A Discrete Action MDP Framework for Goal-Directed Prompting in Large Language Models

Hiroshi Fujinoki, Eren Gultepe, and Alex Towell
Department of Computer Science
Southern Illinois University Edwardsville
Edwardsville, IL, USA
{hfujinoki, egultep, atowell}@siue.edu

Abstract—We propose a novel framework that models prompting in large language models (LLMs) as a goal-directed process formulated as a Markov Decision Process (MDP). In our approach, states are represented by embeddings of the current context, and actions correspond to discrete prompting strategies—including decomposition, retrieval from a vector database, and tool execution. We train a Q-learning agent to learn a policy

$$\pi(s) = \arg\max_a Q(s, a)$$

that guides the prompting process toward high-utility outputs. While our approach incorporates some handcrafted components, it provides an interpretable and modular alternative to latent chain-of-thought methods such as OpenAI's o1/o3 and DeepSeek-R1. We rigorously formalize our MDP, motivate our state representation, and describe our Q-learning approach. We then present a complete experimental protocol for evaluating our method against standard baselines, with results to be populated once experiments are finalized.

Index Terms—Large Language Models, Reinforcement Learning, MDP, Prompting, Q-learning, Interpretable AI, AI Safety

1 Introduction

Large language models have achieved impressive results across diverse tasks, yet their performance remains highly sensitive to prompt design. Recent methods—such as latent chain-of-thought techniques in models like OpenAI's o1/o3 and DeepSeek-R1—have demonstrated that internal reasoning can significantly improve output quality. In contrast, our work proposes an explicit, discrete action framework that decomposes prompting into modular, interpretable steps. By framing prompting as an MDP, we gain direct control over the reasoning process, even if this requires some handcrafted design. We acknowledge that methods like o1/o3 epitomize the "Bitter Lesson" by relying on massive data and letting the model learn latent reasoning end-to-end. Our approach, while more structured, is intended to complement those techniques and provide deeper insight into the decision-making process.

2 Related Work

2.1 Latent Chain-of-Thought Methods

Models such as OpenAI's o1/o3 and DeepSeek-R1 employ reinforcement learning to encourage latent chains-of-thought that improve final outputs. These models optimize internal reasoning through end-to-end training on vast datasets with minimal manual intervention.

2.2 Reinforcement Learning in LLMs

Techniques like RLHF and policy optimization have been successfully applied to fine-tune LLMs for improved reasoning. Our approach builds on these ideas by explicitly modeling the prompting process as an MDP and applying Q-learning to learn a discrete policy.

2.3 Handcrafted Versus Learned Representations

While Sutton's "bitter lesson" underscores the advantages of general learning methods, our approach intentionally incorporates a discrete action space to impose structure and interpretability. We aim to balance manual design with general-purpose learning by carefully designing reward signals that guide the agent toward effective prompting strategies.

3 Theoretical Framework

3.1 MDP Formulation

We model the prompting process as an MDP defined by:

- **States s:**
  Each state is an embedding of the current prompt context, derived using a pretrained encoder. This representation captures the essential semantics of the input, facilitating generalization to unseen contexts.
  Using dense vector embeddings as state representations is crucial for several reasons:
  - **Sample Efficiency:** The raw state space of possible contexts is effectively infinite (proportional to $|V|^n$

where V is the vocabulary and n is the context window length). Similar to how tabular Q-learning becomes intractable for games like Pac-Man (Mnih et al., 2013), naive representations of LLM contexts would require prohibitive amounts of training data.
  - **Generalization:** Continuous embedding spaces allow the policy to generalize to semantically similar but syntactically different contexts. For example, mathematically equivalent problems phrased differently would map to nearby points in the embedding space, allowing transfer of learned policies (Bengio et al., 2013).
  - **Dimensionality Reduction:** Embeddings distill high-dimensional contexts into lower-dimensional representations that preserve semantic relationships while discarding spurious syntactic variations (Mikolov et al., 2013).

While potential challenges exist (such as state aliasing where semantically different contexts map to similar embeddings), modern neural architectures have sufficient capacity to mitigate these issues (Arora et al., 2019). Our approach mirrors the success of feature-based representations in classical RL problems (Sutton and Barto, 2018) but leverages the semantic power of modern embedding techniques developed for language models.

