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 failures is a key capability robots need 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 probabilistic sequential decision-making capability of Relational Dynamic Influence Diagram Language. 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 lack of relevant domain objects; and executes actions to prevent such failures or recover from them. Experimental evaluation in the VirtualHome 3D simulation environment demonstrates substantial improvement in performance compared with state of the art baselines.**

***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 fetching 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. We expect the robot to anticipate the potential for such negative outcomes, i.e., the glass being dropped, and either prevent this 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 executing actions to complete these tasks with a human, and to plan actions that prevent failures or help recover from them. Specifically, the framework enables the robot to:

- Adapt a pretrained Large Language Model (LLM) to anticipate future tasks and convert a description of uncertainty in task outcomes to model parameters of a stochastic planning problem.

[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

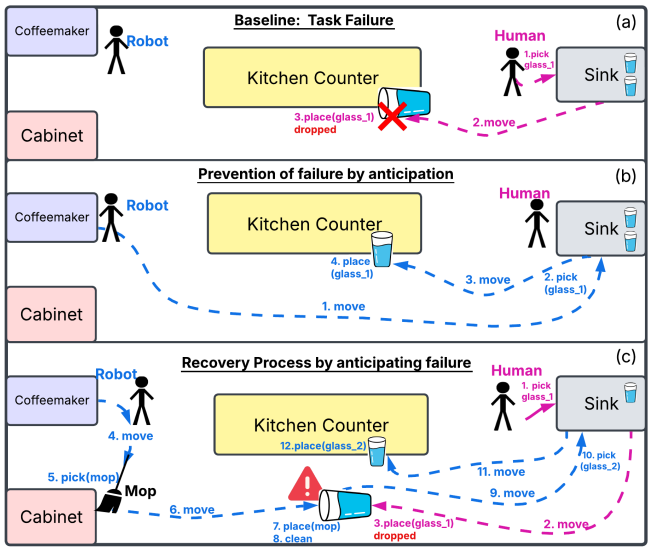

Fig. 1: Illustrative task of fetching a glass of water from the sink to the kitchen counter. In the baseline scenario, the human may end up dropping the water glass due to stability issues. Our framework enables the robot to anticipate such failures; it either prevents the failure by completing the task,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

- Use the Relational Dynamic Influence Diagram Language (RDDL) to encode and solve the stochastic planning problem, computing a sequence of actions to jointly achieve the current and upcoming tasks.
- Encode and reason with a reward mechanism that trades-off task completion probability with the effort needed to prevent or recover from potential failures.

Our key contribution is the smooth integration of the complementary strengths of the three components, i.e., LLM-based task anticipation, RDDL-based probabilistic planning, and proactive failure handling. We experimentally evaluate our framework in the context of human-robot collaboration in 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 just use an LLM or a probabilistic planner.

2 Related Work

There is an extensive body of research in Human-robot interaction (HRI), including recent advances in shared autonomy and collaborative task execution [4], [5]. Despite impressive advancements in perception, reasoning, and learn-

ing, adaptation to failures and collaboration between humans and robots continue to be open problems [6], [7].

Reasoning tasks 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) [8] and using suitable solvers. Other languages such as RDDL [9] 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 [10] 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 for computing plans [11]. The ability of LLMs to predict action sequences to complete tasks has led to claims about their ability to plan and reason [12], [13], although they do not build the models needed for reasoning and their operation does not match the directed search operation involved in planning. There is increasing experimental evidence demonstrating the tendency of LLMs to provide arbitrary responses in novel situations [14], advocating their use in planning frameworks for auxiliary tasks such as knowledge translation [15], task anticipation [16], [17], and goal allocation [18].

Robust Human-Robot Collaboration (HRC) requires the ability to deal with action failures. Planning methods can monitor and adapt to deviations in action outcomes [19] using behavior models encoded in PDDL domains [20] or probabilistic sequential decision making [21]. In the context of probabilistic sequential decision making, existing methods support adaptation to changes in the domain [22], learned models [23], and human behavior [24].

