PLANNING AS EMERGENT BEHAVIOR IN REINFORCEMENT LEARNING WITH RELATIONAL HIDDEN STATES

Armin Sommer
ETH Zurich, Switzerland
sommera@ethz.ch

ABSTRACT

Reinforcement learning is conventionally divided into *model-based* and *model-free* methods. In this taxonomy, model-based methods perform lookahead planning over a learned world model, whereas model-free methods learn a reactive state–action mapping. Recent work, however, has shown that planning can emerge from model-free reinforcement learning alone. The conditions under which this behavior emerges from a pure reward-maximization objective have so far remained unclear. In this paper, we present evidence that, in the observed cases, the hidden-state structure of the neural architecture is the deciding factor. We find that a network of relational hidden states, each anchored to an environment state and exchanging messages along learned relations, acquires a planning mechanism. These hidden states recover the environment's transition structure in their learned relations, and improve the policy at decision time by planning over the learned graph. In a matched control agent that must additionally discover which cells represent which states, no such binding arises, and no planning follows from it. We argue that this explains the observed phenomenon of emergent planning in model-free reinforcement learning and raises the question of how common such emergent planning might be more generally. Finally, we hypothesize that the discovered mechanism could describe how planning emerges from pure reward maximization in the human brain through a neural architectural prior.

1 INTRODUCTION

Reinforcement learning (RL) methods are conventionally divided into two families based on whether the agent learns the dynamics of its environment [31]. *Model-based* methods learn, or are supplied with, a model of the transition dynamics and use it to *plan*: at decision time the agent searches over the model, simulating the consequences of candidate action sequences and selecting an action on that basis. Examples include tree-search agents such as MuZero [25] and its predecessor AlphaGo Zero [28]. *Model-free* methods instead learn their behavior directly from rewards without explicitly modelling their environment's dynamics. The resulting policy is a reactive mapping from states to actions: it is evaluated in a single forward pass, simulates no future, and represents no knowledge of the dynamics. Under this taxonomy, planning is the province of the model-based family, and the model-free agent is characterized precisely by its absence.

However, it has been observed that model-free agents can nonetheless learn to plan. Guez et al. [9] train a Deep Repeated ConvLSTM (DRC), a generic recurrent network composed of convolutional hidden states with no structure specific to planning. Using a standard model-free RL algorithm, they find that it exhibits the characteristics typically associated with a model-based planner: it generalizes across combinatorial, irreversible state spaces, is data-efficient, and, most diagnostically, improves its performance when granted additional test-time computation, or "thinking time," before acting. On combinatorial domains such as Sokoban, it matches or exceeds model-based agents. These behavioural signatures indicate that the model-based/model-free boundary is less sharp than the taxonomy implies: a network with no explicit planning machinery can come to behave as though it deliberates.

Subsequent work has established that this behavior reflects an internal planning process rather than a collection of reactive heuristics. Using concept-based probing, Bush et al. [2] decode from a

Sokoban-playing DRC two spatially-localized, square-level quantities, namely the direction from which the agent will next enter a square and the direction in which it will next push a box, and show these assemble into coherent planning trajectories. The authors find that these representations are progressively refined over the network's internal computation steps, that they transfer to out-of-distribution levels, and that intervening on them causally redirects the agent's behavior. Finally, they show that the decoded procedure resembles a parallelized bidirectional search rather than the forward rollouts of conventional model-based planners. Taufeeque et al. [33] reach a consistent conclusion by reverse-engineering a similar convolutional recurrent agent, identifying an iterative computation that propagates plan information across the spatial state, later refined into a description in terms of path channels and plan-extension kernels [34]. Moreover, the phenomenon is not confined to recurrent architectures: Bush et al. [2] report the same planning signatures in a feedforward 24-layer ResNet, whose plan representations sharpen across successive layers as the DRC's sharpen across recurrent iterations, consistent with the view that residual networks perform unrolled iterative estimation [8].

Taken together, these results demonstrate that planning can emerge under model-free reinforcement learning and characterize *what* algorithm a particular trained agent comes to implement. They leave open a more general question: under what conditions, and in particular owing to which property of the function approximator, does planning emerge at all? Bush et al. [2] raise precisely this question and leave it unresolved. The present work addresses it.

We identify the responsible property as a *relational hidden state*: a recurrent state organized as a set of cells whose updates depend on learned pairwise relations among them. Convolution and attention are two instances of such a relational, graph-structured operation: convolution relates each cell to a fixed local neighbourhood, whereas attention relates cells through learned, content-dependent weights; both are expressible as message passing over a graph defined on the cells Battaglia et al. [1]. Relational structure of this kind has previously been used to improve reasoning in reinforcement learning [37] and, more broadly, to learn the interactions and dynamics of structured systems [24; 16].

We hypothesize that emergent planning requires one architectural prior: an *anchoring* that binds each environment state to its own hidden cell. Given such binding, the messages exchanged between anchored cells can come to respect the environment's transition relation, and successive updates then propagate decision-relevant information along the induced state graph: one hop per update in the convolutional case, a global refinement of the whole field in the attention case. Either way, the aggregate computation is a multi-step lookahead in latent space that deepens with each update, accounting for the gains these agents show under added test-time computation.

In this work we test this hypothesis directly. We train an attention-based agent whose relational core is a dense, unmasked attention recurrence, no cell is privileged as another's neighbour, so that the transition graph, rather than being wired in by a convolution, must be discovered from reward alone. Using mechanistic probes and causal interventions we show that this agent binds board squares to cells, recovers the transition graph in its learned attention, lays out a goal-directed plan over that graph, and refines it with each thinking step. We then isolate the responsible ingredient with a matched control that must additionally learn its binding: no stable binding forms, no transition graph settles into the relations, and planning does not emerge.

Read together, the established convolutional results and the new attention results we present show that the same planning computation arises whether the transition graph is given or learned. A third agent (§5.5) completes the argument: when the given binding is removed, no binding forms, no transition graph settles into the relations, and no planning emerges. It is therefore the relational bias *operating over state-bound cells*, not any built-in graph, that produces planning.

2 Background

2.1 Reinforcement Learning

We consider a deterministic Markov decision process (MDP) $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R})$ [22] with a finite state set $\mathcal{S}$, a finite action set $\mathcal{A}$, a deterministic transition map $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$, and a reward function $\mathcal{R} : \mathcal{S} \to \mathbb{R}$. The objective of reinforcement learning is to find a policy π that maximizes the expected discounted return $\mathbb{E}_\pi\big[\sum_{i\geq 0} \gamma^i \, r_{t+i} \mid s_t = s\big]$, with discount factor $\gamma \in (0, 1)$. ***Model-free*** methods optimize this objective directly, without learning the transition map. Most modern methods use an

actor–critic scheme [17]: a critic

$$V^{\pi}(s) = \mathbb{E}_{\pi}\big[\sum_{i\geq 0} \gamma^{i}\, r_{t+i} \mid s_t = s\big] \tag{1}$$

estimates the return and is learned by temporal-difference (TD) learning [30], while the parameters of the actor π_ϕ are updated by the policy gradient [32]

$$\nabla_{\phi} J(\phi) = \mathbb{E}_{\pi}\Big[\sum_{t} \nabla_{\phi} \log \pi_{\phi}(a_t \mid s_t)\, \delta_t\Big], \tag{2}$$

where δ_t is an advantage estimator, such as the n-step TD error or the generalized advantage estimate [26]. Scalable implementations of this scheme, such as IMPALA [6], are the standard training substrate for the architectures of this paper.

