Thinking is Another Form of Control

Josiah P. Hanna[1] **and Nicholas E. Corrado**[1]
jphanna@cs.wisc.edu, ncorrado@wisc.edu

[1]**Computer Sciences Department, University of Wisconsin – Madison**

Abstract

Dual process theory divides cognitive processing into a fast, intuitive System 1 and a slow, deliberative System 2. In reinforcement learning (RL), model-free learning, in which the agent takes actions with a reactive policy, is reminiscent of System 1, whereas model-based decision-time planning is reminiscent of System 2. This paper presents the view that deliberative, System 2 behaviors ("thinking") can be considered a form of mental action that an agent performs before taking an action that influences its external environment. Under this view, we hypothesize that model-free RL alone would be sufficient to produce deliberation if these mental actions ultimately led to higher value actions being selected. We formalize the notion of a controllable "thought" state, then prove conditions under which "thinking" emerges as a strategy for reward maximization, and discuss how large language models serve as a proof-of-concept for thinking as mental action. Finally, we conclude by discussing new opportunities for research on model-free RL agents that learn both to think and act.

1 Introduction

Dual process theory divides cognitive processing into System 1 and System 2 processing (Wason & Evans, 1974; Kahneman, 2003). System 1 processing is fast, effortless, but potentially imprecise, whereas System 2 is slow, effortful, but potentially leads to better decisions. In artificial intelligence, System 1 capabilities are often associated with sub-symbolic, pattern-matching methods such as neural networks whereas System 2 capabilities are associated with symbolic, rule-based approaches such as planners and logic engines. Specifically, in reinforcement learning (RL), model-free learning is reminiscent of System 1 behavior; the agent observes the state of the environment and immediately computes its next action from a policy or value function. Model-based decision-time planning on the other hand is reminiscent of System 2 behavior; the agent observes the state of the environment and then uses a predictive model of the environment's dynamics to plan its next action. A widely-known example of an RL agent that exhibits both System 1 and System 2 behavior is the AlphaGo agent (Silver et al., 2016) that uses Monte Carlo tree search (Kocsis & Szepesvári, 2006) to plan actions (System 2) while evaluating board positions with a neural network-based value function trained with model-free RL (System 1).

Instead of distinct System 1 and System 2 modules, could a single mechanism produce both fast, reactive behaviors and slower, deliberative behaviors? In this paper, we present the view that deliberating at decision-time is just another form of action and consequently, model-free reinforcement learning (RL) can be sufficient to produce System 2 thinking in the absence of explicit search computations. The idea behind this view is to first understand System 2 thinking as the agent choosing to manipulate its own internal state before selecting an action. Though changing its internal state will not directly affect the agent's total reward, it may lead the agent to take a higher-value action than it would have otherwise. The implication of our view is that no explicit search computation is necessary for deliberative behavior to be realized in RL agents.

Recent work on large language models (LLMs) has produced a proof-of-concept that System 2 behavior can emerge when “thoughts" are simply another form of action (Guo et al., 2025). Specifically, recent work by Guo et al. proposed a unified model whereby System 2 type deliberation emerged as a consequence of model-free RL applied to solve mathematics problems (2025). While this deliberation is anthropomorphized as reasoning or planning, these outputs arise solely for the purpose of reward maximization without any constraints that they represent logic or search operators. The approach is interesting as it suggests that a form of System 2 processing can emerge from simply reinforcing thought patterns that lead to reward.

To formalize our view and show theoretically when model-free RL can produce thinking, we first introduce a minimal extension to the classical Markov decision process (MDP) model that we call a *thought MDP*. A thought MDP is an MDP in which the agent has an internal *thought state* and has access to *thought actions* that manipulate the thought state while leaving the state of the environment unchanged. In thought MDPs, policies choose whether to think or to act based upon both the environment state and the thought state. Thus, while thought actions do not directly produce reward or change the state of the environment, they can help the agent by manipulating the policy’s output for the current state. Under this model, we will show that policy iteration can lead to thinking behavior and this thinking behavior can be viewed as the agent choosing to improve its current policy before acting further. We then show how LLMs match the predictions of our theory. We conclude by discussing open questions raised by our view of deliberative thinking in RL.

2 Related Work

In this section, we discuss the prior literature related to System 2 processing in RL as well as prior models of “mental" actions an agent can take.

Decision-time Planning A hallmark of System 2 processing is deliberative planning as opposed to reflexive action. Thus, a natural way to instantiate System 2 in artificial agents is to use decision-time planning, e.g., Monte Carlo tree search (Kocsis & Szepesvári, 2006), and many works in the RL literature have used this approach (e.g., Anthony et al., 2017; Silver et al., 2016; 2017; Silver, 2009; Schrittwieser et al., 2021; Shah et al., 2020). Decision-time planning is particularly effective when the agent’s policy or value function is sub-optimal but the agent possesses an accurate model of state transitions (Sutton & Barto, 2018). In this case, planning can be viewed as focused policy improvement to improve the decision at the agent’s current state before it acts. As we will show, the idea of thinking as local policy improvement also applies when an RL agent learns to think in a thought MDP. The key difference is that we consider a more abstract model of thinking without any notion of forward prediction and search. There is also no natural answer to the question of when to stop planning and act with most decision-time planning methods. MCTS is an anytime algorithm and the general practice is to allow as much search is possible in the application domain (Silver et al., 2016; 2017). On the other hand, model-free learning in our thought MDP model will directly relate the decision about when to stop thinking to the objective of return maximization. Some work in the (non-learning) planning literature has also studied the question of when to think vs. when to act (Cashmore et al., 2019; Coles et al., 2024).