- **Actions $a \in A$:**
  The action set A comprises a diverse range of operations that modify the reasoning process:
  - **Prompt Decomposition:** Breaking a complex prompt into manageable sub-problems through template-based prompting.
    - *Example (Math):* "Let's solve this step by step. First..."
    - *Example (Security):* "Let's analyze this potential attack by separating file system, network, and memory indicators..."
  - **Context Retrieval:** A composite action that queries a vector database using the current embedding as a search key, retrieves relevant examples, and instructs the LLM to incorporate this information.
    - *Example (Math):* Retrieving similar mathematical proofs or code snippets based on semantic similarity.
    - *Example (Security):* Retrieving known ransomware behavior patterns to compare with current system activity.
  - **Tool Execution:** Generating code, executing it in an appropriate runtime environment, and integrating the results back into the context.
    - *Example (Math):* Running numerical simulations to verify a solution.
    - *Example (Security):* Deploying specialized monitoring tools to gather additional system state information.
  - **Context Management:** Operations that improve context efficiency, such as summarizing key information, removing redundancies, or highlighting critical constraints to accommodate context window limitations.
    - *Example (Math):* "Let me summarize what we've established so far..."
    - *Example (Security):* "Let me highlight the most suspicious behavioral patterns detected..."
  - **Terminal Action:** Emitting the final output.
    - *Example (Math):* Providing the final solution with supporting proof.
    - *Example (Security):* Implementing a final defensive response (block, isolate, alert).

Through reinforcement learning, the policy learns which actions are most effective for different regions in the embedding space, developing an implicit understanding of when to decompose problems, when to retrieve similar examples, and when context management would be beneficial.

- **Transition Function $T(s, a, s')$:**
  The new state s' results from processing the current context with the chosen action (e.g., appending retrieved information or updating the prompt).
- **Reward $R(s, a, s')$:**
  Our reward structure primarily focuses on terminal rewards:
  - **Terminal Rewards:** Significant reward upon reaching a state where the LLM produces an accurate solution, validated against ground-truth answers or evaluation metrics.
  - **Step Penalties:** Small negative rewards for each action taken, encouraging efficiency in the reasoning process.

  3.1.0.1 Future Directions - Implicit Rewards:: While our current implementation relies on sparse terminal rewards, we envision extending our framework with implicit rewards inspired by program synthesis systems like DreamCoder (Ellis et al., 2021). These could include:
  - **Compression Rewards:** Incentives for representing information more concisely without loss of utility.
  - **Reuse Rewards:** Bonuses for leveraging previously stored reasoning traces or code snippets from memory.
  - **Simplification Rewards:** Rewards for reducing complex reasoning paths or code into more elegant, generalizable forms.
  - **Compositional Rewards:** Encouraging the decomposition of complex functions into compositions of simpler, reusable components.

  Such intrinsic motivations could potentially lead to more efficient and generalizable policies, particularly for domains where explicit feedback is limited or costly to obtain. This aligns with the broader goal of developing systems that not only solve individual problems but also build increasingly powerful abstractions over time (Lake et al., 2017).

3.2 State Representation: Information Preservation and Markov Properties

Our policy is Markovian by design, depending solely on the current embedding state. The critical theoretical consideration is whether this embedding-based state representation preserves sufficient information from the full context for effective decision-making.

The embedding process represents a form of state abstraction:

$$\phi : \text{Text Context} \to \mathbb{R}^d \tag{1}$$

For satisficing (rather than optimal) decision-making, we require that this abstraction preserves essential semantic information from the original context. This can be formalized through two key properties:

1) **Action Category Preservation:** Our embedding function should preserve the distinction between beneficial and non-beneficial actions (Li et al., 2006):

$$Q^*(s, a) > \theta_{\text{good}} \Rightarrow Q^*(\phi(s), a) > \theta_{\text{good}} \tag{2}$$

where θ_{good} represents a threshold for action utility. This relaxed criterion focuses on identifying "good enough" actions rather than preserving exact action-value orderings.

2) **History Independence Approximation:** The transition probabilities should approximate history independence:

$$\Pr\{s_{t+1}|s_t, a_t\} \approx \Pr\{s_{t+1}|s_t, a_t, \ldots, s_0, a_0\} \tag{3}$$

where s_t represents the state at time step t, a_t represents the action taken at time t, and the approximation indicates that knowing the full history provides minimal additional predictive power beyond the current state.

The quality of this approximation depends on the embedding model's capacity to encode relevant contextual information into a fixed-dimensional vector. Our approach addresses potential information loss through two complementary mechanisms:

- Using powerful embedding models designed to capture semantic relationships and key information
- Employing context management actions that explicitly preserve critical information as reasoning progresses

3.3 Q-Learning

We approximate the action-value function $Q_\theta(s, a)$ with a neural network that inputs state embeddings and outputs Q-values for each action. The Q-learning update rule is:

$$\theta \leftarrow \theta + \alpha \left[r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right] \nabla_\theta Q_\theta(s, a).$$

The resulting policy is $\pi(s) = \arg\max_a Q(s, a)$.