2.2 Decision-time Planning

In this paper, *planning* refers to *decision-time planning*: selecting the action at the current state by simulating and evaluating the consequences of future actions [31]. In the model-based setting these consequences are simulated under a learned model $\mathcal{M} = (\hat{p}, \hat{r})$, where $\hat{p}(s' \mid s, a)$ approximates the transition dynamics and $\hat{r}(s, a)$ the reward [10; 11; 12]. Such models are typically fit against a predictive loss, which need not align with control performance, creating the objective-mismatch problem [19].

Rather than storing an explicit policy, decision-time planning defines the action at the current state s_t *implicitly*, by optimizing a simulated lookahead of horizon K at the moment the action is required:

$$\pi(s_t) \;=\; \arg\max_{a_t}\; \mathbb{E}_{s_{t+k+1}\sim\hat{p}(\cdot|s_{t+k},a_{t+k})}\left[\sum_{k=0}^{K-1} \gamma^{k}\, \hat{r}(s_{t+k}, a_{t+k}) \;+\; \gamma^{K}\, V^{\pi}(s_{t+K}) \;\middle|\; s_t, a_t\right], \tag{3}$$

where the inner maximization is over the action sequence $a_{t:t+K-1}$ and the terminal estimate V^{π} bootstraps the return beyond the horizon. Only the first action a_t is executed; the agent then re-plans from s_{t+1} in receding-horizon fashion. At $K{=}1$ this reduces to the one-step greedy policy with respect to V^{π}; larger K evaluates progressively longer futures under the model.

3 Relational Hidden States

3.1 Neural Architecture

We study neural networks that move information over their own neural structure. We view such networks as pools of hidden cells exchanging messages, with *no* a priori correspondence between cells and environment states. We refer to this substrate, weight-tied cells coupled by learned relations, as *relational latent states*: the inductive bias whose consequences this paper traces [1].

Definition 3.1 (Relational hidden states)**.** We say that a neural network has relational hidden states if its hidden cells $h_t(i) \in \mathbb{R}^n$ are updated by the other latent cells, indexed by j via

$$h_t^{k+1}(i) \;=\; \bigoplus_{j} F_{\phi}\big(h_t^{k}(i),\, h_t^{k}(j)\big), \tag{4}$$

where $k \in \{0, ..., K-1\}$ is a thinking step and t the environment time. Here, F_ϕ is the learned message for the pair $(j \to i)$ and $\bigoplus$ is any permutation-invariant aggregation over the related cells. The carry between environment steps is the last thinking state, $h_t^0 \equiv h_{t-1}^K$.

Convolutions and the attention mechanism are two instantiations of F_ϕ. A *convolution* places the cells on a grid and lets the message depend only on the sender and its offset to the receiver, with a learned weight $W(j-i)$ for offset on the kernel support and 0 otherwise, applied to the content $h_t^k(j)$. Those weighted messages are then summed over a local neighborhood. *Attention* instead lets the message depend on content: a learned score a_ϕ between the two cells weights each sender's latent content $h_t^k(j)$ by a linear transformation W. Here, the weighted messages are pooled as a normalized average over all pairs. Content-addressed routing of this kind can be read as an associative memory [23] and, when linearized, as a form of recurrence [15], placing convolution, attention, and external-memory architectures [7] on a common relational footing.

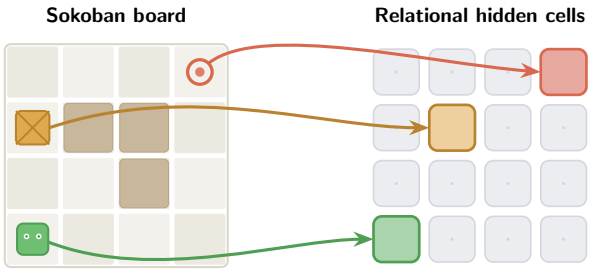


Figure 1: **State-graph binding.** Each Sokoban board square (left) is bound to one relational hidden cell (right) of the neural network via a fixed indexing (color-matched arrows and cells).

3.2 Hidden-State Binding

A relational core (Def. 3.1) carries content over a fixed pool of cells, but nothing in Eq. equation 4 forces those cells to *mean* anything. Suppose the network computes some statistic $g(s)$ of a state, which we call a feature of the state. We use *binding* to denote the process by which training turns the cell pool into an internal, content-addressed copy of the states the agent must reason about, with cells related to one another along the edges the environment permits (Fig. 1). This mirrors the notion of relational memory in cognitive neuroscience, in which distinct experiences are represented as elements linked by their relations rather than as isolated items [3; 5].

Definition 3.2 (State-graph binding)**.** We say a neural network's core exhibits *state-graph binding* if there exists a map $\sigma : \mathcal{S} \to \mathcal{C}$, from the environment's states to the net's hidden cells, such that each state's statistic is carried by a unique cell and recoverable from $h_t(\sigma(s))$. Furthermore, the core's relations *respect the transition graph* if the message in equation 4 from cell $\sigma(s')$ to cell $\sigma(s)$ contributes only when $s' \in \mathcal{N}(s) := \{\mathcal{T}(s, a) : a \in \mathcal{A}\}$.

In a convolution, the cell-to-state binding is learned but the transition-respecting relations are supplied by the fixed local kernel; in attention, both must be learned from reward alone. With both conditions in place, each thinking step moves g one hop along $\mathcal{N}$: after K steps the agent's cell $\sigma(s_t)$ has gathered statistics from its K-hop neighborhood, and the carry $h_t^0 \equiv h_{t-1}^K$ hands that information on to the next environment step, a horizon that recedes step by step and forms the substrate on which planning can emerge.

4 Emergent Planning

We now argue that these two structural properties, state-graph binding and transition-respecting relations, can turn the thinking recursion into decision-time planning. The argument proceeds in three steps. First, we assume that the action statistics localize to the hidden cells. Second, we argue that the learned relations *are* a transition model, so that no separate model needs to be fit. Third, we show that this computation can represent a horizon-K neural planner of equation 3 and stochastic gradient descent can discover decision-time planning from a reward-maximization objective.

4.1 Localized Decision Statistics

We first argue that actions are decodable from the relational core.

Assumption 4.1 (Localized readout)**.** The neural network binds the actions of a state s_t to its corresponding hidden cell, so that after K thinking steps:

$$\pi(\cdot \mid s_t) \;\approx\; \rho_\pi\big(h_t^K(\sigma(s_t))\big), \tag{5}$$

where ρ_π is a learned readout implemented by the next hidden layer or the actor head.

Under state indexing with transition-respecting relations, the cell update equation 4 moves information only along the transition model: a cell's content aggregates the statistics of states adjacent to the

cell's underlying state, and, because the weights vanish across absent transitions, never draws on states unreachable from it. How far along the graph a single step carries information depends on the instantiation. A convolution fetches information exactly from the one-hop successors in our grid-construction. For a learned attention core we find a *soft* reach over successors: per-step influence decays geometrically in graph distance rather than truncating at one hop, and is exactly zero only where transitions are absent. Additional thinking steps refine the resulting computation globally rather than advancing a distance-graded front, and it is this refinement that improves the decision with test-time computation.