Learning to Plan As neural networks are general-purpose function approximators, they can, in principle, learn algorithmic planning or reasoning computations and prior work has attempted to induce planning capabilities through specialized architectures (Tamar et al., 2017; Farquhar et al., 2018; Guez et al., 2019; 2018; Niu et al., 2018; Weber et al., 2018; Sykora et al., 2020; Schleich et al., 2019; Oh et al., 2017). Guez et al. showed that planning-like computations could emerge through model-free learning (2019). Bush et al. (2025) applied mechanistic interpretability approaches to show that the approach of Guez et al. (2019) did in fact learn planning computations (i.e., the network internally learned to propose and evaluate future candidate action sequences). While such works also show that model-free RL can produce thinking behavior, the main difference is that thinking refers to the computation done by the agent’s policy, whereas we study thinking as an action selected by the agent’s policy. More similar to this conception of thinking, Chung et al. (2023) introduce the

thinker architecture in which a policy is trained to select actions both for environment interaction and planning. However, they conduct thinking for a fixed number of steps per environment step and thus the agent is learning *how* to think rather than *when* to think.

Reasoning in LLMs In the past couple of years, a large number of methods have been developed to create System 2 capabilities in LLMs, with particular emphasis on math and coding problems. One prominent approach is chain-of-thought (CoT) prompting in which a user changes their query to include explicit examples of correct responses (Wei et al., 2023). A more basic variant of CoT is just to prompt the model to "think step by step" (Kojima et al., 2023). Many other variations have also been proposed (e.g., Yao et al., 2023a;b; Wang et al., 2023). Another paradigm has been to augment output generation with an explicit search procedure (e.g., Khanov et al., 2024; Liu et al., 2023; Zhou et al., 2023; Zhang et al., 2023; Chen et al., 2024). Finally, recent works have used model-free RL to train LLMs to output CoT reasoning (Guo et al., 2025; OpenAI et al., 2024). Our work takes inspiration from these works but aims to go beyond LLMs in understanding when thinking-like behavior can emerge in RL agents.

Other Forms of Mental Action The idea of thinking as a form of action has a long history in AI and RL. Minsky theorized on the mind as a society of agents whose collective actions produce both cognition and action (Minsky, 1986). Klopf's hedonistic neuron hypothesis modelled neurons as individual RL agents (Klopf, 1982). This hypothesis has led to a line of work studying alternative neural network architectures in which artificial neural activity is produced by the stochastic policies of simple RL agents (Thomas, 2011; Kostas et al., 2019; Gupta et al., 2021). More similar to our work, some prior work has considered augmenting the agent's environment action-space with forms of mental action. These works have focused on the challenge of memory in partially observable domains and using explicit memory read or write actions as an alternative to recurrent neural networks. For instance, Peshkin et al. (2001) augment an RL agent with a set of memory write actions and the contents of the memory are then provided to the agent as an additional input along with the environment state; Zhang et al. (2015) extend this approach to robot manipulation tasks. Various neural architectures have been developed with external memory that can be written to and read from (e.g., Graves et al., 2014; Zaremba & Sutskever, 2016) and Oh et al. (2016) explored these architectures for RL. The thought MDP model differs in that the agent has access to a Markov state representation and so thought actions serve to manipulate the agent's policy as opposed to remembering details of the past.

The Options Framework Finally, thought MDPs are related to the options framework that is often used in hierarchical RL (Sutton et al., 1999). In the options framework, the agent's policy is over a set of options where options are either primitive actions or sub-policies. The crucial difference is that in the options framework, the selection of an option and the execution of that option's first action both occur within a single time step, whereas this execution would take at least two steps in a thought MDP. This difference suggests that thought MDPs might naturally model the cost of switching options, particularly when the agent cannot directly set its thought state and must instead think for multiple steps to select the best option. Thus, thought MDPs might be particularly useful as an alternative to the options with deliberation cost model (Harb et al., 2018).

3 A Formal Model of Thinking as Action

In this section, we first formalize the standard RL problem using the MDP formalism and then introduce the thought MDP model to explicitly model thinking.

3.1 RL in Markov Decision Processes

RL environments are typically modeled as Markov decision processes (MDPs). Here, we will consider an episodic MDP which is a tuple, $\langle \mathcal{S}, \mathcal{A}, p, r, \gamma \rangle$, where $\mathcal{S}$ is the environment state space, $\mathcal{A}$ is the set of actions that the agent can take to influence the environment state, $p : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$

is a stochastic state transition function, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a scalar-valued reward function, and γ is a discount factor. At any moment in time, the agent is in state s_t, takes an action, a_t, receives a reward, and transitions to a new state. Then the process repeats from s_{t+1} until a terminal state, s_∞, is reached. The agent selects actions according to a policy, $\pi : \mathcal{S} \to \Delta(\mathcal{A})$. The value of using a policy from a particular state, s, is defined to be $v_\pi(s) \coloneqq \mathbf{E}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s, a_t \sim \pi(\cdot|s_t)]$. In RL, the agent's objective is to find a policy that maximizes $v_\pi(s)$ in all states.