Despite the existing work, the desired proactive behavior that anticipates failures, and either prevents them or prepares to recover from them, continue to be a problem of interest. 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 prediction capability of LLM, 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 complete a given task specified as the goal G. The sequence of high-level tasks $\{T_1, T_2, \ldots, T_n\}$ to be completed is not known to the robot in advance, and one task is normally assigned as G at a time. Completing each T_i, e.g., *preparing toast*, requires a plan of finer-granularity actions such as *grab bread*, *put-in toaster*, and *switch appliance* to be computed and executed by the robot and the human. Completing a subset of these actions is considered a *subgoal*. The execution of some actions can result in failure, e.g., a heavy plate with food on it may be dropped. Without loss of generality, we limit any such failure (in this paper) to the human's actions due to limitations in their capabilities, e.g., the human may not be able to lift heavy objects. For effective collaboration, the robot has to anticipate such failures based on prior knowledge or observations of the human's abilities. The robot can then 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 for achieving the desired behavior. A robot equipped with this framework prompts an LLM with current task, prior knowledge of user preferences and the scene (if available), and some example task sequences, receiving as output a sequence of anticipated tasks (Section 3.1). The current and next task are assigned as a joint goal to the domain-specific planning component (Section 3.2). Assuming that the domain state is known after each action's execution (which is true in the simulation environment used for testing), 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. The output of this step is a plan of actions to be executed by the robot and the human, although the human's actual 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. Specific components are described in detail below.

3.1 LLM-based Task Anticipation

In our framework, the *llama-3.3-70b-versatile* LLM and the Groq API (temperature 0.2, max_tokens = 500) is used to predict the next four tasks likely to be assigned. At run time, the prompt includes: (i) a *task sample space* of 240 grounded household actions (`master_tasks.json`); (ii) examples of user sequences (`sequence.json`); and (iii) a scene graph of rooms and objects (`virtualhome_categories.json`). The user sequences are sampled from the last three sequences completed by the user, and are updated across runs. During actual deployment, these sequences can be updated over time to adapt prediction to user's preferences.

We used two prompting strategies: (i) few-shot and (ii) chain-of-thought [25]. Both strategies consider a predefined task space and a structured JSON description of the scene (often *kitchen* in our experiments) including the locations of objects and other agents. Specifically, we considered 11 different tasks such as *move* and *grab* with multiple ground instantiations (e.g., move to different locations, grab different objects). The few-shot approach uses 2-3 prior observations of tasks completed by user, while the chain-of-thought method uses two in-context examples with step-by-step reasoning to infer user activity patterns. With either

