the identity on the register, so each of the 2^b register configurations is a fixed point: this is multistability with 2^b attractors, realized by construction rather than learned, and with no transient component during the idle phase. The same reading applies to classical finite-state controllers.

D.3 Horizon generalization and delay-horizon independence

Horizon generalization (HG). Myers et al. (2025) study fully observed, goal-conditioned MDPs. A policy $\pi(a \mid s, g)$ is commanded with a goal g that is available at every step, and the horizon is the distance from s to g, measured by a learned quasimetric. Horizon generalization is the property that a policy optimal for nearby goals remains optimal for distant ones, and it follows from planning invariance: by the triangle inequality, acting toward a distant goal coincides with acting toward a nearer waypoint on the way to it. The horizon-extending operator is path relaxation, which roughly doubles the reachable distance.

Our setting is partially observed. The horizon is the number of steps between a key observation and a later decision, and the optimal action is independent of that number ((A3)). There is no goal and no metric, and the policy is not memoryless: it acts on h_t, a recurrent summary of the history. The horizon-extending operator is the idle map $\Phi_{\bar{x}}$, which adds one step.

HG assumes the goal is available at every step, so the information determining the optimal action is never absent from the observation; in our setting that information is presented once and must be carried. Because HG policies are memoryless, forgetting cannot occur there, whereas forgetting is the entire subject of DHI. Furthermore, Myers et al. (2025) explicitly exclude generalization to unvisited states, whereas in our setting the memory states occupied at long horizons are, by construction, never visited during training. DHI is therefore not HG applied to a sub-case. What the two properties share is a common shape: a property holding at short horizon extends to long horizon when the policy is invariant under the operator that extends the horizon.

Generalization versus independence. HG carries no resource requirement, which is appropriate for a memoryless goal-conditioned policy: there is nothing that could grow with the horizon. Our setting is the opposite case. The whole difficulty is that the information determining the action must be carried across the idle phase, and a policy can always carry it by storing more, so a property that constrained only the decision would be satisfied by the history buffer and would say nothing about memory mechanisms. Requirement (b) of Definition 2.2 closes that door and makes independence different from generalization. Fading-memory models demonstrably generalize over the horizon, over a finite range. They cannot be independent of the horizon in decision and in cost at once. Using one name for both would force every statement of our results into the form of a qualified failure, which is precisely the confusion we wish to avoid.

POMDP structure versus general recurrent memory. We separate what in our results comes from the POMDP structure and what comes from the memory itself. Theorem 3.4 is a statement about the recurrent memory map and holds for any fixed-dimensional recurrent state. The delayed-cue structure of Definition 2.1 makes the readout time-invariant (through (A2)) and makes the optimal action depend on σ (through (A3)), thereby turning the dynamical property into a statement about optimality.

D.4 Relation to state abstraction

Definition 3.1 requires ψ to separate cues only up to optimal-action equivalence. The induced partition of the cue set is exactly a π^*-irrelevance abstraction in the taxonomy of Li et al. (2006), the coarsest of the five abstractions considered there: it constrains only which cues induce distinct optimal actions, preserves no value function, is strictly coarser than a^*-irrelevance (which additionally preserves Q^* at the optimal action), and is not model-preserving.

The caveat that Li et al. (2006) attach to ϕ_{π^*}, namely that value-based learning under this abstraction may converge to a policy that is suboptimal in the ground MDP, is explicitly stated there not to preclude policy-search methods from recovering optimality. Our agents are trained with PPO, which learns π and V_π directly and never forms Q, so the caveat does not bind in our experiments.

The two frameworks answer complementary questions. State abstraction is static: it asks which state distinctions may be discarded without losing optimality. We ask a dynamical question: which recurrent dynamics can carry a retained distinction forward over an unbounded number of steps. The invariance condition on $\Phi_{\bar{x}}$ makes a π^*-irrelevance readout realizable at every horizon rather than only at the horizons seen during training.

E Recurrent Architectures: Dynamics and Steady States

This appendix collects the architectural detail summarized in Sections 2.1, 3.2 and 3.3: the recurrent families we consider, the steady-state computations behind the monostability and multistability claims, and the specific cells used in the experiments. Their update equations are listed in Section N.

E.1 Recurrent memory

RNNs encode time-dependencies in a recurrent hidden state h_t whose update depends on the previous hidden state h_{t-1} and current observation of a time-series x_t. It writes

$$h_t = \Phi(h_{t-1}, x_t), \qquad \forall t \geq 1, \tag{4}$$

where Φ is the update map of Equation (1), which depends on the learnable parameters θ of the network. Nonlinear versions such as LSTMs (Hochreiter & Schmidhuber, 1997) and GRUs (Cho et al., 2014) have dominated sequence modeling for decades, but many variants such as Janet (Van Der Westhuizen & Lasenby, 2018), (n)BRC (Vecoven et al., 2021), and MGU (Zhou et al., 2016) have appeared over the years. Due to their recurrent nature, RNNs can encode time dependencies in their time-varying dynamics (Equation (4)), creating fading memory of past inputs.

Nonlinear RNNs can also exhibit multistability, the coexistence of multiple possible state values $\bar{h}$ at equilibrium for the same constant input $\bar{x}$. An RNN is multistable if its steady-state equation

$$\bar{h} - \Phi(\bar{h}, \bar{x}) = 0, \tag{5}$$

admits multiple solutions for some set of constant input $\bar{x}$. Multistability allows for persistent memory under constant input value through a history-dependent convergence toward one of the multiple equilibria (Vecoven et al., 2021; Lambrechts et al., 2023). Most nonlinear RNNs can reach multistability for some set of learnable parameters θ (Lambrechts et al., 2023). In this work, we use the GRU, BRC and nBRC variants in our experiments (Section N).

The converse reading is equally informative. Widely deployed nonlinear cells such as the GRU and the LSTM retain state-dependent gating and can therefore realize multistability; nothing in our results excludes them from DHI. Our experiments show (Sections 3.3 and M) that multistability is reached only by a fraction of trained models, so that transferring these results to GRU or LSTM policies is a question of *training* procedures that foster multistability, not of architectural capacity.

E.2 Parallelizable architectures are monostable

Recent efforts have led to the emergence of novel RNN architectures that are parallelizable over the sequence length with competitive performance on sequential benchmarks. They are, without exception among the families below, strictly monostable. By Theorem 3.4, these architectures are therefore excluded from DHI *at every parameter value*, and have a finite effective horizon whatever the training procedure. Crucially, this verification is carried out in closed form on the update equation, without training a single model.

