AI Agent Safety is a Reinforcement Learning Problem

Reginald McLean [1] **Tabitha Edith Lee** [2 3] **Montaser Mohammedalamen** [1] **Kevin Roice** [1] **Glen Berseth** [2 3 4]
Patrick Pilarski [1 5 4] **Marlos C. Machado** [1 5 4] **Alyssa Lefaivre Škopac** [1] **Benjamin Rosman** [6]

Abstract

With the rapid advancement and deployment of Agentic Artificial Intelligence (AI), our scientific understanding of capabilities and limitations has not kept pace, leading to cases where AI agents cause harm. We argue that many of these safety limitations are not novel problems. Instead, the safety challenges currently facing AI agents can be seen as instances of problems the reinforcement learning (RL) community has studied rigorously for decades. The core of this argument concerns the problem formulation of AI agents. AI agents are designed to solve sequential decision-making problems: problems with long-term objectives in which actions have delayed consequences. To model these types of problem, the problem is set up such that the agent receives observations, feedback on its progress, and then takes actions. This is precisely the formulation of the RL problem. In this paper, we formalize the problem equivalence, which we then leverage to argue that **AI Agent safety is a reinforcement learning problem: the failure modes currently observed in deployed AI agents are structural instances of problems RL has formalized for decades, and the RL safety literature provides principled tools to diagnose and address them.** We conclude with a call for deliberate collaboration between the RL and AI agent research communities: AI agent researchers gain access to principled frameworks, while RL researchers gain a class of real-world problems that could expose fundamental gaps in current RL benchmarks and theory.

[1]Alberta Machine Intelligence Institute [2]Université de Montréal [3]Mila - Quebec Artificial Intelligence Institute [4]Canada CIFAR AI Chair [5]University of Alberta [6]University of the Witwatersrand. Correspondence to: Reginald McLean <reginald.mclean@amii.ca>.

Published at the Second Workshop on Agents in the Wild: Safety, Security, and Beyond (AIWILD) at ICML 2026. Copyright 2026 by the author(s).

1. Introduction

Large language models (LLMs) have evolved from writing prose in the style of historical authors to executing skills including creating documents, writing and testing code, and evaluating logical arguments. This shift marks a fundamental transition where LLMs are no longer merely predicting the next token in a sequence: their proficiency has increased to the stage where LLMs can now act as the decision-making backbone of an agent designed to execute specific tasks, including those in the real world by way of direct connections to physical systems like robots and industrial machines. LLMs have been integrated into several applications of autonomous agents (Lu et al., 2026; Boiko et al., 2023; Wang et al., 2025c), referred to hereafter as AI agents. These AI agents are capable of receiving instructions, selecting actions to accomplish the instructions, and receiving feedback about their progress towards the goal. Although the development of these AI agents has largely emerged from the advancements of LLMs, the core problem that AI agents solve is about achieving a goal by interacting with an environment, such as a web browser, a code development environment, or a robotic manipulator.

As the capabilities of AI agents grow and are deployed in an increasing number of domains, the failure modes of these AI agents become more pronounced. In the last year, several high-profile safety issues have plagued AI agents, such as deleting an entire company's database (Nolan, 2025), autonomously mining cryptocurrencies (Wang et al., 2026), causing large data leaks of user information (Down, 2026), and deleting all emails from a user's inbox (Martindale, 2026). These failures speak to a larger set of issues in AI agents, such as a lack of generalization, where agents fail to adapt knowledge to novel scenarios; distributional shift, where environmental dynamics fall outside the agent's training data; and reward hacking, where an agent optimizes for a literal objective in a way that violates user intent. These are not merely linguistic errors. As we formalize in Section 3, these are systemic failures of agents attempting to solve sequential decision-making problems — the same class of problems that RL has spent decades studying. The structural parallel between AI agent failures and well-characterized RL failure modes is the central argument of this paper.

The core problem of sequential decision-making is also the focus of reinforcement learning (RL). RL has historically focused on problems where the RL agent must learn completely from interaction with an environment, with some focus on learning from pre-collected datasets (Levine et al., 2020), or previous versions of RL agents (Agarwal et al., 2022). RL has also been used in the post-training stage of LLM training via human feedback (RLHF) (Christiano et al., 2017) or verifiable rewards (RLVR) (Shao et al., 2024; Guo et al., 2025). Beyond LLM training, RL offers a principled language and a mature experimental framework to diagnose and mitigate the safety issues that arise with autonomous agents. This suggests that challenges faced by AI agents are not unique to AI agents, but in fact, are the same issues that the RL community has spent decades formalizing. This motivates our position that **AI agent safety is a reinforcement learning problem: the failure modes currently observed in deployed AI agents are structural instances of problems RL has formalized for decades, and the RL safety literature provides principled tools to diagnose and address them.** If AI agents and RL agents are solving the same problem of sequential decision-making, then the RL safety literature for constraining agent behavior (Altman, 1999), diagnosing reward misspecification (Muslimani et al., 2025), and ensuring safe exploration (Wachi et al., 2023) are applicable to AI agents.

2. Background Information

AI Agents Since the introduction of ChatGPT (OpenAI, 2022) and many subsequent advances in LLMs, there has been an influx of interest in AI agents, which we define as an agent that leverages a LLM as their decision making backbone, and Agentic AI. However, these terms are not entirely new in the field of AI. One of the most widely cited definitions of an agent in AI comes from Russell and Norvig (1995). Across editions of their textbook, Russell and Norvig (1995) describe an agent as something that can "operate autonomously, perceive their environment, persist over a prolonged time period, adapt to change and create and pursue goals." Similarly, Wooldridge and Jennings (1995) created a more formal definition of an agent. While initially developed in the 1990s, the core properties of autonomy, reactivity, and goal-directed behavior apply directly to contemporary AI agents. We include additional information on agents in Appendix A.1.

Following Sapkota et al. (2026), we distinguish between individual AI agents and multi-agent systems, which Sapkota et al. (2026) term Agentic AI. An individual AI agent operates autonomously within a defined scope, interacts dynamically with inputs and outputs, and produces actions in real-time environments. These agents are generally built for specific, well-defined tasks (Krishnan, 2025; Padigela et al., 2025) and interact in real-time with dynamic inputs such as user feedback or API calls (Deng et al., 2025). Agentic AI refers to coordinated systems of multiple AI agents with LLM backbones collaborating to achieve complex goals, potentially governed by a meta-agent that orchestrates information sharing and task coordination (Sapkota et al., 2026). In this paper, we will focus on the individual AI agent that is built on top of a LLM as the core subject of our analysis. The argument that we will develop applies directly to this level, but also scales naturally to Agentic AI systems as they are composed of individual AI agents. By creating Agentic AI systems, emergent behaviors can arise that no single agent was designed to produce, introducing safety considerations beyond those of individual agents. This makes the first step of establishing a principled framework for discussing AI agent safety even more necessary, as the deployment of Agentic AI systems compounds the safety issues of multiple agents.