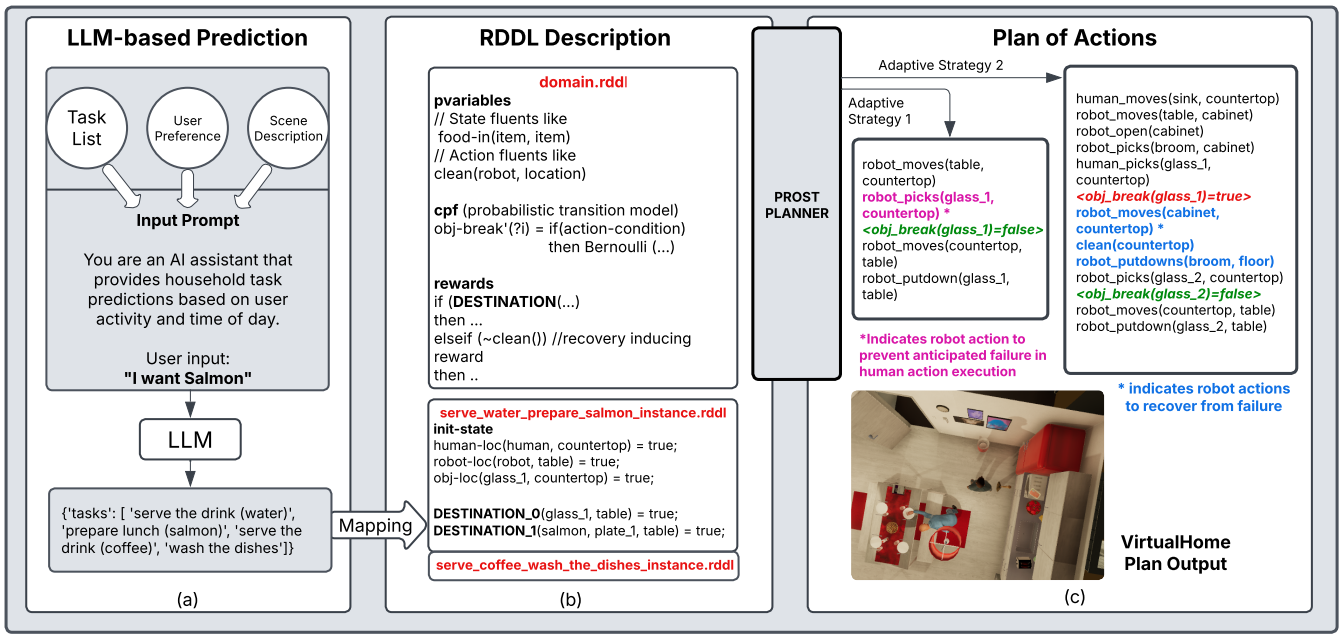

Fig. 2: Our framework's pipeline: (a) LLM takes a prompt of task lists, user preferences, scene description, and current command, to predict upcoming tasks; (b) RDDL description of domain knowledge and joint goals comprising current and predicted tasks fed to PROST planner; and (c) Plan of actions to be executed by robot and human to achieve the goal, including robot's actions to prevent or recover from potential failures due to the human's actions.

strategy, the LLM's output is a sequence of anticipated tasks, which is filtered to remove tasks considered to be invalid. A snapshot of such prompting and the corresponding output is shown in the left part of Figure 2; the user asks for *salmon* 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 through simple template matching (`jsonfiles/rddl_goals.json`). The current and the next (predicted) task (from LLM's output) are mapped to a joint goal G. Recall that 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 set of states, 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 domain description, each task T_i is automatically associated with an instance file defining relevant objects and axioms. These instance files are generated from a common domain file, encoding variables for states (e.g., 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 for task completion. This approach supports modularity, with the domain file being defined once and the instance files defined based on the tasks at hand. In addition, subgoals are defined automatically as logically important stages on the path to any given goal, respecting necessary preconditions and dependencies between relvant states actions. Simulated trials validate these subgoals before they are encoded in the instance files, and used to evaluate partial achievement of the corresponding goal(s).

The reward function is designed to promote adaptive execution by trading off between successful task completion and the effort involved in preventing failures or recovering from them. Positive rewards are assigned for achieving subgoals and goals, while redundant or unsafe actions incur penalties. Additional details about reward specification are in Section 3.4, and an experimental analysis of reward sensitivity is in Figure 5 in Section 4.2. 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 computational efficiency, the original RDDL description is factored to obtain $\langle D_R, D_H \rangle$, where $D_R = \langle S_R, M_R \rangle$ is the robot's description and $D_H = \langle S_H, M_H, B_H \rangle$ is 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 [10] 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 using a heuristic tree search method that maximizes expected cumulative rewards.

3.3 Modeling Human Behavior as State Transitions

The model B_H of human behavior captures the uncertainty in the human's execution of specific actions. Since we were using a simulation environment for experimental evaluation, we had to simulate such uncertainty using empirical probability distributions not known to the robot. Specifically, we introduced noise by sampling from a Gaussian distribution centered on expected outcomes. The magnitude of this noise was adjusted automatically based on task complexity and considered different human behaviors, modeling execution failures as thresholds driven by specific criteria, e.g., a human's attempt to lift an object fails because they are not able to exert sufficient force. As a result, we were able to realistically simulate variability in human action execution, as discussed further in Section 4.1.

Once the noise distributions were determined, observations from 10 simulated trials of each of 11 cooking and cleaning tasks were used to learn an initial model of B_H as the state transition probabilities $P_H(s'|s, a_H)$ for any particular human action a_H. These probabilities were refined over subsequent trials, allowing the robot to predict human action outcomes more accurately. These probabilities were encoded in the domain file, and used to consider uncertainty in human action outcomes during planning and execution.

