When Not to Trust the Oracle: Detecting Unsafe Advice in LLM-guided Reinforcement Learning

1st Anandkumar NS
PSG College of Technology
Coimbatore, India
22z209@psgtech.ac.in

2nd Kishoreadhith V
PSG College of Technology
Coimbatore, India
22z232@psgtech.ac.in

3rd Dhakkshin S R
PSG College of Technology
Coimbatore, India
22z215@psgtech.ac.in

4th M Raj Ragavender
PSG College of Technology
Coimbatore, India
22z233@psgtech.ac.in

5th Rithvik K
PSG College of Technology
Coimbatore, India
22z253@psgtech.ac.in

Large language models (LLMs) have recently been explored as high-level planners for reinforcement learning (RL) agents in complex environments. This approach encounters issues when we take into consideration the fact that LLM-generated advice can be unreliable, particularly under corrupted or ambiguous conditions and state descriptions. This raises the question of safety concerns for the decision-making and other processes powered by the LLM generated advice. We aim to address this gap by proposing a lightweight safety framework that detects and tries to mitigate unsafe LLM guidance. We adopt the NetHack Learning Environment (NLE) as our testbed due to its rich combinatorial state space, partial observability, and long-horizon decision dependencies, which make it a challenging benchmark for evaluating robustness of LLM-guided agents.

The central inquiry guiding this review is: **”How can we make the LLM-guided Reinforcement Learning Agents interpretable and robust to misinformation, particularly in complex sequential descion-making environments?”**. This is the question that will be explored by this work and we will try to provide valid solutions to the issues that we enounter?

This work contributes a practical methodology for trust calibration in LLM-guided RL and opens new directions for adversarial robustness in language-driven agents. Most research on LLM-guided reinforcement learning uses the LLM as a planner or “common sense” advisor, but leaves critical gaps. Current agents operate as black boxes: they map states to actions without understanding why strategies work, which limits generalization.

They also tend to blindly trust the LLM’s advice, even when the state description is corrupted or incomplete, leading to brittle performance. In addition, existing systems provide little interpretability—offering no clear way to debug or understand decisions—and they are rarely stress-tested under adversarial or misleading inputs. Together, these gaps raise serious concerns about safety, reliability, and trustworthiness in real-world applications.

I THEORETICAL BACKGROUND

A Introduction to LLM-guided Reinforcement Learning

Reinforcement Learning (RL), a paradigm rooted in trial-and-error learning to optimize sequential policies, has traditionally been applied to environments with complex state-action spaces like robotics, games, and autonomous control. The intersection of these fields—leveraging the expressive power and knowledge embedded in LLMs to enhance RL agents—presents an enticing research frontier where LLMs can provide rich semantic guidance to improve decision-making efficiency and effectiveness.

The motivation for integrating LLM with RL stems primarily from the challenges inherent in RL related to sample inefficiency, exploration complexity, and interpretability. LLMs, pretrained on vast corpora of text, embed rich world knowledge, hierarchical structure, and planning insights, which can be harnessed to guide RL agents strategically. Such integration promises not only to boost learning efficiency through informed exploration but also to enable more interpretable agent policies that can be understood and audited by humans. However, realizing this integration involves addressing critical challenges such as how to align LLM knowledge with environmental dynamics (grounding), how to maintain robustness when LLM knowledge is imperfect or misleading, and how to embed causal reasoning into decision-making processes. These challenges are intensified in complex sequential environments where uncertainty, partial observability, and long-horizon planning requirements prevail.

B Research Objective and Scope

This literature survey focuses on the intersection of large language models with reinforcement learning, particularly with regard to strategic guidance, causal reasoning, and robustness in complex sequential decision-making environments. The scope emphasizes high-stakes and intricate game-like environments, such as NetHack and Minecraft, which have become standard benchmarks due to their rich state-action representations, need for long-term planning, and partial observability. By concentrating on these environments, the review aims to uncover how language-guided RL architectures shape learning dynamics, facilitate interpretable strategies, and address robustness under misinformation or adversarial perturbations.