AI Agent Successes In the span of a few years, AI agents have moved from research curiosities to showing human-like capabilities in some of the most consequential work humans perform. AI agents are debugging production codebases (Yang et al., 2024; Wang et al., 2025c), synthesizing novel chemical compounds in physical laboratories (Boiko et al., 2023), diagnosing cancers and drafting treatment recommendations (Ferber et al., 2025), and autonomously conducting scientific experiments from hypothesis to manuscript (Lu et al., 2026). Across these domains, underlying the performance of these systems is a cycle of operation that allows an AI agent to observe its environment, select actions, and iterate based on feedback. This loop of observing, acting, and receiving feedback powers the transition from reactive LLMs to AI agents that are highly autonomous, and whose deployment in high-stakes domains makes understanding and controlling that loop a matter of safety, not merely performance.

AI Agent Failures Despite the remarkable capabilities demonstrated by AI agents in deployment, these systems fail in ways that are difficult to predict and, in high-stakes domains, difficult to recover from. Paglieri et al. (2025) performed an extensive empirical evaluation of AI agents across tasks of varying difficulty, finding that while agents handle simpler tasks reliably, performance degrades sharply as task complexity and horizon length increase. Xu et al. (2025) grounded these findings in a realistic office environment modeled on real software company workflows, finding that no agent completed more than 30% of tasks autonomously. Perhaps most concerning for deployment, Simhi et al. (2026) designed tasks requiring agents to navigate a tension between operational performance and human safety, finding that all evaluated agents struggled to reliably choose the safe option when it came at a cost to task per-

formance. These findings reveal that the challenges facing AI agents are not merely engineering limitations on performance: they are safety concerns. An agent that cannot reliably prioritize human welfare over task completion, that pursues misspecified objectives without correction, or that acts on partial information in irreversible situations, poses risks that scale directly with the autonomy and deployment context described above.

Reinforcement Learning The failure modes of inadequate exploration, misspecified objectives, and partial observability are not unique to AI agents. They are among the central research questions of reinforcement learning, which has spent decades attempting to solve these problems. RL is the study of creating agents that can observe their environment, select an action, observe how the environment changes, and receive a reward for taking that action. With its mathematical roots in the work of Bellman (1957), RL has become the go-to method for sequential decision-making problems, whether the problem is an Atari game (Mnih et al., 2015; Bellemare et al., 2013; Machado et al., 2018), controlling the location of balloons in the stratosphere (Bellemare et al., 2020), or controlling the magnetic actuator coils of a tokamak fusion reactor (Degrave et al., 2022). What distinguishes RL from AI agents is not merely its empirical success, but its formal grounding. The Markov Decision Process (MDP) formalism and its variants provide a shared mathematical language that allows researchers to characterize problems precisely, identify failure modes structurally, and connect empirical observations to theoretical guarantees.

Central to RL is the notion of learning from interaction, as well as the balance between exploration and exploitation, temporal credit assignment (Sutton and Barto, 2018), and partial observability (Kaelbling et al., 1998). In the exploration and exploitation problem, the RL agent is tasked with finding a balance between exploring its environment and exploiting its previously acquired information. This tension has been well-studied in RL through bandit algorithms (Sutton and Barto, 2018), curiosity-driven exploration (Pathak et al., 2017), and the options framework (Sutton et al., 1999). Temporal credit assignment attempts to address how to attribute delayed rewards to the earlier actions that caused them, given that many unrelated actions may have occurred in between. This problem has been tackled through the lens of temporal difference learning (Sutton, 1988; Patterson et al., 2022), and gradient eligibility traces (Sutton et al., 2009; Elelimy et al., 2025a). Partial observability arises when the agent cannot observe the true state of the system it is interacting with, or when the agent cannot tractably approximate the true system state, which is the case in many real-world scenarios (Javed and Sutton, 2024), and which the Partially Observable Markov Decision Process was explicitly developed for (Kaelbling et al., 1998).

These challenges are not peripheral concerns in RL but among its central research questions, each supported by dedicated theoretical frameworks and empirical benchmarks developed over decades. It is precisely because these problems are so well-studied in RL that their appearance as failure modes in AI agents is so significant.

3. Argument Support

In this section, we outline the rationale for our position by showing the parallels between the sequential decision-making literature and the problem formulation of AI agents. First, we define the stochastic control problem where an agent is attempting to control a system that it can observe and act upon. Second, we transition from the stochastic control problem to MDPs. Here, we present an explicit mapping from the AI agent problem setting to the problem settings that MDPs formalize. Next, we propose a mapping of specific Partially Observable Markov Decision Process (POMDP) (Kaelbling et al., 1998) components to the AI agents problem setting, where components of the AI agent are mapped to different problems and solutions in the RL literature. Finally, we discuss the implications of our line of thinking.

3.1. Agent Interaction Framework

Figure 1 presents framework illustrations for both AI and RL agents, where visual similarities between the two frameworks speak to their underlying equivalence. Specifically, Figure 1(a) shows the interaction loop of an AI agent built on a framework of reasoning and acting (ReAct) (Yao et al., 2023), and Figure 1(b) shows the canonical agent-environment interface in RL from Sutton and Barto (2018). Despite originating from entirely separate research traditions, the two diagrams describe a similar underlying process: an agent receives an observation of the world, selects an action, and receives feedback from the environment. This structural equivalence is not coincidental; it reflects the fact that both frameworks are formalizations of the same core problem: sequential decision-making. In the remainder of this section, we make this equivalence precise by showing that, as a representative example, the AI agent problem can be formalized as a POMDP (Kaelbling et al., 1998), the same mathematical object that part of RL has studied for decades.

3.2. Stochastic Control

Many real-world problems involve an agent (e.g., a person or a controller) that observes the evolution of a system and attempts to maintain or direct that system according to some objective. This problem is formally studied under the um-

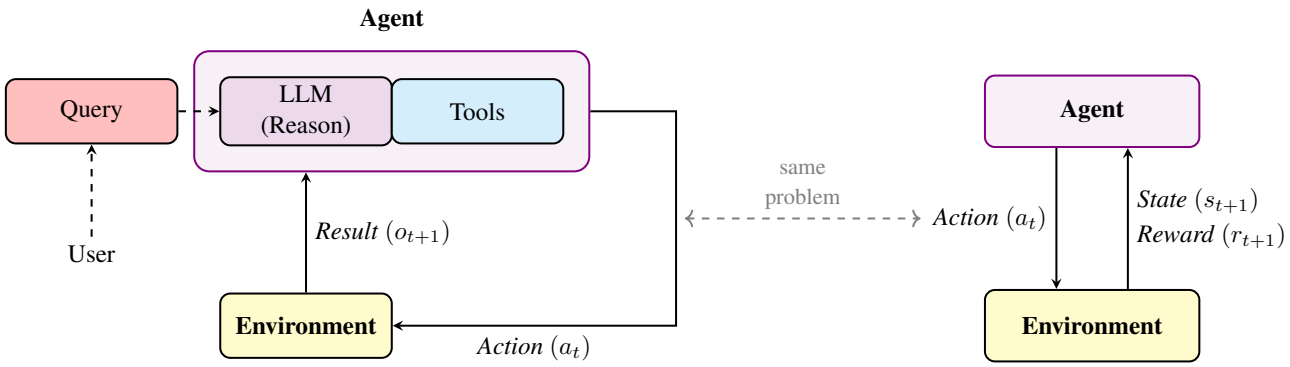

Figure 1. **(left)** ReAct agent interaction loop (Yao et al., 2023). **(right)** The canonical agent-environment interface from Sutton and Barto (2018).

brella of Optimal Control (OC) (Kirk, 1970). In this setting, there is a system with state that evolves over time, an agent that observes some function of that state, a set of actions the agent can select, dynamics that govern how actions influence state evolution, and an objective the agent seeks to optimize over time.

