Explainable Reasoning and Learning in Ad Hoc Teamwork

Hasra Dodampegama and **Mohan Sridharan** [1]

Abstract. An assistive AI agent often has to collaborate with previously unseen agents and humans. Methods considered state of the art for such *ad hoc teamwork* use a large labeled dataset of prior observations to model the behavior of other agents and to determine the ad hoc agent's behavior. These approaches are resource-hungry, and do not support rapid incremental revisions or transparency, with the necessary resources (e.g., training examples, computation) not readily available in practical domains. Our architecture for ad hoc teamwork embeds the principles of refinement, ecological rationality, interactive learning, and explainable agency, leveraging the complementary strengths of knowledge-based and data-driven methods. For any given goal, an ad hoc agent determines its actions through non-monotonic logical reasoning with: (a) prior domain-specific commonsense knowledge; (b) models learned rapidly to predict the behavior of other agents; and (c) anticipated abstract future tasks based on generic knowledge of similar situations. Further, the agent generates relational descriptions as explanations of its decisions and those of other agents. We evaluate the capabilities of our architecture in *VirtualHome*, a realistic 3D simulation environment.

1 Introduction

Consider an AI agent performing household tasks in collaboration with other agents (AI, human) it has not worked with before. Figure 1 shows snapshots of this motivating scenario in a multiagent simulation environment in which an AI agent (blue shirt) and a human (green top) are preparing breakfast and setting up a workstation. The AI agent has to reason with different descriptions of prior knowledge and uncertainty in the form of qualitative statements ("eggs are usually in the fridge") and quantitative statements ("I am 90% sure I saw the eggs on the table"). The agents have a limited view of the environment and do not communicate with each other, although each of them is aware of the domain state and action outcomes, e.g., the location of teammates and objects moved by a teammate. Furthermore, the human may want to query and understand the AI agent's decisions. These characteristics correspond to *Ad hoc Teamwork* (AHT), requiring cooperation "on the fly" without prior coordination [45]. The AHT problem arises in many practical applications such as disaster rescue, exploration, and collaborative games, and poses challenges in knowledge representation, reasoning, and learning [28].

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 [28]. However, it is difficult to gather training samples of different situations in practical domains. Also, these methods are usually opaque, offering limited insight into how or why an agent made a decision. This lack of transparency makes it challenging to fix faults and to foster trust in automated decision-making. In a departure from existing work, our AHT architecture leverages the complementary strengths of knowledge-based and data-driven methods for reasoning and learning to jointly address the underlying challenges. It embeds fundamental principles such as refinement, ecological rationality, interactive learning, and explainable agency, which can be traced back to the early pioneers of AI [41], enabling each ad hoc agent in a team to:

1. Adapt scalably through non-monotonic logical reasoning with prior commonsense domain knowledge and rapidly-learned predictive models of other agents' behavior;
2. Leverage a Large Language Model (LLM) to anticipate future tasks to be completed, adapting the LLM's output based on domain-specific knowledge and experience; and
3. Automatically construct on-demand relational descriptions of its decisions and those of the others as *explanations* in response to different types of questions

Our prior papers have described subsets of these capabilities, e.g., proof of concept reasoning [8], simple explanation generation [9], and scalability [10], and a more detailed description will appear as a journal article [11]. Here, we highlight the benefits of leveraging the interplay between reasoning and learning, particularly in the context of explanation generation. We evaluate the architecture's capabilities in *VirtualHome*, a realistic 3D simulation environment [32], with simulated agents collaborating to perform household tasks.

Figure 1. Screenshots from *VirtualHome* [32] showing a human (female in green top) and an assistive AI agent (male in blue shirt) collaborating.

2 Related Work

Researchers have explored AHT for more than two decades under different names [28]. Initial work used predefined protocols or plays

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

that encoded specific actions for the ad hoc agent in specific scenarios [6]. Subsequent work used probabilistic and sampling-based methods such as Upper Confidence bounds for Trees to determine the ad hoc agent's actions [5, 35]. Recent approaches considered state of the art for AHT have posed it primarily as a learning problem; the key component predicts behavior of other agents and determines the ad hoc agent's behavior using a history of prior interactions with similar agents or agent types. This includes the use of Fitted Q Iteration to learn action selection policies from offline data of each teammate type [4]; attention-based recurrent neural networks for real-time adaptation [7]; graph neural networks to simulate interactions with other agents (agent types) and determine the ad hoc agent's behavior [33]; self-play to learn a cooperation policy and candidate teammate policies [14]; sequential and hierarchical variational auto encoders to model teammates' changing behaviors [47]; and model-based RL methods to learn separate models of the environment and teammates [46]. Also, an LLM-based framework has been developed to compute the ad hoc agent's actions [26].

While the frameworks summarized above offer promising strategies for modeling the behavior of teammates and selecting the actions of the ad hoc agent, they are resource-hungry, requiring substantial time, computation, and training examples to achieve the observed performance. Such resources are not readily available in practical AHT domains, posing questions about the scalability and usability of these methods. Also, they struggle to leverage the commonsense knowledge available in many practical domains, and their internal decision-making mechanisms are often opaque, limiting the ability to understand, justify, and audit the agents' automated decisions.

Embodied AI refers to AI systems operating within physically realistic (simulation) environments such as Habitat [38] and VirtualHome [32]. These interactive platforms support the generation of complex scenarios for evaluating algorithms for single-agent and multi-agent collaboration problems. With the increasing use of embodied AI agents and AI methods in different applications, there is renewed focus on transparency and explainability of decision-making [27, 37], although this has been a well-researched area for decades [34, 30]. With such AI methods being used to interact with other agents and humans, multiple theories and approaches have been developed and adapted to define and use causal relationships to provide explanations [21], categorize methods in terms of their ability to extract causal relationships and provide explanation [31], and consider the recipient's prior knowledge while generating user-centric explanations [13, 15]. Instead of assuming comprehensive prior knowledge or exploring black box models, we seek to design architectures that incorporate interpretable models [36] and automatically support process-level explanations based on knowledge that is evolving over time. To achieve this objective, work in our group has designed architectures that embed the principle of *explainable agency* [24, 25] and build on a theory for explanation generation [44], enabling an agent to construct relational descriptions as explanations in response to different types of questions [29]. The novelty is in the processes used to achieve the desired functional capabilities. In this paper, we extend these ideas and leverage the principle of ecological rationality [19] to provide explanations in the context of multiagent (ad hoc) collaboration.

3 Architecture

Example Domain 1 *[Example Embodied AI Agent Domain]* Consider an embodied AI agent and a human agent collaborating to complete household tasks; Figure 1 shows snapshots while preparing breakfast [32]. The agents can interact with the environment through high-level actions, e.g., move to places, pick up or place objects, switch appliances on or off, and open or close appliances. Completing a task requires a sequence of such actions to be computed and executed by the agents who do not communicate directly with each other. Also, the agent assumes that the other agents have access to the same state information and will make (what it considers as) rational decisions and computes its plan to complete the current task and prepare for the upcoming task(s). Prior commonsense knowledge of the embodied agent includes relational descriptions of some attributes of the domain, ad hoc agent, and the human agent (Section 3.1); a learned (or encoded) graph of information about likely locations of objects in the domain; and default statements that hold in all but a few exceptional circumstances. The knowledge also includes some axioms governing actions and change, e.g., the agent or human can only pick one object at a time, and certain food items require preparation prior to consumption.

3.1 Knowledge Representation and Reasoning

In our architecture, the transition diagram of any given domain is described using an extension of action language $\mathcal{AL}_d$ [17]. 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 domain attributes (statics, fluents) and actions described in terms of these sorts. Basic sorts in our example scenario include $object$, $appliance$, ad_hoc_agent, $human$, and $step$ (for temporal reasoning), and are arranged hierarchically, e.g., $apple$ is a sub-sort of $food$, a sub-sort of $graspable$, a sub-sort of $object$. Actions can be $agent\_actions$ or $exo\_actions$ (exogenous actions). The $agent\_actions$ such as $grab(ad\_hoc\_agent, object)$, $switch\_on(ad\_hoc\_agent, appliance)$, $move(ad\_hoc\_agent, location1, location2)$ are performed by the ad hoc agent. The $exo\_actions$ such as $exo\_grab(other\_agent, object)$, $exo\_switch\_on(other\_agent, appliance)$ are performed by other agents, e.g., human or another AI agent. Statics are domain attributes whose values cannot change. Fluents are attributes whose values can change; they can be *inertial*, which obey inertia laws and are changed by the ad hoc agent's actions, e.g., $at(ad\_hoc\_agent, location)$ is the ad hoc agent's location, or *defined*, which do not obey inertia laws and are not directly changed by the ad hoc agent's actions, e.g., $agent\_at(other\_agent, location)$ is a teammate's location computed by (say) external sensors.

