Generic-to-Specific Reasoning and Learning for Ad Hoc Teamwork

Hasra Dodampegama, Mohan Sridharan

Institute of Perception, Action and Behavior, School of Informatics,
University of Edinburgh, Edinburgh, UK
hasra.dodampegama@ed.ac.uk, m.sridharan@ed.ac.uk

Abstract

Embodied AI agents deployed in assistive roles often have to collaborate with humans or other agents without prior coordination, quickly adapting to new situations and the behavior of others. State of the art methods for such ad hoc teamwork often pursue a data-driven approach that needs a large labeled dataset of prior observations, lacks transparency, and makes it difficult to rapidly revise existing knowledge in response to changes. This paper advocates the combination of knowledge-based reasoning and data-driven learning for ad hoc teamwork. In the context of collaborating with a human, our architecture enables an ad hoc agent to use a Large Language Model to anticipate abstract future tasks and goals of the human. The agent then computes a plan of concrete actions to achieve these goal(s) based on non-monotonic logical reasoning with the predicted future task(s), prior commonsense domain knowledge, and models that are rapidly learned and revised from limited examples to predict the human's concrete action choices. The architecture capabilities are illustrated and evaluated in *VirtualHome*, a realistic 3D simulation environment.

1 Introduction

Consider an *embodied* AI agent completing daily living tasks in collaboration with a human it has not worked with before. Figure 1 shows the agent and human working to prepare breakfast and set up a workstation. The human and the agent have a limited view of the environment at each step and do not communicate directly with each other[1]. While the human is aware of the upcoming tasks in the domain, the agent is not; it has to reason with different descriptions of domain knowledge and uncertainty, adapting its actions to changes in the domain and the human's behavior to collaborate efficiently. These characteristics represent Ad Hoc Teamwork (AHT), where an agent must cooperate with others without prior coordination (Stone et al. 2010).

The state of the art in AHT has moved from using predetermined policies for selecting actions in specific states to methods with a key *data-driven* component that uses a long history of prior experiences to build probabilistic or deep network methods that model the behavior of other agents (or agent types) and optimize the behavior of the ad hoc agent (Mirsky et al. 2022). However, it is difficult to gather large training datasets of different situations in practical domains. Also, these methods lack transparency, and make it difficult to revise existing knowledge over time.

Figure 1: Screenshots from *VirtualHome* (Puig et al. 2018) showing a human (female in green top) and an embodied AI agent (male in blue shirt) collaborating.

In a departure from existing work, we advocate the design of architectures for AHT that bridge knowledge-based and data-driven reasoning and learning methods. Specifically, we build on our recent work on AHT in structured domains (Dodampegama and Sridharan 2023a,b) to explore human-agent collaboration in more complex domains. An ad hoc agent equipped with our architecture:

1. Leverages the strengths of Large Language Models (LLMs) to anticipate future tasks and goals of the human collaborator using partial sequence of tasks for efficient collaboration in dynamic settings; and
2. Considers current and anticipated tasks as joint goals, and performs non-monotonic logical reasoning with relevant commonsense domain knowledge and a rapidly-learned predictive model of the human's behavior to determine its actions toward achieving the goals.

We use Answer Set Prolog (Gelfond and Kahl 2014) for non-monotonic logical reasoning and GPT4o mini for high-level task anticipation. We evaluate our architecture's capabilities in household scenarios in *VirtualHome*, a realistic physics-based 3D simulation environment for multiagent collaboration (Puig et al. 2018).

2 Related Work

Research in AHT evolved from the use of specific protocols defining the agents' behavior in specific scenarios (Bowl-

[1]Our architecture can support communication between agents, but we do not include it here for simplicity.

ing and McCracken 2005), to the use of probabilistic and sampling-based methods (Barrett et al. 2013) over a period of time (Mirsky et al. 2022). Methods currently considered to be state of the art include a data-driven component, using probabilistic, deep-network and/or reinforcement learning (RL) methods to learn *policies* that determine action choices based on a long history of prior observations of different types of agents or situations. For example, RL methods have been used to select a policy for a new teammate from the policies learned for different teammate types (Barrett et al. 2017), and model-based RL has been used to learn environment models and behavior of teammates (Ribeiro et al. 2023). Researchers have used attention-based deep neural networks to jointly learn policies for different agent types (Chen et al. 2020) and different team compositions (Rahman et al. 2021); hierarchical variational auto-encoders and meta-learning to model and infer beliefs over other agents (Zintgraf et al. 2021); Convolutional Neural Networks to detect and adapt to changing teammate types (Ravula, Alkoby, and Stone 2019); sampling strategies combined with learning methods to optimize performance (Zand, Parker-Holder, and Roberts 2022); and meta-RL methods with self-play and perturbed rewards to respond to unknown teammates (Fang et al. 2024). Researchers have also explored communication strategies for AHT, *e.g.,* broadcast messages at a cost or use heuristic methods (Macke, Mirsky, and Stone 2021). Such data-driven methods are resource-hungry, build opaque models, and make it difficult to adapt to changes.

Recent work in AHT has leveraged Large Language Models (LLMs), e.g., an LLM-based hierarchical planner (IRoT) has been used to generate an ad hoc agent's policy to support zero-shot collaboration (Liu et al. 2024), and memory retrieval and code-driven reasoning have been used for AHT in the AvalonPlay benchmark (Shi et al. 2023). In parallel, there has been increased use such data-driven methods for embodied AI systems in physically realistic simulation environments such as Habitat (Savva et al. 2019) and VirtualHome (Puig et al. 2018) that support the generation of complex scenarios for evaluating AI systems.