3.2 Thought MDPs

We now extend the MDP model to explicitly model thinking. We formally define a *thought MDP* as the tuple $\langle \mathcal{S}, \mathcal{A}, p, r, \gamma, \mathcal{T}, \mathcal{C}, p_\mathcal{T} \rangle$, where $\mathcal{S}, \mathcal{A}, p, r, \gamma$ are defined as they are for MDPs. We add $\mathcal{T}$ as the set of *thought states*, $\mathcal{C}$ as the set of *thought actions*, and $p_\mathcal{T} : \mathcal{S} \times \mathcal{T} \times \mathcal{C} \to \Delta(\mathcal{T})$ as the *thought transition function*. We emphasize that thought states and actions do not affect environment state transitions and rewards. The agent's objective remains to maximize cumulative discounted reward across all environment states.

There are different ways to define the agent's policy in a thought MDP. In this work, we formalize policies as a mapping $\pi : \mathcal{S} \times \mathcal{T} \to \Delta(\mathcal{A} \cup \mathcal{C})$. This choice means that the agent can select either an environment action or a thought action at any interaction time-step but not both. We make this choice for initial discussion as it better connects to deliberation in LLMs but an alternative is that the policy is a mapping $\pi : \mathcal{S} \times \mathcal{T} \to \Delta(\mathcal{A} \times \mathcal{C})$, i.e., the agent can think and act at the same time. Such modelling might be more useful in real-time domains where the agent cannot simply sit and think as the rest of the world evolves around it. There, of course, may be other alternatives that could be considered in future work. Below, we will use the notation $\pi(\tau)$ to refer to the agent's state-dependent policy with the thought state fixed at τ.

In a thought MDP, interaction proceeds as follows. Episodes begin in an environment state, s_0, and thought state, τ_0, and the agent chooses either an environment action or thought action according to its policy. If $a_0 \in \mathcal{A}$ then the environment state, s_1, at the next time-step is sampled from $p(s_0, a_0)$, the thought state τ_1 keeps the value of τ_0, and the agent receives reward $r_0 \coloneqq r(s_0, a_0)$. Conversely, if $a_0 \in \mathcal{C}$, then s_1 keeps the value of s_0, τ_1 is sampled from $p_\mathcal{T}(s_0, \tau_0, a_0)$, and the agent receives $r_0 = 0$. Then the process repeats until the agent reaches s_∞, with the agent selecting either an environment or thought action from its policy. For thought MDPs, we define the value function to be $v_\pi(s, \tau) \coloneqq \mathbf{E}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s, \tau_0 = \tau, a_t \sim \pi(\cdot|s_t, \tau_t)]$.

We aim to introduce a generalized model of thinking in MDPs, however, our subsequent analysis will make two assumptions which we state formally here. First, we assume that thought state transitions are deterministic (as is the case for LLM agents).

Assumption 1 (Deterministic Thought Transitions)**.** $\forall s \in \mathcal{S}, \tau \in \mathcal{T}, c \in \mathcal{C}\ p_\mathcal{T}(\tau'|s, \tau, c) = 1$ for one and only one $\tau' \in \mathcal{T}$. We will write $p_\mathcal{T}(s, \tau, c) \in \mathcal{T}$ to denote the thought state that results from taking c in (s, τ).

Second, we assume that all rewards are non-negative.

Assumption 2 (Non-negative Rewards)**.** $\forall s \in \mathcal{S}, a \in \mathcal{A}, r(s, a) \geq 0$.

Without Assumption 2, thinking could emerge as a strategy solely for the purpose of putting off receiving a negative reward, i.e., if all rewards are negative then the agent will be incentivized to just keep taking thought actions rather than environment actions. Finally, we assume reachable positive reward from all states as otherwise thought actions may be just as good as environment ones.

Assumption 3 (Reachable Positive Reward)**.** $\exists s \in \mathcal{S}, a \in \mathcal{A}$ with $r(s, a) > 0$ and $\forall \tilde{s} \in \mathcal{S}$ there exists a policy such that the probability of transitioning from $\tilde{s}$ to s in a finite number of steps is greater than zero.

Modeling of Time This modeling raises important questions about time. First, should the discount factor be applied equally for both thinking and non-thinking time-steps? Equal application discourages thinking as thought actions do not influence reward either directly or indirectly through

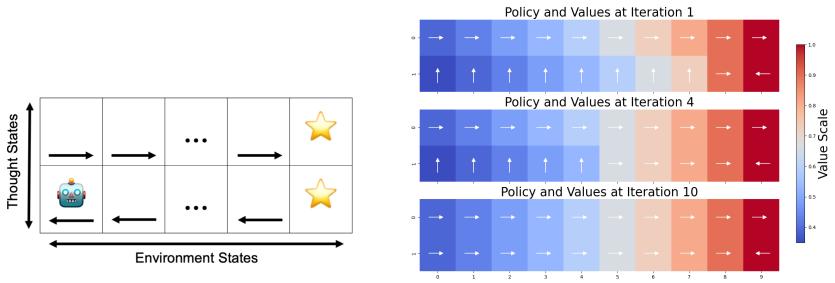

Figure 1: (left) An example thought MDP with $|\mathcal{T}| = 2$. We use $|\mathcal{S}| = 10$ in our illustrative results. The agent receives a reward when it reaches the goal environment state on the far right. The agent can move left or right in the environment state space and up and down in the thought state space. We use $\gamma = 0.9$ for both thinking and non-thinking time-steps. (right) Evolution of the policy and state values for 1, 4, and 10 iterations of policy iteration. The policy is initialized as shown on the left. Colors indicate value and arrows indicate the action that the policy would take.