Given Σ, the domain's dynamics are described in $\mathcal{D}$ using three types of axioms: *causal law*, *state constraint*, and *executablility condition* such as:

$$grab(A, O) \textbf{ causes } in\_hand(A, O) \tag{1a}$$

$$heated(F) \textbf{ if } on(F, E), switched\_on(E) \tag{1b}$$

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

where Statement 1(a), a causal law, implies that grabbing an object causes it to be in the hand of the ad hoc agent; Statement 1(b), a state constraint, implies that a food item placed in an electrical appliance (e.g., microwave) that is switched on gets heated; and Statement 1(c), an executability condition, prevents the ad hoc agent from trying to grab an object from an appliance with a closed door. History $\mathcal{H}$ is a record of observations of the form $obs(fluent, boolean, step)$, and action executions of the form $hpd(action, step)$ at specific time

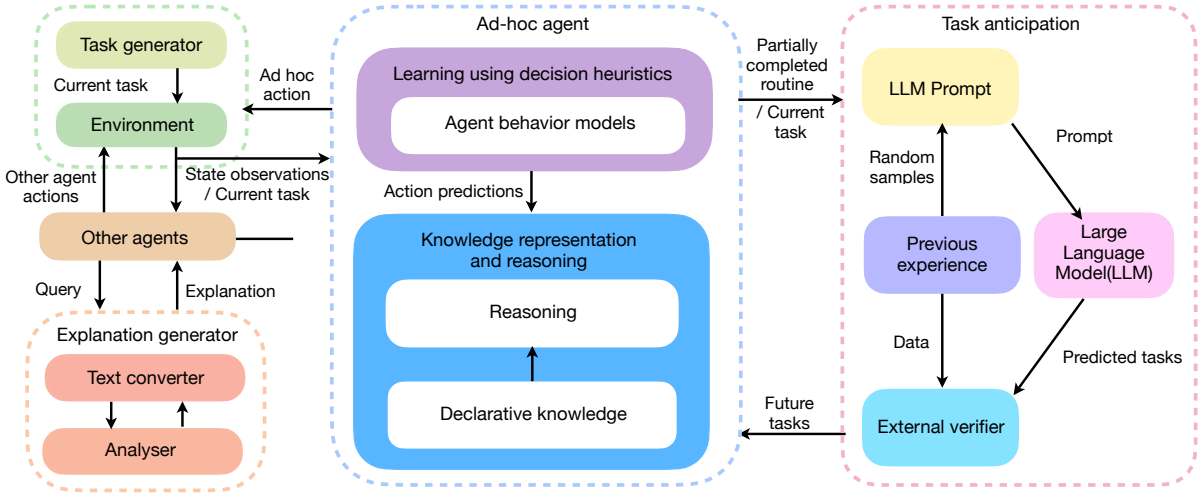

Figure 2. Our architecture leverage the complementary strengths of knowledge-based and data-driven reasoning and learning.

steps. It also includes default statements that are true in all but a few exceptional circumstances, e.g., books are usually in the library but cookbooks are in the kitchen.

To reason with knowledge, our architecture uses a script to automatically construct program $\Pi(\mathcal{D}, \mathcal{H})$ by translating the system description $\mathcal{D}$ and history $\mathcal{H}$ to CR-Prolog [3], 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 encodes *default negation* and *epistemic disjunction*, and supports non-monotonic logic reasoning. This ability to revise previously held conclusions 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 then reduced to computing *answer sets* of Π. We use the SPARC system [2] to solve CR-Prolog programs. Example programs and results are in our open source repository [12].

Our current application domain is substantially more complex than those in our prior work, with many more objects and actions; the corresponding domain description is more complex and the tasks require much longer plans. To enable the use of the KR formalism (above), our architecture incorporates new strategies to constrain the search space. Specifically, a fine-resolution description ($\mathcal{D}_F$) is defined as a *refinement* of a coarse-resolution description ($\mathcal{D}_C$), with the agent now able to reason about aspects of the domain that were previously abstracted away. In our example scenario, the domain is organized into abstract regions in $\mathcal{D}_C$, with each region being refined in $\mathcal{D}_F$ into smaller regions that are components of the larger region, e.g., kitchen in $\mathcal{D}_C$ comprises kitchen table, kitchen counter, refrigerator in $\mathcal{D}_F$. In a similar manner, an object in $\mathcal{D}_C$, e.g., a cup, can comprise multiple parts, e.g., handle and body, in $\mathcal{D}_F$. The signature Σ_F of $\mathcal{D}_F$ is created by expanding Σ_C of $\mathcal{D}_C$ to include the new sorts, actions, fluents, and statics. Next, the axioms of $\mathcal{D}_F$ are defined by adapting some actions from $\mathcal{D}_C$ to the new (expanded) signature, defining new actions, and defining *bridge axioms* that link the two descriptions. For our example scenario, axioms of $\mathcal{D}_L$ include:

$$move^*(A, L) \ \textbf{causes} \ at^*(A, L) \tag{2a}$$

$$at(A, Rg) \ \textbf{if} \ at^*(A, L), \ component(L, Rg) \tag{2b}$$

$$\textbf{impossible} \ grab(A, O) \ \textbf{if} \ at^*(A, L_1), \ at^*(O, L_2) \tag{2c}$$

where L refers to grid-based locations in $\mathcal{D}_F$ that are a *component of* a region location Rg in $\mathcal{D}_C$, and superscript "*" refers to relations introduced in $\mathcal{D}_F$. This coupling between descriptions provides separation of domain-dependent and domain-independent parts of the architecture, and defines conditions under which we can guarantee that any given transition in $\mathcal{D}_C$ can be implemented as a sequence of transitions in $\mathcal{D}_F$. In addition, it allows us to introduce (in $\mathcal{D}_F$) observations and non-deterministic knowledge-producing actions, and to associate probabilities with action outcomes. Prior work in our group has demonstrated how $\mathcal{D}_F$ can be automatically mapped to probabilistic sequential decision-making formalisms for planning [42].

Reasoning with the entire fine-resolution description can quickly become computationally intractable for complex domains. Our architecture addresses this problem by incorporating the related principle of *attention*. Specifically, the formal coupling between the two descriptions is leveraged to enable an ad hoc agent to automatically identify and *zoom to* the part of $\mathcal{D}_F$ that is relevant to any given goal and abstract action under consideration. For example, an agent moving from an *office* to the *kitchen* to fetch a cup of hot tea can focus on just these two rooms and their components while ignoring the other rooms in the domain. Specifically, sorts and ground object instances in the signature Σ_F of $\mathcal{D}_F$ are automatically restricted to those that are relevant to the abstract action (transition) to be implemented, and the axioms of $\mathcal{D}_F$ are restricted to this reduced signature. This process substantially reduces the complexity of reasoning at the finer-granularity, leading to computationally efficient operation.

A common criticism of knowledge-based reasoning methods is that they need comprehensive domain knowledge, but architectures that embed key principles such as refinement have demonstrated the ability to act based on existing knowledge while revising it incrementally [43]. Also, most of the steps for encoding the available knowledge can be automated, and the effort involved in encoding

prior knowledge is much less than the effort involved in designing and training purely data-driven systems.

3.2 Agent Behavior Models

Reasoning with just prior domain knowledge that can be incomplete or inconsistent will lead to poor performance, particularly under AHT settings where the other agent's actions also revise the domain state. Our architecture enables the ad hoc agent to also reason with models that predict the action choices of other agents and are learned (and revised) rapidly. This capability is achieved by embedding the principle of *Ecological Rationality* (ER) [19], which is based on Herb Simon's original definition of Bounded Rationality [39] and the algorithmic theory of decision heuristics [20]. ER explores decision making "in the wild", i.e., under open world uncertainty with the space of possibilities not fully known a priori, and characterizes behavior as a joint function of internal cognitive processes and the environment. It advocates the use of *adaptive satisficing* for making decisions in open worlds: in the absence of comprehensive knowledge, optimal decisions may be unknowable and not just hard to compute. Thus, decision heuristics (e.g., tallying, sequencing, fast and frugal methods) are used to ignore part of the information, and to make decisions more quickly, frugally, and accurately than sophisticated methods with many more free parameters [20]. This approach has been shown to provide better performance than more sophisticated methods in practical applications, but such decision heuristics and their successes do not receive the attention that they deserve [18].

