Hierarchical Reward Design from Language: Enhancing Alignment of Agent Behavior with Human Specifications

Zhiqin Qian
Rice University
Houston, TX, USA
bill.qian@rice.edu

Ryan Diaz
Rice University
Houston, TX, USA
ryandiaz@rice.edu

Sangwon Seo
Rice University
Houston, TX, USA
sangwon.seo@rice.edu

Vaibhav Unhelkar
Rice University
Houston, TX, USA
vaibhav.unhelkar@rice.edu

ABSTRACT

When training artificial intelligence (AI) to perform tasks, humans often care not only about *whether* a task is completed but also *how* it is performed. As AI agents tackle increasingly complex tasks, aligning their behavior with human-provided specifications becomes critical for responsible AI deployment. *Reward design* provides a direct channel for such alignment by translating human expectations into reward functions that guide reinforcement learning (RL). However, existing methods are often too limited to capture nuanced human preferences that arise in long-horizon tasks. Hence, we introduce *Hierarchical Reward Design from Language (HRDL)*: a problem formulation that extends classical reward design to encode richer behavioral specifications for hierarchical RL agents. We further propose *Language to Hierarchical Rewards (L2HR)* as a solution to HRDL. Experiments show that AI agents trained with rewards designed via L2HR not only complete tasks effectively but also better adhere to human specifications. Together, HRDL and L2HR advance the research on human-aligned AI agents.

KEYWORDS

Reward Design; Human-Centered AI; Hierarchical RL

ACM Reference Format:
Zhiqin Qian, Ryan Diaz, Sangwon Seo, and Vaibhav Unhelkar. 2026. Hierarchical Reward Design from Language: Enhancing Alignment of Agent Behavior with Human Specifications. In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25 – 29, 2026*, IFAAMAS, 10 pages. https://doi.org/10.65109/JTHG8732

1 Introduction

Robots and other AI agents are increasingly being deployed in human-centric environments, such as homes, hospitals, and disaster zones [18, 28, 48, 51, 55, 56, 67, 68, 70]. Their usefulness depends not only on task completion, but also on respecting human intentions, operational rules, and safety requirements (collectively referred to as *behavior specifications*).

Aligning agent behavior with these specifications is central to their responsible deployment, with various paradigms being actively explored for conveying such specifications [11]. This work focuses on *reward design*, a paradigm where humans convey specifications through reward functions that guide reinforcement learning (RL), typically before AI deployment. This paradigm is particularly suitable for use cases where specifications are relatively stable and the costs of reward design can be amortized, such as during the repeated use of AI agents by humans in domain-specific contexts.

As AI agents take on increasingly complex tasks, more advanced reward design methods are needed to capture equally complex specifications. Humans rarely reason about tasks and associated specifications as monolithic goals [8, 9, 12, 29, 40, 52]. Instead, we naturally break them into subtasks. Hierarchical frameworks in RL mirror this structure by decomposing tasks into subtasks and organizing them over long horizons [13, 14, 49, 66]. This *hierarchical approach to policy learning* has enabled agents to complete tasks of increasingly longer horizons. However, *the reward design for these hierarchical RL agents remains largely unexplored,* thereby limiting human-AI alignment.

As illustrated in Fig. 1, specifications for long-horizon tasks often include details on *what* subtasks to perform, in *which* order, and *how* they are executed. Existing reward design methods encode these specifications via a flat reward function of the form $\tilde{r}_{flat}(s, a)$. We show *theoretically and empirically that flat rewards are fundamentally limited in capturing specifications for long-horizon tasks.* To address this limitation:

- We introduce the *Hierarchical Reward Design (HRD)* problem, which enables designers to express behavioral specifications inspired by the same structured way people naturally think and teach. Unlike the classical (flat) reward design problem [60], HRD admits reward solutions that enable encoding of complex specifications for long-horizon tasks, capturing both what subtasks to perform and how to execute them. HRD is a general formulation that can be instantiated with multiple input modalities, analogous to how flat reward design has been realized via proxy signals or language [19, 32, 38]. Because natural language is an intuitive medium for specifying layered instructions, we then provide a language-based instantiation called *Hierarchical Reward Design from Language (HRDL, pronounced "hurdle").*

Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), C. Amato, L. Dennis, V. Mascardi, J. Thangarajah (eds.), May 25 – 29, 2026, Paphos, Cyprus. © 2026 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). https://doi.org/10.65109/JTHG8732

An extended version of this paper, which includes the Appendix and supplementary material mentioned in the text, is available at http://tiny.cc/hrdl-appendix

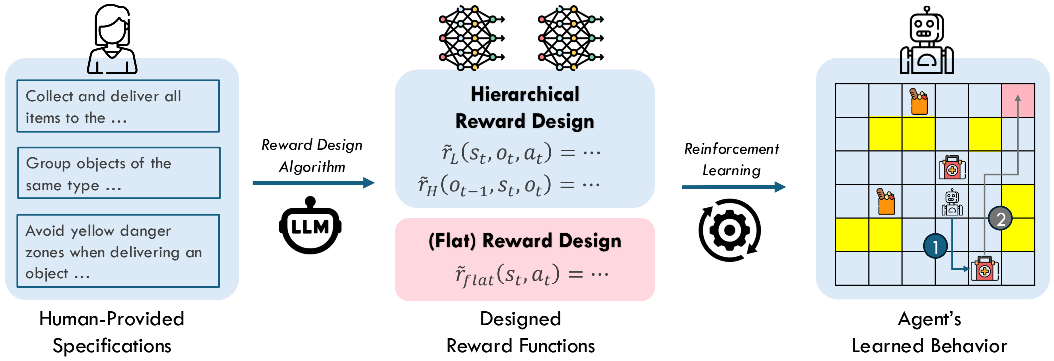


Figure 1: This work introduces the Hierarchical Reward Design from Language (HRDL) problem. Unlike prior work on reward design, HRDL decomposes reward design into low- and high-level components ($\tilde{r}_L, \tilde{r}_H$). Language to Hierarchical Rewards (L2HR), our proposed solution to HRDL, leverages language models to guide the synthesis of these hierarchical rewards, enabling existing RL algorithms to train agents that are better aligned with human specifications.

- We prove that hierarchical rewards of HRDL are strictly more expressive than flat rewards used by prior works, while remaining compatible with standard decision-making frameworks (i.e. Markov and semi-Markov Decision Processes) and RL algorithms.
- We present *Language to Hierarchical Rewards (L2HR)*, an initial solution to HRDL that generates hierarchical rewards directly from natural language specifications, making reward design more accessible while leveraging the reasoning capabilities of large language models [2, 22, 36]. L2HR produces reward structures that guide both high-level subtask selection and low-level execution.

Human subject and numerical experiments demonstrate hierarchical reward design's advantages over flat reward design. Hierarchical reward design (coupled with hierarchical RL) enables AI agents to successfully complete tasks and better align their behavior with language specifications. We view this work as an initial but important step toward aligning AI systems with human expectations through the lens of HRD. Through theoretical analysis and empirical findings, this paper lays the groundwork for future research on designing human-aligned reward structures that employ hierarchies and human input.

2 Background and Related Work

This work focuses on AI agents tasked with sequential decision-making tasks, which are commonly modeled using Markov Decision Processes (MDP) or its variants [50, 65]. An MDP is defined by the tuple $\mathcal{M} \doteq (\mathcal{S}, \mathcal{A}, T, r, \gamma, h)$, where $\mathcal{S}$ and $\mathcal{A}$ are the state and action spaces, $T(s'|s,a)$ the transition dynamics, $r(s,a)$ the reward function, $\gamma \in [0,1]$ the discount factor, and h the horizon. MDPs can be solved using RL, which aims to find a policy $\pi(a|s)$ that maximizes the expected discounted return $\mathbb{E}[\sum_{t=0}^{h} \gamma^t r(s_t, a_t)]$.