Our architecture combines knowledge-based and data-driven methods for reasoning and learning, allowing an ad hoc agent to adapt to the behavior of its teammate in a complex, realistic household environment. We demonstrate that the interplay between reasoning and learning enables the ad hoc agent to utilize prior knowledge, learn and revise predictive models of human behavior with limited data, and accurately anticipate future tasks with minimal feedback.

3 Architecture

In our architecture for human-embodied AI ad hoc collaboration (Figure 2), the ad hoc agent is an assistive embodied AI agent that utilizes non-monotonic logical reasoning with prior commonsense domain knowledge and an incrementally learned behavior model of its human teammate. At each step, both the human and the ad hoc agent receive state observations from the environment, which they independently use to determine and perform their actions. The tasks to be carried out each day are determined by the task generator. The ad hoc agent is only aware of the current task of the human. To anticipate the human's future goals, the ad hoc agent utilize a LLM. The prompt to the LLM is created following different prompt-engineering techniques (Section 3.3) and the LLM's output is validated to ensure the feasibility and correctness of the predictions. Any available previous knowledge of tasks in the domain is used in the prompts and the external validation. Each component of the architecture is described using the following example.

Example Domain 1 *[Embodied AI Agent]*
Consider an AI agent and a human collaborating to complete household tasks; Figure 1 shows snapshots of preparing breakfast and setting up the work-station (Puig et al. 2018). The agent and the human can interact with the environment through actions that involve moving to places, picking up or placing objects, switching appliances on or off, and opening or closing appliances. Completing a task requires a sequence of such actions to be computed and executed by the embodied AI agent and/or the human without direct communication between them. The agent assumes that the human has access to the same information about domain state, predicts the actions that the human will execute over the next few steps, and makes its plan of actions accordingly. The agent's prior commonsense knowledge includes relational descriptions of some attributes of the domain, objects, and human. It also includes axioms governing actions and changes, such as the agent not being able to hold more than two objects at a time or that it is impossible to pick up objects that are not in the same location as the agent.

3.1 Knowledge Representation and Reasoning

In our architecture, any given domain's transition diagram is described using an extension of action language $\mathcal{AL}_d$ (Gelfond and Inclezan 2013). Action languages are formal models of parts of natural language for describing transition diagrams of dynamic systems. The domain representation comprises a system description $\mathcal{D}$, a collection of statements of $\mathcal{AL}_d$, and a history $\mathcal{H}$. $\mathcal{D}$ has a sorted signature Σ with basic sorts and describes the domain attributes (statics, fluents) and actions in terms of the sorts of their arguments. *VirtualHome* domain include basic sorts such as $object$, $appliance$, ad_hoc_agent, $human$, and $step$ (for temporal reasoning). These sorts are arranged hierarchically, e.g., $apple$ is a sub-sort of $food$ that is a sub-sort of $graspable$, a sub-sort of $object$. Actions can be $agent\_actions$, performed by the ad hoc agent, e.g., $grab(ad\_hoc\_agent, object)$ and $switch\_on(ad\_hoc\_agent, appliance)$, or $exo\_actions$, performed by the human, e.g., $exo\_grab(human, object)$ and $exo\_switch\_on(human, appliance)$. Statics are domain attributes whose values cannot be changed and fluents are attributes whose values can be changed. Fluents can be *inertial*, which obey inertia laws and are changed by actions, e.g., $at(ad\_hoc\_agent, location)$ is the ad hoc agent's location; and *defined*, which do not obey inertia laws and are not directly changed by ad hoc agent's actions, e.g., $agent\_at(human, location)$ denotes the human's location.

Based on Σ, the domain dynamics are described in $\mathcal{D}$ using three types of axioms: *causal law*, *state constraint*, and

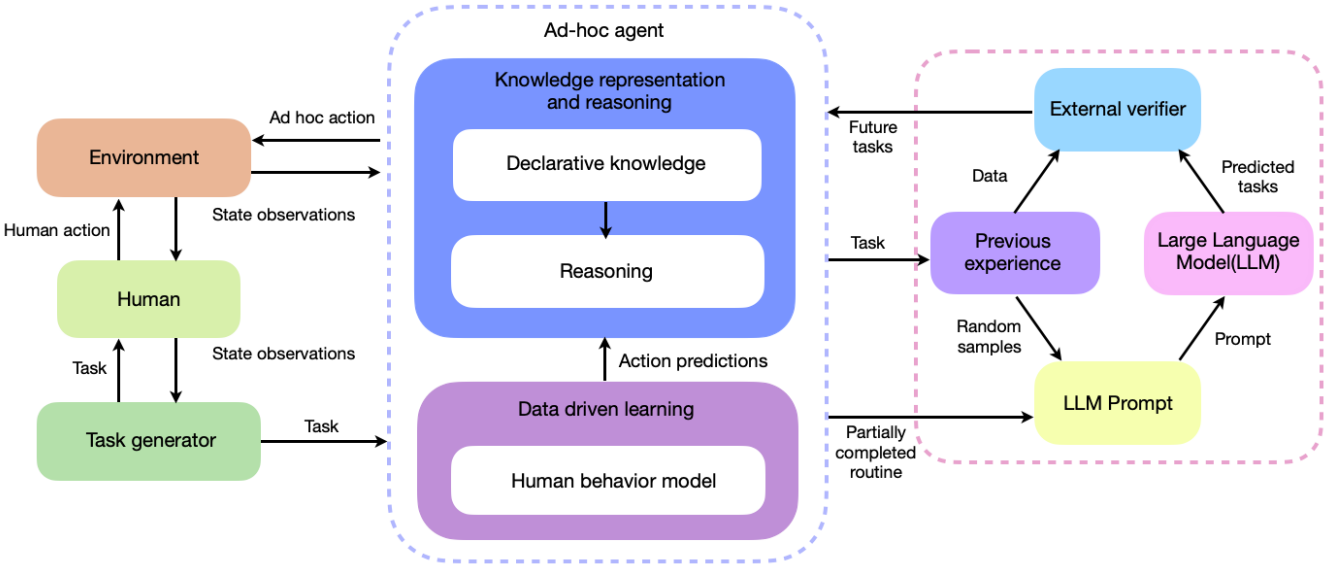

Figure 2: Our architecture combines the complementary strengths of knowledge-based and data-driven reasoning and learning, along with the anticipation capabilities of Large Language Models.

