Anticipate, Adapt, Act: A Hybrid Framework for Task Planning

Nabanita Dash[1], Ayush Kaura[1], Shivam Singh[1], Ramandeep Singh[1], Snehasis Banerjee[2], Mohan Sridharan[3], K. Madhava Krishna[1]

***Abstract*— Anticipating and adapting to potential failures is a key capability that robots require to collaborate effectively with humans in complex domains. This continues to be a challenge despite the impressive performance of state of the art AI planning systems and Large Language Models (LLMs) because of the uncertainty associated with the tasks and their outcomes. Toward addressing this challenge, we present a hybrid framework that integrates the generic prediction capabilities of an LLM with the relational probabilistic sequential decision-making capability of Relational Dynamic Influence Diagram Language (RDDL). For any given task, the robot reasons about the task and the capabilities of the human attempting to complete it; predicts potential failures due to lack of ability (in the human) or relevant domain objects; and executes actions that prevent such failures or help recover from them. Experimental evaluation in the VirtualHome 3D simulation environment demonstrates substantial improvement in task completion, execution time, and collaboration.**

***Index Terms*— Human-Robot Collaboration, Probabilistic Planning, Task Adaptation, Assistive Robotics**

1 Introduction

Consider a robot assisting an elderly human in a kitchen, say with getting a glass of water from the sink to the kitchen counter. Due to mobility and stability limitations, there is uncertainty about whether the human can complete the task successfully; they may end up dropping the water glass. In such situations, we would expect the robot to anticipate the potential for negative outcomes, e.g., the glass being dropped, and either prevent the potential negative outcome, e.g., by fetching the water glass, or prepare to deal with the outcome, e.g., by making sure it has access to the mop needed to clear the water spill; Figure 1 shows some snapshots of these scenarios. State of the art methods for robot planning and human-robot collaboration assume deterministic environments (e.g., with classical planners [1], [2]), or pre-compute and use reactive policies (e.g., with probabilistic planners [3]), and do not fully support the desired proactive decision-making behavior.

The hybrid framework* presented in this paper is inspired by the observation that the desired adaptive behavior needs the ability to anticipate tasks, identify potential failures while trying to complete these tasks, and plan actions that prevent these failures or help recover from them. Specifically, the framework enables the robot to:

- Introduce an anticipatory reward-shaping mechanism that allows a robot to favour plans which proactively minimise the probability of impending human failures;
- Present the first tight LLM-RDDL pipeline that converts free-form language predictions into fully specified stochastic planning problems without manual symbol engineering, enabling plug-and-play use of any modern large language model; and
- Demonstrate online, failure-aware recovery in long-horizon human-robot collaboration (HRC) tasks, achieving state-of-the-art robustness on five VirtualHome scenarios.

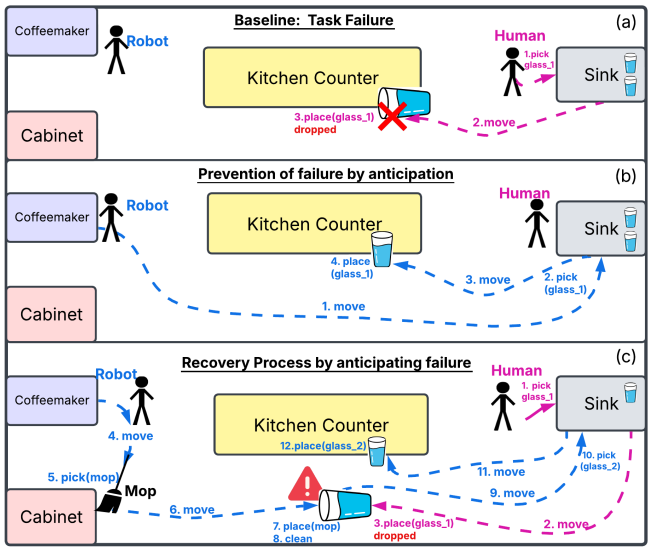

Fig. 1: Illustrative task of fetching a glass of water from the sink to the kitchen counter. In the baseline scenario, the human may end up dropping the water glass due to stability issues. Our framework enables the robot to anticipate such failures; the robot then either prevents failure by completing the task, or prepares to recover from the failure by preparing to clean the potential water spill and complete the task.

Our key contribution is the tight integration of three complementary modules-LLM-based task prediction, RDDL probabilistic planning, and proactive failure handling-into a single closed-loop controller. We experimentally evaluate our framework in the context of household tasks in the realistic VirtualHome simulation environment. We demonstrate an increase in task completion accuracy and a reduction in the number of failures, leading to improved human-robot collaboration compared with baselines that use just an LLM or the probabilistic planner.

2 Related Work

There is an extensive body of research in Human-robot interaction (HRI), including recent advances in shared au-

[1] Robotics Research Center, IIIT Hyderabad, India
[2] TCS Research, Tata Consultancy Services, India
[3] School of Informatics, University of Edinburgh, UK

*https://dnabanita7.github.io/ECMR2025

tonomy and collaborative task execution [4]–[6]. Despite impressive advancements in perception, reasoning, and learning, adaptation to failures and collaboration between humans and robots continue to be open problems [7], [8].

Reasoning problems such as planning and diagnostics have often been addressed by encoding prior domain knowledge as relational logic statements in an action language such as Planning Domain Definition Language (PDDL) [9] and using suitable solvers. Other languages such as RDDL [10] help model a class problems that are difficult to model with probabilistic extensions of PDDL (e.g, due to stochastic effects and unrestricted concurrency). Its semantics are that of a ground Dynamic Bayesian Network, and it can be used for both classical planning and probabilistic sequential decision making (e.g., Markov Decision Process, MDP; Partially Observable MDP, POMDP). It supports both classical tree search planners like PROST [11] and learning-based approaches in RDDL-Gym.