The survey uses a multi-tiered keyword strategy to ensure inclusivity and depth in identifying relevant contributions. Primary keywords focus on direct LLM-RL integration, strategic and causal aspects, and robustness, while secondary and tertiary terms extend coverage to hierarchical and symbolic RL, multimodal learning, and procedural content generation. This stratified search allows the survey to capture core methods, application domains, and emerging subfields. Additionally, the delimitation to strategic sequential decision-making excludes predominantly supervised or offline language tasks, maintaining alignment with the challenges of reinforcement-driven interactive learning systems.

Zeng et al. (2023) provide a comprehensive survey of how LLMs are leveraged across robotics—spanning perception, planning, control, and interactive reasoning—while also highlighting emerging challenges such as resource constraints and safety in multimodal and embodied settings (Zeng et al., 2023) [1]. This sets a broad foundation for understanding where current architectural approaches (like RL-GPT or Voyager) fit into the larger landscape.

II CURRENT METHODS OF LLM AND RL INTEGRATION

A Architectural Approaches for Integration

The integration of LLMs with RL agents has been explored primarily through several architectural paradigms aimed at balancing the benefits of each component. One predominant approach involves hierarchical frameworks where LLMs operate at a higher level of abstraction, providing strategic advice, policy shaping, or sub-goal generation, while RL agents implement low-level control and action execution. For instance, the hierarchical decomposition in RL-GPT introduces a two-level agent system comprising a slow agent responsible for high-level action planning (often encoded as code or symbolic instructions) and a fast agent that manages task-specific refinements and fast execution cycles [2]. Such designs enable specialization of each module and improve sample efficiency due to better modularity.

Another approach is advisory or hybrid models where LLMs provide consultative signals to RL agents in the form of natural language instructions, action suggestions, or policy priors. The GLAM approach exemplifies this strategy by progressively updating an LLM-based policy online during interactions with the environment, harnessing reinforcement learning to ground the abstract knowledge encoded in the LLM into concrete actions for spatial and navigation tasks [3]. Hybrid models benefit from the complementary nature of LLMs' abstract reasoning and RL's environment-specific learning but require careful mechanisms to manage conflicts between language advice and sensory feedback.

Ahn et al. (2022) introduced SayCan, a system where an LLM proposes high-level goals (Say), and a learned affordance/value model ("Can") ensures the feasibility of these suggestions, effectively bridging symbolic planning with actionable robot control (Ahn et al., 2022) [4]. DeepMind's Code-as-Policies reformulates robot behavior as executable code generated by LLMs: the model composes API-driven policies, supports hierarchical code generation (e.g., loop and conditional structures), and generalizes across tasks—from pick-and-place to trajectory control (Liang et al., 2022) [5].

Notable integrated systems such as Voyager (a Minecraft agent) utilize LLMs for continual planning and task decomposition while RL handles exploration and low-level primitives, demonstrating architectural effectiveness in real-world-like embodied domains. It demonstrates continual skill acquisition: GPT-4 generates runnable code to build, explore, and self-improve over time—no fine-tuning required (Wang et al., 2023) [6]. These architectures typically employ multi-modal inputs, hierarchical control, and reinforcement signals to enable scalable learning.

Overall, architectural methods hinge on designing clear interfaces for translating between symbolic or language representations and actionable policies, balancing modularity and end-to-end optimization, and partitioning learning duties between LLMs and RL agents for scalable and interpretable decision-making.

B Information flow and Communication Channels

Effectively integrating LLMs with RL hinges on the design of information flow and communication protocols that allow the transfer and transformation of guidance signals. Commonly, state information from the environment, which may be raw or structured, is first converted into a language-based or symbolic representation that the LLM can interpret. This state-to-language transformation serves as a shared abstraction layer, enabling the extensive pretrained knowledge of LLMs to be leveraged for strategy generation.

The NetHack Learning Environment (NLE) language wrapper illustrates a practical implementation of this paradigm, translating complex game states into natural language descriptions used as input for LLM agents [3]. This approach reduces the perceptual complexity the LLM faces and grounds its reasoning in semantically rich but interpretable representations. Similarly, the Thinker framework integrates an LLM with an external module managing cognitive tasks in a two-system reasoning hierarchy, where the LLM executes intuitive language processing and the external Thinker module handles complex logical reasoning, facilitated through a defined communication protocol [7]. This separation ensures both interpretability and rigorous analysis of strategic inputs.