executablility condition such as:

$$open(A, E) \textbf{ causes } opened(E) \quad (1a)$$

$$\neg(at(A, L1), I) \textbf{ if } at(A, L2), I), L1 \neq L2 \quad (1b)$$

$$\textbf{impossible } grab(A, O) \textbf{ if } on(O, E), \; not\ opened(E) \quad (1c)$$

where Statement 1(a), a causal law, implies that an agent(A) opening an appliances(E) causes it to be opened; Statement 1(b), a state constraint, implies that an agent(A) cannot be in two places($L1, L2$) at the same time; and Statement 1(c), an executability condition, prevents the ad hoc agent(A) from trying to grab an object(O) from an appliance(E) with a closed door.

History $\mathcal{H}$ is a record of statements of the form $obs(fluent, boolean, step)$, which represent observations, and statements of the form $hpd(action, step)$, which represent action execution, at specific time steps. It also includes default statements that are true in the initial state.

To reason with knowledge, we automatically construct program $\Pi(\mathcal{D}, \mathcal{H})$ in CR-Prolog (Balduccini and Gelfond 2003), an extension to ASP that supports consistency restoring (CR) rules. $\Pi(\mathcal{D}, \mathcal{H})$ includes statements from $\mathcal{D}$ and $\mathcal{H}$, inertia axioms, reality check axioms, closed world assumptions for defined fluents and actions, helper relations, e.g., $holds(fluent, step)$ and $occurs(action, step)$ to imply that a fluent is true and an action is part of a plan at a time step, and helper axioms that define goals and guide planning and diagnosis. ASP is based on stable model semantics, and encodes *default negation* and *epistemic disjunction*, i.e., unlike "$\neg a$" that states *a is believed to be false*, "$not\ a$" only implies *a is not believed to be true*, and unlike "$p \vee \neg p$", "p *or* $\neg p$" is not tautologous. Each literal is true, false, or unknown, and the agent only believes what it is forced to believe. ASP supports non-monotonic reasoning, i.e., the ability to revise previously held conclusions, which is essential for agents reasoning and acting in practical domains based on incomplete knowledge and noisy observations. The CR rules allow the agent to make assumptions (e.g., that a default statement does not hold) under exceptional circumstances to recover from inconsistencies. All reasoning tasks, i.e., planning, diagnostics, and inference are reduced to computing *answer sets* of Π. We use the SPARC system (Balai, Gelfond, and Zhang 2013) to solve CR-Prolog programs.

The *VirtualHome* scenario is complex, with many objects scattered in different compartments. Achieving goals often requires a computationally expensive process to compute plans with multiple steps. To ensure scalability, we build on prior work in our group on a refinement-based architecture (Sridharan et al. 2019). The ad hoc agent represents and reasons at two resolutions, with the fine-resolution transition diagram defined as a *refinement* of the coarse-resolution one, automatically selecting and using the resolution and the part of the description relevant to the task at hand.

Knowledge-based reasoning and learning methods are often criticized for their need for comprehensive domain knowledge, but existing methods can leverage incomplete knowledge and revise it over time (Sridharan and Meadows 2018). Also, the effort to encode this knowledge is much less than the effort needed to train purely data-driven systems.

3.2 Agent Behavior Models

Practical human-agent collaboration domains change over time, e.g., due to the human's actions. In addition to prior knowledge, our architecture enables the ad hoc agent to reason with models that predict the human's behavior. We use the Ecological Rationality (ER) principle (Gigerenzer 2020), based on Herb Simon's definition of Bounded Rationality and the algorithmic theory of heuristics, to quickly

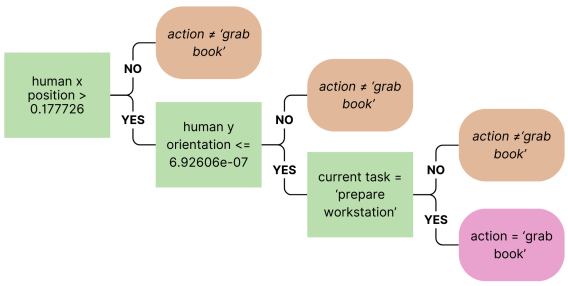

Figure 3: FF tree model of human behavior for the grab_book action in the *VirtualHome* domain.

learn and update predictive behavior models of the human (i.e., the other agent). ER explores decision making "in the wild", i.e., under open world uncertainty with the space of possibilities not fully known, and characterizes behavior as a function of internal cognitive processes and the environment. It prioritizes *adaptive satisficing* because in the absence of comprehensive knowledge, optimal decisions may be unknowable and not just hard to compute. Also, heuristics are used to ignore part of the information to make decisions more quickly, frugally, and/or accurately than complex methods (Gigerenzer and Gaissmaier 2011). This approach has provided better performance that more sophisticated ones in practical applications (Gigerenzer 2016).

Our architecture enables the ad hoc agent to select relevant attributes and learn a model of the human behavior incrementally and from limited data. Specifically, the agent learns an ensemble of *Fast and Frugal* (FF) trees to predict the human's behavior. Each FF tree provides a binary choice for a particular action, and the number of leaves in a tree is limited by the number of attributes (Katsikopoulos et al. 2021). Each level of the tree contains an exit allowing the agent to make quick decisions based on available data. Unlike many current methods for AHT, these predictive models can be learned and revised incrementally and rapidly, which is a useful capability for AHT.