SSMs and linear RNNs exploit linear state dynamics to achieve parallelizable training, either through convolutional representations (Gu et al., 2021; 2022) or the parallel scan algorithm (Blelloch, 1989; Martin & Cundy, 2018; Gu & Dao, 2024). The update equation of arbitrary linear RNNs writes

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

where stability requires all eigenvalues of $A(x_t)$ to be contained within the unit circle in the complex plane. The steady-state solutions

$$\bar{h} = -(A(\bar{x}) - I)^{-1}B(\bar{x})\bar{x}$$

are explicit, meaning that only one *and only one* solution $\bar{h}$ exists for any input $\bar{x}$. Under the constant idle input $\bar{x}$, the idle map is the single affine map $\Phi_{\bar{x}}(h) = A(\bar{x})h + B(\bar{x})\bar{x}$, and when $\rho(A(\bar{x})) < 1$ its fixed point is globally attracting, since $\|\Phi^n_{\bar{x}}(h) - \bar{h}\| = \|A(\bar{x})^n(h - \bar{h})\| \to 0$. The idle map $\Phi_{\bar{x}}$ of such a model is therefore monostable in the sense of Definition 3.3, and Theorem 3.4 applies.

This argument covers selective SSMs such as Mamba (Gu & Dao, 2024) directly: their state update is linear in the state, with parameters that are input-dependent but *not* state-dependent, so it is exactly

of the form $h_t = A(x_t)h_{t-1} + B(x_t)x_t$. Under the constant idle input of Equation (2), A and B are constant and the steady state is unique. Selectivity modulates the *rate* at which information decays, and can make that rate very small, but it cannot create a second attractor: it lengthens $T^\star$ without making it infinite.

Gated RNNs can also achieve parallelizable training by removing the state dependency in their gating mechanisms, *i.e.*, computing gate values from the input alone, $z_t = \sigma(W_z x_t)$. Several such architectures have been proposed, including xLSTM (Beck et al., 2024), HGRN (Qin et al., 2023), MatMul-free LM (Zhu et al., 2025), GILR (Martin & Cundy, 2018), and minGRU (Feng et al., 2024), though some only achieve partial parallelization. It yields an update equation of the form

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t,$$

where $\tilde{h}_t$ depends only on x_t. The steady-state solutions reduce to

$$\bar{h} = \tilde{h} = \phi(W_h \bar{x})$$

where ϕ is an activation function whose structure is determined by the dynamical equation of $\tilde{h}_t$. The steady-state equation maps an explicit solution $\bar{h}$ to any input value $\bar{x}$, so these models are monostable as well. This result extends to gated RNNs with multiple gates, provided that no gate is state-dependent. In this work, we use minGRU as a representative linear gated RNN (Section N).

E.3 Memory recurrent units and the BMRU

Memory recurrent units (MRUs) represent a class of parallelizable RNNs with properties complementary to those of linear RNNs: they exhibit multistability but lack transient dynamics (*i.e.*, they assume instantaneous convergence) (De Geeter et al., 2026; Brandoit et al., 2026). At each timestep t and for a given input x_t, the MRU outputs a solution to the steady-state equation

$$\bar{h}_t - \Phi(\bar{h}_t, x_t) = 0.$$

When multiple solutions exist, the previous state $\bar{h}_{t-1}$ determines the selected solution, effectively modeling history-dependent behavior. A member of this family is the bistable memory recurrent unit (BMRU), which integrates learnable hysteretic dynamics into a gated architecture (De Geeter et al., 2026). In this work, we use the BMRU as a representative multistable model to contrast with the monostable nature of modern gated linear RNNs (Section N).

F T-MAZE ENVIRONMENT

We base our formal definition of the T-maze on the work of Lambrechts et al. (2023). The T-maze environment is a POMDP $(\mathcal{S}, \mathcal{A}, \mathcal{O}, p_0, P, \Omega, R, \gamma)$ parameterized by the maze length $L \in \mathbb{Z}^+$.

State space The discrete state space $\mathcal{S}$ is composed of the position of the agent $\mathcal{P}$ and the position of the goal $\mathcal{G}$, with

$$\begin{cases} \mathcal{S} = \mathcal{P} \times \mathcal{G} \\ \mathcal{P} = \{(0,0), \ldots, (L-1,0)\} \cup \{(L-1,-1), (L-1,1)\} \\ \mathcal{G} = \{-1, 1\}, \end{cases}$$

where the -1 and 1 goal positions represent up and down respectively. Additionally, there are four terminal states given by the set $\mathcal{F} = \{s_t = (p_t, g_t) \in \mathcal{S} \mid p_t \in \{(L-1,-1), (L-1,1)\}, g_t \in \mathcal{G}\}$.

Action space The discrete action space is composed of the four possible directions in which the agent can move and is defined as $\mathcal{A} = \{0, 1, 2, 3\}$, where the values correspond to the right, top, left, and down directions respectively.

Observation space The discrete observation space is composed of the initial observation of the goal at the first timestep, 0 otherwise, and a flag indicating whether the agent has reached the far right of the maze (1) or not (0). Formally, we have

$$\mathcal{O} = \{(g_t, f_t) \mid g_{t=0} \in \mathcal{G}, g_{t>0} = 0, f_t \in \{1, 0\}\}.$$

Initial state distribution The two possible initial states are $s_0^{up} = ((0,0), -1)$ and $s_0^{down} = ((0,0), 1)$. The initial state distribution $p_0 \colon \mathcal{S} \to [0,1]$ is given by

$$p_0(s_0) = \begin{cases} 0.5 & \text{if } s_0 = s_0^{up}, \\ 0.5 & \text{if } s_0 = s_0^{down}, \\ 0 & \text{otherwise.} \end{cases}$$

Transition distribution The transition distribution function $P \colon \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,1]$ is given by

$$P(s_{t+1} \mid s_t, a_t) = \delta_{f(s_t, a_t)}(s_{t+1}),$$

where $s_{t+1} \in \mathcal{S}$, $s_t \in \mathcal{S}$ and $a_t \in \mathcal{A}$, and f is given by

$$f(s_t, a_t) = \begin{cases} s_{t+1} = (p_t + (1,0), g_t) & \text{if } s_t \notin \mathcal{F}, p_t + (1,0) \in \mathcal{P}, a_t = 0, \\ s_{t+1} = (p_t - (1,0), g_t) & \text{if } s_t \notin \mathcal{F}, p_t - (1,0) \in \mathcal{P}, a_t = 2, \\ s_{t+1} = (p_t + (0,1), g_t) & \text{if } s_t \notin \mathcal{F}, p_t + (0,1) \in \mathcal{P}, a_t = 3, \\ s_{t+1} = (p_t - (0,1), g_t) & \text{if } s_t \notin \mathcal{F}, p_t - (0,1) \in \mathcal{P}, a_t = 1, \\ s_{t+1} = (p_t, g_t) & \text{otherwise,} \end{cases}$$

where $s_t = (p_t, g_t) \in \mathcal{S}$ and $a_t \in \mathcal{A}$.