2.1 Hierarchical Reinforcement Learning

While capable, standard RL algorithms struggle with long-horizon tasks. Hierarchical RL (HRL) seeks to solve MDPs with long horizons by decomposing them into simpler subtasks [13, 14, 49, 57, 58, 66]. A widely used HRL paradigm is the options framework [4, 66, 74]. Here, the agent has access to a discrete set of temporally-extended behaviors called options $\mathcal{O}$. Each $o \in \mathcal{O}$ is associated with an intra-option policy $\pi_L(a|s,o)$ and a termination condition $\beta(o^-, s)$. A high-level policy $\pi_H(o|o^-, s)$ selects options, while the low-level policy executes primitive actions until the selected option terminates. The reward model for options computes the expected cumulative reward until option termination. Following [66], we let $\mathcal{E}(o, s, t)$ denote the event where option o is initiated in state s at timestep t, and define the option-level reward as:

$$r_{opt}(s, o) \doteq \mathbb{E}[\Sigma_{i=1}^{k} \gamma^{i-1} r_{t+i} \mid \mathcal{E}(o, s, t)] \tag{1}$$

where k denotes the number of steps after which the initiated option o terminates, determined by its termination condition β.

Another line of HRL research follows the *feudal/goal-conditioned framework* [13, 26, 31, 42, 69], which also decomposes a task into subtasks but differs in how the hierarchical policies are trained and how reward signals are assigned. In this framework, the high-level manager selects subgoals and receives a *task* reward, as in the options framework. However, unlike the options framework, the low-level worker receives a separate *pseudo-reward* $r_p(s, o, a)$ that measures progress toward achieving the current subgoal.

2.2 Reward Design

A core challenge in using MDPs and RL is reward design [60, 65]. Given a well-designed reward function, agents can use RL algorithms to solve the MDP. However, in practice, the design of a reward function is non-trivial and can lead to a host of problems, including poor alignment between humans and agents [3].

2.2.1 Reward Design Problem. To formally study reward design, Singh et al. introduce the (flat) Reward Design Problem (RDP) [60]. RDP is formalized as a tuple $P = (\mathcal{M}_p, \mathcal{R}, \mathcal{A}_{\mathcal{M}_p}, F)$, where

- $\mathcal{M}_p = (\mathcal{S}, \mathcal{A}, T, \gamma, h)$ is the *world model*;
- $\mathcal{R}$ is the space of reward functions;
- $\mathcal{A}_{\mathcal{M}_p}(r) : \mathcal{R} \to \Pi$ is an algorithm to compute policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ that optimizes reward $r \in \mathcal{R}$ in the MDP $(\mathcal{M}_p, r)$; and
- $F : \Pi \to \mathbb{R}$ is the fitness function that produces a scalar evaluation of a policy, only accessible via policy queries.

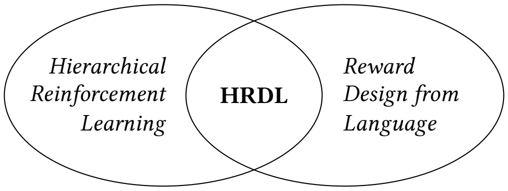


Figure 2: Although prior works have utilized multi-level rewards to train hierarchical agents, the *design* of such rewards remains underexplored and lacks a concrete problem formulation. This work bridges this gap by introducing HRDL.

The RDP seeks a reward function $r \in \mathcal{R}$ such that *the policy* $\pi := \mathcal{A}_{\mathcal{M}_p}(r)$ *that optimizes* r achieves the highest Fitness score $F(\pi)$. Rather than treating the reward as fixed and exogenous, RDP reframes reward design as a *search problem*, a perspective that has profoundly shaped subsequent research. It inspired methods that optimize or evolve reward functions directly [45, 62] as well as formulations that infer or generate them from indirect signals, such as Inverse Reward Design [19], which recovers true rewards from proxy rewards, and Eureka [38], which synthesizes executable rewards from natural language. Many other paradigms can be viewed as RDP instantiations: inverse RL [1, 16, 21, 75] treats expert behavior as evidence for the reward search, while preference-based learning [6, 53] and RLHF [5, 47] extend this to human feedback.

2.2.2 Rewards Design for Hierarchical RL. While RDP provides a unifying framework that has catalyzed advances in both RL and human-AI alignment, *it does not explicitly consider hierarchical RL.* Many prior works have explored the use of hierarchical rewards, from early studies in feudal reinforcement learning [13] and precursors to the options framework [59] to more recent advances in deep HRL [31, 42, 69]. While these works often *assume access* to hierarchical rewards for training agents to complete tasks, **the problem of *designing* such hierarchical rewards has received little attention and, to our knowledge, has yet to be formally defined as a concrete research problem**.

A literature search using the keywords "hierarchical reward design" primarily returns domain-specific studies that discuss how *using* hierarchical rewards enables solving application-level problems [10, 23, 43, 44]. Other works either employ the term *hierarchy* in different contexts [27, 33, 35, 64], such as to express the relative importance of multiple flat reward signals [35]. A related but conceptually distinct line of work utilizes Reward Machines (RMs) [17, 25], which represent rewards using automata or temporal logic. RM-based approaches focus on exploiting known reward structures to improve *learning efficiency*, rather than on studying how such structures can be specified from human input.

Since no previous work formally defines the *design of hierarchical rewards* as a general problem, there is a lack of consistent language and theoretical foundation for studying it. This work addresses this gap at the intersection of hierarchical rewards and reward design (Figure 2) by formalizing the Hierarchical Reward Design problem and introducing a reward structure that is hierarchical, compact, and capable of capturing nuanced behavioral specifications for both *what* subtasks to select and *how* to execute them.

Just as RDP lays the groundwork for studying algorithmic reward design in flat settings, we posit that HRD will provide a principled foundation for reasoning about hierarchical reward structures. In line with the research that originated from RDP, we anticipate that HRD will enable a broad range of problem instantiations (of reward design with different types of human inputs) and solution methods for designing hierarchical rewards.

2.2.3 Reward Design from Language. Early work addressing the RDP focused on designing rewards for intrinsic motivation and reward shaping [61–63]. More recently, research in this area has explored aligning agent behavior more closely with human-provided specifications, using learning or large language models (LLMs) to infer and generate reward functions [19, 32]. In these cases, the human or an oracle, either implicitly or explicitly, serves as the fitness function by evaluating the policy. However, most existing work on reward design or generation focuses exclusively on non-hierarchical (flat) RL settings, producing reward functions of the form $r_{flat}(s, a)$ or $r_{flat}(s)$ [7, 15, 19, 20, 32, 34, 38, 61–63, 72, 73]. While sufficient for certain behaviors, *flat reward functions are fundamentally limited when specifying complex preferences, such as desired subtask sequences or option-conditioned execution strategies, that naturally arise in long-horizon tasks.*

To our knowledge, the only prior work that explicitly considers a hierarchical setting is [39], though its focus differs substantially from ours. Their approach does not formalize the hierarchical reward design problem or analyze the expressivity gap between flat and hierarchical formulations. Moreover, the rewards generated by their LLMs are limited to task completion objectives and do not capture behavioral specifications. In contrast, L2HR generates both high- and low-level rewards that encode natural language behavioral preferences while preserving task feasibility.[1]

3 Hierarchical Reward Design

This section formally introduces the Hierarchical Reward Design problem. We begin by defining the low- and high-level reward functions in HRD and proceed to show that they naturally induce a family of MDPs at the low-level and a semi-MDP (SMDP) at the high-level. Using these insights, we formally define the general HRD problem along with a specific instantiation, the *Hierarchical Reward Design from Language (HRDL)* problem.[2]

3.1 Low-level and High-level Reward Models

Definition 1 (Low-level Reward). *The* ***low-level reward*** *is a function* $r_L : \mathcal{S} \times \mathcal{O} \times \mathcal{A} \to \mathbb{R}$, *which provides feedback for selecting a low-level action* $a \in \mathcal{A}$ *in state* $s \in \mathcal{S}$ *while pursuing option* $o \in \mathcal{O}$.