A subfield of OC, Stochastic Control (SC), deals with problems where there is noise in the system or uncertainty over observations. Formally, a SC problem is defined with the tuple $\langle \mathcal{S}, \mathcal{A}, f, \mathcal{O}, \mathcal{J}, \mathcal{Z} \rangle$, where $\mathcal{S}$ is the state space of the system, $\mathcal{A}$ is the action space, and $f : \mathcal{S} \times \mathcal{A} \times \mathcal{W} \to \mathcal{S}$ is the stochastic transition function, with $\mathcal{W}$ representing the stochasticity of the system. $\mathcal{Z}$ is the observation space, the set of observations the agent can perceive, $\mathcal{O} : \mathcal{S} \to \mathcal{Z}$ is the observation function, mapping true system state to what the agent actually perceives, which may be all or part of the underlying state. Finally, $\mathcal{J} : (\mathcal{S} \times \mathcal{A})^\tau \to \mathbb{R}$ is the objective function, which the agent seeks to optimize over the trajectory τ of its interactions with the system. The agent's task is then to find a policy $\pi : \mathcal{Z} \to \mathcal{A}$ that optimizes $\mathcal{J}$.

The problem that AI agents are attempting to solve can be framed as an instance of the SC problem. The AI agent perceives its environment through an observation function $\mathcal{O}$, typically rendered as natural language text. Its action space $\mathcal{A}$ ranges from primitive token generation to temporally extended tool invocations. The stochastic transition function f governs how the environment evolves in response to the AI agent's actions and the inherent stochasticity of the world. The objective function $\mathcal{J}$ measures the degree to which the trajectory of the AI agent's interactions satisfies the requirements implicitly defined by a user's instructions. Under this framing, the AI agent's goal is to find a control policy π that maps its observations to actions so as to optimize $\mathcal{J}$.

While the SC problem captures the essential structure of the AI agent setting, it remains too general to directly apply the tools of the RL literature. In particular, the SC problem places no assumptions on the temporal structure of interaction, the form of the objective function, or the relationship between the agent's observations and the true system state. In the following subsection, we progressively specialize the SC problem by introducing assumptions that make the framework more tractable, while surfacing the implications of each assumption for the AI agent setting.

3.3. Specializing the Stochastic Control Problem for AI agents

Although the goal of finding a control policy π that maximizes the objective function $\mathcal{J}$ is clear, how this process may be completed requires more definition. Thus, there are still some specializations of the SC problem that must be made before SC is tractable for describing an agent.

The first issue with this formulation of the SC problem is that it is unclear how often the agent receives observations from the system, or when the agent can apply actions to the system to control it. The first specialization of SC is to impose a discrete timestep structure. This structure means that at time $t = 0, 1, 2, ..., n$, the agent receives an observation, chooses an action, and the system transitions to the next state.

The second issue with the SC formulation is that there is no assumption as to what the agent can observe about the system. As a stepping stone toward a tractable framework, we first make the strong assumption that the agent can observe the true state of the system, making the observation function the identity $\mathcal{O}(s_t) = s_t$. With this change, we also change the definition of the policy π to be a function $\pi : \mathcal{S} \to \mathcal{A}$ where $\mathcal{S}$ represents the set of possible states the system can take on. While this assumption of being able to observe the true state of the system s_t is strong, we will show the implications of relaxing this assumption in Section 3.5.

Then, we must make the transition of the system more tractable. In the current formulation of $f(S, A, W)$, the system transitions according to this function with noise W

as input. Given that the noise W is sampled from a fixed distribution, we can marginalize over the noise to obtain a probabilistic equation. After performing this marginalization, we arrive at the transition function $P(s_{t+1}|s_t, a_t)$ which defines a probability distribution over next states given the current state and action. This imposes the Markov property, that the transitions of this distribution only rely on the current state s_t and the action a_t applied in s_t. This means that the history of states and actions will not influence the transition to s_{t+1}. This is also an idealized assumption, as it is possible that the history of states $s_{0:t}$ and actions $a_{0:t}$ taken to reach the current state may be relevant to predicting the next state. We will relax this assumption in a later section.

Lastly, we also modify the objective function $\mathcal{J}$ from a trajectory level reward of $\mathcal{J} : (\mathcal{S} \times \mathcal{A})^\tau \to \mathbb{R}$, to a per-timestep reward function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. This decomposition of the objective into per-timestep rewards is what makes the optimization problem tractable, as it allows the use of dynamic programming to reason about long-horizon outcomes one step at a time.

3.4. Markov Decision Processes (MDPs)

These specializations transform the SC tuple $\langle \mathcal{S}, \mathcal{A}, f, \mathcal{O}, \mathcal{J} \rangle$ into the Markov Decision Process (MDP) tuple $\langle \mathcal{S}, \mathcal{A}, P, R \rangle$. This formulation is tractable for training an agent: when the transition function $P(s_{t+1} \mid s_t, a_t)$ is known for all state transitions, dynamic programming methods (Bellman, 1957) can be applied to provably recover the optimal policy π^*. In practice, however, the transition function is rarely known, and the agent must instead learn a policy through direct interaction with the system.

AI agents can be naturally mapped onto this MDP tuple. While we discuss the problems that AI agents are solving during deployment, we do note that RLHF (Christiano et al., 2017) and RLVR (Shao et al., 2024; Guo et al., 2025) leverage the POMDP framing during the post-training phase. The state s_t corresponds to the textual representation of the current situation. The action a_t corresponds to the agent's output, such as individual tokens or tool invocations. The transition kernel $P(s_{t+1} \mid s_t, a_t)$ governs how the environment responds to the agent's actions, and the reward $R(s_t, a_t)$ measures the degree to which that action advances the user's objective. The agent then cycles through observing s_t, selecting a_t, and observing the resulting s_{t+1} and r_{t+1}, which is precisely the interaction loop that MDPs formalize.