In an attempt to reduce the effort involved in encoding domain knowledge, recent research has explored the use of data-driven frameworks such as LLMs to learn domain models [12]–[14]. The ability of LLMs to predict action sequences to complete tasks has led to claims about their ability to plan and reason [15], [16]. At the same time, the increasing evidence disputing such claims and demonstrating their tendency to provide arbitrary responses in novel situations [17] has led to their use in assisting planning frameworks in auxiliary tasks such as goal translation [18], task anticipation [19], and goal allocation [20].

Robust Human-Robot Collaboration (HRC) requires the ability to deal with action failures. Existing methods monitor and adapt to deviations in action plans [12], [21] using behavior models encoded in PDDL domains [22] or probabilistic sequential decision making [23]. If task planning is modeled as an MDP, existing methods support reasoning about environment states [24], model learning [25], estimating human intentions [26], and encoding human behavior models [27].

Despite existing work, the desired proactive behavior that anticipates failures, and either prevents them or prepares to recover from them, remains an open problem in HRC. We seek to address this problem by leveraging the complementary strengths of knowledge-based and data-driven systems. Specifically, our hybrid framework combines the generic task prediction capability of LLMs, the probabilistic planning capability of RDDL, and a reward mechanism to trade off between task completion and failure recovery.

3 Problem Formulation and Framework

Consider a home environment with a human $\mathcal{H}$ and an assistive robot $\mathcal{R}$ collaborating to any given goal G. The sequence of high-level tasks $\{T_1, T_2, \ldots, T_n\}$ to be completed is not known to the robot, and one task is normally assigned as G at a time. Completing each task T_i, such as *preparing toast*, requires the execution of plan of finer-granularity actions such *grab bread*, *put-in toaster*, and *switch appliance* by the robot and the human; completing a subset of these actions is considered a *subgoal*. The execution of some of these actions can result in failure, e.g., a heavy plate with the bread may be dropped; without loss of generality, we limit any such failure (in this paper) to the human's actions due to limitations in the human's capabilities, e.g., not always able to lift heavy objects. For effective collaboration, the robot has to anticipate such failures based on prior knowledge/experience of the human's abilities, and prevent this failure, e.g., by completing the action instead of the human, or prepare to address this failure, e.g., fetch a broom and dustpan to clear the broken plate.

Figure 2 provides an overview of our hybrid framework. The robot equipped with this framework prompts an LLM with user preferences and input, scene description, and some example task sequences to output a predicted sequence of upcoming tasks (Section 3.1). The current and next task as assigned as a joint goal to the domain-specific planning component (Section 3.2). Since the domain state is known after each action's execution in the VirtualHome simulation environment, domain-specific planning is formulated as a relational MDP, using RDDL to model domain-specific knowledge in the form of fluents, axioms, and a suitable reward structure. This outputs a plan of actions to be executed by the robot and the human; the human's action choices may not match the robot's expectation. This planning also anticipates and accounts for failures (Section 3.4) to complete the tasks reliably and efficiently. Specific components are described in more detail below.

3.1 LLM-based Task Anticipation

We employ the *llama-3.3-70b-versatile* model via the Groq API (temperature 0.2, max_tokens $=$ 500) to predict the robot's next four actions. At run time the driver concatenates three elements into a single prompt: (i) a **task sample space** of 240 grounded household actions (`master_tasks.json`); (ii) the last three user sequences (`sequence.json`); and (iii) a scene graph of rooms, objects, and food items (`virtualhome_categories.json`). The 2-3 user task observations used in few-shot prompting are sampled from the last three completed task sequences by the user. These sequences are stored and updated across runs. In a real deployment, they can be logged over time by the robot during daily activity and used to adapt the prompt in a personalized way.

We used two prompting strategies: (i) few-shot and (ii) chain-of-thought [28]. Both approaches take as input a predefined task space T and a structured JSON scene description that represents the kitchen environment, including the locations of objects and the positions of the agents. Here, T defines the set of valid household tasks, comprising 11 different tasks such as $move$ and $grab$ that can have many different ground instantiations (e.g., moving to different locations, grabbing different objects). The few-shot approach uses 2-3 prior