Specifically, our architecture enables the ad hoc agent to select relevant attributes and learn models of the other agents behavior incrementally and from limited data. The agent learns an ensemble of *Fast and Frugal* (FF) trees to predict the behavior of each teammate (or type of teammate). 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 [23]. Each level of the tree contains an exit allowing the agent to make quick decisions based on available data. Unlike many sophisticated methods for AHT, these predictive models can be learned and revised incrementally and rapidly. Also, the ad hoc agents make decisions efficiently by evaluating the more informative attributes and stopping as soon as a rational option is found. Figure 3 shows an FF tree learned for a human, with Table 1 showing the key attributes used in these trees. Since each FF tree provides a binary choice, we build each predictive model as an ensemble of FF trees with a simple decision tree choosing an action based on the output of the FF trees. As we show later (Section 4.2), reasoning with prior knowledge and these models provides much better performance than methods that just reason with knowledge or use learned models.

Table 1. Attributes used to create behavior models of human.

Description of the attribute
Immediate two previous actions of the human
Immediate two previous objects human interacted
Position of the human (x,y,z)
Orientation of the human (x,y,z)
Distance from human to the kitchen table
Distance from human to the kitchen counter
Number of objects on the kitchen table

Consistent agreement (or disagreement) between actual observations (of outcomes) and the predictions provided by these behavior

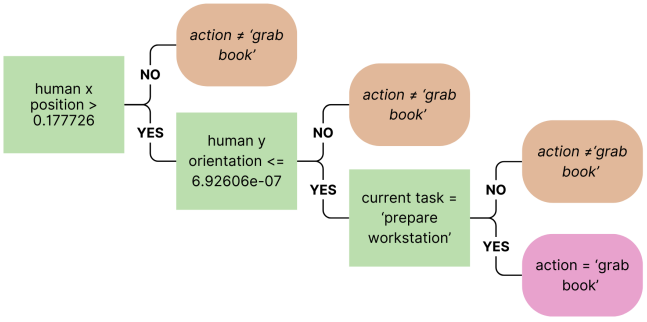

Figure 3. FF tree model of human behavior for the grab_book action in the example scenario from the *VirtualHome* domain.

models triggers the use of a particular model for subsequent steps, or leads to the revision of the existing model(s), allowing the ad hoc agent to quickly adapt to changes in the domain or another agent's behavior over time.

3.3 Task Anticipation

An agent's ability to anticipate future tasks and goals in the domain can enhance collaboration and overall performance. However, encoding such knowledge manually across different domains can be challenging and impractical. On the other hand, LLMs have been shown to be effective when used to translate natural language to domain-specific language, and to generate high-level guidance that is implemented by planning subroutines [22]. We thus enable an ad hoc agent to use an LLM to anticipate the high-level future task (e.g., *prepare dinner*) likely to be assigned once the current task is done. We experimentally demonstrate that jointly planning to complete the current and anticipated tasks improves team performance. The agent uses a combination of prompting strategies with the LLM:

- **Adopting persona:** A specific role or character is assigned to guide the LLM's responses to be more (contextually) consistent with the assigned role.
- **Few-shot prompting:** The prompt includes a few examples of the expected output in specific situations, guiding the use of pretrained knowledge.
- **Chain-of-thought (CoT):** The prompt includes a step-by-step reasoning process that can be followed to arrive at an answer, leading to more accurate responses.

Specifically, a 'system message' guides the LLM to adopt the persona of a household assistant and complete the partial task routine (provided as input) by selecting from the available list of tasks in the example scenario (in *VirtualHome*). With 'few-shot' prompting, the prompt includes two task routines randomly chosen from previous days. Next, CoT prompting is used to explain the reasoning behind each task in these example task routines. Such explanations can be provided by the system designer or generated by the LLM. The system message, few-shot examples with CoT explanation, and the current query (i.e., partially completed or empty task routine for the day) are provided as input to the LLM.

The LLM's output to a prompt is parsed by an *external validator* to check whether the tasks are feasible and in a reasonable order. Since LLMs are trained with large volumes of text from many domains, the anticipated tasks may not be feasible or match the preferences and needs of the human of our domain. Thus the validator compares the LLM's output with domain- and task-specific features extracted

from existing knowledge and recent observations, eliminating tasks that are invalid or irrelevant and reordering tasks according to human preferences. For example, the agent will prioritize preparation of the workstation over packing a lunch box when the human is working from home. Since the list of validated tasks can change over time, the ad hoc agent considers one anticipated task and the current task as the joint goal for which a plan is computed.

3.4 Transparency in decision making

An automated decision-making system's ability to answer questions about its decisions promotes acceptability [1, 16]; this transparency also plays an important role in human reasoning and learning. Unlike existing methods that seek to make a complex learned model interpretable, or to explain (or justify) all the choices made by a reasoning system, KAT seeks to respond to any given question about specific decisions made by the ad hoc agent or the human (teammate) by quickly identifying the relevant information and constructing a relational description as the answer.

The design choice of using knowledge-based reasoning and simple predictive models in KAT is the foundation for the approach described in this paper to provide the desired on-demand descriptions. We build on prior work in our group that demonstrated the ability to identify the axioms and literals relevant to the questions posed to a system making automated decisions [40]. The underlying principle is that of *explainable agency*, which corresponds to enabling specific functional abilities [24, 25]. Specifically, the agent must be able to: (i) provide on-demand justification of decisions made during (or after) plan generation and execution by considering alternative choices; (ii) report actions executed and present information at a suitable level of abstraction; (iii) describe how actual events deviated from a computed plan and how it adapted to these deviations; and (iv) communicate information about decisions and justification such that it makes contact with human concepts such as beliefs and goals.

As described in Section 3.1, KAT is designed to represent and reason with knowledge at different abstractions. We build on this representation and the associated update processes to embed the principle of explainable agency, i.e., to implement the abilities mentioned above, enabling the ad hoc agent to generate relational descriptions in response to four types of questions identified as being important in work on explainable planning [16]. Here we describe the procedure the ad hoc agent follows to generate responses for each of these types of questions. Examples of the use of this procedure to construct explanations on-demand are provided later in Section 4.3.

Causal questions: *Why did you execute a_I, i.e., action a at step I?* As described in Algorithm 1, when asked why an action was executed, the ad hoc agent first checks whether action a_I is the last action of plan P executed by the agent (Line 1, Algorithm 1). If not, the agent extracts the actions $\{A_{af}\} \in P$ that occurred immediately after $a_I \in P$ (Line 2). Next, the agent examines the program $\Pi(\mathcal{D}, \mathcal{H})$ to identify any axioms with the negation of $a_i \in \{A_{af}\}$ in *head*, i.e., axioms encoding conditions that prevent a_i from occurring (Line 3). The agent then checks if each such selected axiom's *body* is satisfied by the *answer set* at time step I (Line 5). If yes, it indicates that action a_i could not be executed unless the fluent literals f in the *body* of the axiom change value. The agent stores the fluent literals f in *body* whose value changed from I to $I+1$ (Line 6). The value of all such literals $\{f\}$ over all identified axioms has been changed by the execution of a_I, the action under consideration. The agent uses these literals to answer the question about a_I (Line 7).

Algorithm 1: Generate Causal Explanations

Input: P: Plan; a_I: action; G: goal; $\Pi(\mathcal{D}, \mathcal{H})$: program; $\mathcal{AS}$: answer set
Output: E: Explanation literals

1 **if** a_I *is not the last action in* P **then**
2 $\quad A_{af} \leftarrow \{a_i \in P \mid i > I\}$
3 $\quad R \leftarrow \{r \in \Pi(\mathcal{D}, \mathcal{H}) \mid head(r) = \neg a_i,\ a_i \in A_{af}\}$
4 $\quad$ **for** $r \in R$ **do**
5 $\quad\quad$ **if** $body(r)$ *satisfied in* $\mathcal{AS}$ *at* I **then**
6 $\quad\quad\quad F \leftarrow \{f \in body(r) \mid val(f, I) \neq val(f, I+1)\}$
7 $\quad\quad\quad E \leftarrow E \cup F$
8 $\quad\quad$ **end**
9 $\quad$ **end**
10 **else**
11 $\quad R \leftarrow \{r \in \Pi(\mathcal{D}, \mathcal{H}) \mid a_I, f' \in body(r)\}$
12 $\quad F' \leftarrow \{f' \mid f' \in body(r),\ r \in R\}$
13 $\quad$ **for** $l \in F'$ **do**
14 $\quad\quad C \leftarrow \{r \in \Pi(\mathcal{D}, \mathcal{H}) \mid head(r) = l\}$
15 $\quad\quad$ **for** $r \in C$ **do**
16 $\quad\quad\quad$ **if** $body(r)$ *satisfied in* $\mathcal{AS}$ *at* I **then**
17 $\quad\quad\quad\quad E \leftarrow E \cup \{l \cup body(r)\}$
18 $\quad\quad\quad$ **end**
19 $\quad\quad$ **end**
20 $\quad$ **end**
21 $\quad E \leftarrow E \cup G$
22 **end**
23 **return** E

If a_I is the last action in P, the agent examines $\Pi(\mathcal{D}, \mathcal{H})$ to identify axioms with a_I in their *body* alongside other additional literals $\{f'\}$, i.e., axioms encoding effects of a_I if additional conditions are true (Line 11). The agent then extracts these literals $\{f'\}$ from the identified axioms and examines $\Pi(\mathcal{D}, \mathcal{H})$ to identify axioms with literal $l \in \{f'\}$ in their *head*, i.e., state constraints (Lines 12 - 14). The agent then checks whether each of the selected axiom's *body* is satisfied by the *answer set* at time step I (Line 16). If yes, the agent use the literal $l \in \{f'\}$ and the *body* literals of the axioms to construct the answer as they have prompted the agent to execute a_I. Furthermore, as the last action in P, a_I directly contributed toward achieving the goal, and the agent also uses the goal G terms to construct the answer.