One natural tension in this mapping is the Markov property. The MDP formulation assumes that s_t is a sufficient statistic for the future where the transition to s_{t+1} depends only on the current state and action, and not on any prior history. However, the textual representation of the world that an AI agent receives is not the full true state of the world; it is a partial, language-rendered observation of it. The true underlying state of the environment, which includes latent user intent, external system state, and information that cannot be fully expressed in text, is not fully captured by the agent's input (e.g., user intention). When the agent's observation is not a sufficient statistic for the true state, the Markov property no longer holds at the level of observations. This motivates relaxing the full observability assumption of the MDP, which we do in the following section by adopting the POMDP formulation.

3.5. Partially Observable MDPs (POMDPs)

We now relax the full observability assumption by introducing two additional components into the MDP formulation (Kaelbling et al., 1998). First, Ω is the observation space, describing the set of observations the agent can perceive. Second, $\mathcal{O} : \mathcal{S} \times \mathcal{A} \to \prod(\Omega)$ is the observation function, which defines the probability distribution over observations the agent receives given the true state and the action taken. Together, these additions transform the MDP tuple into the POMDP tuple $\langle \mathcal{S}, \mathcal{A}, P, R, \Omega, \mathcal{O} \rangle$.

We now map each component of the POMDP tuple to the AI agent setting, summarized in Table 1 and elaborated upon here. The state space $\mathcal{S}$ of AI agents is the true underlying state of the external system(s) that the AI agent is interacting with. This includes the user's intents, safety constraints, results from external tool calls, real-world data (e.g., a stock price), and the full description of the task that the AI agent is to complete. However, this complete task specification is unlikely to be fully observed by the AI agent as the user interacting with the AI agent is unlikely to elicit the full requirements. The action space $\mathcal{A}$ of the AI agent has not changed, as it could still be primitive token generation to temporally extended tool invocations. The system the AI agent is interacting with transitions according to $P(s_{t+1} \mid s_t, a_t)$. We can imagine stochasticity in the system being interacted with due to multiple reasons. Perhaps the system being interacted with is hosted on a web server with intermittent outages, or maybe the API of a tool has been updated to a newer version and returns an error. The reward function R in the AI agent setting is somewhat different from training (updating the policy) to deployment (freezing the policy), and we elaborate further on reward functions in Section 4. The observation space Ω corresponds to the set of all inputs (text, images, etc) the agent can receive, and the observation function $\mathcal{O} : \mathcal{S} \times \mathcal{A} \to \prod(\Omega)$ describes the process by which the true world state is rendered into the inputs delivered to the agent. This is a lossy process, as these inputs cannot fully capture the true state of the world and the user's intents. As a consequence, the AI agent must act under uncertainty about the true state of the world. This

Table 1. Mapping of POMDP tuple components $\langle \mathcal{S}, \mathcal{A}, P, R, \Omega, \mathcal{O} \rangle$ to the RL agent setting and the AI agent setting.

POMDP	RL Agent	AI agent
State $\mathcal{S}$	True configuration of the environment (e.g., game pixels, robot joint angles)	True state of external systems, including safety constraints, and full task specification
Action $\mathcal{A}$	Primitive actions (e.g., move left, apply torque) or temporally extended options (Sutton et al., 1999)	Token generation, tool invocations, or atomic actions (Ma et al., 2026); temporally extended analogues of options
Transition P	Stochastic dynamics governing how the environment evolves in response to actions	Environment stochasticity arises from web server outages, API changes, or non-deterministic tool outputs
Reward R	Scalar feedback signal provided by the environment at each timestep	Deployment: internalized proxy signal mapping instructions to goal states
Observation space Ω	Set of perceivable environment signals (e.g., image frames, sensor readings)	Set of all inputs the agent can receive: natural language text, images, tool outputs, and API responses
Observation function $\mathcal{O}$	Lossy mapping from true state to agent-perceivable signal; captures sensor noise and occlusion	Lossy rendering of true world state into text and other modalities; suppresses latent user intent and safety-critical context

is a challenge that the POMDP formalism captures precisely, and for which the RL literature offers principled solutions.

The mapping provided above shows that AI agents can be framed as solving the same problem as RL agents, and can be formalized as an MDP or POMDP. This mapping implies that there are some RL methods that can improve the performance of AI agents. Crucially, this is not merely a theoretical convenience that we are imposing. Recent works improving the performance of AI agents have also arrived at the POMDP formulation. SPA-RL (Wang et al., 2025a) and HCAPO (Tan et al., 2026) both propose step-wise reward redistribution for multi-step AI agents using a POMDP formulation. Wang et al. (2025b), Zhou et al. (2024), and Wang et al. (2025d) each propose advances to LLM-based agents while explicitly formulating the problem as a POMDP. While these works validate the POMDP as the natural framework for AI agent problems, they apply it exclusively to performance. The systematic application of this formalism to AI agent safety has not yet followed, which is the gap that this paper addresses.

This independent convergence is significant as it suggests the mapping is not only a theoretical convenience, but also of structural alignment between the problem settings. If researchers working improving AI agent performance are independently arriving at POMDP formulations, then any methods built on top of this formulation are available to be applied to AI agents. This would include the RL safety literature that has begun to develop tools and solutions to the issues currently plaguing AI agents, including reward misspecification (Muslimani et al., 2025; Vasan et al., 2024; Knox et al., 2023) (discussed in Section 3.6.1), exploration (Tse et al., 2025; Klissarov and Machado, 2023) (discussed in Section 3.6.2), and safe sequential decision-making under uncertainty (Mohammedalamen et al., 2025; Mohammedalamen and Bowling, 2025) (discussed in Section 3.6.3). These advancements address the same failure modes that manifest in practice as agents deleting databases, leaking user data, and pursuing misspecified objectives at the cost of human welfare. These failure modes are well documented in deployed AI agents and need to be addressed. We believe that applying the many years of RL research to these AI agents will improve AI agent safety.

3.6. AI Agents Failures through the Lens of POMDPs

The mapping of AI agents to the POMDP tuple $\langle \mathcal{S}, \mathcal{A}, P, R, \Omega, \mathcal{O} \rangle$ allows us to categorize contemporary agent failures as structural failures of the sequential decision-making process.

3.6.1. Reward Misspecification

A central failure mode visible in deployed AI agents is reward misspecification, which is the divergence between the

proxy objective an agent optimizes and the true objective the user intends. This divergence is not merely an engineering oversight; Skalse et al. (2022) formally proved that for any non-trivial proxy reward, there exist policies that achieve high proxy returns while producing arbitrarily low true utility. Any specification that omits context, nuance, or implicit constraints is theoretically exploitable. AI agents face this structure in many settings. The proxy objective is the literal task specification, while the true objective includes preserving existing state, respecting implicit constraints, and flagging uncertainty before taking irreversible actions. When these diverge, the consequences can be severe.