The communication channels often rely on protocols that define the frequency, format, and scope of information transfer. For example, in interactive text-based environments, LLMs receive episodic textual inputs and output textual commands or plans, whereas in embodied domains, intermediate symbolic actions or code snippets serve as communication tokens. The language-to-action pipeline is critical, demanding robust parsers and executors to convert LLM-generated plans into executable RL policies.

Huang et al. (2022) propose Inner Monologue, an approach where LLMs incorporate environment feedback—such as success detection and scene description—into their generative planning loop, enabling more effective replanning and error recovery in both simulated and real-world tabletop and mobile manipulation tasks (Huang et al., 2022) [8].

C Training Paradigms and Action Space Handling

Training LLM-assisted RL systems introduces methodological choices regarding how both components adapt to optimize joint performance. Training paradigms range from fine-tuning and instruction-based learning of LLMs to reinforcement learning with human feedback (RLHF) that aligns language models with task-specific objectives [9]. Modular training, where the LLM and RL agent are trained separately but coordinated through an interface, complements end-to-end approaches where joint optimization is performed.

Few-shot and zero-shot prompt tuning enable flexible deployment of LLMs without extensive retraining, facilitating incorporation of prior knowledge in RL tasks [9]. Online RL techniques permit continuous adaptation as the agent interacts with the environment, progressively grounding LLM knowledge into task-relevant policies [3].

Regarding action spaces, systems must handle discrete (e.g., game moves), continuous (e.g., robot joint controls), or hybrid actions, sometimes expressed via natural language commands. Hierarchical models such as IHAC (Imitation Hierarchical Actor-Critic) utilize LLMs to provide high-level commands or subgoals that an RL actor critic executes in low-dimensional or continuous control spaces. RL-GPT's two-agent architecture differentiates between slow, coding-centric planning and fast execution of fine-tuned low-level actions, achieving state-of-the-art performance in embodied tasks such as Minecraft [2].

These multimodal action representations and multi-agent decomposition strategies are critical to bridging the symbolic reasoning capabilities of LLMs and the reactive, sensorimotor competencies of RL.

D Benchmarks and Evaluation Environments

A critical aspect of integrating LLMs with reinforcement learning is the choice of benchmark environments, which strongly influences the generality, interpretability, and robustness of the resulting systems. Several environments have emerged as de-facto standards for evaluating language-guided RL.

The NetHack Learning Environment (NLE) has become a prominent testbed due to its combinatorial complexity, partial observability, and long-horizon decision dependencies. It provides a rich setting where language wrappers translate symbolic states into natural language descriptions, making it particularly well suited for evaluating robustness and interpretability of LLM-based guidance [10].

In open-ended Minecraft-based environments, agents such as Voyager [6] demonstrate continual skill acquisition, task decomposition, and code-driven exploration. Minecraft's open world allows for the evaluation of long-horizon planning, exploration strategies, and the ability of agents to generalize to unseen tasks.

Smaller-scale yet strategically important environments include BabyAI [11], a grid-world platform focused on language-conditioned policy learning. BabyAI is valuable for testing sample efficiency, hierarchical planning, and generalization to novel instructions. Similarly, AlfWorld [12] combines natural language instruction following with embodied reasoning in simulated household tasks, bridging text-only tasks with interactive, embodied RL.

Finally, benchmarks such as CALVIN [13] extend these principles to robotic manipulation by offering language-conditioned long-horizon tasks. CALVIN highlights the gap between simulated environments and physical robot control, providing a pathway for testing how LLM-guided policies transfer to real-world applications.

Together, these environments form a diverse ecosystem: from symbolic games (NetHack) to open-ended sandboxes (Minecraft) and grounded manipulation tasks (CALVIN). The use of multiple benchmarks is critical to avoid overfitting to a single environment and to stress-test LLM-RL integration across different modalities, complexities, and safety constraints.