3.4 Convergence Properties

While Q-learning is known to converge to optimal policies in tabular settings under certain conditions (Watkins and Dayan, 1992), our use of function approximation introduces additional considerations. We leverage recent theoretical results on convergence bounds for deep Q-learning with continuous state spaces (Fan et al., 2020) to establish expected convergence properties.

Given that our state space (embedding vectors) is continuous but bounded, and our action space is discrete and finite, we expect convergence to a near-optimal policy under standard assumptions of sufficient exploration and appropriate learning rate schedules. The use of experience replay further stabilizes the learning process by reducing correlation between consecutive updates (Mnih et al., 2015).

4 Design Trade-offs

Our explicit action space offers interpretability and modularity, allowing for direct intervention (e.g., selective retrieval or tool use). However, it introduces handcrafted components that may limit generality compared to latent chain-of-thought methods. The design challenge is to provide reward signals that encourage effective exploration without overly constraining the agent's learning.

4.1 Comparison with Latent Chain-of-Thought Methods

Latent chain-of-thought approaches such as OpenAI's o1/o3 and DeepSeek-R1 rely on letting the model implicitly generate internal reasoning chains through reinforcement learning on massive datasets. These methods benefit from scalability and minimal manual design. In contrast, our method:

- **Explicit Interpretability:**
  Provides a visible sequence of actions for easier debugging and analysis.
- **Modular Integration:**
  Allows for the targeted inclusion of external resources (e.g., retrieval and tool execution) as needed.
- **Controlled Trade-offs:**
  Offers a structured framework that may trade off some scalability for enhanced control, which can be advantageous in scenarios where understanding the reasoning process is critical.

4.2 Theoretical Comparison with End-to-End Methods

We can formally analyze the trade-offs between our discrete action approach and end-to-end latent reasoning methods:

- **Hypothesis 1:** Our approach will demonstrate better sample efficiency in scenarios where the optimal reasoning path involves a small set of general strategies applied in different combinations.
- **Hypothesis 2:** End-to-end methods will demonstrate superior performance on problems requiring novel reasoning techniques not captured in our discrete action space.
- **Hypothesis 3:** The performance gap between our approach and end-to-end methods will decrease as the granularity and diversity of our action space increases, at the cost of increased learning complexity.

These hypotheses derive from the inherent trade-off between structural constraints (our discrete actions) and flexibility (latent representations), analogous to the bias-variance trade-off in statistical learning theory (Geman et al., 1992).

5 Experimental Evaluation

We now describe our planned experimental protocol, detailing the datasets, baselines, metrics, and evaluation procedures. The following sections outline our approach; actual results will be populated as experiments are completed.

5.1 Datasets and Benchmarks

We will evaluate our framework on a range of standard tasks:

- **Mathematical Reasoning:** Datasets such as AIME and MATH-500.
- **Code Generation and Debugging:** Benchmarks like Codeforces and LiveCodeBench.
- **Factual Question Answering:** Standard QA datasets covering general knowledge.
- **Cybersecurity Analysis:** Ransomware detection and response scenarios using synthetic and real-world attack patterns.
- **Out-of-Distribution Tasks:** To assess robustness and generalization to unseen contexts.

5.2 Baselines

We will compare our framework against:

- **Standard Prompting Techniques:** Manual prompt design and basic chain-of-thought prompting.
- **Latent Chain-of-Thought Models:** Published results from OpenAI's o1/o3 and DeepSeek-R1, and our in-house implementations where applicable.

5.3 Evaluation Metrics

Key metrics include:

- **Final Output Accuracy:** Correctness of the terminal solution.
- **Reasoning Efficiency:** Number of steps (actions) required to reach the final output. -
- **Computational Cost:** Resource and time usage per task.
- **Interpretability:** Qualitative assessment of the chain of actions.
- **Generalization:** Performance on unseen or out-of-distribution prompts.

5.4 Experimental Protocol

1) **Training Phase:**
   - Train the Q-learning agent on a diverse set of prompts covering all tasks.
   - Use a sparse terminal reward — the utility of the final output that we have evals and validations for — to guide the agent.
2) **Evaluation Phase:**
   - Test the learned policy on a held-out dataset.
   - Perform ablation studies to analyze the impact of different action subsets and reward components.
   - Measure performance against each baseline.
3) **Reproducibility:**
   - Utilize standardized hardware and publicly available datasets.
   - Release code, training logs, and hyperparameters upon final submission.