Intuitively, $r_L(s, o, a)$ encodes specifications for *how* the agent should execute the subtask associated with option o in state s.

Proposition 1 (Low-level MDP Models). *Let* $\mathcal{M}_p = (\mathcal{S}, \mathcal{A}, T, \gamma)$ *be a world model,* $\mathcal{O}$ *a set of options, and* $r_L : \mathcal{S} \times \mathcal{O} \times \mathcal{A} \to \mathbb{R}$ *a low-level reward. For a fixed option* $o \in \mathcal{O}$, *the tuple* $\mathcal{M}_{L,o} = (\mathcal{S}, \mathcal{A}, T, r_L(\cdot, o, \cdot), \gamma, h_o)$ *defines an MDP, where* h_o *is the horizon determined by the option's termination condition* $\beta(\cdot, o)$.

---

[1] Please see the Appendix for additional details on related works.

[2] Proofs for all propositions are provided in the Appendix, included in the extended version of this paper and available at http://tiny.cc/hrdl-appendix

Definition 2 (High-level Reward). *The* ***high-level reward*** *is a function* $r_H : \mathcal{O} \times \mathcal{S} \times \mathcal{O} \rightarrow \mathbb{R}$*, which specifies the expected reward for executing option* $o \in \mathcal{O}$ *until termination, given that* o *is initiated in state* $s \in \mathcal{S}$ *and the previous option was* $o^- \in \mathcal{O}$*.*

The high-level reward $r_H(o^-, s, o)$ encodes specifications for *what* subtask should be executed, possibly conditioned on both the current state and prior option. This allows for expressing preferences over subtask *ordering* and dependencies between subtasks.

Proposition 2 (High-level SMDP Model). *Let* $\mathcal{M}_p = (\mathcal{S}, \mathcal{A}, T, \gamma, h)$ *be a world model,* $\mathcal{O}$ *a set of options, and* $r_H : \mathcal{O} \times \mathcal{S} \times \mathcal{O} \rightarrow \mathbb{R}$ *the high-level reward. Then,* $\mathcal{M}_H = (\mathcal{O} \times \mathcal{S}, \mathcal{O}, T_H, r_H, \gamma, h)$ *forms a semi-MDP, where* $T_H : \mathcal{O} \times \mathcal{S} \times \mathcal{O} \rightarrow \Delta(\mathcal{O} \times \mathcal{S} \times \mathbb{N})$ *defines the joint distribution over the next augmented state and transit time, where* $\mathbb{N}$ *is the set of natural numbers.*

Alternatively, the high-level process can be modeled as a standard MDP when *single-step* high-level rewards are used. This flexibility highlights that the HRD framework is compatible with both semi-MDP and MDP formulations, allowing the use of a wide range of existing RL algorithms. We provide the formal MDP definition and corresponding proof in the Appendix of the extended paper.

3.2 The HRD Problem

Definition 3 (Hierarchical Reward Design (HRD)). *The* ***Hierarchical Reward Design (HRD)*** *problem is defined by the tuple* $P = (\mathcal{M}_p, \mathcal{O}, \mathcal{R}, \mathcal{A}_{\mathcal{M}_p}, F)$*, where*

- $\mathcal{M}_p = (\mathcal{S}, \mathcal{A}, T, \gamma, h)$ *is the world model;*
- $\mathcal{O}$ *is a finite option set;*
- $\mathcal{R} = \mathcal{R}_H \times \mathcal{R}_L$ *is the space of candidate reward structures, where* $\mathcal{R}_H = \{r_H : \mathcal{O} \times \mathcal{S} \times \mathcal{O} \rightarrow \mathbb{R}\}$ *and* $\mathcal{R}_L = \{r_L : \mathcal{S} \times \mathcal{O} \times \mathcal{A} \rightarrow \mathbb{R}\}$*;*
- *the learning routine* $\mathcal{A}_{\mathcal{M}_p}(\cdot) : \mathcal{R} \rightarrow \Pi_H \times \Pi_L$ *maps each reward pair* (r_H, r_L) *to a hierarchical policy* (π_H, π_L)*, where* $\pi_H : \mathcal{O} \times \mathcal{S} \rightarrow \Delta(\mathcal{O})$ *optimizes* r_H *in the high-level decision making model* $\mathcal{M}_H$ *and* $\pi_L : \mathcal{S} \times \mathcal{O} \rightarrow \Delta(\mathcal{A})$ *optimizes* r_L *in each underlying MDP* $\mathcal{M}_{L,o}$*; and*
- *the fitness function* $F : \Pi_H \times \Pi_L \rightarrow \mathbb{R}$ *evaluates the quality of hierarchical policies.*

The goal of HRD is to find $(r_H^*, r_L^*) = \arg\max_{(r_H, r_L) \in \mathcal{R}} F(\mathcal{A}_{\mathcal{M}_p}(r_H, r_L))$*.*

Connections to Existing Algorithms. We show in the Appendix that $\mathcal{A}_{\mathcal{M}_p}$ can be instantiated with existing RL algorithms. In our implementation, the low-level policy $\pi_L(a \mid s, o)$ is trained with PPO [54] due to its robustness in control, while the high-level policy $\pi_H(o \mid o^-, s)$ uses DQN-style methods [41], following common practice in SMDP formulations [4, 66]. Stronger structural assumptions on (r_H, r_L) can enable the use of more specialized HRL algorithms. For instance, if the low-level reward depends only on state and action, $r_L(s, o, a) = r_{flat}(s, a)$, and the high-level reward r_H is a single-step reward constructed as $\sum_a r_{flat}(s, a)\pi_L(a|s, o)$, the problem reduces to the two augmented MDPs formulation introduced in [74]. In these cases, algorithms such as double actor-critic [74] and option-critic [4] can be applied to learn hierarchical policies. A detailed exploration of the connections between structural reward assumptions and the applicability of existing HRL algorithms for instantiating $\mathcal{A}_{\mathcal{M}_p}$ is left for future work.

3.3 Hierarchical Reward Design from Language

As discussed in Sec. 1, real-world deployments often require agents to satisfy additional behavioral specifications beyond task completion. In these cases, the task reward can typically be defined once and reused across different behavioral contexts. In contrast, additional rewards must be redesigned for each new behavior specification. While the cost of task reward design is amortized, the cost of designing rewards that match human specifications grows linearly with the number of distinct behaviors desired. This motivates the need for an automated approach to generate rewards to encode behavioral specifications while reusing the existing domain dynamics and task objectives. The challenge of this problem is twofold: (1) The generated rewards should have distinct functional forms – one guiding high-level option selection, and another governing low-level action execution. (2) The generated rewards must remain compatible with existing task rewards, ensuring that agents continue to achieve the original task objectives. We formally define this as a specific instantiation of the HRD problem.