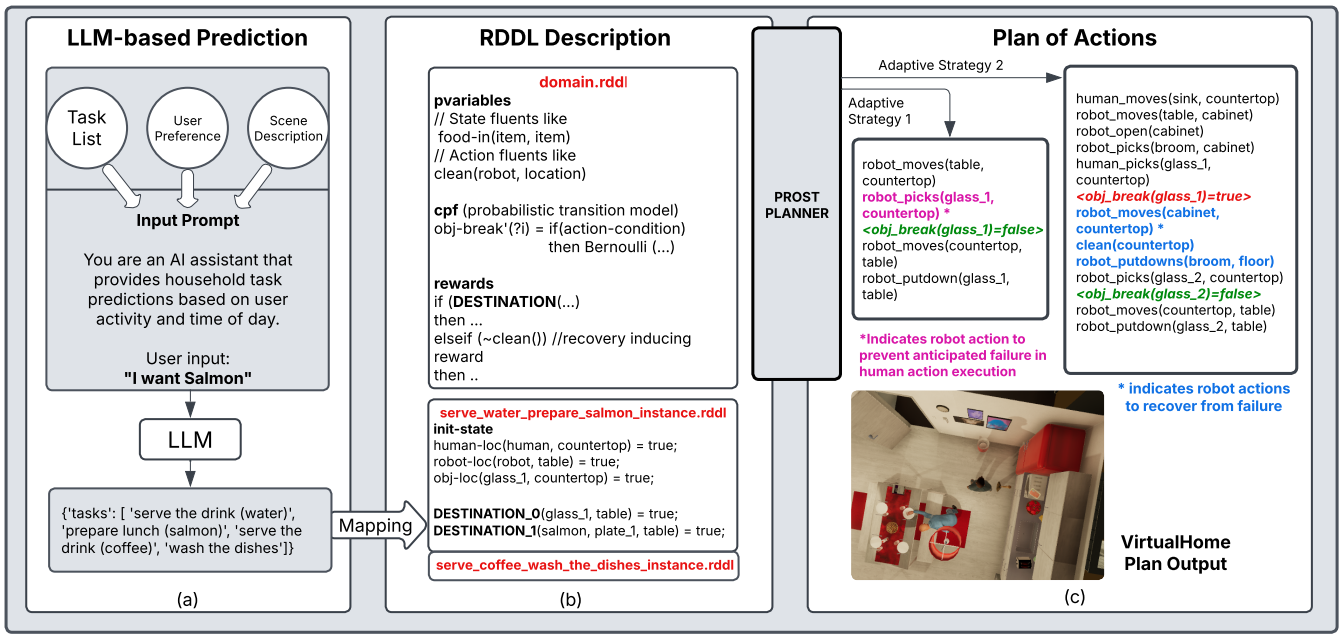

Fig. 2: Framework's pipeline: (a) LLM takes a prompt of task lists, user preferences, scene description, and user input, to predict a task sequence; (b) RDDL description of domain model and joint goal comprising current and predicted tasks is fed to PROST planner; and (c) Plan of actions to be executed by robot (and human) to achieve the goal, including steps to be taken by the robot to prevent or recover from potential failures (e.g., broken objects, spills, item unavailability) in human's action execution.

user task observations, while the chain-of-thought method incorporates two in-context examples with explicit step-by-step reasoning to infer user activity patterns. In both cases, the model generates a sequence of predicted tasks, filtering out tasks that are considered to be invalid (i.e., not similar match found in T) to maintain the reliability of planning. A snapshot of such prompting and the corresponding output is shown in the left part of Figure 2; the user asks for $salmon$ for dinner, and the LLM predicts subsequent tasks to involve serving coffee and washing the dishes.

3.2 *Task Planning*

The JSON snippet is automatically mapped to the corresponding RDDL goal via a template lookup table provided in our repository (`jsonfiles/rddl_goals.json`). The current and the next (predicted) task (from the LLM output) are mapped to a joint goal state G. Recall that the domain-specific planning to achieve G is formulated as a relational MDP: $\langle V, A, P, R, H, s_0 \rangle$, where V is the domain state, A is the set of finer-granularity actions (to be executed by robot or human), P is the state transition function, R is the reward specification, H is the planning horizon, and s_0 is the initial state. In our RDDL description of the domain, each task T_i is automatically associated with an instance file defining relevant objects and axioms, ensuring that execution aligns with the intended task conditions. These instance files are generated from a common domain file, which encodes variables for states (e.g., capturing location of objects, state of appliances) and actions, universal transition dynamics, constraints, and reward structures that incorporate auxiliary incentives to guide the robot through intermediate steps necessary for task completion. This approach supports modularity, with the domain file being invariant and instance files adapting to different task specifications.

Subgoals are defined automatically based on causal dependencies to ensure that tasks progress logically, respecting necessary preconditions and dependencies between actions. Simulated trials validate these subgoals before they are encoded in the domain file. The dynamic nature of the domain file allows the system to adapt task execution while maintaining adherence to the relevant state and action constraints.

The reward function is designed to promote efficient and adaptive execution. Positive rewards are assigned for achieving subgoals and goals (tasks from the LLM), while redundant or unsafe actions incur penalties. Task-specific tuning was not performed. We refer the reader to Section 4.2 and Figure 5 for an empirical analysis of reward sensitivity across tasks.

The planner uses these rewards to generate a fine-grained action sequence by constructing a directed graph representation of possible states; it initializes Q-values to guide decision-making, checks reward locks, ensures that subgoal completion aligns with the overall goal, and prevents unnecessary delays.

For efficient action computation, the original RDDL description is factored to obtain a relational MDP $\langle D_R, D_H \rangle$, where $D_R = \langle S_R, M_R \rangle$ represents the robot's description and $D_H = \langle S_H, M_H, B_H \rangle$ the human's description. Here, S defines types, predicates, and $pvariables$, while M specifies actions, preconditions, and effects. For instance, actions like `human_pick` and `robot_pick` modify the state fluent `obj-loc`. The model predicting human behavior B_H is derived from simulations with added noise impacting state transitions (see Section 3.3 below). A task instance $T =$

$\langle O, I, G \rangle$ consists of objects O, the initial state I, and the goal state G. We use the PROST planner [11] to compute an action sequence $\pi = \langle a_1, \ldots, a_K \rangle$ that transitions the system from I to G as a combination of actions to be executed by the robot and the human. This plan computation is based on heuristic tree search that integrates both decision and chance nodes, with the action selected at each decision node to maximize expected cumulative rewards.

3.3 Modeling Human Behavior as State Transitions

The model B_H predicting human behavior is modeled as probabilistic state transitions that capture uncertainty in the human's execution of specific actions. We first encode such uncertainty in human action outcomes using empirical probability distributions (that are not known to the robot). The added noise governing human behavior is sampled from a Gaussian distribution centered on expected execution parameters. The magnitude of this noise is dynamically adjusted based on task complexity and individual execution patterns. This allows us to model execution failures as threshold-based conditions, i.e., if an action does not meet predefined criteria (e.g., insufficient force to lift an object), the state transition is unsuccessful, prompting re-execution or adaptation. These conditions ensure that the system realistically captures variability in human action execution. We will discuss this further in Section 4.1.