Reward function The reward function $R \colon \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is given by

$$R(s_t, a_t, s_{t+1}) = \begin{cases} 4 & \text{if } s_t \notin \mathcal{F}, s_{t+1} \in \mathcal{F}, p_{t+1} = (L-1, g_0), \\ -0.1 & \text{if } s_t \notin \mathcal{F}, s_{t+1} \in \mathcal{F}, p_{t+1} = (L-1, r), r \in \mathcal{G}, r \neq g_0 \\ 0 & \text{otherwise,} \end{cases}$$

where $s_t = (p_t, g_t) \in \mathcal{S}$, $a_t \in \mathcal{A}$, $s_{t+1} = (p_{t+1}, g_{t+1}) \in \mathcal{S}$.

Observation distribution The observations are deterministic. The observation distribution $\Omega \colon \mathcal{O} \times \mathcal{S} \to [0,1]$ is given by

$$\Omega(o_t \mid s_t) = \begin{cases} 1 & \text{if } o_t = (g_t, \delta(p_t = L-1)), \\ 1 & \text{if } o_t = (0, \delta(p_t = L-1))), \\ 0 & \text{otherwise,} \end{cases}$$

where $o_t \in \mathcal{O}$ and $s_t = (p_t, g_t) \in \mathcal{S}$.

G MiniGrid-Memory Environment

The MiniGrid-Memory (Chevalier-Boisvert et al., 2023) environment is a POMDP $(\mathcal{S}, \mathcal{A}, \mathcal{O}, p_0, P, \Omega, R, \gamma)$ parameterized by a finite set of possible maze sizes $\mathcal{N} \subset \mathbb{Z}^+$ (odd, $n \geq 7$) and a view distance $v \in \mathbb{Z}^+$ (odd, $v \geq 3$). The maze is composed of a 3×3 start room connected by a straight hallway to a goal hallway containing two candidate goal cells, one of which is rewarding. Which goal is rewarding is cued to the agent by a colored marker placed at a fixed cell in the start room, so the agent must remember this cue while traversing the hallway in order to select the correct goal.

State space The discrete state space $\mathcal{S}$ is composed of the maze size n, the agent position p and heading δ, the index k of the cued (rewarding) goal, and the goal layout L, with

$$\mathcal{S} = \left\{ (n, p, \delta, k, L) \;\middle|\; \begin{array}{l} n \in \mathcal{N},\ \delta \in \{0,1,2,3\},\ k \in \{0,1\}, \\ p \in \mathcal{P}(n),\ L\colon \{0,1\} \to \mathcal{K}(n) \text{ a bijection} \end{array} \right\},$$

where, writing $c(n) = \lfloor n/2 \rfloor$,

$$\begin{cases} \mathcal{K}(n) = \{(n-2, c(n)-2), (n-2, c(n)+2)\} \\ \mathcal{P}(n) = \{1,2,3\} \times \{c(n)-1, c(n), c(n)+1\} \\ \qquad\quad \cup\ \{4, \dots, n-3\} \times \{c(n)\} \\ \qquad\quad \cup\ \{n-2\} \times \{1, \dots, n-2\} \end{cases}$$

The three sets composing $\mathcal{P}(n)$ are, respectively, the 3×3 start room, the horizontal hallway, and the vertical goal hallway; together they form the set of reachable cells of the maze. $\mathcal{K}(n)$ is the set of two candidate goal cells, located at the top and bottom of the goal hallway. The bijection L assigns a goal index $i \in \{0,1\}$ to the cell $L(i) \in \mathcal{K}(n)$ it occupies for the episode, and k is the index of the rewarding goal, cued to the agent at the fixed cell $q_{\text{cue}}(n) = (1, c(n)-1)$ in the start room (see Observation distribution below). There are terminal states given by the set $\mathcal{F} = \{s_t = (n_t, p_t, \delta_t, k_t, L_t) \in \mathcal{S} \mid p_t \in \mathcal{K}(n_t)\}$, reached whenever the agent occupies either goal cell.

Action space The discrete action space is composed of the three actions available to the agent and is defined as $\mathcal{A} = \{0, 1, 2\}$, where each value corresponds to turning left, turning right, and moving forward respectively.

Observation space The discrete observation space is composed of an egocentric local window of the cells surrounding the agent and the agent current heading. Formally, we have $\mathcal{O} = \mathcal{O}_c \times \mathcal{O}_\delta$ with $\mathcal{O}_c = \{0,1,2,3\}^{v \times v}$ and $\mathcal{O}_\delta = \{0,1,2,3\}$.

Initial state distribution At the start of an episode, the maze size is drawn uniformly, $n \sim \mathcal{U}(\mathcal{N})$; the goal layout is drawn uniformly among the $2! = 2$ assignments of goal indices to cells, $L \sim \mathcal{U}(\mathfrak{S}(\mathcal{K}(n)))$; the cued goal index is drawn uniformly, $k \sim \mathcal{U}(\{0,1\})$; the agent starts facing right, $\delta_0 = 0$, at a uniformly random cell along the maze central row, $p_0 = (x_0, c(n))$ with $x_0 \sim \mathcal{U}(\{1, \dots, n-2\})$. The initial state distribution $p_0\colon \mathcal{S} \to [0,1]$ is given by

$$p_0(s_0) = \begin{cases} \dfrac{1}{|\mathcal{N}|} \cdot \dfrac{1}{n-2} \cdot \dfrac{1}{2} \cdot \dfrac{1}{2} & \text{if } s_0 = (n, (x_0, c(n)), 0, k, L)\,, \\ 0 & \text{otherwise,} \end{cases}$$

for $n \in \mathcal{N}$, $x_0 \in \{1, \dots, n-2\}$, $k \in \{0,1\}$, $L \in \mathfrak{S}(\mathcal{K}(n))$.

Transition distribution The transition distribution function $P\colon \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,1]$ is deterministic and given by

$$P(s_{t+1} \mid s_t, a_t) = \delta_{f(s_t, a_t)}(s_{t+1}),$$

where $s_{t+1} \in \mathcal{S}$, $s_t = (n, p_t, \delta_t, k, L) \in \mathcal{S} \setminus \mathcal{F}$, $a_t \in \mathcal{A}$, and $f(s_t, a_t) = (n, p_{t+1}, \delta_{t+1}, k, L)$ with

$$\delta_{t+1} = \begin{cases} (\delta_t - 1) \bmod 4 & \text{if } a_t = 0, \\ (\delta_t + 1) \bmod 4 & \text{if } a_t = 1, \\ \delta_t & \text{if } a_t = 2, \end{cases} \qquad p_{t+1} = \begin{cases} p_t + e(\delta_t) & \text{if } a_t = 2 \text{ and } p_t + e(\delta_t) \in \mathcal{P}(n), \\ p_t & \text{otherwise,} \end{cases}$$

and $e\colon \{0,1,2,3\} \to \mathbb{Z}^2$ maps a heading to its unit direction vector, $e(0) = (1,0)$, $e(1) = (0,1)$, $e(2) = (-1,0)$, $e(3) = (0,-1)$ (right, down, left, up respectively). That is, turning changes the agent heading without moving it, and moving forward advances the agent by one cell along its current heading unless that cell lies outside $\mathcal{P}(n)$ (a wall), in which case the agent remains in place.