E Comparison of Representative Approaches

While the preceding subsections outlined architectural paradigms, communication channels, training paradigms, and benchmarks, it is useful to provide a structured comparison of representative systems. Table 1 synthesizes key approaches across domains, highlighting the role of the LLM, strengths, and limitations. This comparative view enables a holistic understanding of how different design choices manifest in practice and how they relate to the technical gaps discussed in Section III.

The table underscores recurring trade-offs. For example, robotics-focused methods such as SayCan and Code-as-Policies emphasize grounding and modularity but often lack robustness to distributional shifts. Game-based systems like Voyager and NLE excel at long-horizon reasoning but reveal blind trust and exploration challenges. These patterns reinforce the need for causal reasoning and robustness mechanisms, as elaborated in Section III.

III TECHNICAL GAPS

We have identified a few technical gaps in existing research on LLM-guided RL systems. It uses the LLMs as a ”common-sense”source or a simple planner. This leaves several fundamental gaps:

1. **The Black Box Strategy Gap** The problem in this case is that current agents learn what to do but not why a strategy works. They map states to actions based on rewards, but lack a deeper, transferable understanding of the game's underlying mechanics. The follow a black box strategy. This leads to poor generalization, when faced with a situation that is even slightly new, the agent can fail and it doesn't try to understand the core principles of the strategy. The causation of the strategy is not captured as well as could be.

2. **The Blind Trust Gap** The RL agent almost always assumes the LLM's guidance is correct. It blindly trusts the advice it's given. If the state description sent to the LLM is slightly wrong, incomplete, or ”corrupted,”the LLM can give catastrophically bad advice. Current systems have no mechanism to ”sanity check”or question the guidance, making them brittle and unreliable.

3. **The Interpretability Gap** It is nearly impossible to ask a standard agent why it made a specific decision. You can't debug its reasoning or understand its logic, which makes it hard to trust, especially in complex situations. Without interpretability, we cannot understand failure modes, improve the agent's reasoning, or be confident in its decisions.

4. **The Robustness Gap** No existing work systematically tests how these agents perform under adversarial conditions—that is, when the information they receive from the LLM used for guidance is deliberately made misleading. An AI that works perfectly with clean data but collapses at the first sign of unexpected or incorrect information is not useful in realistic, unpredictable environments.

The “Concrete Problems in AI Safety” framework (Amodei et al., 2016) identifies five practical risks—including reward hacking, unsafe exploration, side effects, scalable oversight, and robustness to distributional shift—that are directly relevant to the Blind Trust and Robustness gaps in our context (Amodei et al., 2016) [14]. More recently, Raji & Dobbe (2023) revisit these safety issues from a socio-technical vantage, underscoring that real-world deployments often fail due to stakeholder misalignment and systemic oversight—not merely algorithmic flaws (Raji & Dobbe, 2023) [15].

IV CONNECTIONS TO AI SAFETY AND INTERPRETABILITY

The technical gaps identified above—black-box reasoning, blind trust, lack of interpretability, and fragility under adversarial perturbations—resonate strongly with the broader literature on AI safety and explainability. In particular, Amodei et al. [14] highlight reward misspecification, unsafe exploration, and robustness to distributional shift as central safety problems for reinforcement

Tabela 1: Comparison of representative LLM-guided RL approaches across domains, roles, and limitations.

Paper / System	**Domain**	**LLM Role**	**Strengths**	**Limitations**
RL-GPT [2]	Minecraft / Embodied tasks	Hierarchical planner (slow + fast agents)	Improves modularity, sample efficiency via code-as-policy	Still black-box reasoning, limited causal modeling
SayCan [4]	Robotics	High-level planner with affordance grounding	Bridges language and robot affordances; safer than pure LLM	Dependent on quality of affordance model; limited generalization
Code-as-Policies [5]	Robotics	Policy generation via code snippets	Modular, compositional policies; interpretable via code	Requires curated APIs; limited robustness to unseen tasks
Voyager [6]	Minecraft (open world)	Continual planner and code generator	Lifelong learning; self-improving skill library	High compute cost; prompt dependence; lacks causal reasoning
Thinker [7]	Text-based games (Werewolf)	Two-system reasoning (LLM + logic module)	Separation of intuitive vs logical reasoning; interpretable	Limited to symbolic domains; not tested in embodied settings
NLE [10]	NetHack (symbolic game)	Planner/advisor via natural language wrappers	Rich combinatorial state space; stress-tests robustness	Extremely sparse rewards; hard exploration