We simulate 11 cooking and cleaning tasks over 10 trials and use the observations to learn an initial model of the state transition probabilities $P_H(s'|s, a_H)$, the likelihood of moving from state s to s' given human action a_H. As the system collects more observations during action execution, the probability estimates are refined, allowing the robot agent to predict human action outcomes more accurately and adjust accordingly. These learned transition probabilities are explicitly encoded in the domain file, ensuring that human uncertainty is accounted for during planning and execution.

3.4 Anticipation and Collaboration

When humans deviate from expected behavior, e.g., mishandle objects or skip steps, the reward function in our framework is designed such that it drives the robot towards goal completion and avoiding failures to achieve smooth task progression in shared spaces. Figure 3 shows a simplified version of our reward specification for a specific task (*prepare breakfast*). It is based on intuitive subgoals that guide the robot and human agents toward collaboration. Each component of the reward function help to model and decompose tasks into an appropriate sequence of actions. Specifically, rewards are provided for different types of interactions illustrated in the context of preparing breakfast:

- **Appliance interaction:** Rewards are given for `open` and `close` actions performed on accessible appliances.
- **Item collection:** Picking up `FOOD_ITEM` or `CONTAINER` early in the task, laying the groundwork for food preparation, is rewarded.
- **Container placement:** Rewards are given for placing containers (e.g., plates, bowls) at designated `DESTINATION` location.
- **Intermediate food placement:** Placing food items in intermediate processing areas (e.g., stove, toaster) in preparation for cooking is rewarded.
- **Appliance usage:** Actions such as `robot_switch_on` and `robot_switch_off` accrue rewards when appliance supports such actions.
- **Food containment:** Rewards are used to incentivize `put_in` actions that place food items inside containers.
- **Final delivery:** Rewards are provided when container with food is placed in the correct goal location.
- **Goal fulfillment:** Satisfying all conditions of the `GOAL` predicate receives a high reward.

Similar reward functions are populated automatically for other actions and tasks in the domain.

```
reward:
"+ 5 * [robot_open(?r, ?l) ^ APPLIANCE(?l) ^ HAS-SWITCH(?l) ^ robot-loc(?r, ?l)]",
"+ 20 * [pick(?r, ?f, ?l) ^ FOOD_ITEM(?f)]",
"+ 20 * [pick(?r, ?p, ?l) ^ CONTAINER(?p)]",
"+ 5 * [robot_close(?r, ?l) ^ APPLIANCE(?l) ^ HAS-SWITCH(?l) ^ robot-loc(?r, ?l)]",
"+ 40 * [place(?r, ?f, ?l) ^ FOOD_ITEM(?f) ^ DESTINATION_0(?f, ?l)]",
"+ 5 * [robot_switch_on(?r, ?l) ^ APPLIANCE(?l) ^ HAS-SWITCH(?l) ^ robot-loc(?r, ?l)]",
"+ 5 * [robot_switch_off(?r, ?l) ^ APPLIANCE(?l) ^ HAS-SWITCH(?l) ^ robot-loc(?r, ?l) )]",
"+ 40 * [put_in(?r, ?f, ?p) ^ FOOD_ITEM(?f) ^ CONTAINER(?p)]",
"+ 100 * [FOOD_ITEM(?f) ^ CONTAINER(?p) ^ food-in(?f, ?p) ^ GOAL_0(?f, ?p, ?l)]"
```

Fig. 3: Reward for $PrepareBreakfast(Toast)$. Due to space constraints, a partial version is shown.

We also define a set of rewards that promote anticipatory and cooperative behaviors by aligning action choices with capabilities. We illustrate this below in the context of humans performing picking up actions that are likely to result in failure in our experimental setup, and the corresponding actions used to avoid or recover from these failures.

- **Reward for the robot picking up fragile items.**

$$R_1 = \sum_{\substack{r:\text{robot},\\ i:\text{item},\\ l:\text{location}}} \big(\text{pick}(r, i, l) \wedge \text{fragile}(i) \wedge \exists h : \text{human}\,(\text{human-loc}(h, l))\big) \quad (1)$$

  This term rewards the robot for correctly picking up fragile items when a human is present, ensuring safer handling and preventing potential breakage of the item.
- **Reward for proper placement of cleaning mop.**

$$R_2 = \sum_{\substack{r:\text{robot},\\ m:\text{item},\\ l:\text{location}}} \big(\text{mop}(m) \wedge \text{obj-loc}(m, l) \wedge \text{robot-loc}(r, l) \wedge \exists h : \text{human}\,(\text{human-loc}(h, l)) \wedge \exists x : \text{item}\,(\text{fragile}(x) \wedge \text{obj-loc}(x, l))\big) \quad (2)$$

  This reward incentivizes robot to place mop items near locations where a human and fragile items exist.
- **Penalty for humans picking up fragile items without**

a mop nearby.

$$R_3 = \sum_{\substack{l:\text{location},\\ i:\text{item}}} \exists h : \text{human}\left(\text{fragile}(i) \wedge \text{pick\_human}(h,i,l) \wedge \neg\exists m : \text{item}\left(\text{mop}(m) \wedge \text{obj-loc}(m,l)\right)\right) \quad (3)$$

This term applies a penalty in cases when human picks up a fragile item in a location where no mop item is present, increasing the likelihood of accidents.

Other similar rewards can be automatically generated using known or learned knowledge of states and actions likely to result in failures or to help recover from failures. These rewards shape behaviors that are proactive, assistive, and responsive to human presence and task context, with the robot anticipating human behavior, adapting its actions to assist or compensate when needed.

4 Experimental Setup and Results

We experimentally evaluated two hypotheses related to the performance of our framework.