the environment state. Nevertheless, as we shall see, thought actions still might be selected if they ultimately cause the agent to choose a better environment action. Alternatively, we could apply a different discount factor for thinking time-steps to reflect the actual time-delay of thinking compared to acting in a given domain. For example, if thinking takes $(1/k)$ the time of any environment action then we could use a discount of $\gamma^{\frac{1}{k}}$. In this paper, we will assume that the discount is applied the same at both thinking and non-thinking time steps. The second related issue is that the proposed model assumes that the environment state remains constant while the agent takes thought actions. Such an assumption might be reasonable for relatively static environments (such as generative language or board games) but is problematic in dynamic environments (such as autonomous driving) where waiting to act can be consequential. Again, for simplicity, we will focus on the static environment case, but note this issue as another interesting direction to refine the model.

Optimality Thought MDPs are MDPs with state space $\mathcal{S} \times \mathcal{T}$ and action space $\mathcal{A} \cup \mathcal{C}$ and consequently there will always be at least one deterministic optimal policy (Sutton & Barto, 2018). Furthermore, we can show that this optimal policy will never select thought actions.

Proposition 4. *Any optimal policy, $\pi^\star$, for a thought MDP does not take thought actions: $\pi^\star(s, \tau) \in \mathcal{A}$, $\forall s \in \mathcal{S}, \tau \in \mathcal{T}$.*

See Appendix A for proof.

4 Policy Initialization Determines Emergence of Thinking

If the optimal policy would never take a thought action, why should we expect thinking to emerge as a strategy during policy improvement? In this section, we begin to answer this question by showing the key role that policy initialization plays in determining the emergence of thinking. In particular, we first consider exact policy iteration within a thought MDP and provide a formal result and an illustrative example showing how thinking can emerge. We use policy iteration for analysis as essentially all model-free RL algorithms can be understood as instances of generalized policy iteration (Sutton & Barto, 2018). We then provide a second formal result showing that thought actions can reduce the effective horizon (Laidlaw et al., 2023) for the special case of goal-MDPs.

An illustrative example For exposition's sake, suppose $\mathcal{T} = \{\tau_0, \tau_1\}$, in which case we can view the agent's policy, π, as consisting of two sub-policies, $\pi(\tau_0)$ and $\pi(\tau_1)$. Now consider the example depicted in Figure 1 where $\pi(\tau_0)$ is initialized sub-optimally (always takes an action that leads away from the goal) while $\pi(\tau_1)$ is initialized to be optimal. We show the progress of policy iteration in this domain in Figure 1. After the first iteration of policy improvement, the policy has learned to

first change the agent's thought state to τ_1 (top row) from which it then follows $\pi(\tau_1)$ to reach the goal. After four iterations, in environment states close to the goal, the policy just directly moves right without changing its thought state while continuing to first take a thought action in states that are far from the goal. Finally, after ten iterations, the policy converges to the optimal policy and simply moves to the right without thinking. This example shows that, while thinking is sub-optimal in the long-run, it can be beneficial early in learning by allowing the agent to use sub-policies already contained in its policy function.

To formally understand how policy initialization determines the emergence of thinking, we analyze the policy improvement step of policy iteration.

Theorem 5. *Let π be a policy in a thought MDP such that $\pi(s,\tau) \in \mathcal{A}$ for some environment state s and thought state τ. If the policy improvement step of policy iteration sets $\pi'(s,\tau) \leftarrow c$ for $c \in \mathcal{C}$, then $v_\pi(s,\tau') > v_\pi(s,\tau)$, where τ' is the thought state resulting from taking c in (s,τ).*

See Appendix A for the proof. Although, Theorem 5 does not directly say anything about policy initialization, the condition for a thought action to be selected requires $\pi(s,\tau)$ and $\pi(s,\tau')$ to be somehow set up as different such that there could be an advantage in shifting from τ to τ'.

Thinking as a Policy Improvement Operator

One interpretation of Theorem 5 is that thought actions can function as policy improvement operators applied to a particular state. This interpretation is interesting as it aligns with the use of decision-time planning in RL to refine an agent's choice of action in a way that focuses computation on its current state (Sutton & Barto, 2018). While thought actions do not involve look-ahead search, Theorem 5 shows that their utility is also in providing local policy improvement. While Theorem 5 shows this utility for the choice of a single thought action, we also present a corollary showing that if policy improvement produces a policy that thinks for consecutive steps in s then each thinking step will further improve upon the action $\pi(s,\tau)$.

Corollary 6. *Let c be a thought action that leads from τ to τ' and c' be a thought action that leads from τ' to τ''. If, in some environment state s, the policy improvement step of policy iteration sets $\pi'(s,\tau) \leftarrow c$ and $\pi'(s,\tau') \leftarrow c'$, then $v_\pi(s,\tau'') > v_\pi(s,\tau') > v_\pi(s,\tau)$.*

Proof. The inequality $v_\pi(s,\tau') > v_\pi(s,\tau)$ immediately follows from Theorem 5 The inequality $v_\pi(s,\tau'') > v_\pi(s,\tau')$ follows from the same logic as the proof of Theorem 5 except applied to improving the policy in (s,τ'). □

If each step of thinking in s improves the policy, $\pi(s,\cdot)$, when should thinking terminate? From the proof of Theorem 5, we can see that thinking will terminate when the increase in value from thinking another step no longer compensates for the discounting of value caused by waiting a step to begin taking environmental actions. This observation connects the "when to think" vs. "when to act" decision directly to the agent's objective to maximize its total reward.