4.2 Implicit Transition Model

Decision-time planning requires a transition model $\hat{p}$. We now show that the trained relational core implicitly carries such a transition model, in two complementary senses.

Actions as successors. In a deterministic MDP, each action available at s selects a unique successor, so the map $a \mapsto \mathcal{T}(s, a)$ is onto $\mathcal{N}(s)$. A policy over actions is therefore equivalent to a distribution over successor states,

$$P^{\pi}(s' \mid s) \;=\; \sum_{a\,:\,\mathcal{T}(s,a)=s'} \pi(a \mid s), \tag{6}$$

and, conversely, any distribution supported on $\mathcal{N}(s)$ induces a policy. Selecting an action is thus equivalent to selecting a neighbour on the transition graph.

Relations as routing. Let $w(j \to i) \geq 0$ denote the weight that the cell update in equation 4 places on the message from cell j to cell i; for a convolution this is a kernel weight, for attention an attention score. Define the induced model by

$$\hat{p}(s' \mid s) \;\propto\; w\big(\sigma(s') \to \sigma(s)\big). \tag{7}$$

By Definition 3.2, if the relations are transition-respecting then w places all of its mass on genuine successors, so $\operatorname{supp} \hat{p}(\cdot \mid s) \subseteq \mathcal{N}(s)$. Thus $\hat{p}$ is a transition model of the environment read directly off the routing, without being fit against an explicit prediction loss.

4.3 Planning as Message Passing

We write $F^{K}_{\phi,\omega}$ for the K-fold composition of the cell update equation 4, with $\omega(j \to i)$ the aggregation weights of $\bigoplus$ (Eq. equation 7), so that $h^{K}_{t} = F^{K}_{\phi,\omega}(h^{0}_{t})$. If each cell starts from the local statistic $g(s)$ alone, information about distant goals and obstacles can reach the readout cell only through message passing along the learned relations: return is earned exactly to the extent that the relations transport decision-relevant information.

Proposition 4.1 (Message passing can implement decision-time planning)**.** *Under Assumption 4.1 and Definition 3.2, every policy the architecture expresses is a relational readout of the thinking recursion routed by ω,*

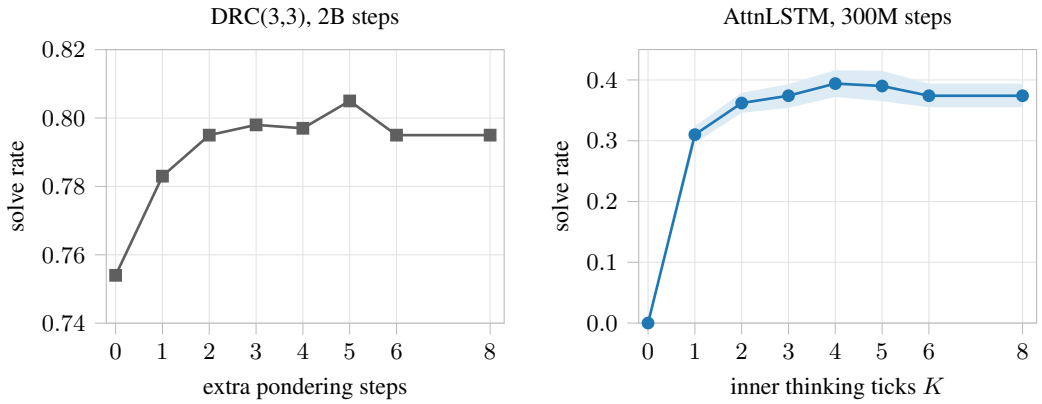
$$\pi(s_t) \;=\; \rho_{\pi}\Big(F^{K}_{\phi,\omega}\big(\{\, x(g(s')) : s' \in \mathcal{S} \,\}\big)\Big), \tag{8}$$

in which each application of $F_{\phi,\omega}$ lets a cell $\sigma(s)$ pull messages only from the successor cells $\sigma(\mathcal{N}(s))$ that ω admits. This computation can implement the horizon-K decision-time planner of Eq. equation 3,

$$\pi(s_t) \;=\; \arg\max_{a_t} \; \mathbb{E}_{s_{t+k+1} \sim \hat{p}(\cdot \mid s_{t+k}, a_{t+k})} \left[\sum_{k=0}^{K-1} \gamma^{k}\, \hat{r}(s_{t+k}, a_{t+k}) \;+\; \gamma^{K}\, V^{\pi}(s_{t+K}) \;\middle|\; s_t, a_t \right], \tag{9}$$

with $\hat{p}$ given by ω as in equation 7. There exist parameters (ϕ, ρ_{π}) under which each application of $F_{\phi,\omega}$ transports information, so that after K steps information about the best leaf state at the horizon has propagated along ω to $\sigma(s_t)$, and ρ_{π} returns the action toward the successor carrying it.

We thus argue that decision-time planning is a solution that gradient descent can discover from reward alone: the model is amortized into the routing, the lookahead into the thinking recursion, and neither is supplied in advance.



(a) DRC with forced no-op pondering steps

(b) AttnLSTM with variable inner recurrent ticks

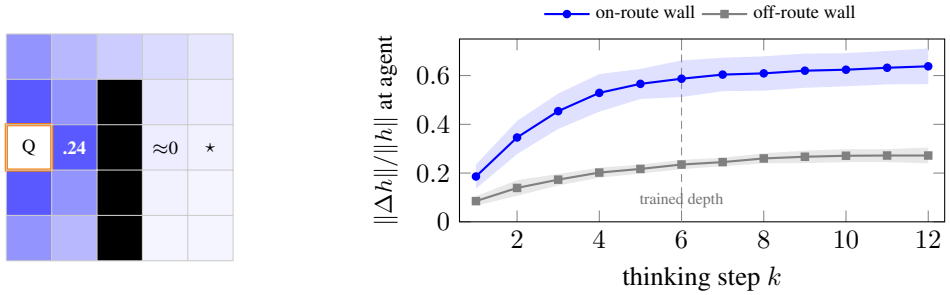
Figure 2: **Additional test-time computation improves solving.** Solve rate on held-out Boxoban `valid_medium`. **(a)** For the pretrained DRC(3,3) of Taufeeque et al. [33], additional computation is provided by forced no-op pondering steps, following Guez et al. [9]. **(b)** For our AttnLSTM, additional computation is provided by increasing the number of inner recurrent thinking ticks before acting. Both agents improve with extra test-time recurrent computation, but the horizontal axes represent different interventions and are therefore shown separately.

5 Empirical Validation

The hypothesis advanced so far is that a *relational hidden state*, a pool of cells coupled by learned relations, is the inductive bias under which decision-time planning emerges from reward alone, and that it does so *without the environment's transition structure being supplied to the agent*. Existing evidence on convolutional agents establishes that such planning occurs but cannot isolate this claim: the Deep Repeated ConvLSTM plans on Sokoban [9], and its plan has been decoded as square-level, causally-effective, iteratively-refined concepts [2; 33], yet because a convolution wires each cell to its spatial neighbours, the transition graph is *built into* the architecture, and the relational bias there cannot be separated from a hand-given graph. This paper removes that crutch: we present an agent whose relational core is a *dense, unmasked attention* recurrence, in which no cell is privileged as another's neighbour and every relation is learned, and we show that decision-time planning emerges, with the transition graph now discovered from reward. Whether the relational bias alone suffices, or whether it must operate over a supplied binding, is exactly what the free-slot control of §5.5 isolates.