Contrastive questions: *Why did you not execute a_I, i.e., action a at step I?* As described in Algorithm 2, when asked why an action was not included in the plan, the ad hoc agent first examines the program $\Pi(\mathcal{D}, \mathcal{H})$ to identify axioms with the negation of a_I in its *head*, i.e., executability conditions (Line 1, Algorithm 2). For each selected axiom, the agent checks whether the axiom's *body* is satisfied by *answer set* at I (Lines 2-3). If yes, the agent collects the fluent literals $\{f\}$ in the *body*, as they prevented consideration of a_I, and uses them to construct the explanation (Line 4).

If no fluent literals have been identified, the agent examines $\Pi(\mathcal{D}, \mathcal{H})$ to identify axioms with a_I in its *body* alongside other literals $\{f'\}$, i.e., causal laws (Line 8). The agent extracts these additional precondition literals $\{f'\}$ and examines $\Pi(\mathcal{D}, \mathcal{H})$ to identify axioms with $l \in \{f'\}$ in its *head*, i.e., state constraints (Lines 9-11). For each selected axiom, the agent then checks if the axiom's *body* is satisfied by *answer set* at step I (Lines 12-13). If not, it uses the literals in the *body* of the axiom, and the literal l, to construct the answer as they prevented the agent from executing action a_I (Line 14).

Algorithm 2: Generate Contrastive Explanations

Input: P: Plan; a_I: action; G: goal; $\Pi(\mathcal{D}, \mathcal{H})$: program; $\mathcal{AS}$: answer set
Output: E: Explanation literals

1 $R \leftarrow \{r \in \Pi(\mathcal{D}, \mathcal{H}) \mid head(r) = \neg a_I\}$
2 **for** $r \in R$ **do**
3 **if** *body*(r) *satisfied in* $\mathcal{AS}$ *at* I **then**
4 $E \leftarrow E \cup \{f \mid f \in body(r)\}$
5 **end**
6 **end**
7 **if** $E = \emptyset$ **then**
8 $R' \leftarrow \{r \in \Pi(\mathcal{D}, \mathcal{H}) \mid a_I, f' \in body(r)\}$
9 $F' \leftarrow \{f' \mid f' \in body(r),\ r \in R'\}$
10 **for** $l \in F'$ **do**
11 $C \leftarrow \{r \in \Pi(\mathcal{D}, \mathcal{H}) \mid head(r) = l\}$
12 **for** $r \in C$ **do**
13 **if** *body*(r) *not satisfied in* $\mathcal{AS}$ *at* I **then**
14 $E \leftarrow E \cup \{l \cup body(r)\}$
15 **end**
16 **end**
17 **end**
18 **end**
19 **if** $E = \emptyset$ **then**
20 $C \leftarrow cost(P)$
21 $C' \leftarrow cost(P' = P \cup a_I)$
22 **if** $C' > C$ **then**
23 $E \leftarrow E \cup \{C, C'\}$
24 **end**
25 **end**
26 **return** E

If the literals to construct an answer have not been identified so far, the agent computes the cost C of P to achieve the goal G, and the cost C' of the P' that also include a_I (Lines 20-21). If the cost of P' is higher than that of P (i.e., $C' > C$), the agent concludes that this prevented it from including a_I in its plan (Line 22). The agent uses the cost terms to construct the answer (Line 23).

Justify beliefs: *Why did you believe l_I, i.e., l at step I?* As described in Algorithm 3, to justify a belief at a specific step, the ad hoc agent first replaces the grounded terms of the belief l_I with their corresponding variables and examines $\Pi(\mathcal{D}, \mathcal{H})$ to identify axioms with l_I in *head*, i.e., state constraints (Line 1, Algorithm 3). The agent then checks whether each selected axiom's *body* is satisfied by the answer set at step I (Lines 2-3). If yes, it collects the fluent literals $\{f\}$ in the *body* as they support belief l_I and uses them to construct the answer (Line 4). If multiple axioms are identified, the agent selects one of them to provide the explanation.

If belief tracing is enabled, the agent creates a tree with l_I as its head and each axiom (identified in Line 1) as a branch (Lines 8-10). With the axiom, the agent also stores the supporting fluents (F). It then repeats this procedure for each fluent literal in F as target belief until no more axioms are identified (Lines 11-19). The paths of the belief tree are then used to construct the explanation.

Counterfactual Questions: *What will be the action of the human in step I?* As described in Algorithm 4, the ad hoc agent first retrieves the most recent state observation of environment S_{I-n} in relation to step I, i.e., start with the current best estimate of world state (Line 1, Algorithm 4). It then uses the predictive behavior model of the human to retrieve their action a^{ot}_{I-n} for step $I - n$; it also uses our

Algorithm 3: Generate Belief Justifications

Input: l_I: belief; $\Pi(\mathcal{D}, \mathcal{H})$: program, $\mathcal{AS}$: answer set
Output: E: Explanation literals

1 $R \leftarrow \{r \in \Pi(\mathcal{D}, \mathcal{H}) \mid head(r) = l_I\}$
2 **for** $r \in R$ **do**
3 **if** *body*(r) *satisfied in* $\mathcal{AS}$ *at* I **then**
4 $E \leftarrow E \cup \{f \mid f \in body(r)\}$
5 $F \leftarrow \{f \mid f \in body(r)\}$
6 **end**
7 **end**
8 **if** *belief tracing enabled* **then**
9 $Tree \leftarrow$ create root node l_I
10 $Tree \leftarrow$ branches (R, F)
11 **for** $f \in F$ **do**
12 $R' \leftarrow \{r \in \Pi(\mathcal{D}, \mathcal{H}) \mid head(r) = f\}$
13 **for** $r \in R'$ **do**
14 **if** *body*(r) *satisfied in* $\mathcal{AS}$ **then**
15 $F \leftarrow F \cup \{f \mid f \in body(r)\}$
16 $Tree \leftarrow$ branches (r, F)
17 **end**
18 **end**
19 **end**
20 **end**
21 **return** $E, Tree$

Algorithm 4: Generate Counterfactual Explanations

Input: I: future step; S_{I-n}: recent observation; FF_h: FF tree of human
Output: E: Explanation literals

1 $S \leftarrow S_{I-n}$
2 **for** $t = I - n$ *to* $I - 1$ **do**
3 $a^{ot}_t \leftarrow predicted\_action(FF_h, S)$
4 $a_t \leftarrow compute\_agent\_action(S)$
5 $S \leftarrow simulate(S, a^{ot}_t, a_t)$
6 **end**
7 $S_I \leftarrow S$
8 $a^o_I \leftarrow predicted\_action(S_I)$
9 $a_I \leftarrow compute\_agent\_action(S_I)$
10 $E \leftarrow$ active branches in FF tree FF_h for a^o_I
11 **return** a^o_I, E

architecture to compute the action a_{I-n} that it should execute (Lines 2-4). The agent then performs a mental simulation of the future step $I - n + 1$ from S_{I-n} using existing knowledge and action choices identified (a^{ot}_{I-n}, a_{I-n}) (Line 5). The agent repeats above n times, i.e., roll out the future and explore effects of the action of ad hoc agent and human until environment reaches the queried step I. It then collects the state information S_I at I (Line 7). The agent then feeds the state information S_I to the predictive behavior model of the human to retrieve the action a^o_I, and use our architecture to retrieve the action for the ad hoc agent a_I in state S_I (Line 7-9). Specifically, the agent traverses the FF tree model of the human (FF_h) to identify branches that were active in selecting a^o_I; these branches and a^o_I are used to construct the answer (Line 10).