5 Language Generation as a Thought MDP

Our work takes inspiration from recent work on large reasoning models (LRMs) that "think" by generating additional text that is not part of the final answer but that somehow serves to improve the final answer. In this section, we review two approaches to imbue LLMs with reasoning capabilities and describe how they can be viewed as instantiating thought MDPs. We then show that forcing the LLM to reason (in a manner similar to zero-shot chain-of-thought Kojima et al. (2023)) increases the expected return from a given state. Thus Theorem 5 predicts that model-free RL applied to this thought MDP would lead to thinking, which is in fact what recent work has found.

We first describe language generation as an MDP and then describe the thought states and actions of LLMs. In language generation, each episode begins with a textual prompt, x and the agent's actions

are possible tokens from a fixed vocabulary. Let y_t be the agent's output at time t. The state at time t is defined as the prompt concatenated with the agent's outputs up to time t, $s_t = (x, y_{1:t})$. Rewards for RL applied to language generation have been defined in different ways. Much of the recent work on reasoning models focuses on math and coding problems, which have verifiable solutions. Thus, the reward is a terminal reward of 1 if a correct solution can be parsed and verified from $y_{1:t}$.

Reasoning with Language as a Thought MDP The main difficulty in mapping language generation to a thought MDP is that thought states are intertwined with environment states in the typical MDP formulation. Similarly, thought actions are simply from the same space as environment actions. To distinguish these components, we redefine the environment state as just the query and tokens that function as part of the query response. Similarly, we define environment actions as just the tokens that are part of the query response. Now, we will say that the entire sequence $(x, y_{1:t})$ is the thought state of the MDP (alternatively, one could view the activations of the transformer for this sequence as the thought state). Thought actions are the tokens in $y_{1:t}$ that are not part of the query response.

Different approaches to inducing reasoning in LLMs use different schemes for determining which tokens are part of the final response and which only serve to improve the final response. For example, Guo et al. (2025) augment the output vocabulary with two special functional tokens, `<think>` and `</think>`. In addition to the sparse verifier reward, Guo et al. train DeepSeek-R1 with reward shaping to encourage valid thinking blocks in which `<think>` is followed by `</think>`. Output text in thinking blocks is not part of the final response that is passed to a verifier to determine reward. Zero-shot prompting is another approach to encourage thinking-like behavior by appending "Let's think step-by-step" to the query (Kojima et al., 2023). In this approach, an answer parser is used to separate the response (environment actions) from the additional prompt and subsequent reasoning tokens (thought actions).

Do thought actions improve expected return in LLMs? Recent work from DeepSeek (and reportedly OpenAI) has shown that thinking-like behavior can emerge from RL. Based on our theory, we hypothesize that a pre-condition for this result is that thought actions increase $v_\pi(s, \tau)$ by changing the thought state τ. Because we lack a value function for each LLM, we instead approximate $v_\pi(s, \tau)$ with the Monte Carlo return or, equivalently, the accuracy of the LLM's response. To test our hypothesis, we take different pre-trained LLMs and apply them to add series of five four-digit numbers. We test two conditions with each model: "No Thinking" and "Thinking." Under both conditions, the prompt begins with "Compute the sum of [a], [b], [c], [d], [e]." where [a], [b], [c], [d], and [e] correspond to four-digit integers. Under the "No Thinking" condition we simply append "[a] + [b] + [c] + [d] + [e]=" to this query, whereas for "Thinking" we append "[a] + [b] + [c] + [d] + [e] = [a + b] + [c] + [d] + [e] = [a + b + c] + [d] + [e]= [a + b + c + d] + [e]" to the original query where [a + b] and [a + b + c] denote the partial sums a+b and a+b+c. In essence, we force the model to first think under the "Thinking" condition. We constuct 100 "Thinking" and 100 "No Thinking" prompts like these using 100 different sequences of four 4-digit integers, each generated uniformly at random. Figure 2 shows the average accuracy for each model, averaged over multiple five-number addition problems. For both models, we see that appending the thinking tokens increases accuracy, which corresponds to increasing $v_\pi(s, \tau)$ by changing τ. While we do not further apply model-free RL to try and learn to think in this way, our theory and these results predict that these models are primed for thinking to further emerge as a strategy.

Model	No Thinking (%)	Thinking (%)
Qwen2.5-7B-Instruct	5.06 ± 3.10	96.10 ± 2.98
Qwen2.5-14B-Instruct	0.90 ± 0.33	95.20 ± 1.33
Tulu-2-7b	0.30 ± 0.33	28.50 ± 2.80
Tulu-2-13b	0.60 ± 0.47	49.00 ± 3.10
Llama-2-7b	0.00 ± 0.00	48.80 ± 3.10
Llama-2-13b	0.20 ± 0.27	60.70 ± 3.03
Gemma-3-4b-it	4.90 ± 1.33	91.50 ± 1.72
Mistral-7B-Instruct-v0.3	1.50 ± 0.74	85.20 ± 2.20

Figure 2: Response accuracy $\pm$ 95% confidence interval when using thinking vs no thinking.