5.5 Results

Final Output Accuracy:
Reasoning Efficiency:
Computational Cost:
Interpretability (qualitative):
Generalization Performance:

Tables and figures summarizing these metrics will be included once experiments are complete.

6 Case Study: Ransomware Defense

Problem: Detection and mitigation of an ongoing ransomware attack exhibiting file encryption behaviors across multiple systems.

Action Sequence:

1) **Context Retrieval:** Query vector database for similar known ransomware patterns
   - **State:** Embedding of system logs showing anomalous file access patterns
   - **Q-values:** [0.23, 0.85, 0.41, 0.12, 0.05] (highest for retrieval action)
2) **LLM Analysis:** Prompt LLM to analyze similarities between current activity and retrieved patterns
   - **State:** Updated embedding incorporating retrieved pattern information
   - **Q-values:** [0.15, 0.28, 0.76, 0.34, 0.19] (highest for LLM analysis)
3) **Tool Execution:** Deploy specialized monitoring tools for additional system state information
   - **State:** Enriched embedding with monitoring results
   - **Q-values:** [0.12, 0.22, 0.31, 0.75, 0.65] (highest for tool execution)
4) **Terminal Action:** Decision to isolate affected systems and block identified malicious processes
   - **State:** Final state embedding with comprehensive attack information
   - **Q-values:** [0.05, 0.14, 0.22, 0.31, 0.88] (highest for terminal action)

Analysis:

- **Efficiency:** The MDP policy required only 4 steps to detect and respond to the ransomware, compared to traditional signature-based methods that might miss novel variants.
- **Decision points:** The policy effectively prioritized pattern matching before executing more resource-intensive monitoring tools.
- **Reward structure impact:** The policy balanced rapid response (secondary rewards) against false positive risks (penalties), achieving containment before significant data loss occurred (primary reward).

This case study demonstrates how the discrete action MDP framework can be applied to cybersecurity domains where rapid, accurate decision-making is critical. The policy's ability to combine pattern recognition with contextual analysis enables more adaptive responses than traditional signature-based detection methods.

7 Limitations and Future Work

Our theoretical framework has several limitations that present opportunities for future research:

- **Action Space Constraints:** Our discrete action space may not capture all effective reasoning strategies, potentially limiting performance on problems requiring novel approaches.
- **Embedding Limitations:** As discussed in our theoretical framework, embeddings necessarily lose information when compressing the full context into a fixed-dimensional vector. This information loss becomes particularly problematic for complex reasoning chains where nuanced details may be critical for optimal decision-making.
- **Reward Sparsity:** Terminal rewards provide limited guidance for credit assignment across long reasoning chains.
- **Computational Efficiency:** The iterative nature of our approach may require more computational resources than end-to-end methods for certain problems.

Future work will explore hierarchical action representations, improved embedding techniques specifically designed for reasoning contexts, and more sophisticated reward shaping approaches that preserve the advantages of our explicit framework while addressing these limitations.

8 Discussion

Our framework presents an interpretable, modular approach to prompting by explicitly modeling reasoning as a sequence of discrete actions. While experimental results are pending, we can discuss several important theoretical implications and potential applications of our approach.

8.1 Interpretability vs. Performance Trade-offs

The discrete action space provides transparency that latent chain-of-thought methods like o1/o3 and DeepSeek-R1 lack. This transparency serves multiple purposes:

- **Debugging and Analysis:** The explicit action sequence makes it possible to pinpoint exactly where reasoning goes wrong, potentially allowing for targeted improvements.
- **Trust and Verification:** In high-stakes domains such as medical diagnosis or financial analysis, the ability to audit reasoning steps could be critical for establishing trust.
- **Educational Applications:** The explicit reasoning steps could be leveraged for explaining complex problem-solving techniques to students, with the model demonstrating metacognitive strategies.

We hypothesize that these benefits may come with performance costs in some domains, and our experiments will help quantify this interpretability-performance trade-off.

8.2 Compositional Generalization

A key theoretical advantage of our discrete action framework is its potential for compositional generalization. By learning when to apply different prompting templates and tools based on state embeddings, our policy may develop the ability to tackle novel problems by composing familiar actions in new ways. This contrasts with end-to-end methods that might struggle with systematic generalization to problems requiring unfamiliar combinations of reasoning steps.

The degree to which this compositional advantage materializes in practice remains an empirical question that our experimental protocol is designed to evaluate.