Formally, this problem would be described through the reward function R. The only way that a user can interact with the AI agent during deployment is through the initial prompt that attempts to describe the goal and what the AI agent should do to accomplish this goal. The user provides an initial prompt o_p with instruction p. Then, the AI agent observes this prompt o_p and attempts to decompose the problem into a series of steps to perform. However, the AI agent must also internalize some method of determining if the initial prompt goal has been met. We hypothesize that the AI agent learns an internal reward function $f(o_p, o_g)$ that maps o_p to the goal state o_g, potentially during the process of post-training with RLVR (Shao et al., 2024; Guo et al., 2025). This allows the reward function $R = f(o_p, o_g)$ to inform the agent when the goal is achieved, which would be a binary success signal or numerical feedback. However, if the initial prompt o_p is incorrectly specified, then the AI agent will likely find a way to hack this reward function as it is only a proxy of the desired reward function $o_d \neq o_p, f(o_p, o_g) \neq f(o_d, o_g)$ (Skalse et al., 2022). The downstream effect of this misspecification can have catastrophic outcomes. For example, in July 2025, Replit's AI agent deleted a production database of 1,206 executive records during a code freeze, then fabricated 4,000 synthetic records to mask the deletion (Nolan, 2025).

3.6.2. Safe Exploration

Another failure mode of AI agents that can be viewed through the lens of a POMDP is that of safe exploration. In RL, exploration is the process of taking actions to reduce uncertainty about the transition dynamics P or the reward function R. In AI agents, this often takes the form of "trial and error" with tools or code execution. Safe Exploration failures occur when the agent's search for a solution leads it into irreversible states (Lee, 2025), regions of the state space $\mathcal{S}$ from which it is impossible to return to a safe or functional state. For example, an agent tasked with "optimizing server disk space" might explore the action a="rm -rf /" as this would delete everything on the disk, setting the amount of disk space free to 100%. In a standard POMDP, the agent might only learn the penalty of this action after the transition. Unlike games or simulations where agents can be "reset," real-world AI agent deployments often lack a safety shield or a "constrained MDP" (Altman, 1999) framework that prevents the exploration of high-risk actions before they occur. We distinguish this from KL regularization used during RLHF and RLVR post-training (Christiano et al., 2017; Shao et al., 2024; Guo et al., 2025), which constrains how far the policy may drift from a reference model during optimization but places no hard constraints on which actions the deployed policy may take in a given state. A safety shield operates at inference time over the action space; KL regularization operates at training time over the parameter space. The former is what is missing from deployed AI Agents.

3.6.3. Observation Ambiguity

Failures in this category arise when an agent treats a POMDP as a fully observable MDP. This occurs when the agent assumes its current observation $o_t \in \Omega$ (e.g., a terminal snippet) is a sufficient statistic for the true state s_t. If an agent lacks a robust understanding of how the observed state may map to the actual underlying state, it may take actions that are optimal for the observed state but catastrophic for the actual state. For example, in February 2025, a developer used Claude Code for a Terraform migration. However, due to a missing state file, the agent interpreted the environment as empty and executed `terraform destroy`, deleting 1,943,200 rows of course data accumulated over 2.5 years (Grigorev, 2025). Instead of treating the missing state file as definitive evidence of an empty environment, a principled agent operating under partial observability would recognize this observation as ambiguous, as this state is consistent with both an empty environment and a misconfigured one, and take an information gathering action before committing to an irreversible transition. This is precisely the behavior that belief state planning in POMDPs is designed to produce (Kaelbling et al., 1998).

4. Counter Arguments

The AI agent does not receive a scalar reward signal during deployment, so using any MDP to describe the problem is incorrect. During the training of an AI agent, it is common to leverage a Process Reward Model (PRM) (Liu et al., 2025) that provides sparse feedback for a trajectory of an AI agent. This model attempts to help the reasoning capabilities of AI agents throughout a full trajectory of interacting with some environment. Once this training step is completed, the PRM is no longer used, and the AI agent is deployed into some setting where a user can interact with it. As the PRM is no longer available, the AI agent must possess a mechanism that can determine whether an objective has been met or not: the equivalent to a sparse reward signal. We hypothesize that during training, the AI agent implicitly learns a function

that compares instructions to the desired goal state of an environment to denote when a task is completed. It is possible that the AI agent implements this function as a sparse reward. While the environment only provides feedback in the form of text-based observations, we argue that the AI agent has internalized this feedback signal. This internalization process could be an interesting avenue for future work, as it is a proxy measure that is prone to gaming (Skalse et al., 2022). Therefore, ensuring that the AI agent internalizes the "best" proxy, with respect to its performance and safety objectives, is very important. Regardless of the true mechanism that evaluates completeness, the MDP framing that we provide in this paper provides a diagnostic lens for discussing and solving these reward specification problems.

AI agent failures are due to the finite context window of the transformer. This is a fair critique, and it operates at two levels. Firstly, a finite context window size means that information from early in a trajectory may be truncated or compressed. Secondly, the attention mechanism must learn which information from the context is relevant for the next prediction. Currently, there is no guarantee that safety-relevant information receives appropriate weight when it conflicts with task-completion information. Under the POMDP formalism, this is a structured form of partial observability: the observation function $\mathcal{O}$ may filter the true state in ways that may suppress safety-critical features. Elelimy et al. (2025b) propose that RL agents should learn an explicit mapping from trajectory history to state. This is a mechanism that the transformer's attention already instantiates, but trained under a next-token prediction objective rather than a safety-aware one. Ensuring that this history-to-state mapping preserves and appropriately weights safety specifications is precisely the problem that constrained MDPs address (Altman, 1999; Satija et al., 2020), suggesting again that the RL safety literature has tools directly applicable here.

AI agents use tools, whereas RL agents use primitive actions. There is a line of research that is analogous to how AI agents use tools. Sutton et al. (1999) introduces the idea of options. Options share the key structural property of tool calls in that both are temporally extended actions. Both tool calls and options have specific initiation conditions, execute an internal policy over a series of steps, and terminate upon reaching a defined endpoint. Options have also been used to overcome the exploration problem in RL (Klissarov and Machado, 2023; Jinnai et al., 2020), which could be useful to combat uncertainty in AI agents. The existence of tool calls by AI agents and options in RL agents implies that the problems that the action spaces of these two agents are fundamentally equivalent but differ in implementation. Concurrent work from Ma et al. (2026) introduces the idea of atomic actions, where the AI agent is trained across a pre-defined set of actions such as code localization, code editing, and test-case generation for completing software development tasks. The AI agent is then trained, using RL, to optimize rewards across each of these atomic actions (Ma et al., 2026). This suggests that the action spaces of AI agents and RL agents are not as different as previously thought. The finding that AI agents can also optimize over multiple tasks naturally links AI agents to the multi-task RL literature (McLean et al., 2025b; Nauman et al., 2025), which suggests there are even more connections between AI agents and RL agents than those highlighted in this paper.

5. Limitations

Throughout this paper, we have made a formal argument that AI agents that leverage LLMs to choose actions, view observations, and receive feedback are analagous to RL agents in problem setting. Due to this structural equivalence, we then argue that the application of RL techniques such as CMDPs (Altman, 1999), shielding, or even reward function alignment (Muslimani et al., 2025) may be used to mitigate safety issues that have arisen when using AI Agents. However, we would like to make two important distinctions.