6 Discussion and Future Work

In this section, we discuss possible extensions for research in thought MDPs. Due to space constraints, we only describe a few directions while briefly noting that further research should consider extensions to partially observable worlds and connections to natural intelligence (**?**Klopf, 1982) and the options framework (Sutton et al., 1999).

Where do thought MDPs come from? This work studied the question of why thinking actions could be useful to an RL agent even though they leave their environment state unchanged and produce no reward. We formalized this abstract notion of thinking with the thought MDP model, but left open the big question of how to define the thought states, actions, and thought dynamics in other RL problems where thinking may be useful. Language generation is one example but it is an open question as to how thought MDPs might arise outside the language domain. There is also the question of where existing thought-conditioned policies come from, as our work showed their existence to be a key enabler of emergent thinking.

Connecting to Models and Planning System 2 processing is reminiscent of decision-time planning using a model of the environment state transition function (Anthony et al., 2017). This work has presented a more abstract model of thinking where the agent simply learns to control an internal thought state. While distinct models, both decision-time planning and thinking in a thought MDP are related in that they amount to locally focused policy improvement (Sutton & Barto, 2018). Furthermore, having thought states and actions that are somehow grounded in environment states and actions could explain how the agents' thought dynamics should be structured.

Thinking in Dynamic and Time-constrained Domains This work has only considered the utility of thinking in relatively static domains where the environment state remains the same during thinking. In reality, the environment state may change due to influences other than the agent's actions, and the choice to stop and think must factor in how the environment might change while it does so. A natural extension would be to have thought states and environment states unfold in parallel to one another, with the agent thinking and acting at the same time.

Agents with Bounded Capacity Could the utility of thinking be enhanced under constraints? We take inspiration from the Big World Hypothesis (Javed & Sutton, 2024) which states that agents will always have less capacity than what is required to learn all possible tasks. Consequently, when an agent is faced with a new task it may have either never seen it or have forgotten how to do it. Thinking might allow rapid repurposing of an agent's present capabilities to learn quickly on a new task. The sub-policies that are more frequently used would be repeatedly reinforced whereas less frequently used sub-policies could be forgotten.

7 Conclusion

In this work, we investigated the question of when model-free RL will lead to "thinking" behavior. We introduced the thought MDP model and then used this model to show that thinking emerges as a strategy to improve an agent's choice of environment action and will thus depend upon how the agent's policy is initialized. We then provided supporting evidence that step-by-step reasoning in LLMs functions similarly to thinking. Finally, we developed a non-language domain in which thinking would emerge as a strategy for reward maximization and discussed the many exciting next steps for developing AI agents that learn to think.

Acknowledgments We thank Adam Labiosa and the workshop reviewer for helpful feedback that shaped the final version of this paper. Josiah Hanna is supported in part by the National Science Foundation (IIS-2410981) and Sandia National Labs through a University Partnership Award.

References

Thomas Anthony, Zheng Tian, and David Barber. Thinking Fast and Slow with Deep Learning and Tree Search, December 2017. URL http://arxiv.org/abs/1705.08439. arXiv:1705.08439 [cs].

Thomas Bush, Stephen Chung, Usman Anwar, Adrià Garriga-Alonso, and David Krueger. Interpreting Emergent Planning in Model-Free Reinforcement Learning, April 2025. URL http://arxiv.org/abs/2504.01871. arXiv:2504.01871 [cs].

Michael Cashmore, Andrew Coles, Bence Cserna, Erez Karpas, Daniele Magazzeni, and Wheeler Ruml. Replanning for Situated Robots. *Proceedings of the International Conference on Automated Planning and Scheduling*, 29:665–673, July 2019. ISSN 2334-0843. DOI: 10.1609/icaps.v29i1.3534. URL https://ojs.aaai.org/index.php/ICAPS/article/view/3534.

Ziru Chen, Michael White, Raymond Mooney, Ali Payani, Yu Su, and Huan Sun. When is Tree Search Useful for LLM Planning? It Depends on the Discriminator, February 2024. URL http://arxiv.org/abs/2402.10890. arXiv:2402.10890 [cs].

Stephen Chung, Ivan Anokhin, and David Krueger. Thinker: Learning to Plan and Act. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, November 2023. URL https://openreview.net/forum?id=mumEBl0arj.

Andrew Coles, Erez Karpas, Andrey Lavrinenko, Wheeler Ruml, Solomon Eyal Shimony, and Shahaf Shperberg. Planning and Acting While the Clock Ticks. *Proceedings of the International Conference on Automated Planning and Scheduling*, 34:95–103, May 2024. ISSN 2334-0843. DOI: 10.1609/icaps.v34i1.31465. URL https://ojs.aaai.org/index.php/ICAPS/article/view/31465.

Gregory Farquhar, Tim Rocktäschel, Maximilian Igl, and Shimon Whiteson. TreeQN and ATreeC: Differentiable Tree-Structured Models for Deep Reinforcement Learning, March 2018. URL http://arxiv.org/abs/1710.11417. arXiv:1710.11417 [cs].

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural Turing Machines, December 2014. URL http://arxiv.org/abs/1410.5401. arXiv:1410.5401 [cs].

Arthur Guez, Théophane Weber, Ioannis Antonoglou, Karen Simonyan, Oriol Vinyals, Daan Wierstra, Rémi Munos, and David Silver. Learning to Search with MCTSnets, July 2018. URL http://arxiv.org/abs/1802.04697. arXiv:1802.04697 [cs].