Reward function The reward function $R\colon \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is given by

$$R(s_t, a_t, s_{t+1}) = \begin{cases} 4 & \text{if } s_t \notin \mathcal{F},\ p_{t+1} = L(k), \\ -0.1 & \text{if } s_t \notin \mathcal{F},\ p_{t+1} = L(1-k), \\ 0 & \text{otherwise,} \end{cases}$$

where $s_t = (n, p_t, \delta_t, k, L) \in \mathcal{S}$, $a_t \in \mathcal{A}$, $s_{t+1} = (n, p_{t+1}, \delta_{t+1}, k, L) \in \mathcal{S}$. A reward of 4 is given for entering the cued goal $L(k)$, a reward of -0.1 for entering the other goal $L(1-k)$, and reaching either goal ends the episode.

Observation distribution The observations are deterministic. Writing $s_t = (n, p_t, \delta_t, k, L) \in \mathcal{S}$, the observation $o_t = (c_t, \delta_t) \in \mathcal{O}$ is given by $\Omega(o_t \mid s_t) = 1$. The heading component of the observation is simply δ_t, i.e. the agent heading is fully observed at every timestep. The local window $c_t \in \{0,\dots,3\}^{v \times v}$ is centered $d = \lfloor v/2 \rfloor$ cells ahead of the agent along its heading, covering the world cells $\{p_t + d \cdot e(\delta_t) + \omega \mid \omega \in \{-d,\dots,d\}^2\}$, each labeled by the labeling function $\kappa(\cdot\,; n, k, L)\colon \mathbb{Z}^2 \to \{0,\dots,3\}$,

$$\kappa(q; n, k, L) = \begin{cases} 2+i & \text{if } q = L(i) \text{ for some } i \in \{0,1\}, \\ 2+k & \text{if } q = q_{\mathrm{cue}}(n), \\ 1 & \text{else if } q \notin \mathcal{P}(n), \\ 0 & \text{otherwise,} \end{cases}$$

i.e. 0 denotes an empty cell, 1 a wall, and $2+i$ an object of color i, either an actual goal cell or, at the fixed cue cell $q_{\mathrm{cue}}(n)$, the marker indicating which color is rewarding this episode. The window is expressed in an *egocentric* frame: it is rotated according to the agent current heading δ_t so that the agent forward direction always maps to the same fixed array axis, irrespective of its true heading in the maze.

H MINIGRID-GOTODOOR ENVIRONMENT

The MiniGrid-GoToDoor (Chevalier-Boisvert et al., 2023) environment is a POMDP $(\mathcal{S}, \mathcal{A}, \mathcal{O}, p_0, P, \Omega, R, \gamma)$ parameterized by a finite set of possible room sizes $\mathcal{N} \subset \mathbb{Z}^+$ and a view distance $v \in \mathbb{Z}^+$ (odd, $v \geq 3$). The agent navigates an $n \times n$ room ($n \in \mathcal{N}$, resampled every episode) with four doors, one at the midpoint of each wall; one door is designated as the goal. The identity (index) of each door is randomized independently of its location every episode, so the agent must explore the room, observe the doors it encounters, and match their index against the goal index given at the start of the episode.

State space The discrete state space $\mathcal{S}$ is composed of the room size n, the agent position p and heading δ, the door layout D, and the goal door index g, with

$$\mathcal{S} = \left\{ (n, p, \delta, D, g) \;\middle|\; \begin{array}{l} n \in \mathcal{N},\ \delta \in \{0,1,2,3\},\ g \in \{0,1,2,3\}, \\ p \in \mathcal{P}(n),\ D\colon \{0,1,2,3\} \to \mathcal{W}(n) \text{ a bijection} \end{array} \right\},$$

where

$$\begin{cases} \mathcal{W}(n) = \{(0, \lfloor n/2 \rfloor), (\lfloor n/2 \rfloor, n-1), (n-1, \lfloor n/2 \rfloor), (\lfloor n/2 \rfloor, 0)\} \\ \mathcal{P}(n) = \{1, \ldots, n-2\}^2 \cup \mathcal{W}(n) \end{cases}$$

$\mathcal{W}(n)$ is the set of four door slots, located at the midpoint of the west, south, east and north walls respectively, and $\mathcal{P}(n)$ is the set of reachable cells, i.e. the interior of the room together with the four door slots. The bijection D assigns a door index $i \in \{0,1,2,3\}$ to the slot $D(i) \in \mathcal{W}(n)$ it occupies for the episode, and g is the index of the rewarding (goal) door, located at $D(g)$. Additionally, there are terminal states given by the set $\mathcal{F} = \{s_t = (n_t, p_t, \delta_t, D_t, g_t) \in \mathcal{S} \mid p_t \in \mathcal{W}(n_t)\}$, reached whenever the agent occupies any door slot. Independently of $\mathcal{F}$, an episode is truncated whenever $t = T$.

Action space The discrete action space is composed of the three actions available to the agent and is defined as $\mathcal{A} = \{0,1,2\}$, where each value corresponds to turning left, turning right, and moving forward respectively.

Observation space The discrete observation space is composed of the initial observation of the goal door index at the first timestep, 0 otherwise, an egocentric local window of the cells surrounding the agent, and the agent current heading. Formally, we have $\mathcal{O} = \mathcal{O}_g \times \mathcal{O}_c \times \mathcal{O}_\delta$ with $\mathcal{O}_g = \{0,1,2,3,4\}$, $\mathcal{O}_c = \{0,\ldots,5\}^{v \times v}$, and $\mathcal{O}_\delta = \{0,1,2,3\}$.

Initial state distribution At the start of an episode, the room size is drawn uniformly, $n \sim \mathcal{U}(\mathcal{N})$; the door layout is drawn uniformly among the $4! = 24$ assignments of door indices to slots, $D \sim \mathcal{U}(\mathfrak{S}(\mathcal{W}(n)))$; the goal door index and initial heading are drawn uniformly, $g \sim \mathcal{U}(\{0,1,2,3\})$ and $\delta_0 \sim \mathcal{U}(\{0,1,2,3\})$; and the agent starts at the center of the room, $p_0 = (\lfloor n/2 \rfloor, \lfloor n/2 \rfloor)$. The initial state distribution $p_0\colon \mathcal{S} \to [0,1]$ is given by

$$p_0(s_0) = \begin{cases} \dfrac{1}{|\mathcal{N}|} \cdot \dfrac{1}{24} \cdot \dfrac{1}{4} \cdot \dfrac{1}{4} & \text{if } s_0 = (n, (\lfloor n/2 \rfloor, \lfloor n/2 \rfloor), \delta, D, g)\,, \\ 0 & \text{otherwise,} \end{cases}$$

for $n \in \mathcal{N}$, $\delta \in \{0,1,2,3\}$, $D \in \mathfrak{S}(\mathcal{W}(n))$, $g \in \{0,1,2,3\}$.