3.4 Anticipation and Collaboration

A key component of our framework is the reward specification that helps the robot trade off between completing the task successfully and the effort involved in preventing or recovering from potential failures in human action execution. Figure 3 shows a simplified version of our reward function for a specific task (*prepare breakfast*) based on subgoals that guide the robot toward task completion. Each component of the reward function models different interactions illustrated here in the context of tasks in the kitchen:

- **Appliance interaction:** successfully performing valid actions, e.g,, `open`, `close`, and `robot_switch_on` on appliances is rewarded.
- **Item collection:** preparing for tasks, e.g., picking up `FOOD_ITEM` or `CONTAINER` early, is rewarded.
- **Container placement:** placing objects at designated locations, e.g., containers such as plates and bowls in their `DESTINATION`, is rewarded.
- **Intermediate placement:** placing objects in intermediate locations in preparation for specific tasks, e.g., stove or toaster for cooking, is rewarded.
- **Final delivery:** placing an object in its correct goal location is rewarded.
- **Goal fulfillment:** satisfying all conditions of the `GOAL` receives a high reward.

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

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

```
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: Partial description of reward specification for the $PrepareBreakfast(Toast)$ task.

- **Reward for preventing failure.** This term rewards the robot for performing an action instead of a human in an attempt to prevent failure, e.g., pick up fragile items that a human may potentially drop and break.

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

- **Reward for preparing to respond to failure.** This term incentivizes the robot to place items that enable it to respond to failure near locations where a failure may occur, e.g., place a mop near location of fragile items that a human may pick up and potentially drop.

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

- **Penalty for not being prepared for failure.** This term penalizes the robot when a human performs an action that may lead to failure and the robot has not prepared for it accordingly, e.g., human allowed to pick up a fragile item without a mop nearby.

$$R_3 = -\sum_{\substack{l:\text{location},\\ i:\text{item}}} \Big(\exists h : \text{fragile}(i) \wedge \text{pick\_human}(h,i,l) \wedge \nexists m : \text{mop}(m) \wedge \text{obj-loc}(m,l)\Big)$$

Rewards can be suitably defined for other actions and domains using knowledge of actions likely to result in failures or help recover from failures. Each action executed by the robot also comes with a cost (i.e., negative reward) based on the effort needed (e.g., time spent, distance traveled) in completing the action. This allows the robot to trade off the need to prevent (or recover from) failures and the effort involved in achieving it.

4 Experimental Setup and Results

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

```
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: Example probabilities of state transition probabilities in human behavior model. The probability of the human: walking to the kitchen from a random location is 0.8; grabbing a bread slice while in the kitchen is 0.625; placing the bread slice in the toaster after grabbing it is 0.6; switching on the toaster while in the kitchen is 0.625.

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

H2: Incorporating the strategy to anticipate failures enables the robot to recover better from such failures compared with not using the strategy.

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

4.1 Experimental Setup

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

Learning the Human Behavior Model. As stated in Section 3.3, 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 also represent preferences (e.g., choosing fragile instead of non-fragile items) and deviations (e.g., leaving a room mid-task). Figure 4 shows probabilities of outcomes of the human's actions related to a particular task.

Encoding Planning Models. We modeled the environment using RDDL as described in Section 3.2. The domain included 11 generic household (cooking and cleaning) tasks based on nine food items, eight appliances, nine cutlery items, and five cleaning items. Human actions were treated as exogenous stochastic transitions that (unknown to the robot) were based on the learned model. We used a predefined JSON file to map natural language LLM outputs into RDDL-specific syntax for goals and rewards.

We used the PROST Planner [10], with the Trial-based Heuristic Tree Search algorithm on a Factored MDP, integrating Upper Confidence Bound for action selection, Unsolved Monte Carlo for handling uncertainty, Partial Bellman Backup for Q-value estimation, and Iterative Deepening Search for heuristic Q-value initialization. We balanced exploration and exploitation by combining the IPC2011 configuration that supports broad exploration and the IPC2014 configuration that improves decision-making by prioritizing informative samples. We set a maximum planning horizon of 60 for the joint goal (two tasks) to limit search. Our reward function rewards task completion and progress, penalizes delays, and discourages failures (Section 3.4). The planner generated joint-action (human and robot actions) sequences that optimize: (a) *distance to target*: nearest agent handles the object; (b) *action prioritization*: robot chooses to perform interactions with fragile objects; (c) *goal relevance*: relevant *object types* are considered; and (d) *plan length*: minimal plan is computed. Recall that human actions are modeled and observed but are not determined by our framework.