8.3 Integration with External Systems

Unlike purely latent approaches, our framework naturally accommodates integration with external tools and resources:

- **Retrieval-Augmented Generation:** Our context retrieval action provides a principled way to incorporate vector database lookups when the policy deems them beneficial.
- **Tool Use:** The tool execution action enables systematic integration of computational resources, potentially extending the model's capabilities beyond its training distribution.
- **Human-in-the-Loop Collaboration:** The discrete action space could facilitate human intervention at specific decision points, enabling more effective human-AI collaboration.

These integration capabilities may be particularly valuable in domains requiring specialized knowledge or computational verification.

8.4 Implications for LLM Training and Architecture

Our approach suggests that adding explicit reasoning modules to LLMs—rather than relying solely on latent mechanisms—may be beneficial for certain applications. This raises interesting questions about hybrid architectures that combine:

- Latent, end-to-end components for fluency and general knowledge
- Explicit, discrete components for complex reasoning and tool integration

Such hybrid approaches might offer a middle path between the "Bitter Lesson" emphasis on general methods and the need for interpretable, reliable systems in high-stakes domains.

8.5 Broader Implications

The broader challenge our work addresses is how to maintain human oversight and understanding of increasingly powerful reasoning systems. As LLMs grow in capability, ensuring that their reasoning processes remain interpretable becomes more important yet more difficult. Our MDP framework represents one approach to this challenge, providing a structured way to guide and interpret LLM reasoning without fully constraining its capabilities.

As our experimental results become available, we will be able to more concretely evaluate whether this approach effectively balances performance with interpretability across different domains and tasks.

9 Conclusion

We have introduced a discrete action MDP framework for goal-directed prompting in LLMs that balances interpretability with performance. Our approach represents the state as an embedding of the current context and defines a discrete action space of prompting strategies, employing Q-learning to optimize a policy that guides the model toward high-utility outputs. This framework offers several key contributions:

First, it provides a theoretically grounded approach to structuring LLM reasoning through explicit action sequences. Second, it demonstrates how traditional reinforcement learning techniques can be adapted to the unique challenges of the prompting domain. Third, it offers a complementary approach to latent chain-of-thought methods that prioritizes transparency and modularity.

While acknowledging the "Bitter Lesson" that general, data-driven methods often outperform handcrafted approaches, our work suggests that hybrid systems incorporating elements of both may be valuable for applications requiring interpretability or safety guarantees. As LLMs continue to advance, frameworks that allow for human oversight without sacrificing capability will become increasingly important. Our experimental protocol will provide empirical validation of these theoretical contributions once experiments are completed.

9.1 Applications in Critical Domains

Beyond general problem-solving, our framework shows particular promise in domains requiring both rapid adaptation and interpretable decision processes:

- **Cybersecurity:** In ransomware defense, our approach offers several advantages over conventional methods:
  - **Adaptive Detection:** Learning to identify novel attack patterns based on behavioral similarities rather than exact signatures
  - **Explainable Responses:** Generating clear justifications for security actions taken, critical for incident response teams
  - **Balanced Decision-Making:** Optimizing the trade-off between false positives (which cause business disruption) and missed detections (which lead to data loss)
  - **Continuous Improvement:** Refining the policy through ongoing exposure to evolving threat patterns

The reward structure for this domain emphasizes preventing data encryption/exfiltration while minimizing system disruption, creating a policy that learns to intervene at the optimal point in a developing attack scenario.

References

Arora, S., Khandeparkar, H., Khodak, M., Plevrakis, O., & Saunshi, N. (2019). A theoretical analysis of contrastive unsupervised representation learning. In *International Conference on Machine Learning* (pp. 5628–5637).

Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8), 1798–1828.

Ellis, K., Wong, C., Nye, M., Sable-Meyer, M., Cary, L., Morales, L., Hewitt, L., Solar-Lezama, A., & Tenenbaum, J. B. (2021). DreamCoder: Growing generalizable, interpretable knowledge with wake-sleep Bayesian program learning. *Proceedings of the National Academy of Sciences*, 118(9), e2015759118.

Fan, J., Wang, Z., Xie, Y., & Yang, Z. (2020). A theoretical analysis of deep Q-learning. In *Learning for Dynamics and Control* (pp. 486–489).

Geman, S., Bienenstock, E., & Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1), 1–58.

Lake, B. M., Ullman, T. D., Tenenbaum, J. B., & Gershman, S. J. (2017). Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40, e253.

Li, L., Walsh, T. J., & Littman, M. L. (2006). Towards a unified theory of state abstraction for MDPs. In *International Symposium on Artificial Intelligence and Mathematics*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems* (pp. 3111–3119).

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing Atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540), 529–533.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Watkins, C. J., & Dayan, P. (1992). Q-learning. *Machine Learning*, 8(3-4), 279–292.