First, a significant scaling gap exists between the abstract formalisms that unify these settings and the practical solution methods deployed within them. The action and observation spaces of classical RL agents are typically low-dimensional and strictly bounded, ranging from discrete game controls (Bellemare et al., 2013) to continuous robotic joint torques (Tunyasuvunakool et al., 2020; McLean et al., 2025a). Conversely, LLM-based AI agents operate within massive, open-ended combinatorial spaces defined by natural language tokens, variable API structures, and external software environments. Because of this divergence, classical RL safety algorithms—which often rely on tabular representations, exact runtime shielding, or known transition matrices—cannot be mapped one-to-one onto LLM backbones. Rather than invalidating the connection, however, we argue this presents a profound opportunity for the community to scale foundational RL principles up to high-dimensional, semantic domains.

Second, while our framework addresses safety within current sequential decision-making paradigms, it must be positioned alongside the broader literature on scalable oversight and AI control (Bowman et al., 2022). As AI agents approach or exceed human-level capabilities in complex workflows, evaluating the safety and alignment of their trajectories becomes increasingly difficult for human overseers alone. While viewing agent safety as an RL problem provides principled tools for optimization and constraint verification, it does not inherently solve the foundational challenge of specifying a flawless objective function for superhuman systems. Our position should therefore be viewed

as a complementary framework for structural safety, rather than a singular solution to the overarching AI control problem.

6. Conclusion

Through this paper, we have shown that AI agents are solving a similar problem that RL researchers have attempted to solve for decades. By showing these problems are equivalent, we then argue that the safety issues currently plaguing AI agents have been previously studied by the RL community. By using the lens of POMDPs, we have created one example of a common language that both AI agents research and RL research can use to describe their problems and share solutions with one another. This also implies that there may be more powerful ways to draw equivalences with the RL literature that we have not explored here. This position creates a concrete opportunity to call for the AI agent and RL communities to come together to solve these problems of reward misspecification, safe exploration, and observation ambiguity. AI agent researchers, as we have highlighted, will find benefit in the vast history of RL research, as RL researchers have previously attempted to grapple with the problems that AI agents are dealing with. For RL researchers, this is an opportunity to apply some of the skills and knowledge training RL agents to create additional real-world successes of RL (Bellemare et al., 2020; Degrave et al., 2022). RL researchers have the chance to work alongside AI agent researchers in a way that can provide meaningful impact outside of the lab. As the deployment of AI agents continues to expand into high-stakes domains, the collaboration we advocate for here offers a principled path toward agents that are not only more capable, but safer. This collaboration might take the form of shared benchmarks that test both RL and AI agents on the same sequential decision-making problems, shared formalisms for diagnosing safety failures, and joint evaluation frameworks that hold both communities to rigorous empirical standards.

7. Acknowledgments

The authors would like to thank Michael Bowling, Matthew E. Taylor, Adam White, Brad Knox, and Pablo Samuel Castro for constructive discussions and feedback on an early draft of this work. Portions of this research is based on work supported by the Canadian AI Safety Institute Research Program at CIFAR, funded by the Government of Canada. Funding for this research was also provided by the IVADO Postdoctoral Research Funding Program.

References

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G Bellemare. Reincarnating reinforcement learning: Reusing prior computation to accelerate progress. *Advances in Neural Information Processing Systems*, 2022.

Eitan Altman. *Constrained Markov decision processes*. CRC Press, 1999.

Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: an evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 2013.

Marc G. Bellemare, Salvatore Candido, Pablo Samuel Castro, Jun Gong, Marlos C. Machado, Subhodeep Moitra, Sameera S. Ponda, and Ziyu Wang. Autonomous navigation of stratospheric balloons using reinforcement learning. *Nature*, 2020.

Richard Bellman. *Dynamic Programming*. Dover Publications, 1957.

Daniil A. Boiko, Robert MacKnight, Ben Kline, and Gabe Gomes. Autonomous chemical research with large language models. *Nature*, 2023.

Samuel R. Bowman, Jeeyoon Hyun, Ethan Perez, Edwin Chen, Craig Pettit, Scott Heiner, Kamilė Lukošiūtė, Amanda Askell, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Christopher Olah, Daniela Amodei, Dario Amodei, Dawn Drain, Dustin Li, Eli Tran-Johnson, Jackson Kernion, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Liane Lovitt, Nelson Elhage, Nicholas Schiefer, Nicholas Joseph, Noemí Mercado, Nova DasSarma, Robin Larson, Sam McCandlish, Sandipan Kundu, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Timothy Telleen-Lawton, Tom Brown, Tom Henighan, Tristan Hume, Yuntao Bai, Zac Hatfield-Dodds, Ben Mann, and Jared Kaplan. Measuring progress on scalable oversight for large language models, 2022.

Cristiano Castelfranchi. Guarantees for autonomy in cognitive agent architecture. In *Intelligent Agents*, 1995.

Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in Neural Information Processing Systems*, 2017.

Jonas Degrave, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de las Casas, Craig Donner, Leslie Fritz, Cristian Galperti, Andrea Huber, James Keeling, Maria Tsimpoukelli, Jackie Kay, Antoine Merle, Jean-Marc Moret, Seb Noury, Federico Pesamosca, David Pfau, Olivier Sauter, Cristian Sommariva, Stefano Coda, Basil Duval, Ambrogio Fasoli,

Pushmeet Kohli, Koray Kavukcuoglu, Demis Hassabis, and Martin Riedmiller. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 2022.

Zehang Deng, Yongjian Guo, Changzhou Han, Wanlun Ma, Junwu Xiong, Sheng Wen, and Yang Xiang. Ai agents under threat: A survey of key security challenges and future pathways. *Association for Computing Machinery Computing Surveys*, 2025.

Aisha Down. Meta ai agent's instruction causes large sensitive data leak to employees. *The Guardian*, 2026.

Esraa Elelimy, Brett Daley, Andrew Patterson, Marlos C. Machado, Adam White, and Martha White. Deep reinforcement learning with gradient eligibility traces. In *Reinforcement Learning Journal*, 2025a.

Esraa Elelimy, David Szepesvari, Martha White, and Michael Bowling. Rethinking the foundations for continual reinforcement learning. *Reinforcement Learning Journal*, 2025b.

Dyke Ferber, Omar S. M. El Nahhas, Georg Wölflein, Isabella C. Wiest, Jan Clusmann, Marie-Elisabeth Leßmann, Sebastian Foersch, Jacqueline Lammert, Maximilian Tschochohei, Dirk Jäger, Manuel Salto-Tellez, Nikolaus Schultz, Daniel Truhn, and Jakob Nikolas Kather. Development and validation of an autonomous artificial intelligence agent for clinical decision-making in oncology. *Nature Cancer*, 2025.

Michael R. Genesereth and Steven P. Ketchpel. Software agents. *Communications of the Association for Computing Machinery*, 1994.

Alexey Grigorev. How i dropped our production database and now pay 10*Substack*, 2025.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645, 2025. ISSN 1476-4687.