The two relational cores. Both cores instantiate the relational recurrence of definition 3.1. They differ only in their relational updates. The *convolutional* core is the Deep Repeated ConvLSTM of Guez et al. [9]: $D{=}3$ ConvLSTM layers whose cells sit on the 10×10 board and exchange messages only with their spatial neighbours through a fixed convolutional kernel (plus a pooled global channel). In this work, we analyse the pretrained agent studied by Bush et al. [2]; Taufeeque et al. [33]. The *attention* core, which we call AttnLSTM, replaces the fixed kernel with learned, content-based attention: a convolutional encoder maps the board to 100 cells (one per square), which each thinking step relates by dense, multi-head entmax attention with no locality mask. Thus, every hidden cell may attend to every other and which relations matter must be learned by the agent. These relations are followed by an LSTM gate; $D{=}3$ stacked cells run up to $K{=}6$ thinking steps, the thinking depth sampled per rollout so the agent must act well at every thinking budget. The AttnLSTM is trained from reward alone (IMPALA on Boxoban, 300M steps). Full architectural descriptions and hyperparameters are in App. A.

Previous work established that the DRC improves its solve rate with additional thinking steps [9]. We find that the attention core shows the same improvement (Fig. 2). However, improvement with computation is only a necessary signature of planning, not a sufficient one, so it could reflect reactive heuristics sharpened over iterations as well. We make use of mechanistic interpretability techniques to show that the attention core also implements decision-time planning over the transition graph. We establish four core properties predicted by the theoretical framework developed earlier.



(a) successors a query connects to

(b) a route-blocking wall re-plans the move

Figure 3: **The attention routing is an amortized transition model. (a)** A cell Q's settled attention concentrates on feasible one-step successors (≈ 0.24 at one hop) and decays geometrically with *graph* distance ($\rho \approx 0.61$). Its attention to cells made unreachable by a wall is approximately zero. **(b)** Turning a floor tile on the agent's route into a wall shifts the current position's hidden cell $2.3\times$ more than turning an off-route tile at a matched distance into a wall. Additionally, the intervention flips the agent's greedy move on 19% of route-blocked boards, compared with 7.2% of off-route boards: a change to the dynamics propagates along the graph and rewrites the plan.

(i) **A localized decision statistic**: the policy statistic $\pi_t(s)$ that determines the action at a given state s_t is localized to the cell $\sigma(s_t)$ bound to the given state.

(ii) **Amortized transition dynamics**: the relations among cells concentrate on the feasible one-step transitions, so the routing *is* a transition model, given by locality in the DRC, discovered from reward in the attention core.

(iii) **A decodable plan**: a goal-directed plan is decodable over that hidden-cell graph after the final thinking step, before the next action is taken.

(iv) **Plan refinement with thinking**: the plan is refined with each additional thinking step, turning extra computation into a better decision.

These constitute the structural content of decision-time planning: a state-indexed decision (i), read from a learned model of the dynamics (ii), assembled into a lookahead over reachable futures (iii) that deepens with computation (iv).

Sections 5.1–5.4 establish (i)–(iv) on the attention core. Section 5.5 then tests the chain's first link by ablation: an otherwise-matched core that must learn its binding fails to form one, and neither the transition graph nor planning emerges.

5.1 Action Localization

Because the environment is deterministic, selecting an action is selecting a successor; the statistic that determines the action at s_t is therefore equivalently a *next-state statistic*, and Assumption 4.1 predicts that the action statistic is localized at the cell bound to the current state. For the convolutional core this is established by prior work: concept probes decode, per square, the direction from which the agent will next enter it and the direction in which it will next push a box, so that at the agent's own square the decoded concept is its next move [2; 33]. The attention core, in which nothing places the decision at the agent's cell a priori, shows the same signature: a linear probe reading the agent-square cell alone decodes the model's own next action in 60% of cases, compared with a chance level of 25%. The decision statistic is thus localized in the representation and readable at the current state's cell, as Section 4.1 requires—though not perfectly. We suspect the residual gap reflects an undertrained agent rather than a delocalized readout, and would expect the coupling to tighten alongside further gains in return with longer training; verifying this lies beyond our compute budget, and we leave it open.

5.2 Amortized transition dynamics

A convolutional core gets its transition model for free: its kernel passes messages only between grid neighbours, so information flows only along the grid graph [33]. The attention core is unconstrained,

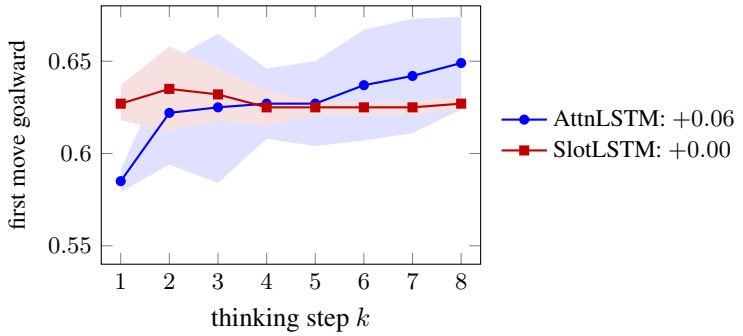


Figure 4: **Thinking improves the decision only in the bound core.** At each thinking step k, the greedy action is read from the intermediate hidden state through the model's own trained actor head, on the initial observation of each of 512 held-out boards, and scored by whether the first move is *goalward*. We restrict to hard levels with agent-to-goal distance of 8–12 hops, where lookahead matters most. Here, the attention core keeps improving through the *final* thinking step ($+0.06$); the slot control is flat ($+0.00$).

its dense relations can route anywhere, however we find it recovers the transition graph from reward alone. Its settled attention concentrates on a cell's one-step successors and decays geometrically by a factor 0.61 with k-hop *graph* reachability, vanishing on cells a wall renders unreachable even one pixel away (Fig. 3a). We also find the routing to be causal: placing a wall on the agent's path to the goal, thereby severing an edge it must traverse, shifts the hidden state at the agent's own cell $2.3\times$ more than an identical off-path wall at matched distance, and flips its greedy move on 19% of route-blocked boards versus 7.2% off-route (Fig. 3b). Cutting an edge the plan depends on reroutes information to the decision and rewrites it; cutting an irrelevant one does not. This suggests that the attention-core's routing is an amortized causal transition model of the Sokoban dynamics.

5.3 A Decodable Plan

Choosing the next state from a single cell is one step; planning requires that the choice reflect a multi-step, goal-directed plan over the graph. In the convolutional core such a plan is decoded directly: the square-level entry- and push-directions of Bush et al. [2] assemble into coherent trajectories from start to goal and transfer to unseen layouts. The attention core carries the same plan as a *field* over its cells: a linear probe recovers, per cell, a move direction that points towards the goal in 61% of cases (chance 25%) laying the plan out across the whole board rather than only at the agent's square.