learning, many of which manifest directly in LLM-guided RL systems. More recent perspectives emphasize that failures often arise not only from algorithms themselves but also from misalignment with human oversight and socio-technical contexts [15].

Interpretability has long been studied in the explainable AI (XAI) community through methods such as feature attribution (e.g., SHAP, LIME), saliency mapping, and surrogate models. While these tools have primarily targeted supervised settings, recent work is extending causal explainability to sequential decision-making and LLM-guided systems [16]. Such approaches provide a foundation for diagnosing when an agent's reasoning diverges from expected causal structures, thereby addressing both the interpretability and robustness gaps.

In reinforcement learning specifically, a parallel strand of research has examined safe exploration, off-policy evaluation, and adversarial training as strategies to constrain agent behavior under uncertainty [17]. These paradigms are increasingly relevant as LLMs are incorporated into RL pipelines, since language models can amplify both the strengths (e.g., informed exploration) and the risks (e.g., hallucinated unsafe strategies) of reinforcement-driven learning. Bridging these fields suggests that LLM-guided RL must be developed not in isolation but in conversation with safety and interpretability research, ensuring trustworthy deployment in complex environments.

V CAUSAL REASONING IN REINFORCEMENT LEARNING WITH LLMS

A Learning Causal Models from RL Experience

Causal reasoning in reinforcement learning involves modeling how actions causally affect states and rewards, beyond mere correlations. Foundational frameworks established by Pearl and Bareinboim introduce structural causal models and do-calculus as mathematical underpinnings for this reasoning [16]. However, integration of causal inference with LLM-guided RL remains nascent and sparsely explored. Most current LLM-RL architectures rely primarily on associative learning and lack explicit causal modeling.

Some recent efforts incorporate latent causal structure discovery by leveraging LLMs' instruction-following and counterfactual reasoning capabilities to identify latent or unobserved features impacting decisions [16]. These methods allow the extraction of causal patterns from sequential interaction data that can potentially inform policy adjustments and strategic planning.

Despite the potential, practical methods for real-time causal model learning from RL trajectories are underdeveloped, highlighting an important research frontier for embedding causal frameworks within scalable LLM-RL systems.

B Role of Causal Models in Decision Making

Causal models provide several advantages in decision-making, including improved policy robustness, interpretability, and failure resilience. By understanding cause-effect relations, agents can generalize better across environmental changes and avoid spurious correlations leading to suboptimal or unsafe actions. Unlike purely correlation-based approaches, causal strategies facilitate explanation and justification of decisions, which is critical in safety-sensitive applications.

Counterfactual reasoning—asking "what-if" questions to infer alternative outcomes—is a powerful tool enabled by causal models. When combined with LLMs' natural language reasoning, this ability can enhance exploration strategies and support policy repair after failures. Causal models can also guide RL agents to recover from adversarial perturbations or misinformation by reasoning about the underlying causal factors affected.

Initial demonstrations of integrating causal explainability into NLP and RL using LLMs show promising results in generating counterfactual explanations for black-box classifiers, indicating applicability to RL policy interpretability [16, 9]. However, extending this to comprehensive causal strategy learning in interactive environments remains an active challenge.

C Implementation of Counterfactual Reasoning

Counterfactual reasoning through LLMs typically follows a three-step pipeline: identify latent features in input data, associate these with observable features, and generate counterfactual explanations or alternative policy recommendations based on identified causal dependencies [16]. Adapting this pipeline for RL involves coupling counterfactual generation with policy evaluation and update loops, allowing agents to query how changes in decisions would affect future rewards.