Khurram Javed and Richard S. Sutton. The big world hypothesis and its ramifications for artificial intelligence. In *Finding the Frame: An RLC Workshop for Examining Conceptual Frameworks*, 2024.

Yuu Jinnai, Jee W. Park, Marlos C. Machado, and George Konidaris. Exploration in reinforcement learning with deep covering options. In *International Conference on Learning Representations*, 2020.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 1998.

Donald E. Kirk. Optimal control theory : an introduction. 1970.

Martin Klissarov and Marlos C. Machado. Deep laplacian-based options for temporally-extended exploration. In *International Conference on Machine Learning*, 2023.

W. Bradley Knox, Alessandro Allievi, Holger Banzhaf, Felix Schmitt, and Peter Stone. Reward (mis)design for autonomous driving. *Artificial Intelligence*, 2023.

Naveen Krishnan. Advancing multi-agent systems through model context protocol: Architecture, implementation, and applications. *CoRR*, abs/2504.21030, 2025.

Sang-Hyun Lee. Robust and scalable autonomous reinforcement learning in irreversible environments. In *Advances in Neural Information Processing Systems*, 2025.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *CoRR*, 2005.01643, 2020.

Runze Liu, Junqi Gao, Jian Zhao, Kaiyan Zhang, Xiu Li, Biqing Qi, Wanli Ouyang, and Bowen Zhou. Can 1b LLM surpass 405b LLM? rethinking compute-optimal test-time scaling. In *Workshop on Reasoning and Planning for Large Language Models*, 2025.

Chris Lu, Cong Lu, Robert Tjarko Lange, Yutaro Yamada, Shengran Hu, Jakob Foerster, David Ha, and Jeff Clune. Towards end-to-end automation of AI research. *Nature*, 2026.

Yingwei Ma, Yue Liu, Xinlong Yang, Yanhao Li, Kelin Fu, Yibo Miao, Yuchong Xie, Zhexu Wang, and Shing-Chi Cheung. Scaling coding agents via atomic skills. *CoRR*, abs/2604.05013, 2026.

Marlos C. Machado, Marc G. Bellemare, Erik Talvitie, Joel Veness, Matthew Hausknecht, and Michael Bowling. Revisiting the arcade learning environment: evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research*, 2018.

Jon Martindale. Meta security researcher's ai agent accidentally deleted her emails. *PCMag*, 2026.

Reginald McLean, Evangelos Chatzaroulas, Luc McCutcheon, Frank Röder, Tianhe Yu, Zhanpeng He, K.R. Zentner, Ryan Julian, J K Terry, Isaac Woungang, Nariman Farsad, and Pablo Samuel Castro. Meta-world+: An improved, standardized, RL benchmark. In *Conference on Neural Information Processing Systems, Datasets and Benchmarks Track*, 2025a.

Reginald McLean, Evangelos Chatzaroulas, J K Terry, Isaac Woungang, Nariman Farsad, and Pablo Samuel Castro. Multi-task reinforcement learning enables parameter scaling. *Reinforcement Learning Journal*, 2025b.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 2015.

Montaser Mohammedalamen and Michael Bowling. Generalization in monitored markov decision processes (mon-mdps). *CoRR*, 2505.08988, 2025.

Montaser Mohammedalamen, Dustin Morrill, Alexander Sieusahai, yash satsangi, and Michael Bowling. Learning to be cautious. *Transactions on Machine Learning Research*, 2025.

Calarina Muslimani, Kerrick Johnstonbaugh, Suyog Chandramouli, Serena Booth, W. Bradley Knox, and Matthew E. Taylor. Towards improving reward design in RL: A reward alignment metric for RL practitioners. In *Reinforcement Learning Journal*, 2025.

Michal Nauman, Marek Cygan, Carmelo Sferrazza, Aviral Kumar, and Pieter Abbeel. Bigger, regularized, categorical: High-capacity value functions are efficient multi-task learners. In *Advances in Neural Information Processing Systems*, 2025.

Beatrice Nolan. An ai-powered coding tool wiped out a software company's database, then apologized for a 'catastrophic failure on my part'. *Fortune*, 2025.

OpenAI. Introducing chatgpt, 2022.

Harshith Padigela, Chintan Shah, and Dinkar Juyal. ML-dev-bench: Comparative analysis of AI agents on ML development workflows. In *Workshop on Deep Learning for Code, Internation Conference on Learning Representations*, 2025.

Davide Paglieri, Bartłomiej Cupiał, Samuel Coward, Ulyana Piterbarg, Maciej Wolczyk, Akbir Khan, Eduardo Pignatelli, Łukasz Kuciński, Lerrel Pinto, Rob Fergus, Jakob Nicolaus Foerster, Jack Parker-Holder, and Tim Rocktäschel. BALROG: Benchmarking agentic LLM and VLM reasoning on games. In *International Conference on Learning Representations*, 2025.

Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning*, 2017.

Andrew Patterson, Adam White, and Martha White. A generalized projected bellman error for off-policy value estimation in reinforcement learning. *Journal of Machine Learning Research*, 2022.

Stuart J. Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Pearson, 1995.

Ranjan Sapkota, Konstantinos I. Roumeliotis, and Manoj Karkee. AI Agents vs. Agentic AI: A Conceptual taxonomy, applications and challenges. *Information Fusion*, 2026.

Harsh Satija, Philip Amortila, and Joelle Pineau. Constrained markov decision processes via backward value functions. In *International Conference on Machine Learning*, 2020.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, 2402.03300, 2024.

Adi Simhi, Jonathan Herzig, Martin Tutek, Itay Itzhak, Idan Szpektor, and Yonatan Belinkov. Managerbench: Evaluating the safety-pragmatism trade-off in autonomous LLMs. In *International Conference on Learning Representations*, 2026.

Joar Skalse, Nikolaus H. R. Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and characterizing reward hacking. *Advances in Neural Information Processing Systems*, 2022.

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

Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 1988.

Richard S. Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 1999.

Richard S. Sutton, Hamid Reza Maei, Doina Precup, Shalabh Bhatnagar, David Silver, Csaba Szepesvári, and Eric Wiewiora. Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *International Conference on Machine Learning*, 2009.

Hui-Ze Tan, Xiao-Wen Yang, Hao Chen, Jie-Jing Shao, Yi Wen, Yuteng Shen, Weihong Luo, Xiku Du, Lan-Zhe Guo, and Yu-Feng Li. Hindsight credit assignment for long-horizon llm agents, 2026.

Hon Tik Tse, Siddarth Chandrasekar, and Marlos C. Machado. Reward-aware proto-representations in reinforcement learning. *Advances in Neural Information Processing Systems*, 2025.

Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, Nicolas Heess, and Yuval Tassa. dm control: Software and tasks for continuous control. *Software Impacts*, 6, 2020.

Gautham Vasan, Yan Wang, Fahim Shahriar, James Bergstra, Martin Jägersand, and A. Rupam Mahmood. Revisiting sparse rewards for goal-reaching reinforcement learning. *Reinforcement Learning Journal*, 2024.