5.4 Plan Refinement with Thinking

Finally, if these thinking steps perform planning, they should *revise* the decision, the revisions should *converge*, and they should *improve* the outcome. In the convolutional core, all three hold: the decoded plan concepts sharpen progressively across the recurrent iterations [2]. We find that the attention core refines the same way: over its thinking steps it revises the chosen action on roughly a quarter of boards, and this revision rate falls to 0 as the decision settles, while the policy's margin on the chosen action grows $1.7\times$ and the move stays goal-consistent ($\approx$60% goalward, Fig. 5). The AttnLSTM collects most of this mean gain in the first thinking step, which is to be expected, since dense routing can span the board in a single step, but on the hardest boards (agent-to-goal distance 8–12 hops) the gain instead accrues gradually through the final step ($+0.06$, Fig. 4), the signature of deepening lookahead. [1]

[1]Throughout, a move is *goalward* if it equals the BFS-greedy first step toward the target square, computed over non-wall tiles (chance 0.25). This is a navigation proxy that ignores box dynamics, as the correct Sokoban move is often box-directed rather than target-directed, so even the settled policy scores only $\sim$0.6. For us, goal-directedness is thus used only as a measure of whether a policy is reactive or adaptive under computation, which is a traditional planning signature.

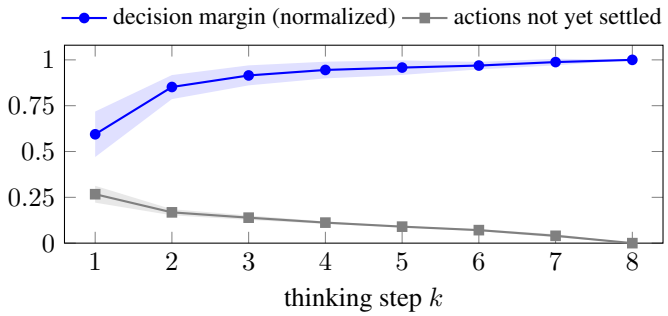


Figure 5: **Each thinking step refines the decision.** At every thinking step k of the AttnLSTM we read the policy from the intermediate state h_t^k through the trained actor head. The margin on the chosen action, the top-1 minus top-2 action logit, is shown normalized to its converged value (blue) and rises monotonically to 1, while the fraction of boards whose greedy action has not yet reached its settled value (grey) falls from 27% to 0: the decision sharpens and converges by step 8.

5.5 The Binding is Load-Bearing: a Free-Slot Control

Both cores explored until now share one strong architectural prior: the convolutional encoder pins each board square to its own cell, so the binding σ of Definition 3.2 is supplied by construction and only the relations must be learned. This leaves open which ingredient the emergence actually rests on: just the relational message passing, or the message passing *over that given binding*. To separate them, we train a third, otherwise-matched core in which the binding itself must be discovered. In the *slot* core (SlotLSTM), the 100 cells are free slots with no positional identity: at every thinking step each slot first *binds* by a slot-attention competition over the encoder's board tokens (a softmax across slots, so the slots compete to explain each square), then *routes* by the same dense slot-to-slot entmax attention as the AttnLSTM, and integrates both messages through the same LSTM gate (App. A). Position remains available as content, the board tokens carry a learned positional embedding, so what is removed is exactly the fixed cell-to-square correspondence, nothing else.

No state-graph binding emerges. Definition 3.2 asks for a map from states to cells under which each state's statistic is carried by its own, recoverable cell, and, for the weight-tied routing to use it, a map that holds across boards. We find that the slot core does not discover this. Its binding reads are diffuse rather than one-to-one: a slot places only 0.13 of its read mass on its top square, 3–4 slots share each bound square, and only 43% of navigable squares are any slot's primary read. Furthermore, the assignment is not stable across Sokoban levels and even the most agent-specialized slot reads the agent on only 29% of boards. However, the slots are not empty: a square's tile still decodes from its winning slot at 0.66 vs. 0.50 from a random one; what fails to form is the unique, board-independent correspondence Def. 3.2 requires.

Without binding, neither a transition graph nor planning emerges. While the anchored attention core's routing decays geometrically with graph distance between the squares its cells hold ($\rho \approx 0.61$ per hop, Fig. 3a), the slot core's is nearly distance-blind ($\rho \approx 0.94$): 67% of its mass connects squares more than one transition apart, it carries no asymmetry toward the goal, and a perturbation at *any* graph distance 1–9 reaches the agent's slot within the first thinking step (arrival slope 0.00 steps per hop). The broadcast still delivers information: a route-blocking wall shifts the decoded value at the agent's slot $2.9\times$ more than a matched off-route wall, a ratio at least as large as the attention core's. But this shift arrives instantaneously, and extra computation yields no better decision, the opposite of what planning predicts. Read through the actor head at each step, the first move's goalward fraction is flat ($+0.004$ from step 1 to 8, vs. $+0.06$ for the AttnLSTM), the move is revised on 9% of boards (vs. 27%, Fig. 5), the route-blocking wall flips the move on only 11% of route-blocked boards (vs. 19%, Fig. 3b), and eight extra recurrent steps on the initial observation (the pondering protocol of 9) leave the slot core's held-out solve rate essentially unchanged (0.067 with 0 extra steps, 0.063 with 8); under the same pondering protocol the DRC gains $0.754 \rightarrow 0.805$ (Fig. 2a), while the AttnLSTM under its inner-tick sweep gains $0.31 \rightarrow 0.39$ (Fig. 2b). We further investigated whether the difference in variable-depth and fixed-depth computation could be at fault here. However, we found that an AttnLSTM trained at the same $K{=}4$ gains $+0.080$ in goal-directedness gradually over

steps 1–5, whereas the slot core is flat within *every* distance band (± 0.03, sign-incoherent). We further find that a 50-slot variant replicates every finding: routing barely decays over successors ($\rho \approx 0.97$), and the solve rate increases marginally at best ($0.051 \rightarrow 0.059$).

6 Discussion & Future Work

Our results show that a neural network with a graph-like backbone can be trained to learn decision-time planning from reward maximization alone, without an explicit model or a prescribed search procedure. Because the convolutional and attention-based relational cores we study are common among modern deep-learning architectures, this raises the possibility that emergent planning is a fairly general phenomenon in reinforcement learning. Several works have investigated whether planning mechanisms can emerge in transformers, and some have found indications that they do, such as search- and lookahead-like computation across Othello and chess transformers [20; 13] and emergent planning in language models [21; 4].

We caution against over-generalizing: in both planning cores, each environment state is pinned to a single cell by a convolutional encoder, and only the hidden-cell relations are learned. The slot control (§5.5) shows that this strong inductive bias is not incidental but load-bearing: when the binding must itself be discovered, our agent fails to form one, only a diffuse, per-board assignment emerges. From this, no transition structure settles into the networks' relations, and planning does not emerge. Most large-scale models lack the one-state-per-cell structure. For example, a transformer's residual stream mixes many features per position, and positions need not correspond to environment states at all. On our account, the operative question for such models is whether a stable-enough binding is realized elsewhere in their representations, and whether planning emerges exactly where it is: emergent planning may be common in architectures that supply this binding cheaply and rarer elsewhere. Whether a different learned-binding mechanism, longer training, or greater scale can stabilize σ where slot competition did not, we regard as open and leave to further research.