If the counterfactual question is about the outcome of a specific action executed by the human at a specific time step, this action is used directly instead of the predicted action Line 8 of Algorithm 4. In a similar manner, if the question is about the outcome of a hypothetical

action by the ad hoc agent at a specific future time step, this action is considered in Line 9. In addition, actions at a specific time step in the past can also be considered by examining the recorded history and then performing a mental simulation. Furthermore, if there are multiple agents (other than the ad hoc agent), the ad hoc agent would consider the corresponding predictive models as input and compute a set of actions $\{A^{ot}_{I-n}\}$ in Line 3 for all these agents.

Across the different types of questions, any information obtained can be used for further training, especially the human behavior prediction models. For all types of questions, the identified literals are processed using existing tools and templates to generate textual descriptions provided as responses (i.e., explanations) before, during, or after planning or execution.

4 Experimental Setup and Results

We experimentally evaluated the following hypotheses regarding our architecture's capabilities:

H1 Reasoning with prior knowledge and rapidly-learned behavior models improves performance;
H2 Using task anticipated by LLM as joint goal improves performance compared with not planning for joint goal;
H3 Incrementally-updated prompts and validators improve task anticipation and team performance;
H4 Using LLM to directly output a sequence of low-level actions to complete tasks results in poor performance; and
H5 Our architecture enables the ad hoc agent to generate relational descriptions as explanations of its decisions and beliefs and those of the human.

We evaluated these hypotheses in the *VirtualHome* and recorded the number of steps (plan length) and the total time taken to complete the task(s) as the *performance measures*. All prompts to the LLM were through OpenAI GPT4o mini API with default generation (temperature = 1.0). Details about experiments, baselines, and measures are provided below.

4.1 Experimental Setup

In the VirtualHome domain, we modeled the human as a simulated entity that chooses its actions based on an ASP program that considered prior knowledge and runtime observations but did not have models predicting teammates' actions. The human's ASP program also include actions such as eating and drinking that were not available to the ad hoc agent. Furthermore, the human's ASP program encoded priorities and preferences, e.g., when preparing breakfast, the human toasted the bread first before preparing the cereal. The sequence of tasks generated by the task generator were assigned to all agents one task at a time. The agents were not aware of the complete sequence and received one task at a time. The human was assigned the same goal as the ad hoc agent(s). All agents received the same observations of the domain at each step, which they used to plan their respective actions without direct communication with each other.

When an ad hoc agent equipped with our architecture received a task, it prompted the LLM (Section 3.3). The anticipated tasks were validated and mapped to ASP literals, with the next anticipated task and current task set as joint goals for this agent. During planning, the ad hoc agent also used the learned behavior prediction model to predict each teammate's actions for a few steps (Section 3.2). These predictive models were built using only 1000 examples of prior traces of actions and domain state and were incrementally revised based on teammates observed behavior. The ASP program of the ad hoc agent included additional axioms for reasoning about these predicted actions of each teammate, which were mapped to exogenous actions. The ad hoc agent's plan anticipated preconditions of some intermediate steps to be satisfied by a teammate's actions, even though the teammate did not always execute that action. The ad hoc agent hence had to respond to unexpected action outcomes.

For evaluating **H1** and **H2**, in **Exp1**, we randomly selected 100 task routines sampled from predefined sequences and measured the ability of a team (human and an ad hoc agent) to complete these tasks. Performance measures were the number of steps and time taken. We used three baselines: **Base1**: LLM for anticipating future tasks, but no behavior models to predict human's actions; **Base2**: no LLM to anticipate future tasks, but behavior models to predict the human's actions; and **Base3**: did not use LLM for task anticipation or behavior models to predict human's actions. In the absence of a framework that supports all (or most) of our architecture's capabilities, these baselines allowed us to conduct ablation studies and evaluate the contribution of each key component. Since the time taken and the number of actions in a plan can vary substantially based on the task, the average of these values across trials may not be meaningful. We instead *ran paired trials and computed performance measure values for baselines as a fraction of these values for our architecture in each trial*. We then reported the average of these ratios.

To evaluate **H3**, in **Exp2**, we computed the precision and recall of tasks anticipated by the LLM, before and after applying the validator, over the 100 task routines. We also computed precision and recall of a simple statistical model that replaced the LLM and anticipated the next task based on past experience. Further, in **Exp3**, we randomly selected 20 task routines and recorded the LLM's performance with and without prompting methods, using four baselines: **Base4**: no prompting strategy or validator; **Base5**: few-shot prompting but no validator; **Base6**: CoT prompts but no validator; and **Base7**: validator but no prompt-engineering.

For evaluating **H4**, we conducted **Exp4**, with the LLM being used to directly compute sequences of actions for specific tasks (**Base8**). The prompt included details of actions available in our example scenario in *VirtualHome*, and their intended purpose. We also supplied the LLM some **Action Feasibility Rules**: (a) *movement limitation* (critical): must only move to adjacent locations defined by the next_to relationships; (b) *object location*: must be in the same location as an object to act on it (e.g., grab, put); (c) *carry limit*: cannot hold more than two objects, and actions like open, close, switch-on, or switch-off require putting at least one object down if holding two objects; (c) *appliance safety*: you should not open appliance doors when they are switched on; (d) *avoid conflict*: a human holding an object will perform actions involving object, so focus on other parts of the goal. The LLM had access to the current state, including location of agents, objects, and appliances, and each appliance's state. The problem specification described the task to be performed; the immediate prior actions of the human and ad hoc agent; and other relevant information (e.g., human working from home). Additionally, the prompt included a detailed example of selecting an action, and asked the LLM to generate an action sequence for the ad hoc agent to complete the task. Feedback was provided for errors in the LLM's output (e.g., action to grab an object without moving to suitable location) up to three times per trial; LLM was told why this choice was incorrect, and allowed to output another action. We measured the number of steps and time taken by the team to complete the selected 100 task routines.

To evaluate **H5** in **Exp5**, we randomly selected 10 configurations

(from 100 in **Exp1**) and saved the corresponding answer sets (with our architecture, baseline) to provide ground truth. Then, we posed 32 different questions (split between four types of questions) about some chosen steps in each trial corresponding to one of these 10 configurations, with answers computed as described in Section 3.4. We recorded precision and recall of the literals in the answers.

4.2 Experiment Results

Table 2 summarizes the results of **Exp1**. When the ad hoc agent reasoned with anticipated tasks and predicted human actions, it provided the best observed performance with lowest number of action steps and least amount of time taken to complete the routines. The prediction accuracy of the behavior models learned by the ad hoc agent for the human using the ensemble of FF trees was 85%. We observe that the model does make errors, but it supports rapid learning and revision. Also, reasoning with prior knowledge and the output of these predictive models significantly improves the performance. When the ad hoc agent used task anticipation but did not use the behavior prediction models (**Base1**), the number of steps and time taken to complete tasks increased because the inability to predict the actions of teammates may lead the ad hoc agent to execute the same actions as a teammate, hindering collaboration. These results emphasize the importance of using the behavior prediction models, supporting **H1**.

When the ad hoc agent used the behavior models to predict the human's actions, but did not anticipate and plan for future tasks (**Base2**), the performance worsened, with a further increase in the number of steps and the time taken to complete the tasks. Recall that this setting corresponded to the agent only planning for one goal at a time without anticipating future goals. Planning actions jointly for the current task and the anticipated (next) task saved time and effort. For example, when the agent visited the bedroom to retrieve a board game to entertain guests, it also picked up bottles of wine from the cellar that is on the way. Making two separate trips for these tasks extended the length and duration of the plans. These results indicate the impact that planning for joint goals has on performance, supporting **H2**. When the ad hoc agent did not use task anticipation or the behavior prediction models (**Base3**), it resulted in the worst observed performance with the highest value for number of steps and time taken to complete the tasks. These results support **H1** and **H2**. All results were statistically significant ($p < 0.0001$).

Table 2. Average number of steps and time taken to complete task routines, with values for baselines computed as a fraction of these values for our architecture in each trial; for comparison, the average absolute values are 26.8 steps and 361 seconds for our architecture. Task anticipation (LLM) and teammates' behavior/action prediction (FF trees) substantially improved performance.

Architecture	*Steps*	*Time(s)*
Ours (anticipate tasks, predict actions)	1.0	1.0
Base1 (anticipate tasks)	1.1	1.1
Base2 (predict actions)	1.3	1.2
Base3	1.4	1.4

Table 3 shows the results from **Exp2**, i.e., the precision and recall of the tasks anticipated by a simple statistical model, and by the LLM before and after applying the validator. The errors in the LLM's outputs were substantially reduced by the validator. The recall values do not change substantially as the validator did not introduce new tasks; it only reordered the tasks that are out of order and removed irrelevant tasks. These results support **H3**.