H1: Reasoning with learned/encoded models of human behavior improves performance and collaboration with a human compared to not using such models.

H2: Anticipating failure enables the robot agent to execute efficient recovery strategies, in contrast to scenarios where the robot agent lacks failure anticipation.

The experimental setup used for evaluation and the corresponding results are discussed below.

4.1 Experimental Setup

Our experimental setup involved three main components: learning a stochastic human behavior model, encoding domain knowledge in RDDL, and selecting appropriate baselines and evaluation measures.

Learning the Human Behavior Model. Recall from Section 3.3 that we learned a probabilistic state transition model of human behavior in the VirtualHome simulator by decomposing tasks into actions and introducing noise sampled from a normal distribution ($\mu = 0, \sigma = 0.1$) filtered with a 0.5σ threshold. For each task, we ran 10 noisy simulations to compute conditional probabilities over state transitions. These probabilities capture patterns such as preferences (e.g., choosing fragile vs. non-fragile items) and deviations (e.g., leaving a room mid-task). Figure 4 illustrates example probabilities, such as a 0.8 likelihood of moving to the kitchen from a random location and 0.6 likelihood of placing bread in the toaster when in the kitchen.

Encoding Planning Models. We modeled the environment using RDDL, representing states, actions, and rewards; see Section 3.2. The domain includes 11 generic food-related household tasks that involve nine food items, eight appliances, nine cutlery items, and five cleaning items. Human actions were treated as exogenous stochastic transitions based on the learned model, while robot actions were planned to maximize expected cumulative reward.

We used the PROST Planner [11], with the Trial-based Heuristic Tree Search (THTS) algorithm on a Factored MDP, integrating Upper Confidence Bound (UCT) for action selection, Unsolved Monte Carlo (UMC) for handling uncertainty, Partial Bellman Backup (PB) for Q-value estimation, and Iterative Deepening Search (IDS) for heuristic Q-value initialization. To optimize planning efficiency, we combined the IPC2011 and IPC2014 configurations. IPC2011 version supports broad exploration via Monte Carlo backups, while IPC2014 improves decision-making through *UMC* and *PB*, prioritizing informative samples; the combination balances exploration and exploitation. We set a maximum planning horizon of 60 for the joint goal (two tasks), limiting excess computation and action concurrency to prevent unnecessary search depth expansion. Our reward function promotes goal-directed behavior by rewarding task completion, penalizing delays, offering intermediate rewards for progress, and discouraging errors (Section 3.4). The planner generated joint-action (human and robot actions) sequences that optimized:

- *Distance to target*: nearest agent handles the object.
- *Action prioritization*: robot preempts fragile or unavailable object interactions.
- *Goal relevance*: relevant *objects types* prioritized.
- *Task efficiency*: compute minimal plan.

We map natural language LLM outputs into RDDL specific goals and rewards syntax using a predefined JSON file. The reward function incentivizes successful sub-goal completion, safe handling, anticipation of failures, and overall efficiency and penalties discourage inefficiencies, such as random movements, unnecessary toggling, or repeated actions. Robot actions are explicitly rewarded or penalized; human actions are modeled and observed but are not directly influenced.

Evaluation Measures and Baselines. To evaluate **H1** and **H2**, we performed 30 simulation rollouts, each with five collaborative tasks in our household domain. We use the following evaluation measures:

- **Average number of actions**: computed as the mean number of actions for completing entire task(s).
- **Number of failures**: count of critical failures (e.g., unsafe handling); fewer unresolved failures (no prevention/recovery) indicates better performance.
- **Number of failures prevented**: count of instances in which robot's proactive actions reduced failures in human action outcomes.
- **Recovery through anticipation**: count of instances in which robot recovered from failure by anticipating it.
- **Task completion rate**: fraction of tasks completed successfully; higher indicate better performance.
- **Goal completion percentage**: fraction of subgoals achieved, measuring adherence to overall task(s).

Each simulation included robot and human actions such as `pick`, `place`, `move`, `switch`, `open`, and `put_in`. The robot evaluated multiple trajectories rather than committing to a single plan, optimizing for expected reward under uncertainty. The environment includes fragile and non-fragile

```
Task: prepare_breakfast (toast)
P(walk_to_kitchen) = 0.800
P(in_hand | kitchen) = 0.625
P(put_in | in_hand, kitchen) = 0.600
P(switch_on | kitchen) = 0.625
```

Fig. 4: For any given task, we monitor the simulation agent over ten simulations, each with added noise in the sub-goals. The probability of the human agent walking to the kitchen when starting from a random location is 0.8. The probability of grabbing a bread slice while in the kitchen is 0.625. The probability of the human agent placing the bread slice in the toaster after grabbing it in the kitchen is 0.6. The probability of the human switching on the toaster while in the kitchen is 0.625.

objects (e.g., fruits, cereal, mop, bread, milk) and dynamic constraints like path obstructions or unavailable items. As described later (Table II), we assessed the robot's ability to: (a) Anticipate human delays or mistakes; (b) recover from missteps such as tool misuse or incorrect object selection; and (c) adapt to constraints via rewards which consider length of plans, safety violations, and failed subgoals. As **baselines** for comparison, we consider just the LLM (**L**) and just the RDDL-based planner (**R**). Like the SOTA LLM/VLM-only task planners [12] which demonstrate recovery from plan deviation, we define a LLM baseline (with a fixed number of max_replans). The LLM baseline directly computes a sequence of actions for the joint goal (current and predicted next task). The LLM baseline struggles to replan from failures (Sec. 4.2) in the five composite household tasks mimicking real-world scenarios consisting of long-horizon action sequences and high chances of action failure. The RDDL-based baseline does **not** incorporate any model of human behavior of the corresponding anticipatory rewards. It follows a fixed goal-conditioned plan derived from an LLM-predicted task description. It lacks anticipatory strategies, which results in brittle execution when the human diverges from expected behaviors.