Our empirical claims rest on three independently trained AttnLSTMs and three independently trained slot controls (the original run plus two additional random seeds each), together with the single pretrained DRC checkpoint [2], evaluated on one test set. Across the three AttnLSTM seeds the mechanism is stable rather than a single-seed artifact: the routing recovers the transition graph in every run (graph-distance decay $\rho = 0.61$, and blocked-by-wall influence far below the open baseline in all three), the decision statistic stays localized to the agent's cell (60%), and the decision measurably refines over thinking steps in each run (first-step revision rate 27%, falling to zero as it settles). The negative result is equally stable: across the three slot controls the binding stays diffuse (0.13 read mass on the top square; the most agent-specialized slot reads the agent on 29% of boards) and the routing stays a near-uniform broadcast ($\rho = 0.94$, 67% of mass beyond one hop, arrival slope 0.00), with no plan refinement. Point estimates in the main text are 3-seed means; the per-seed values and standard deviations for every reported quantity are given in Table 1 (App. B), and the figures show the corresponding mean$\pm$sd bands. The binding nonetheless remains architectural in the AttnLSTM and ConvLSTM cores—the convolutional encoder pins each square to its own cell—so it is not itself an emergent quantity; and a systematic sweep over initialization scale, training length, and alternative learned-binding mechanisms that might stabilize σ where slot competition did not remains future work.

We close by noting that many accounts of how the human brain plans rely on a similar relational-graph mechanism. In cognitive science, the hippocampus is often described as a relational graph [35; 29; 36] that binds abstract elements by their relations [18; 5]. If a biological system already supplies a relational state graph, our account suggests reward-driven learning over it could yield planning by the same route. However, we offer this only as a hypothesis raised by our framework, not a claim we establish.

7 Acknowledgements

The author is supported by an Excellence Scholarship from the State of Lower Austria. The author would like to express his gratitude to Jannik Schilling for his feedback and suggestions on initial drafts.

REFERENCES

[1] Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Caglar Gulcehre, Francis Song, Andrew Ballard, Justin Gilmer, George Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matthew Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks, 2018. URL https://arxiv.org/abs/1806.01261.

[2] Thomas Bush, Stephen Chung, Usman Anwar, Adrià Garriga-Alonso, and David Krueger. Interpreting emergent planning in model-free reinforcement learning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=DzGe40glxs.

[3] Neal J. Cohen and Howard Eichenbaum. *Memory, Amnesia, and the Hippocampal System*. MIT Press, Cambridge, MA, 1993.

[4] Zhichen Dong, Zhanhui Zhou, Zhixuan Liu, Chao Yang, and Chaochao Lu. Emergent response planning in LLMs, 2025. URL https://arxiv.org/abs/2502.06258.

[5] Howard Eichenbaum. On the integration of space, time, and memory. *Neuron*, 95(5):1007–1018, 2017. doi: 10.1016/j.neuron.2017.06.036.

[6] Lasse Espeholt, Hubert Soyer, Rémi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: Scalable distributed deep-RL with importance weighted actor-learner architectures. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1407–1416. PMLR, 2018. URL https://proceedings.mlr.press/v80/espeholt18a.html.

[7] Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, Adrià Puigdomènech Badia, Karl Moritz Hermann, Yori Zwols, Georg Ostrovski, Adam Cain, Helen King, Christopher Summerfield, Phil Blunsom, Koray Kavukcuoglu, and Demis Hassabis. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626): 471–476, 2016. doi: 10.1038/nature20101.

[8] Klaus Greff, Rupesh K. Srivastava, and Jürgen Schmidhuber. Highway and residual networks learn unrolled iterative estimation, 2017. URL https://arxiv.org/abs/1612.07771.

[9] Arthur Guez, Mehdi Mirza, Karol Gregor, Rishabh Kabra, Sébastien Racanière, Théophane Weber, David Raposo, Adam Santoro, Laurent Orseau, Tom Eccles, Greg Wayne, David Silver, and Timothy Lillicrap. An investigation of model-free planning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2464–2473. PMLR, 2019. URL https://proceedings.mlr.press/v97/guez19a.html.

[10] David Ha and Jürgen Schmidhuber. World models, 2018. URL https://arxiv.org/abs/1803.10122.

[11] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2555–2565. PMLR, 2019. URL https://proceedings.mlr.press/v97/hafner19a.html.

[12] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models, 2023. URL https://arxiv.org/abs/2301.04104.

[13] Erik Jenner, Shreyas Kapur, Vasil Georgiev, Cameron Allen, Scott Emmons, and Stuart Russell. Evidence of learned look-ahead in a chess-playing neural network. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://arxiv.org/abs/2406.00877.

[14] Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. An empirical exploration of recurrent network architectures. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2342–2350. PMLR, 2015. URL https://proceedings.mlr.press/v37/jozefowicz15.html.

[15] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention, 2020. URL https://arxiv.org/abs/2006.16236.

[16] Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for interacting systems. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2688–2697. PMLR, 2018. URL https://proceedings.mlr.press/v80/kipf18a.html.

[17] Vijay Konda and John Tsitsiklis. Actor-critic algorithms. In *Advances in Neural Information Processing Systems*, volume 12, 1999. URL https://proceedings.neurips.cc/paper_files/paper/1999/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf.

[18] Alex Konkel and Neal J. Cohen. Relational memory and the hippocampus: Representations and methods. *Frontiers in Neuroscience*, 3:166–174, 2009. doi: 10.3389/neuro.01.023.2009. URL https://www.frontiersin.org/journals/neuroscience/articles/10.3389/neuro.01.023.2009.

[19] Nathan O. Lambert, Brandon Amos, Omry Yadan, and Roberto Calandra. Objective mismatch in model-based reinforcement learning. In *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, volume 120 of *Proceedings of Machine Learning Research*, pages 761–770. PMLR, 2020. URL https://proceedings.mlr.press/v120/lambert20a.html.

[20] Kenneth Li, Aspen K. Hopkins, David Bau, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Emergent world representations: Exploring a sequence model trained on a synthetic task. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=DeG07_TcZvT.

[21] Jack Lindsey, Wes Gurnee, Emmanuel Ameisen, Brian Chen, Adam Pearce, Nicholas L. Turner, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Tom Henighan, Adam Jermyn, Andy Jones, Andrew Persic, Zhenyi Qi, T. Ben Thompson, Sam Zimmerman, Kelley Rivoire, Thomas Conerly, Chris Olah, and Joshua Batson. On the biology of a large language model. Transformer Circuits Thread, 2025. URL https://transformer-circuits.pub/2025/attribution-graphs/biology.html.

[22] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, New York, NY, 1994. doi: 10.1002/9780470316887.

[23] Hubert Ramsauer, Bernhard Schäfl, Johannes Lehner, Philipp Seidl, Michael Widrich, Thomas Adler, Lukas Gruber, Markus Holzleitner, Milena Pavlović, Geir Kjetil Sandve, Victor Greiff, David Kreil, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. Hopfield networks is all you need, 2020. URL https://arxiv.org/abs/2008.02217.

[24] Alvaro Sanchez-Gonzalez, Nicolas Heess, Jost Tobias Springenberg, Josh Merel, Martin Riedmiller, Raia Hadsell, and Peter Battaglia. Graph networks as learnable physics engines for inference and control. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4470–4479. PMLR, 2018. URL https://proceedings.mlr.press/v80/sanchez-gonzalez18a.html.