Definition 4 (Hierarchical Reward Design from Language (HRDL)). *The* ***HRDL*** *problem is an instance of the HRD problem with additional inputs: (1) a task reward function* $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$*, (2) a subtask completion reward (pseudo-reward)* $r_p : \mathcal{S} \times \mathcal{O} \times \mathcal{A} \rightarrow \mathbb{R}$*, and (3) behavior specifications* $l \in \Sigma^*$*, provided as a natural language description.* l *guides the reward generation during training, and the fitness function* F *is accessible* ***only*** *during evaluation. The objective of HRDL is to generate high- and low-level designed rewards,* $R^* = (\tilde{r}_H^*, \tilde{r}_L^*) \in \mathcal{R}$*, such that the resulting hierarchical policy* (π_H^*, π_L^*)*, trained under the composite rewards* $(r_{opt} + \tilde{r}_H^*, r_p + \tilde{r}_L^*)$ *using* $\mathcal{A}_{\mathcal{M}_p}$*, maximizes the fitness score:* $(\tilde{r}_H^*, \tilde{r}_L^*) = \arg\max_{(\tilde{r}_H, \tilde{r}_L) \in \mathcal{R}} F(\mathcal{A}_{\mathcal{M}_p}(r_{opt} + \tilde{r}_H, r_p + \tilde{r}_L))$*.*

If a *non-hierarchical* reward design method is used, the designed reward has the flat form $\tilde{r}_{flat}(s, a)$. To integrate this flat reward into the hierarchical setting, we must decompose it into high- and low-level rewards:

$$r_L(s, o, a) = r_p(s, o, a) + \tilde{r}_{flat}(s, a) \tag{2}$$

$$r_H(s, o) = r_{opt}(s, o) + \tilde{r}_{flat,H}(s, o) \tag{3}$$

where $\tilde{r}_{flat,H}(s, o)$ aggregates $\tilde{r}_{flat}(s, a)$ using the same expression as Eq. 1. While flat designed rewards $\tilde{r}_{flat}(s, a)$ can encode some behavior specifications, the definitions of high- and low-level rewards in HRD provide a significantly more expressive mechanism for specifying agent behavior:

$$r_L(s, o, a) = r_p(s, o, a) + \tilde{r}_L(s, o, a) \tag{4}$$

$$r_H(o^-, s, o) = r_{opt}(s, o) + \tilde{r}_H(o^-, s, o) \tag{5}$$

In fact, the flat reward is a special case of hierarchically designed rewards, where $\tilde{r}_L(s, o, a) = \tilde{r}_{flat}(s, a)$ and $\tilde{r}_H(o^-, s, o) = \tilde{r}_{flat,H}(s, o)$. The hierarchical formulation is strictly more general than the flat formulation, offering greater expressiveness in the following ways:

Property 1. *Certain specifications on sub-task selection can be expressed through* $\tilde{r}_H(s, o^-, o)$*, but they cannot be expressed by flat function:* $\tilde{r}_{flat}(s, a)$*.*

Property 2. *Certain specifications on sub-task execution can be expressed through* $\tilde{r}_L(s, o, a)$*, but they cannot be expressed by flat function:* $\tilde{r}_{flat}(s, a)$*.*

Proofs of these properties are provided in the Appendix. In the following sections, we introduce an algorithm for generating hierarchical rewards $(\tilde{r}_H, \tilde{r}_L)$ from natural language specifications and empirically compare its performance against flat reward design that generates alignment rewards of the form $\tilde{r}_{flat}(s, a)$.

4 Language to Hierarchical Rewards (L2HR)

We now present L2HR, an algorithm designed to solve HRDL by leveraging large language models (LLMs). L2HR represents reward functions as programs, which it generates directly from natural language specifications. Specifically, it first employs a structured prompting strategy to capture specifications as prompts, which are then used as inputs to a training module that produces hierarchical reward functions. An illustration of L2HR's input and output is provided in Fig. 3, with more details available in the Appendix.

4.1 Prompting Strategy

To generate executable reward functions from natural language, L2HR requires the specifications to be provided as a structured prompt. We design a prompting strategy for L2HR inspired by Eureka, a recent approach for LLM-based reward generation [38]. However, unlike Eureka, L2HR's prompt design focuses on producing feasible reward functions without relying on access to a fitness F during reward code generation. Specifically, within L2HR, the LLM is provided with a structured prompt comprising:

(1) *Task Description*: A natural language description of the overall task objective, including the approximate scale of the task reward. Since RL is sensitive to reward magnitudes, this information helps ensure that the generated preference rewards are neither too small to influence learning nor so large as to destabilize training. The task reward code itself is intentionally withheld to reflect realistic settings in which only the reward signal (but not the code) is accessible and to prevent overfitting to implementation details.
(2) *Environment Code Context*: Snippets of environment source code that expose the state and action spaces without revealing simulation internals, following the methodology of [38].
(3) *Relevant Action-Related Spaces*: Descriptions of the option space $\mathcal{O}$ and action space $\mathcal{A}$, including the semantic role of each. We include these descriptions to help the LLM correctly distinguish between high-level and low-level decision spaces.
(4) *Behavior Specification*: A natural language string that describes the desired agent behaviors beyond task completion.
(5) *Formatting and Reward Design Tips*: Coding constraints (e.g., avoiding global variables) and guidance on balancing designed rewards with the underlying task reward.

4.2 Training Procedure

Given the structured prompt, state-of-the-art LLMs can generate plausible reward code in a zero-shot manner. However, we find that zero-shot generation often yields reward code with syntax errors and invalid variable references in practice. To mitigate these issues, L2HR incorporates a training procedure for reward code generation that couples LLMs with RL. Specifically, L2HR generates k reward candidates independently from the LLM and apply a lightweight filtering process to ensure validity. During filtering, L2HR verifies whether the code compiles without syntax errors, and whether it references only permitted state, option, and action variables exposed in the environment prompt. We find that at least one sample in the batch passes these checks. As a result, we forego more complex iterative refinement strategies, such as evolutionary search or reward reflection [38], leaving their integration into L2HR as a direction for future work.

Task description: The objective is to pick up all apples and eggs on the dining table and place them in the sink...
Environment context:

```
'''
Background: PnP_LL_Actions = [...], PnP_HL_Actions = [...] ...
'''
class ThorPickPlaceEnv(gym.Env):
    def __init__(...): ...
```

Relevant task spaces: The agent's option/subtask space consists of picking up and placing the two types of objects...
Low-level user preference: The agent should avoid the stool while on its way to pick up or drop an egg...
High-level user preference: The agent should pick up an object type that is different from what was previously picked...
Additional info: Do not use function attributes or global variables...

(a) Natural Language Specifications

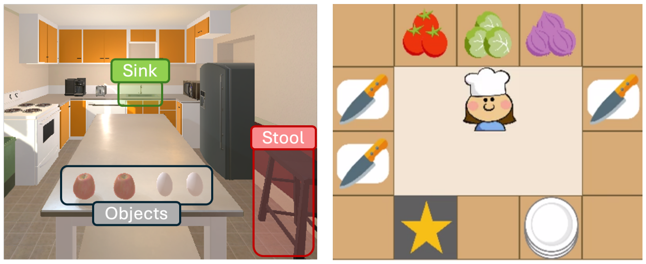
```
# High-level preference reward
def get_high_level_pref_gpt(state: Dict, prev_option: int, option: int) ->
    Tuple[float, Dict[str, float]]:
    ...
    return reward, reward_components

# Low-level preference reward
def get_low_level_pref_gpt(state: Dict, option: int, action: int) ->
    Tuple[float, Dict[str, float]]:
    ...
    return reward, reward_components
```

(b) Reward Functions designed by L2HR

Figure 3: Illustration of L2HR input and output.

The full two-stage training procedure of **L2HR** is provided in the Appendix. In the first stage, L2HR uses the LLM to generate k candidate low-level alignment functions $\tilde{r}_L^{(1)}, \ldots, \tilde{r}_L^{(k)}$ from the specification l. The low-level prompt differs from the high-level one only in its description of the action space and level-specific behavior specifications. Each $\tilde{r}_L^{(i)}$ is then used to train a corresponding low-level policy $\pi_L^{(i)}$ with the combined objective of pseudo-rewards plus the LLM-generated $\tilde{r}_L^{(i)}$. Only the policies $\pi_L^{(i)}$ that surpass a predefined threshold of subgoal completion, based on cumulative pseudo-rewards, are considered as *valid*.

In the second stage, L2HR generates k candidate high-level alignment functions $\tilde{r}_H^{(1)}, \ldots, \tilde{r}_H^{(k)}$ and uses them to train high-level policies $\pi_H^{(j)}$, each conditioned on a *valid* $\pi_L^{(i)}$ from the first stage. These policies are optimized using with the combined objective of option-level task reward plus the LLM-generated $\tilde{r}_H^{(j)}$. Then, L2HR returns all designed rewards $(\tilde{r}_H^{(j)}, \tilde{r}_L^{(i)})$ and corresponding trained policy pairs $(\pi_H^{(j)}, \pi_L^{(i)})$ that achieve cumulative task rewards above a threshold. This two-stage structure promotes modularity and allows for selective reuse of subtask policies.