Table 3. Precision and recall values of the anticipated tasks with and without the LLM and the validator.

Architecture	*Precision*	*Recall*
Simple statistical model	0.75	0.76
LLM without validator	0.78	0.76
LLM with validator	0.99	0.78

Table 4 shows the results from **Exp3**, which explored the use of different prompting strategies and the validator with the LLM to anticipate tasks. We observed a significant performance improvement with the external validator and a combination of prompting methods. The results were statistically significant for **Base4-6** ($p < 0.001$). **Base7** showed mixed results; reduction in steps was statistically significant ($p < 0.05$), while reduction in time was not ($p = 0.09$). This outcome matched expectations as Base7 used the validator; it emphasizes the importance of the validator and the need for further exploration of more complex scenarios to determine the importance of the prompting strategies. These results support hypothesis **H3**.

Table 4. Average number of steps and time taken by the team (human, ad hoc agent) to complete tasks with prompting strategies and/or external validator; performance measure values for baselines computed as a fraction of values for our architecture in each trial, with average absolute values of 27.5 steps and 372.7 seconds for our architecture.

Architecture	*Steps*	*Time(s)*
Proposed (all prompting, with validator)	1.00	1.00
Base4 (no prompting, no validator)	1.21	1.15
Base5 (few-shot prompting, no validator)	1.17	1.18
Base6 (chain-of-thoughts, no validator)	1.16	1.16
Base7 (no prompting, with validator)	1.05	1.04

Next, Table 5 shows the results of **Exp4**, where we used the LLM to directly compute the low-level actions for the ad hoc agent in the example scenario in *VirtualHome*. The number of steps and time taken to successfully complete the task routines are significantly higher than our architecture as well as all other baselines (**Base1-3**) in Table 2. These results support **H4**.

Table 5. Average number of steps and time taken by the team (human, ad hoc agent) to complete task routines when the LLM directly outputs sequence of low-level actions to be executed. Performance measure values computed as a fraction of values obtained with our architecture in Table 2.

Architecture	*Steps*	*Time(s)*
Base8 (LLM predict low-level actions)	1.5	1.5

Table 6 shows results of **Exp5**, with precision and recall calculated relative to all the literals expected to be included in the explanation. We observed high precision and recall indicate the agent's ability to automatically extract the correct literals to provide relational descriptions as explanations. These results support hypothesis **H5**.

4.3 Execution Trace

We also provide some execution traces as a qualitative evaluation of **H3**. Figure 4 shows an execution example where the ad hoc agent used the LLM to anticipate future tasks, with and without the prompting strategies and validator. The example was set on a weekday where the human was working from home and no guests were expected. The

Table 6. Precision and recall of retrieving relevant literals for providing explanations.

Question type	*Precision*	*Recall*
Action justification	1.00	1.00
Contrastive	0.89	0.99
Belief justification	0.88	0.94
Counterfactual	1.00	0.78

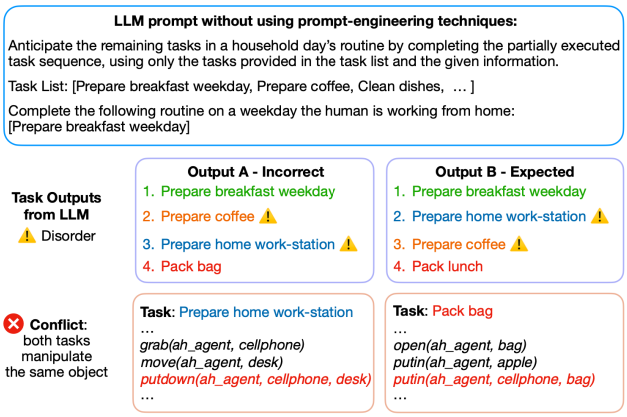

Figure 4. Execution example: using LLM without prompting strategies or external validator causes conflicts during execution, having a negative impact on performance.

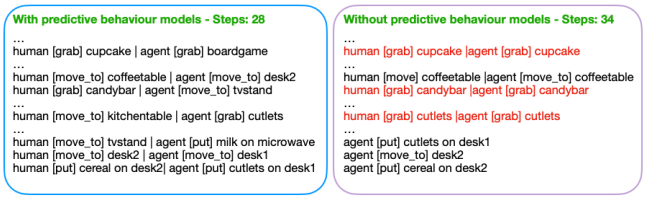

Figure 5. Execution trace for task routine: *[Prepare breakfast, Prepare activities, Serve snacks, Clean kitchen]*. When an ad hoc agent is not allowed to predict and reason about the human's actions, it may choose to execute same action(s) as the human, leading to longer plans.

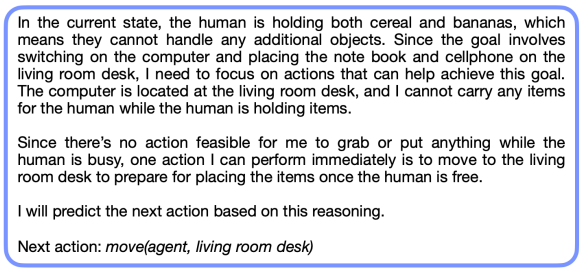

Figure 6. Example output when ad hoc agent uses the LLM for computing sequences of actions for specific tasks (**Base8**).

correct task routine in this context was: *Prepare breakfast, Prepare home work-station, Prepare coffee, Prepare lunch*.

When the ad hoc agent queried the LLM without the prompt engineering techniques or validator (Section 3.3), the anticipated task list was different from the expected output. The prompt to the LLM without using the prompt engineering strategies is shown in Figure 4. The LLM output in this scenario was [*Prepare breakfast, Prepare coffee, Prepare home work-station, Pack bag*]. This output failed to align with the human preferences and priorities as:

- Higher priority was assigned to making coffee than setting up the workstation. This would delay the human for work and lead to coffee not being hot when needed.
- Packing the bag was an unnecessary task as the human was not leaving the house; it would have been filtered by the validator.

These preferences and priorities were captured by the agent through its experience with the human across multiple iterations, and used by the validator to filter out irrelevant tasks or reorder them. When the ad hoc agent used the prompt engineering and validation strategies, the prompt to the LLM was automatically generated while incorporating context, as described in Section 3.3. The LLM's output was [*Prepare breakfast, Prepare home work-station, Prepare coffee, Prepare lunch*]. This matched the expected routine. i.e., making breakfast and setting up the workstation were considered high priority tasks, and irrelevant tasks such as *pack bag* were filtered out by the validator. These results demonstrate the importance of using a combination of prompting techniques and the external validator, supporting **H3**.

We observed similar situations when three agents, one human and two ad hoc agents, were considered to be collaborating together on a weekend when guests were expected. The correct task routine in this context was: *Prepare breakfast, Prepare table for guests, Prepare lunch, Clean dishes*. When the first ad hoc agent queried the LLM with the prompting strategies but without the validator, the anticipated task list by the LLM was *Prepare breakfast, Prepare table for guests, Prepare lunch, Serve snacks*. For the second ad hoc agent the LLM output was *Prepare breakfast, Prepare table for guests, Prepare lunch, Serve snacks, Clean dishes*. When the validator was used, the outputs to both the agents were refined by incorporating context. Since the human usually did not require snacks after lunch, the task *Serve snacks* was removed. For the first agent the refined task list was *Prepare breakfast, Prepare table for guests, Prepare lunch*. For the second agent the task list was *Prepare breakfast, Prepare table for guests, Prepare lunch, Clean dishes*; since *Clean dishes* was a defined task in this domain, it was retained by the validator. This example further demonstrates the importance of using the validator in our architecture, and and supports **H3**.