Figure 3 shows one such FF tree learned for the human. Our trees are built while minimizing false positives. The initial version of these trees were built using only 1000 traces of human action choices and domain state from the *VirtualHome* domain, with the corresponding attributes listed in Table 1. Since each FF tree provides a binary choice we build an ensemble of trees with a simple decision tree considering the output from the FF trees. Furthermore, consistent agreement (or disagreement) between observations and model predictions can trigger model selection or revision of the existing models, allowing the ad hoc agent to quickly adapt to changes in the domain or the human's behavior.

3.3 Task Anticipation with LLMs

While LLMs have demonstrated impressive statistical prediction capabilities, they (by themselves) are not capable of effective planning or self-validation (Kambhampati et al. 2024). They have been shown to be much more effective when they are used to generate generic (high-level) plans

Description of the attribute
Immediate two previous actions of the human
Position of the human (x,y,z)
Orientation of the human (x,y,z)
Any objects in the hand of the agent
Current and previous tasks
Flags (day of the week, going to office, guests expected)

Table 1: Attributes used to create the behavior models of the human in the *VirtualHome* domain.

that are then validated externally to ensure correctness and feasibility before using suitable subroutines to implement each plan step (Kambhampati et al. 2024). LLMs can thus also be viewed as good translators between natural language and domain-specific languages that help refine and correct planning models (Guan et al. 2023).

Motivated by the above findings, we use LLMs in our architecture to anticipate high-level future tasks to be executed. In the absence of the LLM, the ad hoc agent and the human come to know of tasks (to be completed in in *VirtualHome*) one at a time. When the LLM is included in the architecture, the ad hoc agent obtains the anticipated next task/goal that is likely to be assigned once the current task is done. By jointly considering the current and next tasks, the ad hoc agent can come up with a better plan of actions that can enhance the overall team performance.

Figure 4 show an example input prompt for the LLM in *VirtualHome* by the ad hoc agent. We employ three prompt engineering strategies for refining LLM performance.

1. **Adopting a persona:** Assign a specific role or character to the LLM model to guide its responses. This helps the model generate responses that are consistent with the assigned role and contextually appropriate.
2. **Few-shot prompting:** The model is provided some examples within the prompt, guiding the use of pretrained knowledge to perform any given task with limited data.
3. **Chain-of-thought (CoT):** Step-by-step reasoning process that could be followed when arriving at an answer. This helps the model generate intermediate reasoning steps leading to more accurate and relevant responses.

More specifically, we asked the LLM to adopt the persona of an intelligent household assistant for more accurate and contextually appropriate responses. Then we utilized 'few-shot' prompting by including two example task routines from previous days in the prompt. These examples were dynamically selected from a list of past tasks, randomly chosen from two different days to provide as examples to the LLM. These example routines can be at various stages of completion: one with no completed task for the LLM to predict, and another partially completed routine for the LLM to continue. Finally we utilized the CoT method to describe the reasoning behind each of the selected tasks of the examples in a day. This provided the LLM with additional context and guidance allowing for rapid adaptation, which is critical in AHT settings. We provided the LLM with a list of possible high-level tasks

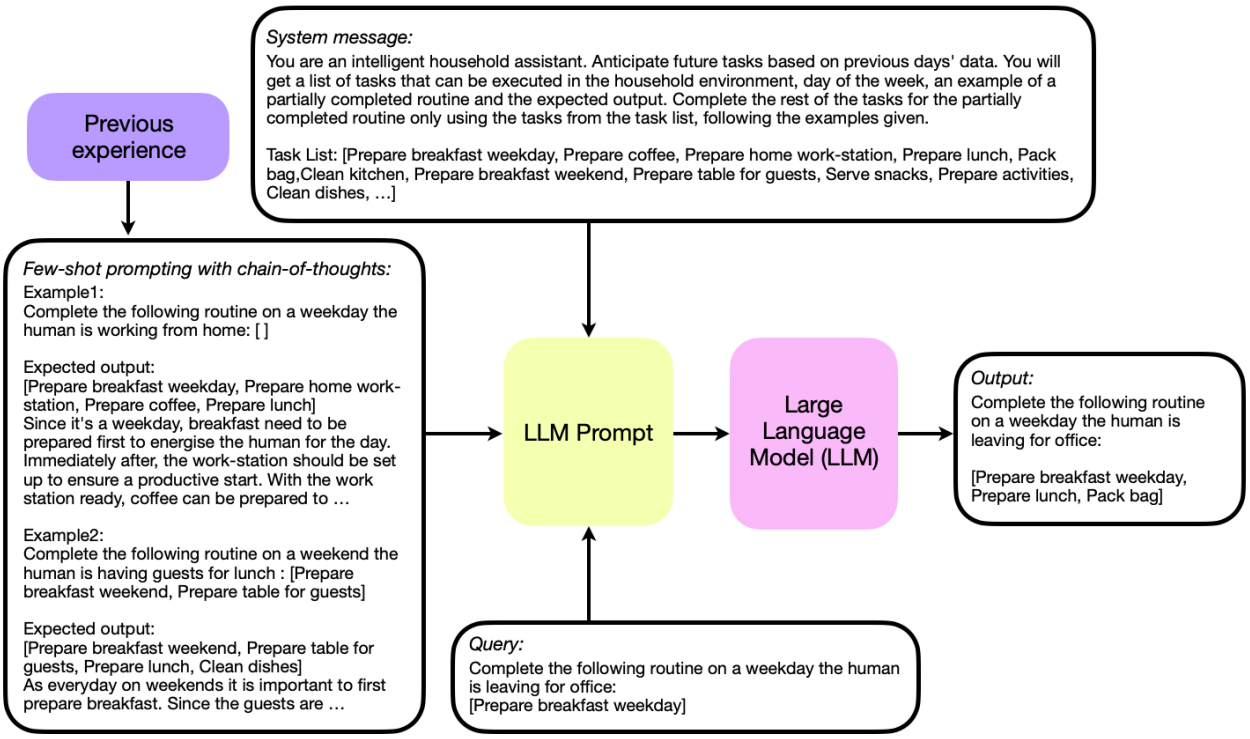

Figure 4: Example prompt sent to the LLM to anticipate the human's future tasks. Each prompt includes a list of possible tasks, two partially completed example routines, and a partially completed routine for the LLM to complete.

in the *VirtualHome* domain. This enabled the LLM to provide more actuate and contextual responses. We then asked the LLM to complete a routine for a given day.