[25] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature*, 588:604–609, 2020. doi: 10.1038/s41586-020-03051-4.

[26] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation, 2015. URL https://arxiv.org/abs/1506.02438.

[27] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Advances in Neural Information Processing Systems*, volume 28, 2015. URL https://proceedings.neurips.cc/paper/2015/hash/07563a3fe3bbe7e3ba84431ad9d055af-Abstract.html.

[28] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, 2017. doi: 10.1038/nature24270.

[29] Kimberly L. Stachenfeld, Matthew M. Botvinick, and Samuel J. Gershman. The hippocampus as a predictive map. *Nature Neuroscience*, 20(11):1643–1653, 2017. doi: 10.1038/nn.4650.

[30] Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1):9–44, 1988. doi: 10.1007/BF00115009.

[31] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 2 edition, 2018.

[32] Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, volume 12, pages 1057–1063. MIT Press, 2000. URL https://papers.nips.cc/paper/1713-policy-gradient-methods-for-reinforcement-learning-with-function-approximat

[33] Mohammad Taufeeque, Philip Quirke, Maximilian Li, Chris Cundy, Aaron David Tucker, Adam Gleave, and Adrià Garriga-Alonso. Planning in a recurrent neural network that plays Sokoban, 2024. URL https://arxiv.org/abs/2407.15421.

[34] Mohammad Taufeeque, Aaron David Tucker, Adam Gleave, and Adrià Garriga-Alonso. Path channels and plan extension kernels: A mechanistic description of planning in a Sokoban RNN. In *Mechanistic Interpretability Workshop at NeurIPS 2025*, 2025. URL https://openreview.net/forum?id=z025sIH4Nq.

[35] Edward C. Tolman. Cognitive maps in rats and men. *Psychological Review*, 55(4):189–208, 1948. doi: 10.1037/h0061626.

[36] James C. R. Whittington, Timothy H. Muller, Shirley Mark, Guifen Chen, Caswell Barry, Neil Burgess, and Timothy E. J. Behrens. The Tolman-Eichenbaum machine: Unifying space and relational memory through generalization in the hippocampal formation. *Cell*, 183(5):1249–1263.e23, 2020. doi: 10.1016/j.cell.2020.10.024.

[37] Vinicius Zambaldi, David Raposo, Adam Santoro, Victor Bapst, Yujia Li, Igor Babuschkin, Karl Tuyls, David Reichert, Timothy Lillicrap, Edward Lockhart, Murray Shanahan, Victoria Langston, Razvan Pascanu, Matthew Botvinick, Oriol Vinyals, and Peter Battaglia. Relational deep reinforcement learning, 2018. URL https://arxiv.org/abs/1806.01830.

A Appendix: Architecture and Training Details

Shared relational recurrence. A convolutional encoder of two 4×4 same-padded conv layers, no nonlinearity, $C{=}32$ channels; 9 maps the observation to a feature map $e \in \mathbb{R}^{H \times W \times C}$. Here $H{=}W{=}10$ is read as $S{=}HW{=}100$ cells indexed by board position with the playable region being the inner 8×8. A core of D stacked, weight-tied cells is applied for K "thinking" ticks within one environment step t, and the settled state is carried across steps, $h_t^0 \equiv h_{t-1}^K$. Each cell layer d holds an LSTM state (c_d, h_d) and, at repeat k, updates it from its own previous state h_d^{k-1}, the encoder features e, and the layer below h_{d-1}^k, via a relation over the S cells. The two instantiations below differ only in that relation.

Convolutional core: ConvLSTM [9]. The relation is a fixed local kernel, realised by a convolutional LSTM [27]. The layer-d input at repeat k concatenates the encoder features, the hidden state below, a pool-and-inject summary, and a boundary-indicator channel,

$$x_d^k \;=\; \big[\, e\,;\; h_{d-1}^k\,;\; \mathrm{p}(h_d^{k-1})\,;\; \mathbf{1}_\partial \,\big], \tag{10}$$

$$\mathrm{p}(h) \;=\; \mathrm{broadcast}\big(W_p[\,\mathrm{maxpool}_{HW} h\,;\, \mathrm{meanpool}_{HW} h\,]\big), \tag{11}$$

where $\mathbf{1}_\partial$ marks the grid boundary (ones on the border, zeros inside; not zero-padded) to keep convolution edge effects from corrupting the recurrence. A 3×3 ConvLSTM then updates the cell, with $*$ a 2-D convolution and $\odot$ the Hadamard product:

$$i = \sigma\big(W_i * x_d^k + U_i * h_d^{k-1}\big), \tag{12}$$

$$f = \sigma\big(W_f * x_d^k + U_f * h_d^{k-1} + b_f\big), \tag{13}$$

$$o = \tanh\big(W_o * x_d^k + U_o * h_d^{k-1}\big), \tag{14}$$

$$g = \tanh\big(W_g * x_d^k + U_g * h_d^{k-1}\big), \tag{15}$$

$$c_d^k = f \odot c_d^{k-1} + i \odot g, \tag{16}$$

$$h_d^k = o \odot \tanh\big(c_d^k\big). \tag{17}$$

The $\tanh$ output gate follows Jozefowicz et al. [14] and the forget bias b_f is initialised to 1. Because messages are gathered by the 3×3 kernel, each cell s draws only from its eight spatial neighbours; the transition graph is fixed by the architecture, while pool-and-inject adds a coarse global summary (a whole-board max/mean), not a learned pairwise relation. We use $D{=}3$, $K{=}3$ (the DRC$(3,3)$ of 9) and the pretrained checkpoint of Bush et al. [2].

Attention core (AttnLSTM). The relation is implemented as learned, content-based attention over all S cells (no locality mask). With $\tilde{h}_s = \mathrm{RMSNorm}\big(h_s + W_{\text{in}}[\,e_s;\, h_{s,\text{below}}\,]\big)$ projected to n_h heads of width $d_h = C/n_h$,

$$q_s,\; k_s,\; v_s = W_q \tilde{h}_s,\; W_k \tilde{h}_s,\; W_v \tilde{h}_s, \tag{18}$$

$$L_{s,r} = \tfrac{1}{\sqrt{d_h}}\, q_s^\top k_r + b_{\,s-r}, \tag{19}$$

$$w_{s,\cdot} = \mathrm{entmax}_{1.5}\big(L_{s,\cdot}\big), \tag{20}$$

$$a_s = W_o \textstyle\sum_r w_{s,r}\, v_r, \tag{21}$$

where $b_{\,s-r}$ is an offset-tied relative-position bias (zero-initialised). Thus, $w_{s,\cdot}$ is a learned, 1.5-entmax-sparse distribution over the cells from which s draws; its support defines the recovered graph $\mathcal{N}$. The message a_s drives the same LSTM gates as above with $x_d^k = [\,e_s;\, a_s\,]$, with 1.5-entmax being an architectural choice to incentivize sparse routing in the attention. Our experiments use the default hyperparameters of $D{=}3$, $n_h{=}4$, $C{=}32$, with thinking depth K ranging up to 6.