Next, Figure 5 compares two plans executed by a team comprising a human and an ad hoc agent for completing a different set of tasks: *[Prepare breakfast, Prepare activities, Serve snacks, Clean kitchen]*, with and without the behavior prediction models (Section 3.2). In the first plan, when the ad hoc agent used the behavior prediction model to predict the future actions of the human, the team successfully completed all tasks for the given day in just 28 steps. On the other hand, when the ad hoc agent did not use behavior prediction models, it often selected the same actions as the human for any particular task, leading to unnecessary delays in completing the tasks. For example, in the second plan the agent frequently selected the same action as the human—simultaneously picking up the cupcake, candy bar, and cutlets, introducing redundant behavior and prolonging task execution. As a result, the overall plan was extended to 34 steps. These results

Table 7. Part of the ad hoc agent's plan for task in Execution Example 1.

occurs(find(ahagent,breadslice),0)	occurs(putin(ahagent,cutlets,fryingpan),8)
occurs(grab(ahagent,breadslice),1)	occurs(put(ahagent,cutlets,kitchentable),11)
occurs(put(ahagent,breadslice,kitchentable),2)	occurs(grab(ahagent,poundcake),13)
occurs(switchon(ahagent,stove),4)	occurs(open(ahagent,microwave),15)
occurs(grab(ahagent,cutlets),6)	occurs(put(ahagent,poundcake,microwave),16)

Table 8. Part of plan generated by ad hoc agent to avoid conflict with human.

occurs(find(ahagent,cutlets),1)	occurs(put(ahagent,poundcake,microwave),9)
occurs(grab(ahagent,cutlets),2)	occurs(close(ahagent,microwave),10)
occurs(find(ahagent,fryingpan),3)	occurs(switchon(ahagent,microwave),11)
occurs(putin(ahagent,cutlets,fryingpan),4)	occurs(open(ahagent,microwave),13)
occurs(grab(ahagent,poundcake),6)	occurs(put(ahagent,poundcake,kitchentable),16)
occurs(open(ahagent,microwave),8)	occurs(put(ahagent,cutlets,kitchentable),24)

demonstrate that using the behavior prediction models enables the ad hoc agent to coordinate efficiently by avoiding action conflicts with the human. These further supports **H1**.

When the ad hoc agent used the LLM for directly computing sequences of actions for specific tasks **Base8**, the prompt was automatically constructed following the procedure described in Section 4.1. Figure 6 shows part of the LLM output during the task routine: [*Prepare breakfast, Prepare home work-station, Prepare coffee, Prepare lunch*]. The selected action *move(agent, living room desk)* violated the 'Movement Limitation (Critical)' rule in 'Action Feasibility Rules', which constrained the agent to only move to locations adjacent to its current location. This example demonstrates that the LLM may not respect constraints even when they are provided as input, and highlights that an LLM is not designed for computing plans for non-trivial tasks; using an LLM to directly output a sequence of low-level actions to complete tasks can lead to infeasible actions, resulting in poor performance. These results support hypothesis **H4**.

Next, we provide some execution traces illustrating our architecture's explanation generation capabilities. Consider the scenario in which the task is to *prepare breakfast* and the world state is: bread slice is inside the toaster; cutlets are on the kitchen counter; poundcake is on the kitchen table; water glass is in the bedroom; microwave is closed and switched off; frying pan is on the stove that is switched off; and the human and ad hoc agent are in the kitchen.

Consider the scenario in which the: bread slice is inside toaster; cutlets are on kitchen counter; poundcake is on kitchen table; water glass is in bedroom; microwave is closed and switched off; frying pan is on stove that is switched off; and the human and ad hoc agent are in the kitchen. To help the human prepare breakfast, the ad hoc agent generated a plan with 23 steps, some of which are shown in Table 7; as usual, the agent expects the human collaborator to complete some intermediate steps.

Execution Example 1. [Action Justification, Contrastive, Counterfactual] Consider an exchange with the ad hoc agent after executing first plan step.

- **Questioner:** "Why did you find bread slice in step 1?"
- **Ad hoc Agent:** "Because I had not found the bread slice yet and I wanted to grab it in step 2".
  The agent's response draws attention to the target action's outcome being a requirement for executing a subsequent action. Specifically, since the *grab* action occurred immediately after *find*, the relevant axioms identified included:

$$\neg occurs(grab(R,O),I) \leftarrow \neg holds(found(R,O),I), \quad agent(R), graspable(O). \tag{3}$$

  Next, the ad hoc agent explored the relevant answer sets and identified that the literal $found(ad\_hoc\_agent, bread\_slice)$ was present in step 2 but not in step 1. This literal was then selected to justify the action execution. The agent can also be asked why it did not consider picking up a different object.
- **Questioner:** "Why did you not find the water glass in step 1?"
- **Ad hoc Agent:** "Because I predicted that the human will find the water glass in step 1."
  The agent first tried to identify the axioms with *find* in the head.

$$\neg occurs(find(R,O),I) \leftarrow \neg holds(agent\_found(T,O),I), \quad other\_agents(T), graspable(O). \tag{4}$$

  Next, the agent ground the body of the axiom and verified whether it was included in the answer set. Since the $agent\_found(human, waterglass)$ was valid in step 1, the agent identified it as a literal that prevented it from considering the *find* action in step 1. This is due to the ad hoc agent's model of the human predicting that the human will execute action $exo\_find(human, waterglass)$ in step 0. The agent may also be asked about the human's action choices further.
- **Questioner:** "Why do you think the human will pick up the water glass in step 2?"
- **Ad hoc Agent:** "Because my prediction is that the human wants to bring the glass to the table."
- **Questioner:** "What will happen if the human decided to pick up the cutlets in step 2?"
- **Ad hoc Agent:** "If the human picks up the cutlets in step 2, they will be in the human's hands in step 3."
  To answer questions about hypothetical situations, the ad hoc agent has to simulate the evolution of state, and the execution of actions by the human and the agent for a few steps in order to identify and report the motivation for specific action choices. Specifically, the agent first retrieved the current state estimate and

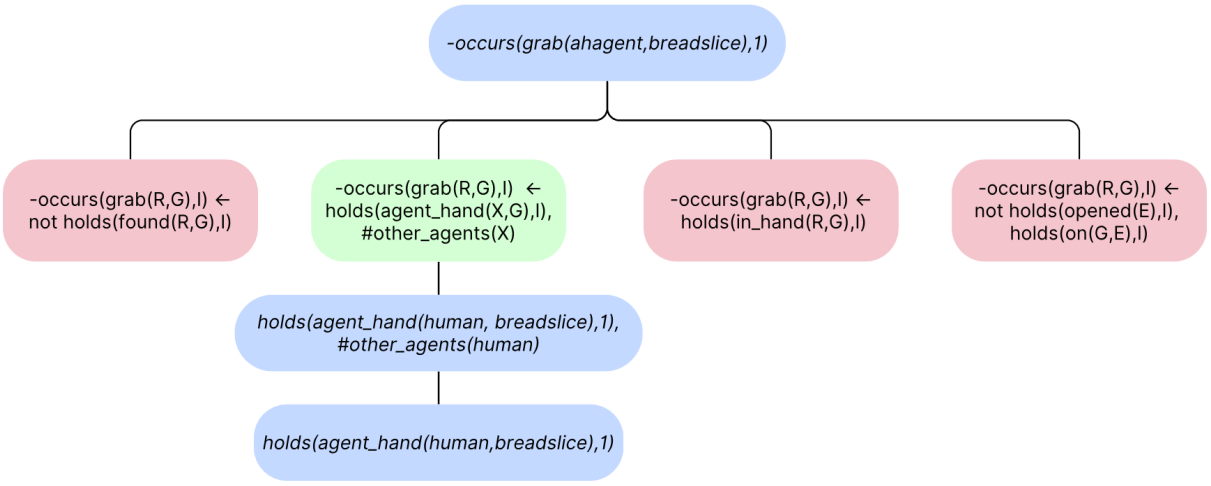

Figure 7. Belief tracing to explore why ad hoc agent did not grab bread slide in step 1.

initialized a new *VirtualHome* environment. Then it simulated the environment for the desired future time steps while using the human behavior model to predict the future actions of the human. Next, it went through the FF tree model predicting the human behavior and identified rules that caused it to believe the human will *grab* the water glass in step 2. this information was then used to generate the responses (above) to the questions posed.

Execution Example 2. [Action Justification, Contrastive, Belief Tracing] The accuracy of the model predicting human action choices (85%) indicates that the predictions can be incorrect, particularly in situations in which the agent's understanding of the human is limited to a few observations. Consider a variant of the scenario above, in which the human decided to find and grab a slice of bread as the first action. Since the ad hoc agent decided to do the same action, it created a conflict. Some key steps of the plan computed by the ad hoc agent to overcome this conflict are shown below. The subsequent conversation with the ad hoc agent was as follows:

- **Questioner:** "Why did you find cutlets in step 1 ?"
- **Ad hoc Agent:** "Because I have not found the cutlets and I wanted to grab them in time step 2".
- **Questioner:** "Why did you not grab bread slice in step 1?"
- **Ad hoc Agent:** "Because the human was holding the bread slice."

This exchange demonstrates the ability of the ad hoc agent to change its plan in order to prevent a conflict with the human, and to justify this decision. Similar to the previous example, the ad hoc agent used the identified axioms and literals to generate the answer to the questions posed. Figure 7 shows the belief tree created for tracing the ad hoc agent's beliefs and justifying why it did not grab the bread slice in step 1. The green and blue boxes represent the satisfied axioms and their grounded literals, while the red boxes represent the axioms that were not satisfied in this example.