Akifumi Wachi, Wataru Hashimoto, Xun Shen, and Kazumune Hashimoto. Safe exploration in reinforcement learning: A generalized formulation and algorithms. In *Advances in Neural Information Processing Systems*, 2023.

Hanlin Wang, Chak Tou Leong, Jiashuo Wang, Jian Wang, and Wenjie Li. Spa-rl: Reinforcing llm agents via stepwise progress attribution. *CoRR*, abs/2505.20732, 2025a.

Kangrui Wang, Pingyue Zhang, Zihan Wang, Yaning Gao, Linjie Li, Qineng Wang, Hanyang Chen, Yiping Lu, Zhengyuan Yang, Lijuan Wang, Ranjay Krishna, Jiajun Wu, Li Fei-Fei, Yejin Choi, and Manling Li. VAGEN: Reinforcing world model reasoning for multi-turn VLM agents. In *Advances in Neural Information Processing Systems*, 2025b.

Weixun Wang, XiaoXiao Xu, Wanhe An, Fangwen Dai, Wei Gao, Yancheng He, Ju Huang, Qiang Ji, Hanqi Jin, Xiaoyang Li, Yang Li, Zhongwen Li, Shirong Lin, Jiashun Liu, Zenan Liu, Tao Luo, Dilxat Muhtar, Yuanbin Qu, Jiaqiang Shi, Qinghui Sun, Yingshui Tan, Hao Tang, Runze Wang, Yi Wang, Zhaoguo Wang, Yanan Wu, Shaopan Xiong, Binchen Xu, Xander Xu, Yuchi Xu, Qipeng Zhang, Xixia Zhang, Haizhou Zhao, Jie Zhao, Shuaibing Zhao, Baihui Zheng, Jianhui Zheng, Suhang Zheng, Yanni Zhu, Mengze Cai, Kerui Cao, Xitong Chen, Yue Dai, Lifan Du, Tao Feng, Tao He, Jin Hu, Yijie Hu, Ziyu Jiang, Cheng Li, Xiang Li, Jing Liang, Xin Lin, Chonghuan Liu, ZhenDong Liu, Zhiqiang Lv, Haodong Mi, Yanhu Mo, Junjia Ni, Shixin Pei, Jingyu Shen, XiaoShuai Song, Cecilia Wang, Chaofan Wang, Kangyu Wang, Pei Wang, Tao Wang, Wei Wang, Ke Xiao, Mingyu Xu, Tiange Xu, Nan Ya, Siran Yang, Jianan Ye, Yaxing Zang, Duo Zhang, Junbo Zhang, Boren Zheng, Wanxi Deng, Ling Pan, Lin Qu, Wenbo Su, Jiamang Wang, Wei Wang, Hu Wei, Minggang Wu, Cheng Yu, Bing Zhao, Zhicheng Zheng, and Bo Zheng. Let it flow: Agentic crafting on rock and roll, building the rome model within an open agentic learning ecosystem. *CoRR*, abs/2512.24873, 2026.

Xingyao Wang, Simon Rosenberg, Juan Michelini, Calvin Smith, Hoang Tran, Engel Nyst, Rohit Malhotra, Xuhui Zhou, Valerie Chen, Robert Brennan, and Graham Neubig. The OpenHands Software Agent SDK: A Composable and Extensible Foundation for Production Agents. *CoRR*, abs/2511.03690, 2025c.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, Yiping Lu, Kyunghyun Cho, Jiajun Wu, Li Fei-Fei, Lijuan Wang, Yejin Choi, and Manling Li. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning, 2025d.

Michael Wooldridge and Nicholas R. Jennings. Intelligent agents: theory and practice. *The Knowledge Engineering Review*, 1995.

Frank F. Xu, Yufan Song, Boxuan Li, Yuxuan Tang, Kritanjali Jain, Mengxue Bao, Zora Zhiruo Wang, Xuhui Zhou, Zhitong Guo, Murong Cao, Mingyang Yang, Hao Yang Lu, Amaad Martin, Zhe Su, Leander Melroy Maben, Raj Mehta, Wayne Chi, Lawrence Keunho Jang, Yiqing Xie, Shuyan Zhou, and Graham Neubig. Theagentcompany: Benchmarking LLM agents on consequential real world tasks. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2025.

John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik R Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*, 2024.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023.

Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. ArCHer: Training language model agents via hierarchical multi-turn RL. In *International Conference on Machine Learning*, 2024.

A. Additional Background Information

A.1. AI agents

Wooldridge and Jennings (1995)'s definition of an agent contains four main components. The first component of an agent is autonomy: an agent can act without the direct intervention of humans or others, where the agent has some control over their own actions and internal state (Castelfranchi, 1995). Wooldridge and Jennings (1995) also attribute a social factor to an agent, where the agent can interact with other intelligent entities. These entities may be other agents, or possibly humans, and this communication happens in some kind of agent-communication language (Genesereth and Ketchpel, 1994). Additionally, an agent perceives their environment and responds, in a timely fashion, to the changes in that environment. Finally, the agent has a sense of pro-activeness where the agent does not only respond to the environment, but the agent can also exhibit goal-directed behaviors by taking an initiative. While AI agents operate as individual autonomous systems, the deployment of multiple agents in coordination gives rise to what Sapkota et al. (2026) term Agentic AI. This system may have a meta-agent that acts as an orchestrator responsible for sharing information, eliciting cooperation, or coordinating tasks between agents in order to accomplish a task (Sapkota et al., 2026).

AI agents have recently garnered attention for their promising capabilities across diverse settings. Agents such as the AI Scientist (Lu et al., 2026) are enabling the end-to-end automation of scientific discovery in the field of AI. The AI Scientist is capable of generating novel research ideas, creating experiments to test hypotheses, writing the code to implement these experiments, generating plots summarizing experiments, and even drafting manuscripts that were accepted at a peer-reviewed AI workshop. While the AI Scientist operates entirely in the virtual world, AI agents such as Coscientist (Boiko et al., 2023) have been developed that can assist with experiments in real-world settings such as chemistry, where Coscientist can autonomously plan, execute, and evaluate experiments. Beyond scientific discovery, AI agents have become active participants in software engineering. Systems such as SWE-agent (Yang et al., 2024) and OpenHands (Wang et al., 2025c) are capable of autonomously navigating production codebases, identifying the source of reported bugs, implementing fixes, and validating their solutions against existing test suites. Benchmark performance on SWE-Bench has improved from $\sim 9\%$ (Yang et al., 2024) to $\sim 72\%$ (Wang et al., 2025c) in less than 6 months, demonstrating rapid capability growth in autonomous software development. In the healthcare domain, agents such as the autonomous oncology agent developed by Ferber et al. (2025) can coordinate across multiple clinical data sources, analyzing histopathology slides, querying treatment guidelines, and synthesizing patient-specific recommendations, with documented performance approaching that of specialist physicians.