4.2 Experimental Results

Recall that we evaluated our framework and baselines using five collaborative household tasks in simulation environment. Each simulation of these tasks involved multiple potential failures in the form of undesired human action outcomes, and the robot adapted through anticipation and planning. The evaluation used the measures described in Section 4.1. Recall that each task is a high-level goal, such as "Prepare and Serve Toast and Coffee." This task can be split into subgoals, like "Toast the Bread," and "Place items on the table." Each subgoal is carried out through a series of actions, such as "Open Toaster," "Put in Toaster," "Switch on Toaster," "Pick up Cup," and "Pour Liquid."

Evaluating H1. We first evaluated the impact of reasoning with human behavior models across the different task and computed the values of the six measures, with the results summarized in **Table I** and **Table II** for an illustrative set of tasks. The results show notable differences between our framework, denoted by **(O)** compared with two baselines: RDDL/PROST without anticipatory rewards, denoted by **(R)**, and LLM-only plans, denoted by **(L)**.

As seen in **Table I**, our framework consistently outperforms both baselines in subgoal and task completion. For instance, in the task *Prepare and Serve Salmon + Water*, our method achieves 85% subgoal completion, while the RDDL baseline without anticipation reaches 63.3% and the LLM achieves only 46.67%. Similarly, in *Prepare and Serve Pizza + Wash Dish*, our framework attains 84.2% subgoal completion, compared to the RDDL baseline's 66.7% and the LLM baseline's 35.56%. The average subgoal completion rate across all tasks is 85.5% for our framework, compared to 68.26% for the RDDL baseline and 44.55% for the LLM-only baseline. These results highlight our system's effectiveness in decomposing complex tasks (vis-à-vis the LLM only plans) and executing them reliably using the human behavior model (vis-à-vis the RDDL baseline).

Table II presents failure rates, prevention statistics, and recovery times across tasks. Our framework shows a significant advantage in proactive failure prevention and recovery. For example, in *Prepare and Serve Cereal + Coffee*, our method prevents 20 out of 30 failures, while the RDDL baseline prevents 17 failures, and LLM baseline prevent only 5 failures. Furthermore, our framework averages 28 seconds per task, while the RDDL baseline fails to complete the composite tasks in most scenarios. In tasks like *Prepare and Serve Toast + Coffee*, the baseline suffers from high failure rates with limited recovery, while our framework reduces failure occurrences and improves recovery efficiency.

Evaluating H2. Next, we evaluated the role of failure anticipation and recovery strategies. The comparison between our framework and the baseline(s), as shown in **Table I**, reveals that our method consistently achieves higher task completion rates. Tasks like *Prepare and Serve Coffee + Wash Dish* (87.5%) and *Prepare and Serve Pizza + Wash Dish* (83.4%) show significant improvements over the RDDL baseline's (R) completion rates (75% and 66.7%, respectively). This suggests that the integration of human behavior modeling and failure anticipation in our framework played a key role in ensuring higher task success.

In contrast, the baseline(s) struggled with achieving high task completion, often leaving subgoals incomplete. For example, in *Prepare and Serve Salmon + Water*, the RDDL baseline without anticipatory behavior achieved only 55% task completion; LLM-only baseline managed 30% task completion, while our framework reached 80%. The baselines' lack of anticipatory planning and recovery results in more frequent task interruptions and failures.

The statistics from **Table II** emphasize the advantages of our approach in recovery and failure prevention. For instance, in *Prepare and Serve Toast + Coffee*, our method demonstrated more effective recovery (9/12) by using anticipatory behavior models as compared to LLM-only baseline (6/27) and the RDDL baseline without anticipatory behavior (0/9). Our proactive strategies, such as adjusting actions and preventing incorrect tool usage, resulted in fewer failures

Task	SubGoals Completion %			Task Completion %		
	(LLM)	(RDDL)	(Ours)	(LLM)	(RDDL)	(Ours)
Prepare and Serve Salmon + Water	46.67%	63.3%	**85%**	30%	55%	**80%**
Prepare and Serve Coffee + Wash Dish	47.5%	75%	**86.7%**	20%	55%	**87.5%**
Prepare and Serve Cereal + Coffee	52.5%	70%	**88.3%**	25%	60%	**85%**
Prepare and Serve Toast + Coffee	42.5%	68.3%	**83.3%**	15%	58.3%	**84%**
Prepare and Serve Pizza + Wash Dish	35.56%	66.7%	**84.2%**	10%	57%	**83.4%**
Average%	44.55%	68.26%	**85.5%**	20%	57.06%	**84.78%**

TABLE I: Evaluation measures for selected composite tasks (Prepare + Serve) between LLM-only planner, RDDL/PROST without anticipatory rewards, and our framework. Percentages reflect subgoal and task completion rates over 30 simulations.