The tasks predicted by the LLM are parsed by an external validator. Specifically, the future task list from the LLM output is compared with the existing knowledge in the ASP program and previous experience (i.e., observations) to ensure that the tasks are feasible (for the given day) and that the task order is reasonable. Given that the LLM is not trained in the specific domain, there may be a difference between the task order preferred by a human in *VirtualHome* and the order suggested by the LLM. Thus, we re-order the tasks if they are not in a reasonable (or preferred) order. For example, the agent may need to prioritize preparing the workstation for work over packing a bag for shopping when the human is working from home, or the human may prefer breakfast before starting on coffee. This ensures plan feasibility and prioritizes tasks that require more attention.

The validated high-level LLM-based task predictions can be set as joint goals to be achieved. In our architecture, the ad hoc agent considers one anticipated (future) task along with the current task and the predicted human actions toward the current task, planning a sequence of actions that it then executes. We show that team performance is much better with the LLM-based anticipation than without it.

4 Experimental Setup and Results

We experimentally evaluated the following hypotheses regarding the capabilities of our architecture:

- **H1:** Reasoning with prior domain knowledge and predictive behavior models improves team performance;
- **H2:** Considering anticipated tasks from the LLM as joint goals improves team performance compared with pursuing one goal at a time;
- **H3:** Incrementally-updated prompts of recent experiences and external validators lead to better LLM-based task anticipation and improved team performance; and
- **H4:** Using LLMs to predict low-level actions(planning) in complex domains results in poor performance.

We evaluated these hypotheses in *VirtualHome* with the latest GPT-4o-mini LLM. Each episode continued until the human-agent team completed all tasks for the given day. We recorded the number of steps (plan length) and the total time taken by the team (i.e., human and embodied AI agent) to complete the task as the *performance measures*. More details of the experiments and baselines are provided below.

4.1 Experimental Setup

In the *VirtualHome* domain, we modeled the human as a simulated entity that selects its actions based on an ASP program. This program (and thus the human) does not reason about the agent's capabilities. An external task generator

computes a sequence of tasks, but the human and the ad hoc agent only receive one task (to be completed) at a time. Both the human and the ad hoc agent received the same observations from the domain at each step, which they used to plan their respective actions in the domain.

When the ad hoc agent received a task from the task generator, it prompted the LLM for any future tasks. As described in Section 3.3 (also see Figure 4), these anticipated tasks were validated to check their feasibility and execution order. The validated goals were then mapped to a ASP goal and set as the goal of the ad hoc agent together with the goal received from the task generator. In our implementation, the ad hoc agent planned jointly for the goal from the task generator and the next (anticipated) task from the LLM. During this planning process, the ad hoc agent also used the learned human behavior prediction model to predict the possible future actions of the humans, as described in Section 3.2. Recall that these predictive models were built only using 1000 examples of prior traces of human actions and domain state.

The ASP program of the ad hoc agent included additional axioms for reasoning about these predicted actions (of the human) that are mapped to exogenous actions. As a result, the ad hoc agent's plan anticipates that the preconditions for some intermediate steps will be created by the human's actions, even though the human may not always do so. On the other hand, the human cannot predict the ad hoc agent's actions, and the ASP program governing the human's action choices (in simulation) does not include axioms for reasoning about these actions, leading to the human's actions being determined primarily by the current state and goal.

For evaluating **H1** and **H2** in **Exp1**, we randomly selected 100 task routines and measured the ability of the team comprising a human and an embodied AI agent to successfully complete these tasks in *VirtualHome*. The performance measures were the number of steps and time taken to complete the tasks. We used three specific baselines to evaluate the contribution of each component of our architecture.

- **Base1**: used the LLM for anticipating future tasks, but did not use the predictive behavior models to anticipate the future actions of the human.
- **Base2**: did not use the LLM to predict the future tasks, but used the predictive behavior models to anticipate the human's future actions.
- **Base3**: did not use the LLM for task anticipation or behavior models to predict the human's future actions.

The agent only planned for current task in **Base2** and **Base3**.

Next, for evaluating **H3** in **Exp2**, we randomly selected 20 task routines from the domain and recorded the team performance (number of steps and task completion time) when the LLM used different prompt-engineering methods (adopting a persona, few-shot prompting, CoT; Section 3.3) and when the LLM did not use the prompt-engineering methods. We used three baselines to evaluate the importance of each prompt-engineering method and the external validator.

- **Base4**: did not use any specific prompt-engineering method or external validator.
- **Base5**: used few-shot prompting (i.e., two examples of partially completed routines and their expected answers included in the prompt) but no external validator.
- **Base6**: used CoT prompting i.e., each example was accompanied by a step-by-step explanation on arriving at the answer, but no external validator.
- **Base7**: used external validator to filter (and reorder) LLM output but no prompt-engineering methods.