(a) iTHOR Domain **(b) Kitchen Domain**

Figure 4: Rendered scenes from the iTHOR and Kitchen domains. Fig. 1 (right) illustrates the Rescue World domain.

5 Experiments

We empirically evaluate whether framing reward generation as a hierarchical problem offers advantages over the traditional flat (non-hierarchical) formulation. Specifically, we aim to evaluate:

Q1. Are the generated alignment rewards syntactically correct?
Q2. Do they preserve task feasibility?
Q3. Do they successfully induce behaviors that match the provided specifications?

To investigate **Q3**, we additionally conduct user studies. The Appendix includes further experimental details, including more information about the domains, metrics, LLM prompts, hyperparameters, and user study protocol.

5.1 Baselines

To evaluate L2HR (referred to as *Hier* in the tables), we consider the following baselines. For all LLM-based experiments, we generate $k = 8$ reward function candidates per trial using GPT-4o [24] and repeat this process 3 times, resulting in a total of 24 reward candidates per configuration.

Language to Flat Reward (Flat). This baseline generates flat alignment reward $\tilde{r}_{flat}(s, a)$ from language and incorporates it into the hierarchical setting using Eq. 4 and 5 to maintain a consistent two-level training framework. Prompts are identical to those used for L2HR, except that the prompts for flat reward generation exclude option-related instructions (since flat rewards are independent of options) and express user preferences as a single combined description rather than separate high- and low-level specifications.

No Reward Design (Task). This baseline does not utilize LLMs and uses only the task reward r and pseudo-reward r_p without incorporating any behavioral specifications (i.e., $\tilde{r}_H = \tilde{r}_L = 0$). As this setting is less noisy, we run it for 5 trials.

5.2 Domains

We evaluate the approaches on three long-horizon, multi-subtask domains: Rescue World, iTHOR, and Kitchen. Each domain poses distinct challenges in subtask sequencing and low-level execution. Furthermore, as we see later, flat rewards cannot in principle capture all specifications in Rescue World and iTHOR, whereas they can in Kitchen. This contrast allows us to investigate, *in practice*, how effectively language specifications can be translated into flat and hierarchical rewards across both types of scenarios. Additional details are provided in the Appendix and Fig. 4.

Rescue World. This is a variant of the Rescue World for Teams (RW4T) domain [46], where the agent must collect and deliver all supplies in the environment. This domain features a large state space represented by a 407-dimensional vector and poses a long-horizon challenge, requiring the agent to complete 8 subtasks, each lasting up to around 10 steps. Behavioral specifications include: (1) a high-level *persistence* specification for delivering all supplies of one type before switching to another and (2) a low-level *safety* specification for avoiding hazardous zones while carrying supplies.

iTHOR. Built upon the Unity game engine, iTHOR is an environment within the AI2-THOR [30] framework that features several realistic household scenes in which an agent navigates and interact with everyday objects. Here, we focus on a long-horizon pick and place task within a kitchen setting consisting of 8 subtasks, each requiring up to approximately 30 steps to complete. The agent must deliver a set of apples and eggs located on the dining table to the sink on the other side of the room. The state space is represented by a 30-dimensional vector that contains object and agent positions and object states. Behavioral specifications include: (1) a high-level *diversity* specification that requires delivering a different item from the one previously delivered and (2) a low-level *avoidance* specification that prevents the agent from going near a stool placed in the environment while picking up or delivering an egg.

Kitchen. This is a single-agent variant of Overcooked, an environment originally developed for studying human-AI collaboration in kitchen tasks [37, 71]. In our setting, the agent needs to prepare a salad with lettuce, tomatoes, and onions. This domain features an even larger state space, represented by a 699-dimensional continuous vector that captures various ingredient states. It also involves a long-horizon task requiring the completion of 5 subtasks in a strict sequence, with the final 2 subtasks dependent on the successful completion of all preceding ones. The high-level behavioral specification is a preferred *chopping* sequence (e.g., tomatoes → onions → lettuce). Since the environment uses fixed low-level policies, we skip low-level reward design in this domain.

5.3 Numerical Experiments

As a precursor to evaluting solutions to HRDL, we also conducted experiments where hand-crafted flat and hierarchical rewards were used directly to train policies *without requiring reward generation from language specifications*. These experiments serve as a proof-of-concept and demonstrate that:

- given expert-specified hierarchical rewards $(\tilde{r}^*_H, \tilde{r}^*_L)$, existing RL algorithms can effectively learn hierarchical policies (π^*_H, π^*_L) that achieve high task performance and strong alignment with designer specifications; and
- while expert-specified flat rewards $\tilde{r}^*_{flat}$ can capture some behavioral specifications, they fail to express ones that require knowledge of the previous subtask (e.g., the *persistence* specification in Rescue World and the *diversity* specification in iTHOR).

We report these preliminary results in the Appendix. Now, we return to the core HRDL setting, where designed rewards must be synthesized from natural language inputs.

Table 1: Table showing the performance of policies trained with the task reward alone or combined with LLM-generated flat or hierarchical rewards. For each metric, we report both the cumulative reward returns and the percentage of policies at expert-level alignment (attaining the maximum possible cumulative return for that metric). Means and standard deviations are computed over all runs for the *Task* baseline, and only over the LLM-generated reward candidates that successfully complete the task for *Flat* and *Hier*.

Domain	Method	High-Level		Low-Level		Total	
		Rewards ↑	Expert % ↑	Rewards ↑	Expert % ↑	Rewards ↑	Expert % ↑
Rescue	Task	11.22 ± 5.57	20.00	-16.46 ± 5.49	0.00	73.80 ± 5.70	0.00
	Flat	9.38 ± 7.02	12.50	-2.62 ± 5.19	62.50	85.13 ± 9.33	12.50
	Hier	**16.65 ± 6.93**	**76.92**	**-0.69 ± 1.58**	**76.92**	**93.98 ± 9.01**	**69.23**
iTHOR	Task	4.10 ± 1.34	0.00	-23.38 ± 1.86	0.00	12.31 ± 0.66	0.00
	Flat	7.67 ± 4.48	0.00	-35.20 ± 12.42	0.00	3.27 ± 9.31	0.00
	Hier	**14.19 ± 2.23**	**87.50**	**-3.75 ± 8.14**	**75.00**	**37.68 ± 6.68**	**62.50**
Kitchen	Task	0.00 ± 0.00	0.00	–	–	0.75 ± 0.00	0.00
	Flat	0.06 ± 0.13	10.00	–	–	0.80 ± 0.12	10.00
	Hier	**0.39 ± 0.05**	**92.86**	–	–	**1.08 ± 0.05**	**92.86**

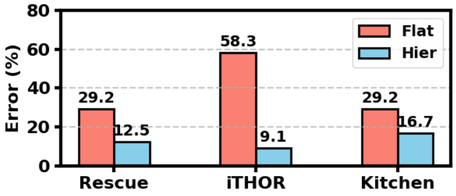


Figure 5: Syntax error rates for LLM-based reward generation, computed over 24 candidates per configuration.

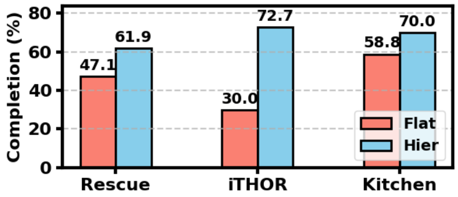


Figure 6: Task completion rates for LLM-generated rewards, calculated as the proportion of designed rewards that preserve task feasibility among syntactically valid candidates.