Task	Failures			Prevention			Recovery			Avg. Actions			Time Taken	
	L	R	O	L	R	O	L	R	O	L	R	O	R	O
Prepare and Serve Salmon + Water	18/30	17/30	14/30	12/30	13/30	16/30	6/18	0	11/14	–	–	38	–	28s
Prepare and Serve Coffee + Wash Dish	9/30	11/30	8/30	21/30	19/30	22/30	0	2/11	6/8	24	–	26	–	22s
Prepare and Serve Cereal + Coffee	25/30	13/30	10/30	5/30	17/30	20/30	9/25	0	8/10	–	–	42	–	32s
Prepare and Serve Toast + Coffee	27/30	9/30	12/30	3/30	21/30	18/30	6/27	0	9/12	–	–	48	–	34s
Prepare and Serve Pizza + Wash Dish	18/30	11/30	9/30	12/30	19/30	21/30	3/18	0	7/9	–	–	30	–	24s
Average	20.0	12.2	10.6	10.6	17.8	19.4	4.8	–	8.2	–	–	36.8	–	28s

TABLE II: Failure prevention and recovery statistics across 30 simulations for selected composite (Prepare + Serve) tasks. *Prevention* indicates proactive avoidance of likely human failures, while *Recovery* refers to corrective intervention after failure has occurred. *Avg. Actions* and *Time Taken* represent the average number of steps and time required to complete the tasks. We compare LLM-only plans (**L**), the RDDL/PROST without considering human behavior models or anticipatory rewards (**R**), and our framework (**O**). In most scenarios, both baseline methods fail to complete the composite tasks, making *Avg. Actions* and *Time Taken* not applicable, denoted as '–'. Our framework consistently completes tasks through anticipation and recovery.

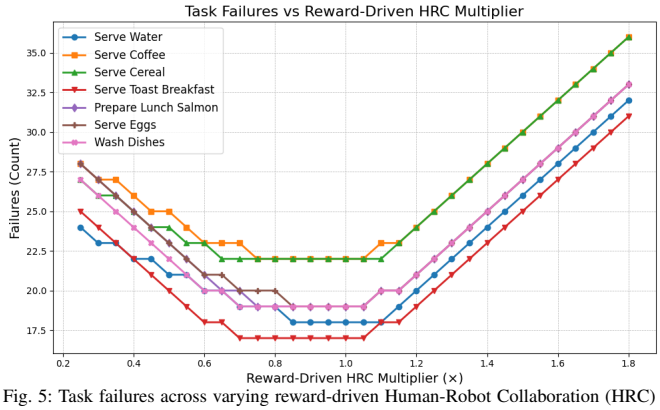

Fig. 5: Task failures across varying reward-driven Human-Robot Collaboration (HRC) multipliers. Tasks such as Serve Water, Serve Coffee, and Serve Cereal show stable failure rates near baseline, while Prepare Toast, Prepare Lunch Salmon, Serve Eggs, and Wash Dishes exhibit clear minima, highlighting improved task robustness at optimal collaboration sensitivity.

overall. Overall, our framework resulted in a more reliable solution for human-robot collaboration. They also highlight the significance of integrating human-aware reasoning and anticipatory reasoning in robotic planning, thus supporting both **H1** and **H2**. We also measured the planning latency for recovery actions, which averaged 0.28 seconds (std = 0.4), suggesting that reactive replanning can be done online without noticeable delays.

Effect of collaboration sensitivity on task failures. To further understand the robustness of our framework under different collaboration scenarios, we analyze task failures as a function of changes to the extent to which the reward-base strategy prioritizes task completion or failure recovery, e.g., preparing for recovering from likely failure takes time away from completing the task(s). Figure 5 presents the number of failures observed across several tasks under different reward multipliers. Tasks such as Serve Water, Serve Coffee, and Serve Cereal demonstrated stable failure rates close to the baseline, indicating low sensitivity to changes in collaborative incentive structure. In contrast, tasks like Prepare Toast, Prepare Lunch Salmon, Serve Eggs, and Wash Dishes exhibited pronounced minima in failure rates at specific multipliers. This behavior highlighted that tuning of reward mechanism significantly enhances robust task completion. These results suggest that our reward structure can be adapted dynamically based on task complexity to maximize the benefits of proactive assistance.

5 Conclusion

In this work, we have introduced a hybrid framework for a Human-Robot Collaboration, where the robot provides anticipatory assistance to the potential failures in the actions executed by the human participant, enabling task planning which is adaptive to changes in the environment. The initial part of framework uses an LLM to predict upcoming tasks based on user preferences and current environment description. These tasks serve as joint high-level goals for relational MDP planner based on RDDL domain description, using the

PROST planner to generate action plans that achieve these goals. This planning also consider a simple learned model predicting probabilistic state transitions caused by human actions, and a reward function designed to smoothly and automatically trade-off task completion and failure anticipation (and recovery). Experimental results show the benefits of our framework, demonstrating a **27.72%** and **17.24%** increase in task completion and sub-goals completion (respectively) compared with the RDDL baseline that does not include a failure anticipation or recovery strategies. All experiments assume full observability of the environment and human actions, as provided by the VirtualHome simulator. While this enables controlled evaluation, it does not capture the uncertainty of real-world settings. In future work, we aim to deploy our framework on a physical robot in a mock kitchen, explore multi-agent collaboration, and handle partial observability using learned belief models. We also plan to reduce reliance on handcrafted rewards through offline reinforcement learning.

REFERENCES

[1] R. I. Brafman, D. Tolpin, and O. Wertheim, "Probabilistic programs as an action description language," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 13, pp. 15 351–15 358, Jul. 2024. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/26790

[2] Y.-q. Jiang, S.-q. Zhang, P. Khandelwal, and P. Stone, "Task planning in robotics: an empirical comparison of pddl- and asp-based systems," *Frontiers of Information Technology Electronic Engineering*, vol. 20, no. 3, pp. 363–373, 2019. [Online]. Available: https://doi.org/10.1631/FITEE.1800514

[3] R. S. Novin, A. Yazdani, A. Merryweather, and T. Hermans, "Risk-aware decision making for service robots to minimize risk of patient falls in hospitals," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 3299–3305.

[4] A. Smith and B. Jones, "Advancements in human-robot interaction for household tasks: A review," *International Journal of Robotics Research*, vol. 41, no. 3, pp. 345–367, 2022.