For evaluating **H4**, we conducted a special experiment where we extended our architecture to utilize the LLM for low-level planning (**Base8**). Specifically, we extended our prompt to include information on the low-level actions that can be executed in *VirtualHome* along with their intended purposes, such as: move(agent, location): move the agent to an adjacent location; grab(agent, object): pick up an object. We also supplied the LLM with **Action Feasibility Rules**:

- *Movement Limitation* (critical): you must only move to adjacent locations defined by the next_to relationships. Any move to a destination that is not adjacent is prohibited. Always check adjacency before predicting a move.
- *Object Location*: you must be in the same location as an object to perform any action on it (e.g., grab or put).
- *Carrying Limit*: you cannot hold more than two objects at a time. Also, when holding two objects, actions like open, close, switch-on, or switch-off require you to put down at least one object first.
- *Appliance Safety*: for safety, you should not open appliance doors when they are switched on.
- If the human is holding an object, assume they will handle all tasks involving the object. Do not attempt to grab or interact with the object the human is holding. Instead, focus on parts of the goal unrelated to that object.

Then we included information about adjacent places in the domain emphasizing the fact that the agent can only move between the defined adjacent places.

$$next\_to(counter\_one, kitchen\_table)$$
$$next\_to(livingroom\_desk, livingroom\_coffeetable)$$

The LLM had access to the current world state, including the location of the human, agent, objects, and appliances, each appliance's state, and information about the objects held by the agent or human. The problem specification also described the task to be performed; the immediate previous actions of the human and the agent; any specific information to be considered on any given day (e.g., human working from home or the office); whether it is a weekday or the weekend; and whether the human is expecting guests. In addition, the prompt included a detailed example of selecting an action, and asked the LLM to generate a plan for achieving the assigned goal and specify the next action to execute.

The action choice by the LLM was then assigned as the action of the ad hoc agent. As a recovery mechanism, we corrected identified errors in the LLM output up to three times per trial. For example, if the LLM selected an action that was not feasible in its current location, such as attempting to grab an object without moving to the appropriate location or trying to move to a non-adjacent place, we provided

feedback explaining why the action choice was incorrect. The feedback allowed the LLM to predict another action for that step, and we measured the ability of the human-ad hoc agent team to achieve success in completing the tasks in the previously selected 100 task routines. As before, the performance measures were the number of steps and time taken to complete the set of tasks.

4.2 Experiment Results

Table 2 summarizes the results of **Exp1**. When the ad hoc agent used task anticipation from the LLM and the human action prediction based on the behavior prediction models, it resulted in the best performance with lowest number of steps and time to complete the task routines. When the ad hoc agent used the LLM for future task anticipation but not the output from the behavior prediction models (**Base1**), both the number of steps and time taken to complete the tasks increased. This can be attributed to the fact that being unable to anticipate the actions of the human teammate may lead the ad hoc agent to select the same actions as the human. Such conflicts prevent effective collaboration between the human and the ad hoc agent. These results emphasize the importance of using the behavior prediction models in planning and support hypothesis **H1**.

When the ad hoc agent used the behavior prediction models to predict the future actions of the human, but did not use the LLM to anticipate possible future tasks (**Base2**), the performance worsened further, with a further increase in the number of steps and the time taken to complete the task. Recall that this setting corresponds to the agent only planning for one goal at a time and not anticipating future goals. Planning and executing actions jointly for the current goal and the goal likely to be assigned in the immediate future often saves both time and effort, and this benefit is lost when the ad hoc agent does not anticipate future tasks. For example, when the agent visited the bedroom to retrieve a board game to entertain guests, it could have also picked up bottles of wine from the cellar that is on the way. Instead, making two separate trips for these tasks unnecessarily extends the length and duration of the the plans. These results indicate the impact that planning for joint goals has on overall performance, supporting hypothesis **H2**.

Base3 used neither the future task anticipation from the LLM nor the predictions of human's action based on the behavior models. This led to worst observed performance with the highest value for both number of steps and time taken by the team to complete the assigned tasks. These results further support hypotheses **H1** and **H2**.

Table 3 show the results from **Exp2**, where we used the LLM to predict the future tasks in *VirtualHome* with and without the external validator and the prompt-engineering methods discussed in Section 3.3. We observed a marked improvement in performance in the form of lower number of steps and time taken to successfully complete the task routines when the external validator and a combination of prompting methods were used. In particular, the use of external validator to check and correct the output from the LLM based on domain-specific information had a significant impact on performance. These results support **H3**.

Architecture	*Steps*	*Time(s)*
Proposed (anticipate tasks, predict actions)	26.8	361.0
Base1 (anticipate tasks)	29.3	385.8
Base2 (predict actions)	34.1	443.0
Base3	37.5	487.4

Table 2: Average number of steps and time taken by team (human, ad hoc agent) to complete the task routines; LLM-based task anticipation and FF trees-based human action prediction substantially improve performance.

Architecture	*Steps*	*Time(s)*
Proposed (all prompting, with validator)	27.5	372.7
Base4 (no prompting, no validator)	33.1	427.7
Base5 (few-shot prompting, no validator)	32.1	441.1
Base6 (chain-of-thoughts, no validator)	31.7	430.8
Base7 (no prompting, with validator)	28.9	387.5

Table 3: Average number of steps and time taken by the team (human and ad hoc agent) to successfully complete the tasks when using the prompt-engineering methods in conjunction with the external validators.

Table 4 shows the results of **Exp3**, where we used the LLM to directly anticipate the low-level actions for the ad hoc agent in the *VirtualHome* domain. The number of steps and time taken to successfully complete the task routines are significantly higher than our architecture as well as all the baselines (**Base1-3**). These results support **H4**.