Slot core (SlotLSTM). The SlotLSTM replaces the S position-indexed cells with $N{=}100$ free slots with no spatial identity. The carry between recurrent steps is $(c, h) \in \mathbb{R}^{N \times C}$, reset to zero at episode boundaries; a learned per-slot identity μ_i—the only thing distinguishing slot i from slot j—enters additively at every step, $\tilde{h} = \mathrm{RMSNorm}(h + \mu)$. Each thinking step computes two messages. The

binding read queries the S encoder tokens (which carry a learned positional embedding, so position is available as content): with $q_i = W_q^b \tilde{h}_i$ and $k_s, v_s = W_k^b e_s,\ W_v^b e_s$,

$$A_{is} \;=\; \frac{\exp\big(q_i^\top k_s/\sqrt{d_h}\big)}{\sum_{i'} \exp\big(q_{i'}^\top k_s/\sqrt{d_h}\big)}, \qquad m_i^{\text{bind}} \;=\; W_o^b \textstyle\sum_s \bar{A}_{is}\, v_s, \tag{22}$$

where the softmax runs over *slots*. This can be seen as the slot-attention competition: slots compete to explain each square, which breaks the permutation symmetry and ties slots to states, with $\bar{A}$ renormalizing each slot's row over the tokens. The *routing* message is the same dense, multi-head 1.5-entmax attention as the attention cell, applied slot-to-slot (q, k, v from $\tilde{h}$; no mask and no positional term). Both messages drive the same LSTM gates with $x_d^k = [\, m^{\text{bind}};\, m^{\text{route}} \,]$, and the same flatten-MLP readout is applied to the slot field. We use $N{=}100$ (matching the 100 board squares), $D{=}3$, $n_h{=}4$, $C{=}32$, at a fixed thinking depth $K{=}4$, trained with the identical IMPALA recipe for 3×10^8 steps (a $N{=}50$ variant behaves the same throughout §5.5). The undertaken measurements are: *binding*, by decoding a square's tile from the slot whose binding read $\bar{A}$ peaks on that square (against a random slot), and by the across-board consistency of that assignment; *routing*, by binning the slot-to-slot weights by the graph distance between the squares the two slots currently bind; *reach*, by the arrival step of a perturbation's effect at the agent's slot as a function of graph distance; and *decision refinement*, by the per-step actor readout of Figs. 5 and 4 and by the solve rate with 0 vs. 8 extra recurrent steps on the initial observation.

Readout. After K repeats, the top layer's settled state, with the encoder skip, is flattened and passed through a one-hidden-layer MLP (256 units) shared by actor and critic,

$$u = \mathrm{relu}\big(W_d\, \mathrm{vec}(h_{\text{top}}^K + e) + b_d\big), \tag{23}$$

$$\pi(\cdot \mid s_t) = \mathrm{softmax}(W_a u), \tag{24}$$

$$V = W_c u. \tag{25}$$

Training. We train the attention core with IMPALA, an actor–critic method with V-trace, on Boxoban unfiltered-train for 3×10^8 environment steps, using $\gamma = 0.97$, gradient clipping, and entropy regularisation [6]. The attention core is trained with *variable thinking depth*: each rollout samples $K \sim \mathcal{U}\{1, \ldots, 6\}$ and the learner replays the same depth, so the agent must act at every budget (a fixed-depth $K{=}4$ variant is used where noted). No model-prediction or reconstruction auxiliary loss is used; the transition graph is shaped by reward alone.

We trained the AttnLSTM for 300 million environment steps, achieving a 91.9% solve rate on the training set. The ConvLSTM was pretrained by Bush et al. [2] for 2 billion environment steps, achieving a 98.4% solve rate on the training set. The slot control was trained with the same recipe and budget at fixed $K{=}4$, reaching a 65–68% training solve rate.

Faithful recompute. The model's forward pass triggers a tracer error on the offset-tied relative-bias gather under `nn.scan` (JIT/GPU). All interpretability probes therefore recompute the $D{=}3$ attention stack in plain JAX from the trained parameters, and apply the head matrix multiplications for logits and value; this reproduces the hidden state to within 1.8×10^{-7} and is used for all results in Section 5.

B Per-Seed Values and Run-to-Run Variance

Every quantitative claim in the main text is reported as the mean over three independently trained models: the original run (seed 4242) and two additional random seeds (1, 2). Table 1 lists the per-seed value and the mean $\pm$ standard deviation for each reported quantity; the corresponding bands are drawn on Figs. 2, 3, 4 and 5. Each probe uses the same protocol across seeds (and the original seed reproduces the previously reported single-seed value). The pretrained DRC$(3, 3)$ is a single external checkpoint and is not seed-varied.

Quantity	source	seed 4242	seed 1	seed 2	mean $\pm$ sd
AttnLSTM — emergent planning					
Action decode @ agent cell (chance 0.25)	§5.1	0.615	0.577	0.615	0.60 ± 0.02
Routing decay ρ (graph distance)	§5.2, Fig. 3a	0.659	0.642	0.529	0.61 ± 0.06
One-hop attention mass	Fig. 3a	0.206	0.244	0.256	0.24 ± 0.02
Wall→agent shift ratio	§5.2, Fig. 3b	2.23	2.39	2.41	2.34 ± 0.08
On-route flip rate	Fig. 3b	0.200	0.183	0.200	0.19 ± 0.01
Off-route flip rate	Fig. 3b	0.067	0.067	0.083	0.072 ± 0.008
Hard-band goalward gain over ticks	§5.4, Fig. 4	$+0.081$	$+0.022$	$+0.089$	$+0.06 \pm 0.03$
First-step revision rate	§5.4, Fig. 5	0.346	0.244	0.234	0.27 ± 0.05
Thinking-curve solve rate (peak)	Fig. 2b	0.421	0.394	0.368	0.39 ± 0.02
Training-set solve rate	App. A	0.923	0.917	0.916	0.919 ± 0.003
SlotLSTM — free-slot control					
Routing decay ρ (near-uniform)	§5.5	0.946	0.959	0.912	0.94 ± 0.02
Routing mass beyond one hop	§5.5	0.721	0.649	0.651	0.67 ± 0.03
Winning-slot tile decode (chance 0.50)	§5.5	0.665	0.675	0.624	0.66 ± 0.02
Read mass on top square	§5.5	0.112	0.149	0.118	0.13 ± 0.02
Agent-slot stability across boards	§5.5	0.32	0.28	0.27	0.29 ± 0.02
Wall→slot value-shift ratio	§5.5	2.43	3.50	2.65	2.9 ± 0.5
Flip rate	§5.5	0.070	0.083	0.167	0.11 ± 0.04
Goalward gain over ticks	§5.5, Fig. 4	$+0.002$	$+0.010$	$+0.000$	$+0.004 \pm 0.004$
First-step revision rate	§5.5, Fig. 5	0.102	0.061	0.119	0.09 ± 0.02
Pondering solve rate (0 steps)	§5.5	0.073	0.051	0.076	0.067 ± 0.011
Pondering solve rate (8 steps)	§5.5	0.070	0.051	0.069	0.063 ± 0.009
Training-set solve rate	App. A	0.681	0.648	0.672	0.667 ± 0.014

Table 1: **Per-seed values and run-to-run variance** over three independently trained models. Main-text numbers are the mean column; the figures show mean$\pm$sd bands. The AttnLSTM planning signatures and the SlotLSTM control's failure both hold across all three seeds.