[5] J. Park and H. Kim, "Collaborative task planning for human-robot interaction: Recent advances and future directions," *IEEE Robotics and Automation Magazine*, vol. 31, no. 2, pp. 77–94, 2024.

[6] Q. Chen and Z. Liu, "Collaborative task planning for household service robots: A review of approaches and challenges," *International Journal of Social Robotics*, vol. 16, no. 2, pp. 189–207, 2024.

[7] H. Guo, F. Wu, Y. Qin, R. Li, K. Li, and K. Li, "Recent trends in task and motion planning for robotics: A survey," *ACM Comput. Surv.*, vol. 55, no. 13s, Jul. 2023. [Online]. Available: https://doi.org/10.1145/3583136

[8] S. Honig and T. Oron-Gilad, "Understanding and resolving failures in human-robot interaction: Literature review and model development," *Frontiers in Psychology*, vol. 9, 2018. [Online]. Available: https://www.frontiersin.org/journals/psychology/articles/10.3389/fpsyg.2018.00861

[9] C. Aeronautiques, A. Howe, C. Knoblock, I. D. McDermott, A. Ram, M. Veloso, D. Weld, D. W. Sri, A. Barrett, D. Christianson *et al.*, "Pddl| the planning domain definition language," *Technical Report, Tech. Rep.*, 1998.

[10] S. Sanner, "Relational dynamic influence diagram language (rddl): Language description," 2010, available: http://users.cecs.anu.edu.au/~ssanner/IPPC_2011/RDDL.pdf.

[11] T. Keller and P. Eyerich, "PROST: Probabilistic planning based on UCT," in *Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling (ICAPS 2012)*. AAAI Press, 2012.

[12] Y. Guo, Y.-J. Wang, L. Zha, and J. Chen, "Doremi: Grounding language model by detecting and recovering from plan-execution misalignment," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024, pp. 12 124–12 131.

[13] T. Birr, C. Pohl, A. Younes, and T. Asfour, "Autogpt+p: Affordance-based task planning using large language models," in *Robotics: Science and Systems XX*, ser. RSS2024. Robotics: Science and Systems Foundation, Jul. 2024. [Online]. Available: http://dx.doi.org/10.15607/RSS.2024.XX.112

[14] Z. Zhao, W. S. Lee, and D. Hsu, "Large language models as commonsense knowledge for large-scale task planning," in *Proceedings of the 37th International Conference on Neural Information Processing Systems*, ser. NIPS '23, 2023.

[15] E. Hirsch, G. Uziel, and A. Anaby-Tavor, "What's the plan? evaluating and developing planning-aware techniques for language models," 2024. [Online]. Available: https://arxiv.org/abs/2402.11489

[16] F. Joublin, A. Ceravola, P. Smirnov, F. Ocker, J. Deigmoeller, A. Belardinelli, C. Wang, S. Hasler, D. Tanneberg, and M. Gienger, "Copal: Corrective planning of robot actions with large language models," 2025. [Online]. Available: https://arxiv.org/abs/2310.07263

[17] K. Valmeekam, K. Stechly, and S. Kambhampati, "Llms still can't plan; can lrms? a preliminary evaluation of openai's o1 on planbench," *ArXiv*, vol. abs/2409.13373, 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:272770270

[18] Y. Xie, C. Yu, T. Zhu, J. Bai, Z. Gong, and H. Soh, "Translating natural language to planning goals with large-language models," 2023. [Online]. Available: https://arxiv.org/abs/2302.05128

[19] R. Arora, S. Singh, K. Swaminathan, A. Datta, S. Banerjee, B. Bhowmick, K. M. Jatavallabhula, M. Sridharan, and M. Krishna, "Anticipate act: Integrating llms and classical planning for efficient task execution in household environments," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 14 038–14 045.

[20] S. Izquierdo-Badiola, G. Canal, C. Rizzo, and G. Alenyà, "Plancollabnl: Leveraging large language models for adaptive plan generation in human-robot collaboration," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 17 344–17 350.

[21] D. Nau, M. Ghallab, and P. Traverso, *Automated Planning: Theory & Practice*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2004.

[22] S. Izquierdo-Badiola, G. Canal, C. Rizzo, and G. Aleny, "Improved task planning through failure anticipation in human-robot collaboration," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 7875–7880.

[23] S. Yang, X. Mao, Q. Wang, and Y. Huang, "A hybrid planning approach for accompanying information-gathering in plan execution monitoring," *Journal of Intelligent Robotic Systems*, vol. 103, 09 2021.

[24] S. Zhang, P. Khandelwal, and P. Stone, "icorpp: Interleaved commonsense reasoning and probabilistic planning on robots," *Robotics and Autonomous Systems*, vol. 174, p. 104613, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S092188902300252X

[25] R. Karia, P. Verma, A. Speranzon, and S. Srivastava, "Epistemic exploration for generalizable planning and learning in non-stationary settings," *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 34, p. 310âĂŞ318, May 2024. [Online]. Available: http://dx.doi.org/10.1609/icaps.v34i1.31489

[26] M. Cramer, K. Kellens, and E. Demeester, "Probabilistic decision model for adaptive task planning in human-robot collaborative assembly based on designer and operator intents," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7325–7332, 2021.

[27] O. C. Görür, B. Rosman, F. Sivrikaya, and S. Albayrak, "Social cobots: Anticipatory decision-making for collaborative robots incorporating unexpected human behaviors," in *Proc. ACM/IEEE Int. Conf. Human-Robot Interaction (HRI '18)*, 2018, pp. 398–406. [Online]. Available: https://doi.org/10.1145/3171221.3171256

[28] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou, "Chain-of-thought prompting elicits reasoning in large language models," 2023. [Online]. Available: https://arxiv.org/abs/2201.11903