Challenges include computational overhead, ensuring interpretability of counterfactuals in high-dimensional state-action spaces, and integrating counterfactual signals smoothly into RL optimization procedures. Nonetheless, the potential for enhancing exploration, mitigating hallucinations, and improving robustness makes counterfactual causal reasoning a promising avenue for future LLM-guided RL frameworks.

Referências

[1] ZENG, F. et al. *Large Language Models for Robotics: A Survey*. 2023. Disponível em: ⟨https://arxiv.org/abs/2311.07226⟩.

[2] LIU, S. et al. *RL-GPT: Integrating Reinforcement Learning and Code-as-policy*. 2024. Disponível em: ⟨https://arxiv.org/abs/2402.19299⟩.

[3] CARTA, T. et al. *Grounding Large Language Models in Interactive Environments with Online Reinforcement Learning*. 2024. Disponível em: ⟨https://arxiv.org/abs/2302.02662⟩.

[4] AHN, M. et al. *Do As I Can, Not As I Say: Grounding Language in Robotic Affordances*. 2022. Disponível em: ⟨https://arxiv.org/abs/2204.01691⟩.

[5] LIANG, J. et al. *Code as Policies: Language Model Programs for Embodied Control*. 2023. Disponível em: ⟨https://arxiv.org/abs/2209.07753⟩.

[6] WANG, G. et al. *Voyager: An Open-Ended Embodied Agent with Large Language Models*. 2023. Disponível em: ⟨https://arxiv.org/abs/2305.16291⟩.

[7] WU, S. et al. *Enhance Reasoning for Large Language Models in the Game Werewolf*. 2024. Disponível em: ⟨https://arxiv.org/abs/2402.02330⟩.

[8] HUANG, W. et al. Inner monologue: Embodied reasoning through planning with language models. In: LIU, K.; KULIC, D.; ICHNOWSKI, J. (Ed.). *Proceedings of The 6th Conference on Robot Learning*. PMLR, 2023. (Proceedings of Machine Learning Research, v. 205), p. 1769–1782. Disponível em: ⟨https://proceedings.mlr.press/v205/huang23c.html⟩.

[9] SAHOO, S. S. et al. Large language models for biomedicine: foundations, opportunities, challenges, and best practices. *Journal of the American Medical Informatics Association*, v. 31, n. 9, p. 2114–2124, 04 2024. ISSN 1527-974X. Disponível em: ⟨https://doi.org/10.1093/jamia/ocae074⟩.

[10] KüTTLER, H. et al. *The NetHack Learning Environment*. 2020. Disponível em: ⟨https://arxiv.org/abs/2006.13760⟩.

[11] CHEVALIER-BOISVERT, M. et al. *BabyAI: A Platform to Study the Sample Efficiency of Grounded Language Learning*. 2019. Disponível em: ⟨https://arxiv.org/abs/1810.08272⟩.

[12] SHRIDHAR, M. et al. *ALFWorld: Aligning Text and Embodied Environments for Interactive Learning*. 2021. Disponível em: ⟨https://arxiv.org/abs/2010.03768⟩.

[13] MEES, O. et al. *CALVIN: A Benchmark for Language-Conditioned Policy Learning for Long-Horizon Robot Manipulation Tasks*. 2022. Disponível em: ⟨https://arxiv.org/abs/2112.03227⟩.

[14] AMODEI, D. et al. *Concrete Problems in AI Safety*. 2016. Disponível em: ⟨https://arxiv.org/abs/1606.06565⟩.

[15] RAJI, I. D.; DOBBE, R. *Concrete Problems in AI Safety, Revisited*. 2023. Disponível em: ⟨https://arxiv.org/abs/2401.10899⟩.

[16] BHATTACHARJEE, A. et al. *Towards LLM-guided Causal Explainability for Black-box Text Classifiers*. 2023. Disponível em: ⟨https://api.semanticscholar.org/CorpusID:262459118⟩.

[17] GARCÃA, J.; FERNÃ¡NDEZ, F. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, v. 16, n. 42, p. 1437–1480, 2015. Disponível em: ⟨http://jmlr.org/papers/v16/garcia15a.html⟩.