Q1. Are the designed rewards syntactically correct? Figure 5 shows that hierarchical reward generation achieves substantially lower code generation error rates than flat reward generation in all three domains, *suggesting that formulating the HRL reward design problem hierarchically can simplify reward synthesis for LLMs.* In Rescue World and iTHOR, flat rewards cannot capture the high-level specifications, as doing so requires access to the previously executed option. As a result, the LLM often hallucinates unavailable variables related to previously delivered object types. In Kitchen, higher error rates stem from the complexity of reasoning over low-level actions. For example, correctly checking if the agent is chopping an onion on the low-level requires inspecting coordinate-level state variables. In contrast, having access to the options space in HRD enables direct reasoning over high-level behaviors (e.g., `chop onion`), greatly simplifying reward generation.

Q2. Do the designed rewards preserve task feasibility? *Hier* consistently better preserves task feasibility than *Flat* across all domains (Figure 6), which is essential for real-world deployment. In Rescue World and iTHOR, flat rewards often rely on spurious heuristics to infer past behavior (e.g. inferring the last delivered item type from the agent's current location), leading to unintended behaviors. In iTHOR, flat rewards tend to misidentify the type of objects being picked up, resulting in incorrect reward accumulation. In Kitchen, flat rewards frequently struggle to reason over low-level actions (e.g., which ingredient the agent is interacting with), producing alignment rewards that interfere with task completion. Hierarchical rewards, by contrast, avoid these issues by conditioning on temporally extended options rather than raw state transitions.

Q3. Do the designed rewards actually lead to agent behavior that match the behavioral specifications? Table 1 summarizes alignment performance using handcrafted high- and low-level ground-truth rewards ($\tilde{r}_H, \tilde{r}_L$). The *Total* metric combines task and alignment rewards and serves as a proxy for the overall fitness F. In Rescue World, *Hier* substantially outperforms both *Task* and *Flat* baselines on high-level alignment, achieving expert-level performance in 76.92% of successful runs. This reflects its ability to encode the *persistence* specification, which flat rewards fundamentally cannot represent. While *Flat* occasionally (12.50%) attains expert-level high-level alignment, these instances are coincidental. Both methods perform comparably on low-level alignment, as the agent's carrying status can be inferred from observable states without explicit option conditioning. Overall, *Hier* achieves the highest total return, with 69.23% of policies attaining expert-level alignment.

In iTHOR, *Hier* again outperforms *Flat* on high-level alignment, as the *diversity* specification cannot be fully captured by the flat reward without access to the agent's current option. In this case, *Hier* also significantly outperforms *Flat* on low-level alignment as well. Although the LLM is explicitly asked to penalize the agent's

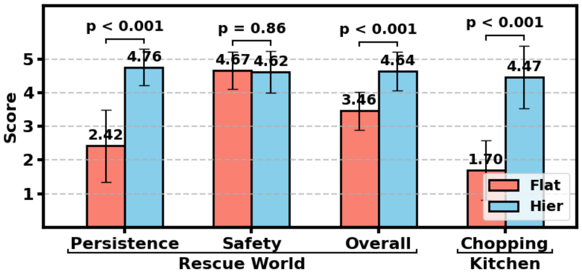


Figure 7: Human-provided ratings for agent alignment.

proximity to the stool when *on its way* to pick up or deliver, the generated flat rewards only apply the penalty in the timestep that the agent is specifically performing the pick or place action, leading the agent to not avoid the stool. This makes sense, as it is difficult to discern the agent's intent in picking up or dropping an egg from just the state without option information.

In Kitchen, *Hier* achieves substantially higher alignment with the chopping specification (92.86% vs. 10.00%). Although flat rewards are theoretically capable of encoding this behavior, doing so requires complex and brittle logic: only 1 flat reward candidate successfully implemented the specification. This demonstrates a key advantage of designing rewards for HRL with HRD: even when flat rewards are theoretically sufficient, hierarchical rewards can simplify reward design through high-level abstractions and lead to better alignment with behavioral specifications. Example videos of policies for all domains are provided in the supplementary material.

5.4 Evaluations with Human Participants

In real-world applications, manually designed ground-truth rewards are rarely available. To better reflect practical deployment scenarios, we conducted an IRB-approved user study on Rescue World and Kitchen using human participants recruited via Prolific. The goal was to assess whether *Hier* agents are perceived as better aligned with behavioral specifications than *Flat* agents.

In this study, non-expert participants effectively served the role of fitness function F, providing human-centered evaluations. Participants viewed videos of agent behaviors produced by both methods and rated their alignment with textual specifications on a scale from 1 (least aligned) to 5 (most aligned), similar to the scale employed in the evaluation methodology of [32]. Participants were not aware of the underlying reward design methods of the policies. We collected usable responses (e.g., those that passed attention checks) from 30 participants, evenly split across the two domains. Further details of the study design are provided in the Appendix.

As shown in Fig. 7, participants consistently rated *Hier* agents higher than *Flat* for high-level alignment. In Rescue World, *Hier* significantly outperforms *Flat* on the *persistence* preference (4.76 vs. 2.42), a specification that flat rewards struggle to capture due to the lack of previous option information. While both methods have similar ratings for the low-level *safety* specification, *Hier* achieves significantly higher *overall* alignment scores (4.64 vs. 3.46).

In Kitchen, the advantage of *Hier* is even more pronounced for the *chopping* specification (4.47 vs. 1.70). This larger gap arises because, unlike in Rescue World, the preferred behavior in Kitchen rarely occurs accidentally, as aligning with the *chopping* specification requires taking additional steps in the environment. Notably, across both domains, *Hier* policies consistently receive average ratings above 4, indicating a strong perception of alignment. These findings suggest that, in practice, when task completion is used to filter out unsuccessful policies, the remaining *Hier* candidates are consistently well-aligned with behavioral specifications.

Table 2: Candidate agent policies (%) that received a perfect alignment score from all human participants.

	Rescue World			Kitchen
Method	Persistence ↑	Safety ↑	Overall ↑	Chopping ↑
Flat	12.50%	50.00%	12.50%	10.00%
Hier	**76.92%**	**61.54%**	**53.85%**	**71.43%**

Table 2 shows that over half of the policies produced by *Hier* achieve perfect human ratings across all behavioral specifications, substantially outperforming those generated by *Flat*. Overall, *Hier* consistently outperforms *Flat* in capturing language-based behavioral specifications across domains in both simulated evaluations and user studies.

6 Conclusion

This paper introduces the **Hierarchical Reward Design (HRD) problem**, which (1) formulates a more expressive reward structure than flat rewards, (2) integrates seamlessly with existing decision-making frameworks and RL algorithms, and (3) better encodes behavioral specifications for long-horizon tasks, with an initial solution to this problem (**L2HR**) achieving considerably better or comparable results in both numerical and human evaluations.

While our results provide strong motivation for HRD, several limitations and interesting areas for future investigation remain. First, our experiments focus on complex but simulated domains; evaluating HRD in real-world settings, such as robotics and interactive AI systems, is an important next step. Second, more advanced reward generation techniques, including evolutionary optimization or human feedback, may further improve performance. Finally, since option discovery remains a core challenge in Hierarchical RL, jointly addressing option discovery and reward design could substantially enhance practical applicability.

Finally, *we emphasize that human-agent alignment is inherently challenging and HRD should not be viewed in isolation.* Rather, it complements a broader ecosystem of alignment approaches, including learning from demonstrations, rankings, and user corrections. Understanding how HRDL and L2HR interact with these approaches is an important avenue for future work, bringing us closer to AI agents that are reliably aligned with human needs, values and objectives.

ACKNOWLEDGMENTS

We thank the anonymous reviewers and the members of the Ken Kennedy Institute at Rice University for their constructive feedback. This research was supported in part by NSF award #2205454 and Rice University funds.

REFERENCES

[1] Pieter Abbeel and Andrew Y Ng. 2004. Apprenticeship learning via inverse reinforcement learning. In *International Conference on Machine Learning*. PMLR.