Transition distribution The transition distribution function $P\colon \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,1]$ is deterministic and given by

$$P(s_{t+1} \mid s_t, a_t) = \delta_{f(s_t, a_t)}(s_{t+1}),$$

where $s_{t+1} \in \mathcal{S}$, $s_t = (n, p_t, \delta_t, D, g) \in \mathcal{S} \setminus \mathcal{F}$, $a_t \in \mathcal{A}$, and $f(s_t, a_t) = (n, p_{t+1}, \delta_{t+1}, D, g)$ with

$$\delta_{t+1} = \begin{cases} (\delta_t - 1) \bmod 4 & \text{if } a_t = 0, \\ (\delta_t + 1) \bmod 4 & \text{if } a_t = 1, \\ \delta_t & \text{if } a_t = 2, \end{cases} \qquad p_{t+1} = \begin{cases} p_t + e(\delta_t) & \text{if } a_t = 2 \text{ and } p_t + e(\delta_t) \in \mathcal{P}(n), \\ p_t & \text{otherwise,} \end{cases}$$

and $e\colon \{0,1,2,3\} \to \mathbb{Z}^2$ maps a heading to its unit direction vector, $e(0) = (1,0)$, $e(1) = (0,1)$, $e(2) = (-1,0)$, $e(3) = (0,-1)$ (right, down, left, up respectively). That is, turning changes the

agent heading without moving it, and moving forward advances the agent by one cell along its current heading unless that cell lies outside $\mathcal{P}(n)$ (a wall), in which case the agent remains in place.

Reward function The reward function $R\colon \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is given by

$$R(s_t, a_t, s_{t+1}) = \begin{cases} 1 & \text{if } s_t \notin \mathcal{F},\ s_{t+1} \in \mathcal{F},\ p_{t+1} = D(g), \\ 0 & \text{otherwise,} \end{cases}$$

where $s_t = (n, p_t, \delta_t, D, g) \in \mathcal{S}$, $a_t \in \mathcal{A}$, $s_{t+1} = (n, p_{t+1}, \delta_{t+1}, D, g) \in \mathcal{S}$. A unit reward is given only for entering the goal door $D(g)$; reaching any other door also ends the episode but yields no reward.

Observation distribution The observations are deterministic. Writing $s_t = (n, p_t, \delta_t, D, g) \in \mathcal{S}$, the observation $o_t = (g_t, c_t, \delta_t) \in \mathcal{O}$ is given by $\Omega(o_t \mid s_t) = 1$ where

$$g_t = \begin{cases} g + 1 & \text{if } t = 0, \\ 0 & \text{if } t > 0. \end{cases}$$

The heading component of the observation is simply δ_t, i.e. the agent heading is fully observed at every timestep. The local window $c_t \in \{0, \dots, 5\}^{v \times v}$ is centered $d = \lfloor v/2 \rfloor$ cells ahead of the agent along its heading, covering the world cells $\{p_t + d \cdot e(\delta_t) + \omega \mid \omega \in \{-d, \dots, d\}^2\}$, each labeled by the labeling function $\kappa(\cdot\,; n, D)\colon \mathbb{Z}^2 \to \{0, \dots, 5\}$,

$$\kappa(q; n, D) = \begin{cases} 2 + i & \text{if } q = D(i) \text{ for some } i \in \{0, 1, 2, 3\}, \\ 1 & \text{else if } q_x \le 0 \text{ or } q_x \ge n - 1 \text{ or } q_y \le 0 \text{ or } q_y \ge n - 1, \\ 0 & \text{otherwise,} \end{cases}$$

i.e. 0 denotes an empty cell, 1 a wall, and $2 + i$ door i. The window is expressed in an *egocentric* frame: it is rotated according to the agent current heading δ_t so that the agent forward direction always maps to the same fixed array axis, irrespective of its true heading in the room.

I Training Procedure and Hyperparameters

Design of the experiments. For each environment we use two complementary approaches to test the connection between DHI and RNN dynamics. In the first, we train nonlinear RNNs in the usual way and compare performance and generalization with multistability measured a posteriori, which keeps the evaluation unbiased. We then reproduce the experiments with minGRU as a parallelizable monostable RNN with transient dynamics and BMRU as a parallelizable bistable RNN without transient dynamics, which lets us assess the role of each separately.

Network size. The recurrent layers are small by design: 5 units on the T-maze, 64–128 elsewhere. This is a choice, not a limitation of the analysis. It keeps the dynamical characterization (VAA, basin structure) interpretable, and it makes the point that multistability yields DHI at very small cost, whereas fading memory needs substantially larger networks to buy a longer, and still bounded, effective horizon. Control experiments with larger networks are reported in Section 3.2.

Training behavior. Models trained on short T-mazes reach an optimal policy within a few training steps and with almost no variance across seeds. Models trained on T-mazes of length 100 struggle to learn the task and show high variance; learning curves are given in Section M. This is expected in an RL setting, since exploration is far harder in a long corridor, which destabilizes training.

Given that the environments used in this work are partially observable Markov decision processes, the optimal policies and value functions are dependent on the sequence of all observations and actions (*i.e.*, the history). We use the proximal policy optimization (PPO) (Schulman et al., 2017) algorithm, an advantage actor-critic (Mnih et al., 2016b) method, to approximate the optimal policy and the value function. The policy and the value function (π and VF respectively) are implemented with RNNs. Nothing is shared between the policy and the value function, so the history is encoded separately by both RNNs.

The policy neural network outputs a probability distribution over the action space from which the next action is sampled to encourage exploration at the beginning of the training and exploitation at the end as the distribution entropy decreases.

All models were trained using a modified version of the LSTM PPO implementation from CleanRL (Huang et al., 2022).

The architecture shown in Figure 5 was considered for the T-maze experiments shown in Figure 2 with the exception of the deep minGRU.