We also performed a qualitative evaluation of our architecture's capabilities, particularly of hypothesis **H3**. Figure 5 shows an execution example from the *VirtualHome* domain using the LLM without the prompt-engineering methods or external validator. In this scenario, as explained in the prompt, the LLM was asked to predict a list of tasks a human would perform on a weekday while working from home. The expected task list from the LLM was [*Prepare breakfast weekday, Prepare home work-station, Prepare coffee, Prepare lunch*]; i.e., making breakfast and setting up the workstation were considered to the time-critical tasks to be completed first before proceeding to other tasks. When not using the prompt-engineering methods and external validator, the task list predicted by the LLM was [*Prepare breakfast weekday, Prepare coffee, Prepare home work-station, Pack bag*]. This task routine did not align with the human preferences; more importantly, it led to a conflicting situation in which making coffee was given higher priority than

Architecture	*Steps*	*Time*
Base8 (LLM predict low-level actions)	39.7	520.0

Table 4: Average number of steps and time taken by the human-ad hoc agent team to successfully complete the task routines when using the LLM to predict low-level actions that can be directly executed in the *VirtualHome* domain.

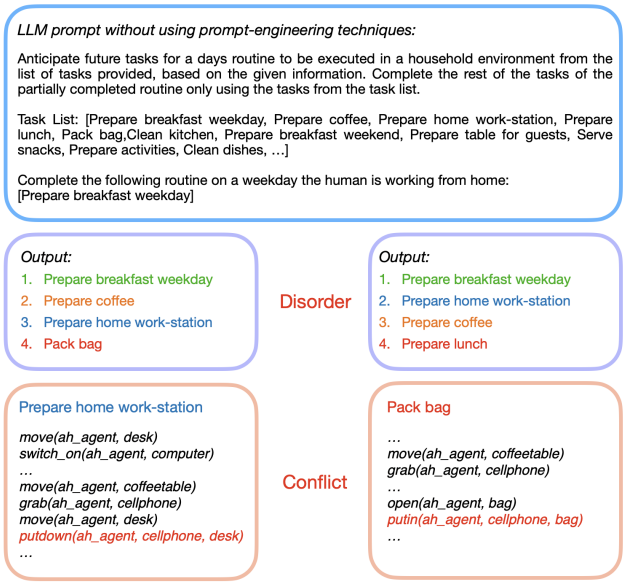

Figure 5: Execution example illustrating how the use of LLM in the absence of suitable prompt-engineering methods or an external validator causes conflicts during plan execution.

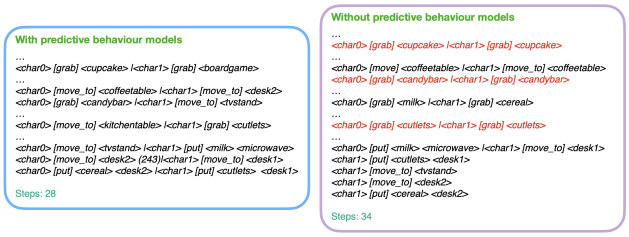

Figure 6: Execution example illustrating the use of behavior prediction models to improve performance. The tasks executed are *[Prepare breakfast weekend, Prepare activities, Serve snacks, Clean kitchen]*. When the ad hoc agent is unable to predict the human's actions, it frequently chooses to execute same action as the human leading to longer plans.

setting up the workstation. This would lead to the human being late for work and the coffee not being hot when the human was ready to drink it. Additionally, packing the bag was an unnecessary action that prevented the ad hoc agent from making proper use of its time in collaborating with the human. In addition, when the agent used the prompt-engineering methods described in Section 3.3, the task list predicted by the agent was [*Prepare breakfast weekday, Prepare home work-station, Prepare coffee, Clean dishes*]. This task routine was then further filtered by the external validator to remove the non-feasible task of *Clean dishes*. These results further support **H3**.

Figure 6 shows an execution example of the ad hoc agent executing the tasks *[Prepare breakfast weekend, Prepare activities, Serve snacks, Clean kitchen]*. When the agent did not use behavior prediction models, it often selected the same actions as the human when pursuing any particular task/goal, leading to unnecessary delays in completing the assigned tasks. However when the agent use the FF trees-based behavior prediction models, it avoided such unnecessary actions and delays, resulting in more efficient plans. These results further support hypothesis **H1**.

5 Conclusions

This paper described an architecture for Ad Hoc Teamwork (AHT), enabling an embodied AI agent to collaborate with a human by reasoning with prior commonsense domain knowledge and incrementally learned models that predict the human behavior. The architecture integrates principles of non-monotonic logical reasoning and ecological rationality, automatically identifying and reasoning with relevant information. It leverages the generic knowledge and predictive capabilities of LLMs for high-level task and goal anticipation based on limited prompts, translating them into concrete actions executed within a realistic physics-based simulation environment (*VirtualHome*). We demonstrate the architecture's improved performance compared with various baselines, highlighting the significance of each component in our design. In future work, we aim to explore the scalability of our architecture to multiple ad hoc agents and humans, as well as its implementation and evaluation on physical robots in AHT settings.

References

Balai, E.; Gelfond, M.; and Zhang, Y. 2013. Towards Answer Set Programming with Sorts. In *Conference on Logic Programming and Nonmonotonic Reasoning*.

Balduccini, M.; and Gelfond, M. 2003. Logic Programs with Consistency-Restoring Rules. In *AAAI Spring Symposium on Logical Formalization of Commonsense Reasoning*.

Barrett, S.; Rosenfeld, A.; Kraus, S.; and Stone, P. 2017. Making friends on the fly: Cooperating with new teammates. *Artificial Intelligence*, 242: 132–171.

Barrett, S.; Stone, P.; Kraus, S.; and Rosenfeld, A. 2013. Teamwork with Limited Knowledge of Teammates. In *AAAI Conference on Artificial Intelligence*, volume 27.