[2] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. GPT-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).

[3] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. 2016. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565* (2016).

[4] Pierre-Luc Bacon, Jean Harb, and Doina Precup. 2017. The option-critic architecture. In *AAAI Conference on Artificial Intelligence*, Vol. 31.

[5] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862* (2022).

[6] Chandrayee Basu, Mukesh Singhal, and Anca D Dragan. 2018. Learning from richer human guidance: Augmenting comparison-based learning with feature queries. In *ACM/IEEE International Conference on Human-Robot Interaction*. 132–140.

[7] Siddhant Bhambri, Amrita Bhattacharjee, Durgesh Kalwar, Lin Guan, Huan Liu, and Subbarao Kambhampati. 2024. Extracting Heuristics from Large Language Models for Reward Shaping in Reinforcement Learning. *arXiv preprint arXiv:2405.15194* (2024).

[8] Matthew M Botvinick, Yael Niv, and Andew G Barto. 2009. Hierarchically organized behavior and its neural foundations: A reinforcement learning perspective. *Cognition* 113, 3 (2009), 262–280.

[9] Richard Catrambone. 1998. The subgoal learning model: Creating better examples so that students can solve novel problems. *Journal of Experimental Psychology: General* 127, 4 (1998), 355.

[10] Ruihai Chen, Hao Li, Guanwei Yan, Haojie Peng, and Qian Zhang. 2023. Hierarchical reinforcement learning framework in geographic coordination for air combat tactical pursuit. *Entropy* 25, 10 (2023), 1409.

[11] Sonia Chernova and Andrea L Thomaz. 2022. *Robot learning from human teachers.* Springer Nature.

[12] Carlos G Correa, Mark K Ho, Frederick Callaway, Nathaniel D Daw, and Thomas L Griffiths. 2023. Humans decompose tasks by trading off utility and computational cost. *PLoS Computational Biology* 19, 6 (2023), e1011087.

[13] Peter Dayan and Geoffrey E Hinton. 1992. Feudal reinforcement learning. *Advances in Neural Information Processing Systems* 5 (1992).

[14] Thomas G Dietterich. 2000. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research* 13 (2000), 227–303.

[15] Yuqing Du, Olivia Watkins, Zihan Wang, Cédric Colas, Trevor Darrell, Pieter Abbeel, Abhishek Gupta, and Jacob Andreas. 2023. Guiding pretraining in reinforcement learning with large language models. In *International Conference on Machine Learning*. PMLR, 8657–8677.

[16] Justin Fu, Katie Luo, and Sergey Levine. 2018. Learning Robust Rewards with Adverserial Inverse Reinforcement Learning. In *International Conference on Learning Representations*.

[17] Daniel Furelos-Blanco, Mark Law, Anders Jonsson, Krysia Broda, and Alessandra Russo. 2023. Hierarchies of reward machines. In *International Conference on Machine Learning*. PMLR, 10494–10541.

[18] Juan Angel Gonzalez-Aguirre, Ricardo Osorio-Oliveros, Karen L Rodriguez-Hernandez, Javier Lizárraga-Iturralde, Ruben Morales Menendez, Ricardo A Ramirez-Mendoza, Mauricio Adolfo Ramirez-Moreno, and Jorge de Jesus Lozoya-Santos. 2021. Service robots: Trends and technology. *Applied Sciences* 11, 22 (2021), 10702.

[19] Dylan Hadfield-Menell, Smitha Milli, Pieter Abbeel, Stuart J Russell, and Anca Dragan. 2017. Inverse reward design. *Advances in Neural Information Processing Systems* 30 (2017).

[20] Xu Han, Qiannan Yang, Xianda Chen, Zhenghan Cai, Xiaowen Chu, and Meixin Zhu. 2024. Autoreward: Closed-loop reward design with large language models for autonomous driving. *IEEE Transactions on Intelligent Vehicles* (2024).

[21] Jonathan Ho and Stefano Ermon. 2016. Generative adversarial imitation learning. *Advances in Neural Information Processing Systems* 29 (2016).

[22] Jie Huang and Kevin Chen-Chuan Chang. 2023. Towards reasoning in large language models: A survey. In *Findings of the Association for Computational Linguistics*. 1049–1065.

[23] Xiaohui Huang, Jiahao Ling, Xiaofei Yang, Xiong Zhang, and Kaiming Yang. 2023. Multi-agent mix hierarchical deep reinforcement learning for large-scale fleet management. *IEEE Transactions on Intelligent Transportation Systems* 24, 12 (2023), 14294–14305.

[24] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. 2024. GPT-4o System Card. *arXiv preprint arXiv:2410.21276* (2024).

[25] Rodrigo Toro Icarte, Toryn Q Klassen, Richard Valenzano, and Sheila A McIlraith. 2022. Reward machines: Exploiting reward function structure in reinforcement learning. *Journal of Artificial Intelligence Research* 73 (2022), 173–208.

[26] Yiding Jiang, Shixiang Shane Gu, Kevin P Murphy, and Chelsea Finn. 2019. Language as an abstraction for hierarchical deep reinforcement learning. *Advances in Neural Information Processing Systems* 32 (2019).

[27] Kexin Jin, Guohui Tian, Bin Huang, Yongcheng Cui, and Xiaoyu Zheng. 2024. Reward Design Framework Based on Reward Components and Large Language Models. In *International Conference on Artificial Intelligence, Robotics, and Communication (ICAIRC)*. IEEE, 278–282.

[28] Cory D Kidd and Cynthia Breazeal. 2008. Robots at home: Understanding long-term human-robot interaction. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 3230–3235.

[29] Juho Kim, Robert C Miller, and Krzysztof Z Gajos. 2013. Learnersourcing subgoal labeling to support learning from how-to videos. In *CHI'13 Extended Abstracts on Human Factors in Computing Systems*. 685–690.

[30] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Matt Deitke, Kiana Ehsani, Daniel Gordon, Yuke Zhu, et al. 2017. AI2-THOR: An interactive 3d environment for visual AI. *arXiv preprint arXiv:1712.05474* (2017).

[31] Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. 2016. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Advances in Neural Information Processing Systems* 29 (2016).

[32] Minae Kwon, Sang Michael Xie, Kalesha Bullard, and Dorsa Sadigh. 2023. Reward design with language models. In *International Conference on Learning Representations*.

[33] Yuhang Lai, Siyuan Wang, Shujun Liu, Xuan-Jing Huang, and Zhongyu Wei. 2024. Alarm: Align language models via hierarchical rewards modeling. In *Findings of the Association for Computational Linguistics*. 7817–7831.

[34] Hao Li, Xue Yang, Zhaokai Wang, Xizhou Zhu, Jie Zhou, Yu Qiao, Xiaogang Wang, Hongsheng Li, Lewei Lu, and Jifeng Dai. 2024. Auto MC-Reward: Automated dense reward design with large language models for minecraft. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 16426–16435.

[35] Meng Li, Zhibin Li, Bingtong Wang, and Shunchao Wang. 2024. A Bounded Rationality-Aware Car-Following Strategy for Alleviating Cut-In Events and Traffic Disturbances in Traffic Oscillations. *IEEE Transactions on Intelligent Transportation Systems* (2024).

[36] Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. 2022. Competition-level code generation with AlphaCode. *Science* 378, 6624 (2022), 1092–1097.

[37] Jijia Liu, Chao Yu, Jiaxuan Gao, Yuqing Xie, Qingmin Liao, Yi Wu, and Yu Wang. 2024. LLM-Powered Hierarchical Language Agent for Real-time Human-AI Coordination. In *International Conference on Autonomous Agents and Multiagent Systems*. 1219–1228.

[38] Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2024. Eureka: Human-level reward design via coding large language models. In *International Conference on Learning Representations*.

[39] Shinya Masadome and Taku Harada. 2025. Reward Design Using Large Language Models for Natural Language Explanation of Reinforcement Learning Agent Actions. *IEEJ Transactions on Electrical and Electronic Engineering* (2025).