Performance Measures and Baselines. To evaluate **H1** and **H2**, we performed 30 simulation rollouts, each with five collaborative tasks in our household domain. Each task was 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." To achieve each (sub)goal, the robot had to execute a sequence of actions such as "Open Toaster," "Put bread in Toaster, and " "Switch on Toaster." Each simulation involved multiple potential failures in the form of undesired human action outcomes. We used the following performance measures:

- **Average number of actions**: the mean number of actions for completing entire task(s).
- **Number of failures**: count of failures during execution of the assigned task(s).
- **Number of failures prevented**: count of instances of robot preventing failure in human action outcomes.
- **Number of failures recovered**: count of instances in which robot recovered from failure by anticipating it.
- **Task completion rate**: fraction of tasks completed successfully; higher value indicates better performance.
- **Subgoal completion rate**: fraction of subgoals achieved, measuring adherence to task(s).

Each simulation included robot and human actions such as `pick`, `place`, `move`, `switch`, `open`, and `put_in`. The robot evaluated multiple trajectories, optimizing for expected reward. The environment includes fragile and non-fragile objects (e.g., fruits, cereal, mop, bread, milk) and dynamic constraints in the form of obstructions and unavailable items. As summarized later (in 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 based on rewards that consider plan length and failed subgoals.

As **baselines** for comparison, we considered just the LLM (**L**) and just the RDDL-based planner (**R**). Similar to existing literature [26], the LLM baseline directly computed a sequence of actions for the joint goal (current and predicted next task). The RDDL-based baseline did not include the predictive model of human behavior in its reward specification; its reward structure directed the robot to follow a goal-conditioned plan to complete the task(s).

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: Task and subgoal completion performance of our framework is substantially better than that of the two baselines, LLM-only and RDDL/PROST without anticipatory rewards, over selected composite tasks.

Task	Failures			Prevention			Recovery			Avg. Actions		
	L	R	O	L	R	O	L	R	O	L	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
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
Prepare and Serve Cereal + Coffee	25/30	13/30	10/30	5/30	17/30	20/30	9/25	0	8/10	–	–	42
Prepare and Serve Toast + Coffee	27/30	9/30	12/30	3/30	21/30	18/30	6/27	0	9/12	–	–	48
Prepare and Serve Pizza + Wash Dish	18/30	11/30	9/30	12/30	19/30	21/30	3/18	0	7/9	–	–	30
Average	20.0	12.2	10.6	10.6	17.8	19.4	4.8	–	8.2	–	–	36.8

TABLE II: Failure prevention and recovery statistics for selected composite tasks. *Prevention* indicates proactive avoidance of likely human failures, while *Recovery* refers to corrective action after failure has occurred. In most scenarios, the LLM-only baseline (**L**) or RDDL/PROST baseline (**R**) failed to complete the composite tasks, with *Avg. Actions* and *Time Taken* reported as '–'. Our framework (**O**) consistently completes tasks through anticipation and recovery.

4.2 Experimental Results

Table I and **Table II** summarize results for a set of representative composite tasks. Our framework (**Ours, O**) consistently outperformed the two baselines: RDDL/PROST without anticipatory rewards (**RDDL, R**) and LLM-only plans (**LLM, L**). For example, in **Table I**, average subgoal completion rate across all tasks was 85.5% for our framework, compared with 68.26% for the RDDL baseline and 44.55% for the LLM baseline; substantial performance improvements were observed for overall task completion as well. Also, there was considerable variation in the results obtained with the baselines depending on the type and complexity of the tasks, whereas the performance remained consistently good with our framework. In addition, the task (or subgoal) completion rate was not 100% with our framework only because the robot ran out of time to complete the tasks; this limit was relaxed for the numbers shown in **Table II** below. These results indicate the importance of the human behavior prediction model and the reward mechanism that leverages this model.

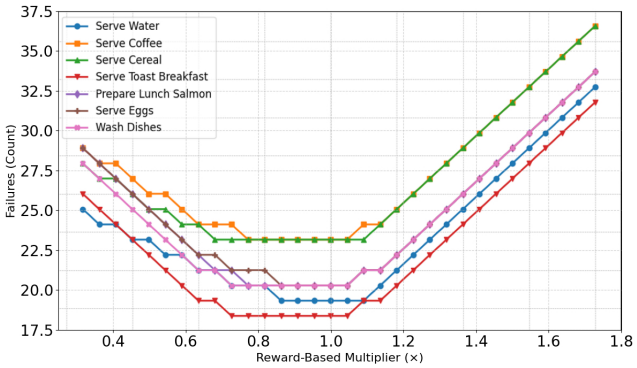

Fig. 5: Task failures as a function of the reward-based multiplier that determines extent to which failure prevention and recovery is prioritized over task completion. With an increase in the multiplier, failures reduce up to a point before increasing again. A trade off between failure prevention and task completion leads to good performance.