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, May 2019. URL http://arxiv.org/abs/1901.03559. arXiv:1901.03559 [cs].

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Dhawal Gupta, Gabor Mihucz, Matthew K. Schlegel, James E. Kostas, Philip S. Thomas, and Martha White. Structural Credit Assignment in Neural Networks using Reinforcement Learning. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Jean Harb, Pierre-Luc Bacon, Martin Klissarov, and Doina Precup. When waiting is not an option: Learning options with a deliberation cost. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Khurram Javed and Richard S. Sutton. The Big World Hypothesis and its Ramifications for Artificial Intelligence. In *The Finding the Frame Workshop at the Reinforcement Learning Conference*, July 2024. URL https://openreview.net/forum?id=Sv7DazuCn8.

Daniel Kahneman. Maps of bounded rationality: Psychology for behavioral economics. *American economic review*, 93(5):1449–1475, 2003.

Maxim Khanov, Jirayu Burapacheep, and Yixuan Li. ARGS: Alignment as Reward-Guided Search, January 2024. URL http://arxiv.org/abs/2402.01694. arXiv:2402.01694 [cs].

A. Harry Klopf. *The Hedonistic Neuron: A Theory of Memory, Learning, and Intelligence*. Hemisphere Publishing Corporation, 1982. ISBN 978-0-89116-202-5. Google-Books-ID: kMxqAAAAMAAJ.

Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *European conference on machine learning*, pp. 282–293. Springer, 2006.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large Language Models are Zero-Shot Reasoners, January 2023. URL http://arxiv.org/abs/2205.11916. arXiv:2205.11916 [cs].

James E. Kostas, Chris Nota, and Philip S. Thomas. Asynchronous Coagent Networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2019. URL https://arxiv.org/abs/1902.05650v4.

Cassidy Laidlaw, Stuart J Russell, and Anca Dragan. Bridging rl theory and practice with the effective horizon. *Advances in Neural Information Processing Systems*, 36:58953–59007, 2023.

Jiacheng Liu, Andrew Cohen, Ramakanth Pasunuru, Yejin Choi, Hannaneh Hajishirzi, and Asli Celikyilmaz. Don't throw away your value model! Making PPO even better via Value-Guided Monte-Carlo Tree Search decoding, October 2023. URL http://arxiv.org/abs/2309.15028. arXiv:2309.15028 [cs].

Marvin Minsky. *Society of mind*. Simon and Schuster, 1986.

Sufeng Niu, Siheng Chen, Hanyu Guo, Colin Targonski, Melissa Smith, and Jelena Kovačević. Generalized Value Iteration Networks:Life Beyond Lattices. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), April 2018. ISSN 2374-3468. DOI: 10.1609/aaai.v32i1.12081. URL https://ojs.aaai.org/index.php/AAAI/article/view/12081. Number: 1.

Junhyuk Oh, Valliappa Chockalingam, and Honglak Lee. Control of memory, active perception, and action in minecraft. In *International conference on machine learning*, pp. 2790–2799. PMLR, 2016. URL http://proceedings.mlr.press/v48/oh16.html.

Junhyuk Oh, Satinder Singh, and Honglak Lee. Value Prediction Network, November 2017. URL http://arxiv.org/abs/1707.03497. arXiv:1707.03497 [cs].

OpenAI, Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, and et al. OpenAI o1 System Card, December 2024. URL http://arxiv.org/abs/2412.16720. arXiv:2412.16720 [cs].

Leonid Peshkin, Nicolas Meuleau, and Leslie Kaelbling. Learning policies with external memory. *arXiv preprint cs/0103003*, 2001.

Daniel Schleich, Tobias Klamt, and Sven Behnke. Value Iteration Networks on Multiple Levels of Abstraction. In *Robotics: Science and Systems XV*, June 2019. DOI: 10.15607/RSS.2019.XV.014. URL http://arxiv.org/abs/1905.11068. arXiv:1905.11068 [cs].

Julian Schrittwieser, Thomas Hubert, Amol Mandhane, Mohammadamin Barekatain, Ioannis Antonoglou, and David Silver. Online and Offline Reinforcement Learning by Planning with a Learned Model. In *Advances in Neural Information Processing Systems*, volume 34, pp. 27580–27591. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/hash/e8258e5140317ff36c7f8225a3bf9590-Abstract.html.

Devavrat Shah, Qiaomin Xie, and Zhi Xu. Non-Asymptotic Analysis of Monte Carlo Tree Search. In *Abstracts of the 2020 SIGMETRICS/Performance Joint International Conference on Measurement and Modeling of Computer Systems*, pp. 31–32, Boston MA USA, June 2020. ACM. ISBN 978-1-4503-7985-4. DOI: 10.1145/3393691.3394202. URL https://dl.acm.org/doi/10.1145/3393691.3394202.

David Silver. *Reinforcement learning and simulation-based search*. PhD thesis, University of Alberta, 2009.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, January 2016. ISSN 1476-4687. DOI: 10.1038/nature16961. URL https://www.nature.com/articles/nature16961. Publisher: Nature Publishing Group.

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, October 2017. ISSN 1476-4687. DOI: 10.1038/nature24270. URL https://www.nature.com/articles/nature24270. Publisher: Nature Publishing Group.

Richard Sutton and Andrew Barto. *Reinforcement Learning: an Introduction*. MIT Press, 2018.

Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

Quinlan Sykora, Mengye Ren, and Raquel Urtasun. Multi-Agent Routing Value Iteration Network. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 9300–9310. PMLR, November 2020. URL https://proceedings.mlr.press/v119/sykora20a.html. ISSN: 2640-3498.

Aviv Tamar, Yi Wu, Garrett Thomas, Sergey Levine, and Pieter Abbeel. Value Iteration Networks, March 2017. URL http://arxiv.org/abs/1602.02867. arXiv:1602.02867 [cs].

Philip S. Thomas. Policy Gradient Coagent Networks. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2011.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-Consistency Improves Chain of Thought Reasoning in Language Models, March 2023. URL http://arxiv.org/abs/2203.11171. arXiv:2203.11171 [cs].

Peter C Wason and J St BT Evans. Dual processes in reasoning? *Cognition*, 3(2):141–154, 1974.

Théophane Weber, Sébastien Racanière, David P. Reichert, Lars Buesing, Arthur Guez, Danilo Jimenez Rezende, Adria Puigdomènech Badia, Oriol Vinyals, Nicolas Heess, Yujia Li, Razvan Pascanu, Peter Battaglia, Demis Hassabis, David Silver, and Daan Wierstra. Imagination-Augmented Agents for Deep Reinforcement Learning, February 2018. URL http://arxiv.org/abs/1707.06203. arXiv:1707.06203 [cs].

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models, January 2023. URL http://arxiv.org/abs/2201.11903. arXiv:2201.11903 [cs].

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of Thoughts: Deliberate Problem Solving with Large Language Models, December 2023a. URL http://arxiv.org/abs/2305.10601. arXiv:2305.10601 [cs].

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing Reasoning and Acting in Language Models, March 2023b. URL http://arxiv.org/abs/2210.03629. arXiv:2210.03629 [cs].

Wojciech Zaremba and Ilya Sutskever. Reinforcement Learning Neural Turing Machines - Revised, January 2016. URL http://arxiv.org/abs/1505.00521. arXiv:1505.00521 [cs].

Marvin Zhang, Zoe McCarthy, Chelsea Finn, Sergey Levine, and Pieter Abbeel. Learning Deep Neural Network Policies with Continuous Memory States, September 2015. URL http://arxiv.org/abs/1507.01273. arXiv:1507.01273 [cs].

Shun Zhang, Zhenfang Chen, Yikang Shen, Mingyu Ding, Joshua B. Tenenbaum, and Chuang Gan. Planning with Large Language Models for Code Generation, March 2023. URL http://arxiv.org/abs/2303.05510. arXiv:2303.05510 [cs].

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language Agent Tree Search Unifies Reasoning Acting and Planning in Language Models, December 2023. URL http://arxiv.org/abs/2310.04406. arXiv:2310.04406 [cs].

A Proofs of Theoretical Results

Proposition 4. *Any optimal policy, $\pi^\star$, for a thought MDP does not take thought actions: $\pi^\star(s,\tau) \in \mathcal{A}$, $\forall s \in \mathcal{S}, \tau \in \mathcal{T}$.*

Proof. The proof is by contradiction. Suppose $\pi^\star$ is an optimal policy and $\exists s \in \mathcal{S}, \tau \in \mathcal{T}$ such that $\pi(s,\tau) \in \mathcal{C}$. Because thought actions cannot produce reward, the optimal policy must eventually reach a thought state, $\tilde{\tau}$, where $\pi^\star(s,\tilde{\tau}) \in \mathcal{A}$. If that happens after k thought actions, then $v_{\pi^\star}(s,\tau) = \gamma^k v_{\pi^\star}(s,\tilde{\tau})$. But then the policy could be strictly improved by setting $\pi^\star(s,\tau) \leftarrow \pi^\star(s,\tilde{\tau})$. This contradicts the assumption that $\pi^\star$ was optimal and completes the proof. □

Theorem 5. *Let π be a policy in a thought MDP such that $\pi(s,\tau) \in \mathcal{A}$ for some environment state s and thought state τ. If the policy improvement step of policy iteration sets $\pi'(s,\tau) \leftarrow c$ for $c \in \mathcal{C}$, then $v_\pi(s,\tau') > v_\pi(s,\tau)$, where τ' is the thought state resulting from taking c in (s,τ).*

Proof. Policy iteration sets $\pi'(s,\tau) \leftarrow \arg\max_{x \in \mathcal{A} \cup \mathcal{C}} q_\pi(s,\tau,x)$ where

$$q_\pi(s,\tau,x) \coloneqq \begin{cases} r(s,x) + \gamma \mathbf{E}_{s'}[v_\pi(s',\tau)] & \text{if } x \in \mathcal{A} \\ \gamma v_\pi(s,\tau') & \text{if } x \in \mathcal{C}. \end{cases}$$

Since the policy improvement step is selecting a thought action, we have that:

$$\forall a \in \mathcal{A}, q_\pi(s,\tau,a) < \gamma v_\pi(s,\tau').$$

Since $\pi(s,\tau)$ was in $\mathcal{A}$ before the update we have that:

$$v_\pi(s,\tau) = q_\pi(s,\tau,\pi(s,\tau)) < \gamma v_\pi(s,\tau') < v_\pi(s,\tau') = q_\pi(s,\tau',\pi(s,\tau')).$$

Thus, a thought action is selected when the policy is such that $\pi(s,\tau')$ is expected to lead to more reward than $\pi(s,\tau)$. □