Bowling, M.; and McCracken, P. 2005. Coordination and Adaptation in Impromptu Teams. In *National Conference on Artificial Intelligence*, 53–58.

Chen, S.; Andrejczuk, E.; Cao, Z.; and Zhang, J. 2020. AATEAM: Achieving the Ad Hoc Teamwork by Employing the Attention Mechanism. In *AAAI*.

Dodampegama, H.; and Sridharan, M. 2023a. Back to the Future: Toward a Hybrid Architecture for Ad Hoc Teamwork. In *AAAI Conference on Artificial Intelligence*.

Dodampegama, H.; and Sridharan, M. 2023b. Knowledge-based Reasoning and Learning under Partial Observability in Ad Hoc Teamwork. *Theory and Practice of Logic Programming*, 23(4): 696–714.

Fang, Q.; Zeng, J.; Xu, H.; Hu, Y.; and Yin, Q. 2024. Learning Ad Hoc Cooperation Policies from Limited Priors via Meta-Reinforcement Learning. *Applied Sciences*, 14(8).

Gelfond, M.; and Inclezan, D. 2013. Some Properties of System Descriptions of AL_d. *Applied Non-Classical Logics, Special Issue on Equilibrium Logic and ASP*, 23(1-2).

Gelfond, M.; and Kahl, Y. 2014. *Knowledge Representation, Reasoning and the Design of Intelligent Agents*. Cambridge University Press.

Gigerenzer, G. 2016. *Towards a Rational Theory of Heuristics*, 34–59. London: Palgrave Macmillan UK.

Gigerenzer, G. 2020. What is Bounded Rationality? In *Routledge Handbook of Bounded Rationality*. Routledge.

Gigerenzer, G.; and Gaissmaier, W. 2011. Heuristic Decision Making. *Annual Review of Psychology*, 62.

Guan, L.; Valmeekam, K.; Sreedharan, S.; and Kambhampati, S. 2023. Leveraging Pre-trained Large Language Models to Construct and Utilize World Models for Model-based Task Planning. In Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and Levine, S., eds., *Advances in Neural Information Processing Systems*, volume 36, 79081–79094. Curran Associates, Inc.

Kambhampati, S.; Valmeekam, K.; Guan, L.; Verma, M.; Stechly, K.; Bhambri, S.; Saldyt, L.; and Murthy, A. 2024. LLMs Can't Plan, But Can Help Planning in LLM-Modulo Frameworks. arXiv:2402.01817.

Katsikopoulos, K.; Simsek, O.; Buckmann, M.; and Gigerenzer, G. 2021. *Classification in the Wild: The Science and Art of Transparent Decision Making*. MIT Press.

Liu, X.; Li, P.; Yang, W.; Guo, D.; and Liu, H. 2024. Leveraging Large Language Model for Heterogeneous Ad Hoc Teamwork Collaboration. arXiv:2406.12224.

Macke, W.; Mirsky, R.; and Stone, P. 2021. Expected Value of Communication for Planning in Ad Hoc Teamwork. In *AAAI Conference on Artificial Intelligence*.

Mirsky, R.; Carlucho, I.; Rahman, A.; Fosong, E.; Macke, W.; Sridharan, M.; Stone, P.; and Albrecht, S. 2022. A Survey of Ad Hoc Teamwork: Definitions, Methods, and Open Problems. In *European Conference on Multiagent Systems*.

Puig, X.; Ra, K.; Boben, M.; Li, J.; Wang, T.; Fidler, S.; and Torralba, A. 2018. Virtualhome: Simulating household activities via programs. In *International Conference on Computer Vision and Pattern Recognition*, 8494–8502.

Rahman, M. A.; Hopner, N.; Christianos, F.; and Albrecht, S. V. 2021. Towards Open Ad Hoc Teamwork Using Graph-based Policy Learning. In *International Conference on Machine Learning*, 8776–8786.

Ravula, M.; Alkoby, S.; and Stone, P. 2019. Ad Hoc Teamwork With Behavior Switching Agents. In *International Joint Conference on Artificial Intelligence*.

Ribeiro, J. G.; Rodrigues, G.; Sardinha, A.; and Melo, F. S. 2023. TEAMSTER: Model-based reinforcement learning for ad hoc teamwork. *Artificial Intelligence*, 324: 104013.

Savva, M.; Kadian, A.; Maksymets, O.; Zhao, Y.; Wijmans, E.; Jain, B.; Straub, J.; Liu, J.; Koltun, V.; Malik, J.; Parikh, D.; and Batra, D. 2019. Habitat: A Platform for Embodied AI Research. *CoRR abs/1904.01201*.

Shi, Z.; Fang, M.; Zheng, S.; Deng, S.; Chen, L.; and Du, Y. 2023. Cooperation on the Fly: Exploring Language Agents for Ad Hoc Teamwork in the Avalon Game. arXiv:2312.17515.

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

Sridharan, M.; and Meadows, B. 2018. Knowledge Representation and Interactive Learning of Domain Knowledge for Human-Robot Collaboration. *Advances in Cognitive Systems*, 7: 77–96.

Stone, P.; Kaminka, G.; Kraus, S.; and Rosenschein, J. 2010. Ad Hoc Autonomous Agent Teams: Collaboration without Pre-Coordination. In *AAAI Conference on Artificial Intelligence*, 1504–1509.

Zand, J.; Parker-Holder, J.; and Roberts, S. J. 2022. On-the-Fly Strategy Adaptation for Ad-Hoc Agent Coordination. In *International Conference on Autonomous Agents and Multiagent Systems*, 1771–1773.

Zintgraf, L.; Devlin, S.; Ciosek, K.; Whiteson, S.; and Hofmann, K. 2021. Deep Interactive Bayesian Reinforcement Learning via Meta-Learning. In *International Conference on Autonomous Agents and Multiagent Systems*.