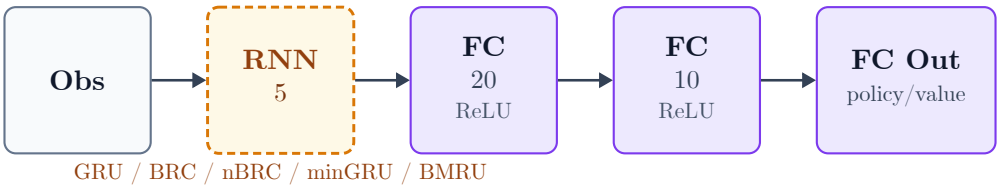

Figure 5: Architecture used for the T-maze experiments shown in Figure 2 (except for the deep minGRU). FC is a fully connected (or dense) layer. RNN is a recurrent layer and is either GRU, BRC, nBRC, minGRU, or BMRU depending on the model. Since the policy and the value function use non-shared distinct networks, the FC Out layer contains 4 neurons for the policy and 1 for the value function.

For the deep minGRU experiment shown in Figure 2, the deeper architecture shown in Figure 6 was considered with $N = 4$. This architecture was also used for the experiments shown in Figure 3 and Figure 4 with $N = 1$ for minGRU and BMRU, and $N = 4$ for deep minGRU.

The hyperparameters used for each experiment are shown in Table 1.

For the T-maze experiments shown in Figure 2, the following additional hyperparameters were used: $\text{LR}_\pi = 0.005$, $\text{LR}_{\text{VF}} = 0.005$, PPO clip range $\epsilon = 0.2$, Entropy coefficient $c_2 = 0.01$. For the other experiments, the hyperparameter sweeps are detailed in Section J.

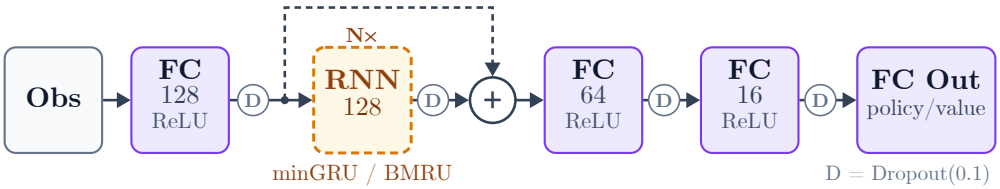

Figure 6: Architecture used for the deep minGRU in Figure 2 and for all models in Figure 3 and Figure 4. FC is a fully connected (or dense) layer. RNN is N recurrent layers in sequence which are either minGRUs or BMRUs depending on the model, with $N = 1$ for minGRU and BMRU, and $N = 4$ for deep minGRU. Since the policy and the value function use non-shared distinct networks, the FC Out layer contains various neurons depending on the environment for the policy and 1 for the value function. Each layer is followed by a Dropout layer with probability $p = 0.1$.

Table 1: Hyperparameters used during training for the three benchmarks.

Hyperparameter	T-maze	MiniGrid (Memory / GoToDoor)
π epochs	20	10
VF epochs	10	10
π optimizer	Adam	Adam
VF optimizer	Adam	Adam
LR scheduler	Cosine annealing ($T_{max} = 250$)	Cosine annealing ($T_{max} = 250$)
Grad. clip norm	1	1
VF coeff. c_1	1	1
GAE (λ)	0.98	0.98
Discount (γ)	0.998	0.998
Parallel envs	50	2048
Minibatch (N $\times$ size)	2×25	8×256
KL_{max} early stop	0.2	0.2
Sample steps	1400	100
Total iterations	250	250

Compute resources All training experiments were performed on an academic HPC cluster, using a single NVIDIA Tesla A100 GPU (40 GB VRAM) per run on a node with AMD EPYC Milan CPUs and up to 512 GB of system RAM. The total compute footprint of the project, including preliminary experiments not reported in this paper, remains modest by modern deep learning standards.

J HYPERPARAMETER SWEEP

To check that our conclusions do not depend on a particular hyperparameter choice, we performed a random search over the four hyperparameters most likely to affect how much information a recurrent policy retains: the policy and value function learning rates, the PPO clip range ϵ, and the entropy coefficient c_2. All other settings were held at the values of Table 1. The ranges of the hyperparameters are shown in Table 2. For each environment and model, we selected the hyperparameter configuration achieving the highest average reward, breaking ties by the highest average instantaneous reward. The resulting values are shown in Table 3.

Table 2: Random-search space used for the hyperparameter sweep.

Hyperparameter	Distribution	Range
Learning rate (π and VF)	log-uniform	$[10^{-6}, 10^{-2}]$
Clip range ϵ	uniform	[0.1, 0.3]
Entropy coefficient c_2	log-uniform	$[10^{-4}, 2 \times 10^{-2}]$

Table 3: Hyperparameters selected via random search for each environment–model pair.

Environment	Model	π_{lr}	VF_{lr}	ϵ	c_2
Memory	minGRU	4.423×10^{-3}	2.015×10^{-3}	0.134	1.930×10^{-4}
	BMRU	1.408×10^{-3}	8.917×10^{-3}	0.191	3.438×10^{-3}
	Deep minGRU	9.202×10^{-4}	5.585×10^{-4}	0.155	1.209×10^{-2}
GoToDoor	minGRU	3.708×10^{-3}	1.219×10^{-3}	0.198	3.204×10^{-3}
	BMRU	1.423×10^{-3}	5.254×10^{-5}	0.233	2.603×10^{-4}
	Deep minGRU	7.313×10^{-4}	1.620×10^{-3}	0.247	4.261×10^{-3}

The outcome is the one predicted by Theorem 3.4: hyperparameter optimization changes where a monostable model stops generalizing, but not whether it stops. No sampled configuration produced a monostable model with an unbounded effective horizon, which is expected, since monostability is a property of the update equation and not of the training procedure.

K MEASURING THE EFFECTIVE HORIZON

Definition 2.3 defines the effective horizon $T^\star$ of a policy as the largest horizon up to which it remains optimal. Since returns are stochastic and optimality is not exactly attained in practice, the following empirical surrogate can be used. Let $R(T)$ be the mean return of a policy evaluated at horizon T over n episodes, R_{opt} the optimal return, and R_{rand} the return of a uniformly random decision at the junction. We define the normalized score

$$\hat{R}(T) = \frac{R(T) - R_{\mathrm{rand}}}{R_{\mathrm{opt}} - R_{\mathrm{rand}}} \in (-\infty, 1],$$

and set

$$T^\star = \max \left\{ T \in \mathcal{G} \mid \hat{R}(T') \geq 1 - \kappa \text{ for all } T' \in \mathcal{G}, T' \leq T \right\},$$

where $\mathcal{G}$ is the evaluated grid of horizons and κ a tolerance. A model whose score never drops below the threshold on the whole evaluated grid is reported as $T^\star > \max \mathcal{G}$; this is a lower bound, and is the expected outcome for a model achieving DHI, since by Theorem 3.7 its score is horizon-independent.

For the experiments shown in Figure 3, we evaluated the policies at every horizon in $\mathcal{G} = \{1, \ldots, 10^6\}$ with $\kappa = 0$. Each policy was run for 2 episodes, one per goal direction.