Table II summarizes statistics of failures, failure prevention, and recovery across tasks. The robot prevented failures in many more trials when using our framework than when it used the baselines; it also recovered from a larger percentage of the errors that were not prevented. Recall that neither baseline explicitly reasons about potential failures to prevent or recover from them. Any prevention of failure or recovery from failure with our baselines was hence purely happenstance, e.g., the robot using the RDDL baseline chose to fetch the glass of water that a human may have dropped, or the LLM baseline happened to direct the robot to clear the mess created by the glass of water dropped by the human. As observed in **Table I**, there was variation in the performance of the baselines based on the type and complexity of the tasks, but the performance remained consistent with our framework. Furthermore, our framework resulted in tasks being completed in each trial, including when the robot did not prevent or recover from errors. With the two baselines, on the other hand, the tasks remained incomplete in most trials. These results indicate the robot's ability to collaborate effectively with the human; it anticipated and prevented failures, recovered from failures, and/or found other plans

to complete the task, thus supporting **H1** and **H2**.

We also explored the trade off between task completion and failure prevention or recovery. Figure 5 presents the number of failures for different tasks and different reward multipliers; for a higher value of the multiplier, the reward structure assigns higher weight to failure prevention and recovery. The sensitivity of performance to this multiplier depended on the type and complexity of the task, but the increased focus on failures improved performance up to a point before having a negative impact. These results suggest that our reward structure can be adapted to the tasks and domain.

5 Conclusion

This paper described a hybrid framework that enables a robot collaborating with a human to anticipate upcoming tasks, reason about potential failures due to actions executed by the human based on a learned model of their action capabilities, and to act to prevent these failures or to recover from them. The framework combines the complementary strengths of knowledge-based and data-driven methods. In particular, it combines the LLM-based statistical prediction and the RDDL-based probabilistic relational sequential decision making capabilities. Experimental results demonstrate the substantial improvement in performance compared with LLM-based and RDDL-based baselines. Future work will explore the use of this framework on a physical robot and its extension to support multiagent collaboration.

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. Zhang, P. Khandelwal, and P. Stone, "Task planning in robotics: an empirical comparison of pddl- and asp-based systems," *Frontiers of Information Technology and 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 *IEEE International Conference on Robotics and Automation*, 2021, pp. 3299–3305.

[4] 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.

[5] 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.

[6] 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

[7] 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

[8] 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.

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

[10] T. Keller and P. Eyerich, "PROST: Probabilistic planning based on UCT," in *International Conference on Automated Planning and Scheduling*. AAAI Press, 2012.

[11] Z. Zhao, W. S. Lee, and D. Hsu, "Large language models as commonsense knowledge for large-scale task planning," in *International Conference on Neural Information Processing Systems*, 2023.

[12] 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

[13] 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

[14] 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

[15] 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

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

[17] T. Fu, B. Jauw, and M. Sridharan, "Combining LLM, Non-monotonic Logical Reasoning, and Human-in-the-loop Feedback in an Assistive AI Agent," in *IEEE International Conference on Robot and Human Interactive Communication*, Eindhoven, The Netherlands, August 2025.

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

[19] D. Nau, M. Ghallab, and P. Traverso, *Automated Planning: Theory and Practice*. Morgan Kaufmann Publishers, 2004.

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

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

[22] M. Sridharan, M. Gelfond, S. Zhang, and J. Wyatt, "REBA: A Refinement-Based Architecture for Knowledge Representation and Reasoning in Robotics," *Journal of Artificial Intelligence Research*, vol. 65, pp. 87–180, May 2019.

[23] R. Karia, P. Verma, A. Speranzon, and S. Srivastava, "Epistemic Exploration for Generalizable Planning and Learning in Non-Stationary Settings," in *International Conference on Automated Planning and Scheduling*, 2024. [Online]. Available: http://dx.doi.org/10.1609/icaps.v34i1.31489

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

[25] 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

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