[40] George A Miller, Galanter Eugene, and Karl H Pribram. 2017. Plans and the Structure of Behaviour. In *Systems Research for Behavioral Science*. Routledge, 369–382.

[41] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529–533.

[42] Ofir Nachum, Shixiang Shane Gu, Honglak Lee, and Sergey Levine. 2018. Data-efficient hierarchical reinforcement learning. *Advances in Neural Information Processing Systems* 31 (2018).

[43] Yusei Naito, Tomohiko Jimbo, Tadashi Odashima, and Takamitsu Matsubara. 2025. Task-priority Intermediated Hierarchical Distributed Policies: Reinforcement Learning of Adaptive Multi-robot Cooperative Transport. In *2025 IEEE/SICE International Symposium on System Integration (SII)*. IEEE, 1556–1562.

[44] Charles Newton, Christopher Ballinger, Michael Sloma, and Keith Brawner. 2022. Hierarchical, Discontinuous Agent Reinforcement Learning Rewards in Complex Military-Oriented Environments. In *The International FLAIRS Conference Proceedings*, Vol. 35.

[45] Scott Niekum, Andrew G Barto, and Lee Spector. 2010. Genetic programming for reward function search. *IEEE Transactions on Autonomous Mental Development* 2, 2 (2010), 83–90.

[46] Liubove Orlov Savko, Zhiqin Qian, Gregory Gremillion, Catherine Neubauer, Jonroy Canady, Vaibhav Unhelkar, and Catherine Neubauer. 2024. RW4T Dataset: Data of Human-Robot Behavior and Cognitive States in Simulated Disaster Response Tasks. In *ACM/IEEE International Conference on Human-Robot Interaction*. 924–928.

[47] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems* 35 (2022), 27730–27744.

[48] Caroline Pantofaru, Leila Takayama, Tully Foote, and Bianca Soto. 2012. Exploring the role of robots in home organization. In *ACM/IEEE International Conference on Human-Robot Interaction*. 327–334.

[49] Ronald Parr and Stuart Russell. 1997. Reinforcement learning with hierarchies of machines. *Advances in Neural Information Processing Systems* 10 (1997).

[50] Martin L Puterman. 2014. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

[51] Peizhu Qian, Filip Bajraktari, Carlos Quintero-Peña, Qingxi Meng, Shannan Hamlin, Lydia Kavraki, and Vaibhav Unhelkar. 2025. ASTRID: A Robotic Tutor for Nurse Training to Reduce Healthcare-Associated Infections. In *Robotics: Science and Systems*. Los Angeles, CA, USA.

[52] Preeti Ramaraj. 2023. *Analysis of Situated Interactive Non-Expert Instruction of A Hierarchical Task to a Learning Robot*. Ph.D. Dissertation. University of Michigan.

[53] Dorsa Sadigh, Anca Dragan, Shankar Sastry, and Sanjit Seshia. 2017. Active Preference-Based Learning of Reward Functions. In *Robotics: Science and Systems*. Cambridge, Massachusetts.

[54] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[55] Massimiliano Scopelliti, Maria Vittoria Giuliani, Alexandra M D'amico, and Ferdinando Fornara. 2004. If I had a robot at home... Peoples' representation of domestic robots. In *Designing A More Inclusive World*. Springer, 257–266.

[56] Sangwon Seo, Bing Han, Rayan Harari, Roger Dias, Marco Zenati, Eduardo Salas, and Vaibhav Unhelkar. 2025. Socratic: Enhancing Human Teamwork via AI-enabled Coaching. In *International Conference on Autonomous Agents and Multi-Agent Systems*.

[57] Sangwon Seo and Vaibhav Unhelkar. 2024. IDIL: Imitation Learning of Intent-Driven Expert Behavior. In *International Conference on Autonomous Agents and Multi-Agent Systems*.

[58] Sangwon Seo and Vaibhav Unhelkar. 2025. Hierarchical Imitation Learning of Team Behavior from Heterogeneous Demonstrations. In *International Conference on Autonomous Agents and Multi-Agent Systems*.

[59] Satinder Singh. 1992. Transfer of learning by composing solutions of elemental sequential tasks. *Machine Learning* 8 (1992), 323–339.

[60] Satinder Singh, Richard L Lewis, and Andrew G Barto. 2009. Where do rewards come from. In *Annual Conference of the Cognitive Science Society*. Cognitive Science Society, 2601–2606.

[61] Satinder Singh, Richard L Lewis, Andrew G Barto, and Jonathan Sorg. 2010. Intrinsically motivated reinforcement learning: An evolutionary perspective. *IEEE Transactions on Autonomous Mental Development* 2, 2 (2010), 70–82.

[62] Jonathan Sorg, Richard L Lewis, and Satinder Singh. 2010. Reward design via online gradient ascent. *Advances in Neural Information Processing Systems* 23 (2010).

[63] Jonathan Sorg, Satinder P Singh, and Richard L Lewis. 2010. Internal rewards mitigate agent boundedness. In *International Conference on Machine Learning*. PMLR, 1007–1014.

[64] Liting Sun. 2019. *Intelligent and High-performance Behavior Design of Autonomous Systems via Learning, Optimization and Control*. University of California, Berkeley.

[65] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement Learning: An Introduction*. MIT press.

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

[67] Vaibhav V Unhelkar, Przemyslaw A Lasota, Quirin Tyroller, Rares-Darius Buhai, Laurie Marceau, Barbara Deml, and Julie A Shah. 2018. Human-Aware Robotic Assistant for Collaborative Assembly: Integrating Human Motion Prediction with Planning in Time. *IEEE Robotics and Automation Letters* 3, 3 (2018), 2394–2401.

[68] Vaibhav V Unhelkar, Shen Li, and Julie A Shah. 2019. Semi-Supervised Learning of Decision-Making Models for Human-Robot Collaboration. In *Conference on Robot Learning*.

[69] Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. 2017. Feudal networks for hierarchical reinforcement learning. In *International Conference on Machine Learning*. PMLR, 3540–3549.

[70] Huaxiaoyue Wang, Kushal Kedia, Juntao Ren, Rahma Abdullah, Atiksh Bhardwaj, Angela Chao, Kelly Y Chen, Nathaniel Chin, Prithwish Dan, Xinyi Fan, et al. 2024. MOSAIC: A Modular System for Assistive and Interactive Cooking. *arXiv preprint arXiv:2402.18796* (2024).

[71] Xihuai Wang, Shao Zhang, Wenhao Zhang, Wentao Dong, Jingxiao Chen, Ying Wen, and Weinan Zhang. 2025. ZSC-Eval: An evaluation toolkit and benchmark for multi-agent zero-shot coordination. *Advances in Neural Information Processing Systems* 37 (2025), 47344–47377.

[72] Tianbao Xie, Siheng Zhao, Chen Henry Wu, Yitao Liu, Qian Luo, Victor Zhong, Yanchao Yang, and Tao Yu. 2024. Text2Reward: Reward shaping with language models for reinforcement learning. In *International Conference on Learning Representations*.

[73] Wenhao Yu, Nimrod Gileadi, Chuyuan Fu, Sean Kirmani, Kuang-Huei Lee, Montserrat Gonzalez Arenas, Hao-Tien Lewis Chiang, Tom Erez, Leonard Hasenclever, Jan Humplik, et al. 2023. Language to Rewards for Robotic Skill Synthesis. In *Conference on Robot Learning*. PMLR, 374–404.

[74] Shangtong Zhang and Shimon Whiteson. 2019. DAC: The double actor-critic architecture for learning options. *Advances in Neural Information Processing Systems* 32 (2019).

[75] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, Anind K Dey, et al. 2008. Maximum entropy inverse reinforcement learning. In *AAAI Conference on Artificial Intelligence*, Vol. 8. Chicago, IL, USA, 1433–1438.