L COMPUTATION OF THE VAA

Classifying trained models. Architecture fixes the dynamical regime for affine and input-gated cells, but a trained nonlinear cell must be classified after the fact. We do so with the VAA: a value at the bottom of the range indicates a single attractor, that is, monostability, and a value at the top indicates that distinct initial conditions settle on distinct attractors. The monostable population reported in Section 3.2 is therefore of two kinds, minGRU, monostable by construction, and those GRU, BRC and nBRC seeds whose trained parameters land in the monostable regime; the multistable population of Section 3.3 is the mirror image, with BMRU multistable by construction.

The variability among attractors (VAA) quantifies the level of multistability of an RNN Φ by measuring the number of different stable states reached by this model from a set of initial states $\mathcal{H}_0$ under a constant input x. It returns the normalized number of different stable states, *i.e.*, a scalar between $\frac{1}{|\mathcal{H}_0|}$, when there is only one stable state, and 1, when all stable states are different. Mathematically, the VAA is defined as:

$$\mathrm{VAA}(\Phi, \mathcal{H}_0, x) = \frac{1}{|\mathcal{H}_0|} \sum_{i=1}^{|\mathcal{H}_0|} \frac{1}{\sum_{j=1}^{|\mathcal{H}_0|} \mathbf{1}\left(\limsup_{n\to\infty} \|\Phi_x^n(h_i) - \Phi_x^n(h_j)\| = 0\right)},$$

where $\mathbf{1}(\cdot)$ is the indicator function and Φ_x^n is the n-fold composition of $\Phi_x = \Phi(\cdot, x)$.

However, the rigorous definition of the VAA implies computing limits to infinity, making it unusable in practice. Therefore, an approximation of the VAA is used in the experiments. This approximation introduces two parameters: M, the number of iterations of Φ_x, and ϵ, the tolerance to consider two states as identical. Mathematically, it gives:

$$\mathrm{VAA}(\Phi, \mathcal{H}_0, x; M, \epsilon) = \frac{1}{|\mathcal{H}_0|} \sum_{i=1}^{|\mathcal{H}_0|} \frac{1}{\sum_{j=1}^{|\mathcal{H}_0|} \mathbf{1}\left(\|\Phi_x^M(h_i) - \Phi_x^M(h_j)\| \leq \epsilon\right)}.$$

When computing VAA to a specific task, the set of initial states $\mathcal{H}_0$ and the constant input x have to be defined. In this work, we consider the classic T-maze task. For the classic T-maze, we simply used the hidden states obtained at the first timestep after giving the position of the reward to the agent. As only two positions are allowed, $|\mathcal{H}_0| = 2$. The constant input x is simply the observation given to the agent while it has not reached the junction. In this work, we used $M = 3000$ and $\epsilon = 10^{-5}$, values large and small enough, respectively, that the convergence analysis of the hidden state was unaffected by further adjustment.

M LEARNING CURVES AND RNN DYNAMICS

Once horizon independence is secured, the training horizon can be chosen for convenience rather than for coverage, and a short one is the better choice: it is cheaper, and it avoids the slow transients that a multistable model trained at $T = 100$ develops and never needs, which then cost it accuracy on the short settings. This appendix reports this comparison.

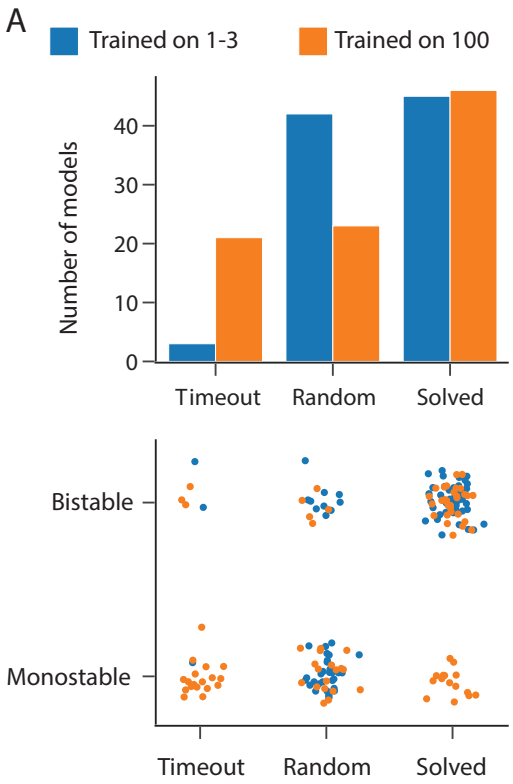

Figure 7: **Training horizon and the incidence of multistability.** Top, number of models that timed out, reached the end but acted randomly, or solved the T-maze of size 100. Bottom, the same models separated by their performance and by their monostable/bistable character. Blue, models trained on T-mazes of size 1–3; orange, models trained on T-mazes of size 100.

Training horizon and the incidence of multistability. Figure 7 (bottom) categorizes each model by its performance on the long T-maze (timeout, random or solved) and by its stability properties. For models trained on short T-mazes, the correlation between multistability and solving the environment is clear: no monostable model trained at $T \in [1, 3]$ solves $T = 100$, whereas most bistable ones do. Models trained directly at $T = 100$ solve it using either slow transient dynamics or multistability, and fewer of them reach bistability at all (36/90, against 57/90 at the short horizon), indicating that training on long horizons does not itself promote encoding through multistability.

Transient dynamics and the training horizon. Multistable models trained at the two horizons behave differently at the short end of the evaluation range (Figure 2A, bottom). Models trained at $T \in [1, 3]$ are optimal over the whole range. Models trained at $T = 100$ remain optimal at long horizons, as Theorem 3.7 requires, but lose accuracy on the shortest mazes. The reason is that even a multistable cell needs transient dynamics to settle into a basin, and those transients are considerably slower when the model is trained at a long horizon, where nothing penalizes a slow approach to equilibrium (Figure 8B). At short horizons, the decision step then arrives before the state has settled, in a regime the model never saw during training, and the readout misfires. Training at the short horizon avoids the problem entirely: the transients are fast because they had to be, and the multistability that carries the cue is unaffected by the horizon at which it was learned.

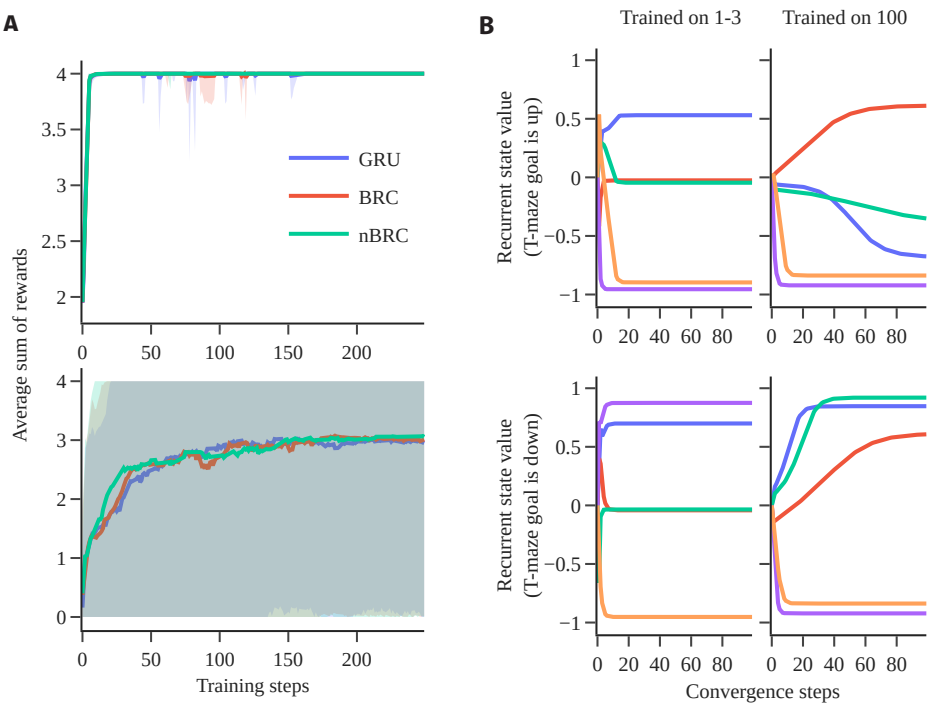

Figure 8: **A.** Evolution of the average sum of rewards (with 2σ) obtained by the agents during their training on T-mazes of lengths 1-3 (top), and of length 100 (bottom). **B.** Time evolution of the internal states of two bistable models, one trained on small T-mazes (left), the other trained on long T-mazes (right), for the two possible cases on a T-maze of length 100. The figure shows that models trained on long T-mazes have slower transient dynamics before reaching equilibrium.

N EQUATIONS FOR THE DIFFERENT RNNS USED IN THIS WORK

The equations for the different models used in this paper are provided below. In each equation, W are learnable weight matrices, b and w are learnable parameter vectors, x_t is the input, h_t is the state, and r_t, z_t and n_t are gate variables. $\mathrm{H}(\cdot)$ denotes the Heaviside step function, and $\mathrm{S}(\cdot)$ is the sign function.

N.1 GRU (CHO ET AL., 2014)

$$
\begin{aligned}
r_t &= \sigma(W_{xr}x_t + W_{hr}h_{t-1} + b_r) \\
z_t &= \sigma(W_{xz}x_t + W_{hz}h_{t-1} + b_z) \\
n_t &= \tanh(W_{xn}x_t + r_t \odot W_{hn}h_{t-1} + b_n) \\
h_t &= z_t \odot h_{t-1} + (1 - z_t) \odot n_t
\end{aligned}
$$

N.2 BRC (VECOVEN ET AL., 2021)

$$
\begin{aligned}
r_t &= 1 + \tanh(W_r x_t + w_r \odot h_{t-1} + b_r) \\
z_t &= \sigma(W_z x_t + w_z \odot h_{t-1} + b_z) \\
n_t &= \tanh(W_n x_t + r_t \odot h_{t-1} + b_n) \\
h_t &= z_t \odot h_{t-1} + (1 - z_t) \odot n_t
\end{aligned}
$$

N.3 NBRC (VECOVEN ET AL., 2021)

$$
\begin{aligned}
r_t &= 1 + \tanh(W_{xr}x_t + W_{hr}h_{t-1} + b_r) \\
z_t &= \sigma(W_{xz}x_t + W_{hz}h_{t-1} + b_z) \\
n_t &= \tanh(W_n x_t + r_t \odot h_{t-1} + b_n) \\
h_t &= z_t \odot h_{t-1} + (1 - z_t) \odot n_t
\end{aligned}
$$

N.4 MINGRU (FENG ET AL., 2024)

$$
\begin{aligned}
z_t &= \sigma(W_z x_t + b_z) \\
n_t &= W_n x_t + b_n \\
h_t &= z_t \odot h_{t-1} + (1 - z_t) \odot n_t
\end{aligned}
$$

N.5 BMRU (DE GEETER ET AL., 2026)

$$
\begin{aligned}
n_t &= W_n x_t + b_n \\
\beta_t &= |W_\beta x_t + b_\beta| \\
z_t &= \mathrm{H}(|n_t| - \beta_t) \\
h_t &= z_t \odot \mathrm{S}(n_t) \odot \alpha + (1 - z_t) \odot h_{t-1}
\end{aligned}
$$

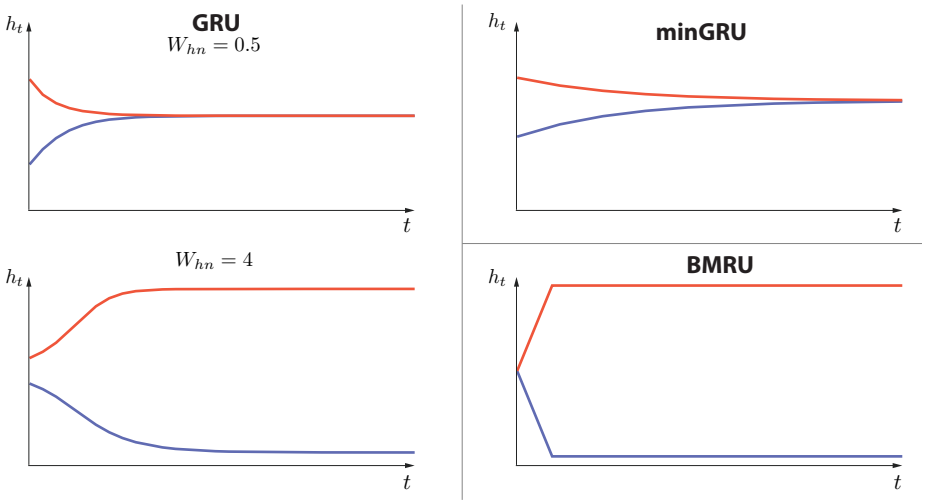

Figure 9: Left, time evolution of the internal state of a single GRU for two different initial conditions (red and blue curves) and two different values of W_{hn}. The top graph shows a monostable behavior: both initial conditions reach the same equilibrium. The bottom graph shows a bistable behavior: each initial condition reaches a different equilibrium. Right, time evolution of the internal state of a single minGRU (top) and BMRU (bottom) for two different initial conditions (red and blue curves). minGRU exhibits monostable behavior with transient dynamics, whereas BMRU exhibit bistable behavior with a one step convergence.