Overall, these results support hypothesis **H5**. Additional results in the form of videos are available in our open-source code repository [12].

5 Conclusions

This paper described an AHT architecture that embeds the principles of refinement, ecological rationality, interactive learning, and explainable agency, enabling each ad hoc agent to: leverage the generic knowledge in a pretrained LLM for high-level task anticipation; rapidly learn models predicting teammates' actions; perform non-monotonic logical reasoning with relevant prior knowledge and behavior models to plan and execute actions that jointly achieve current and anticipated tasks; and generate on-demand explanations of the agent's decisions and those of others as answers to queries. Experiments in a realistic simulation environment demonstrated performance improvements compared with baselines, highlighting the importance of each component and the ability to scale to additional agents. Future work will explore larger, heterogeneous teams in AHT settings.

REFERENCES

[1] Sule Anjomshoae, Amro Najjar, Davide Calvaresi, and Kary Framling, 'Explainable agents and robots: Results from a systematic literature review', in *International Conference on Autonomous Agents and Multiagent Systems*, Montreal, Canada, (2019).

[2] Evgenii Balai, Michael Gelfond, and Yuanlin Zhang, 'Towards Answer Set Programming with Sorts', in *Conference on Logic Programming and Nonmonotonic Reasoning*, (2013).

[3] Marcello Balduccini and Michael Gelfond, 'Logic Programs with Consistency-Restoring Rules', in *AAAI Spring Symposium on Logical Formalization of Commonsense Reasoning*, (2003).

[4] Samuel Barrett, Avi Rosenfeld, Sarit Kraus, and Peter Stone, 'Making friends on the fly: Cooperating with new teammates', *Artificial Intelligence*, **242**, 132–171, (2017).

[5] Samuel Barrett, Peter Stone, Sarit Kraus, and Avi Rosenfeld, 'Teamwork with limited knowledge of teammates', in *AAAI Conference on Artificial Intelligence*, volume 27, (2013).

[6] Michael Bowling and Peter McCracken, 'Coordination and adaptation in impromptu teams', in *National Conference on Artificial Intelligence*, p. 53–58, (2005).

[7] Shuo Chen, Ewa Andrejczuk, Zhiguang Cao, and Jie Zhang, 'AATEAM: Achieving the ad hoc teamwork by employing the attention mechanism', in *AAAI*, (2020).

[8] Hasra Dodampegama and Mohan Sridharan, 'Knowledge-based Reasoning and Learning under Partial Observability in Ad Hoc Teamwork', *Theory and Practice of Logic Programming*, **23**(4), 696–714, (2023).

[9] Hasra Dodampegama and Mohan Sridharan, 'Reasoning and Explanation Generation in Ad hoc Collaboration between Humans and Embodied AI', in *International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*, Dallas, USA, (October 11-14, 2024).

[10] Hasra Dodampegama and Mohan Sridharan, 'Reasoning with Commonsense Knowledge and Decision Heuristics for Scalable Ad hoc Human-Agent Collaboration', in *Distributed AI Conference (DAI)*, London, UK, (November 21-24, 2025).

[11] Hasra Dodampegama and Mohan Sridharan, 'Collaborate and Explain On-the-Fly: Knowledge-based Reasoning and Learning in Ad Hoc Teamwork', *Frontiers in Artificial Intelligence: Logic and Reasoning in AI*, to appear, (2026).

[12] Hasra Dodampegama and Mohan Sridharan. `https://github.com/hharithaki/KAT-VH`, 2026.

[13] Upol Ehsan, Samir Passi, Q. Vera Liao, Larry Chan, I-Hsiang Lee, Michael Muller, and Mark O Riedl, 'The who in xai: How ai background shapes perceptions of ai explanations', in *CHI Conference on Human Factors in Computing Systems*, (2024).

[14] Qi Fang, Junjie Zeng, Haotian Xu, Yue Hu, and Quanjun Yin, 'Learning ad hoc cooperation policies from limited priors via meta-reinforcement learning', *Applied Sciences*, **14**(8), (2024).

[15] Bettina Finzel, David E. Tafler, Stephan Scheele, and Ute Schmid, 'Explanation as a Process: User-Centric Construction of Multi-level and Multi-modal Explanations', in *KI 2021: Advances in Artificial Intelligence*, ed., Stefan Edelkamp and Ralf Möller and Elmar Rueckert, volume 12873, 80–94, Springer International Publishing, (2021).

[16] Maria Fox, Derek Long, and Daniele Magazzeni, 'Explainable Planning', in *IJCAI Workshop on Explainable AI*, (2017).

[17] Michael Gelfond and Daniela Inclezan, 'Some Properties of System Descriptions of AL_d', *Applied Non-Classical Logics, Special Issue on Equilibrium Logic and ASP*, **23**(1-2), (2013).

[18] Gerd Gigerenzer, *Towards a Rational Theory of Heuristics*, 34–59, Palgrave Macmillan UK, London, 2016.

[19] Gerd Gigerenzer, 'What is Bounded Rationality?', in *Routledge Handbook of Bounded Rationality*, Routledge, (2020).

[20] Gerd Gigerenzer and Wolfgang Gaissmaier, 'Heuristic Decision Making', *Annual Review of Psychology*, **62**, (2011).

[21] Joseph Y. Halpern and Judea Pearl, 'Causes and explanations: A structural-model approach. part i: Causes', *The British Journal for the Philosophy of Science*, **56**(4), 843–887, (2005).

[22] Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Mudit Verma, Kaya Stechly, Siddhant Bhambri, Lucas Saldyt, and Anil Murthy, 'Position: LLMs Can't Plan, But Can Help Planning in LLM-Modulo Frameworks', in *International Conference on Machine Learning*, (2024).

[23] Konstantinos Katsikopoulos, Ozgur Simsek, Marcus Buckmann, and Gerd Gigerenzer, *Classification in the Wild: The Science and Art of Transparent Decision Making*, MIT Press, 2021.

[24] Pat Langley, 'Explainable, Normative, and Justified Agency', in *AAAI Conference on Artificial Intelligence*, pp. 9775–9779, (January 2019).

[25] Pat Langley, Ben Meadows, Mohan Sridharan, and Dongkyu Choi, 'Explainable Agency for Intelligent Autonomous Systems', in *Innovative Applications of Artificial Intelligence*, San Francisco, USA, (February 4-9, 2017).

[26] Xinzhu Liu, Peiyan Li, Wenju Yang, Di Guo, and Huaping Liu, 'Leveraging Large Language Model for Heterogeneous Ad Hoc Teamwork Collaboration', in *Robotics: Science and Systems*, Delft, The Netherlands, (2024).

[27] Tim Miller, 'Explanations in Artificial Intelligence: Insights from the Social Sciences', *Artificial Intelligence*, **267**, 1–38, (February 2019).

[28] Reuth Mirsky, Ignacio Carlucho, Arrasy Rahman, Elliot Fosong, William Macke, Mohan Sridharan, Peter Stone, and Stefano Albrecht, 'A Survey of Ad Hoc Teamwork: Definitions, Methods, and Open Problems', in *European Conference on Multiagent Systems*, (2022).

[29] Tiago Mota, Mohan Sridharan, and Ales Leonardis, 'Integrated Commonsense Reasoning and Deep Learning for Transparent Decision Making in Robotics', *Springer Nature CS*, **2**(242), (2021).

[30] Judea Pearl, *Causality*, Cambridge University Press, 2 edn., 2009.

[31] Judea Pearl and Dana Mackenzie, *The book of why : the new science of cause and effect*, Allen Lane, 2018.

[32] Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba, 'Virtualhome: Simulating household activities via programs', in *International Conference on Computer Vision and Pattern Recognition*, pp. 8494–8502, (2018).

[33] Muhammad A Rahman, Niklas Hopner, Filippos Christianos, and Stefano V Albrecht, 'Towards open ad hoc teamwork using graph-based policy learning', in *International Conference on Machine Learning*, pp. 8776–8786, (18–24 Jul 2021).

[34] Raymond Reiter, 'A Theory of Diagnosis from First Principles', *Artificial Intelligence*, **32**, 57–95, (1987).

[35] João G. Ribeiro, Gonçalo Rodrigues, Alberto Sardinha, and Francisco S. Melo, 'Teamster: Model-based reinforcement learning for ad hoc teamwork', *Artificial Intelligence*, **324**, 104013, (2023).

[36] Cynthia Rudin, 'Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead', *Nature Machine Intelligence*, **1**, 206–215, (2019).

[37] Cynthia Rudin, Chaofan Chen, Zhi Chen, Haiyang Huang, Lesia Semenova, and Chudi Zhong, 'Interpretable machine learning: Fundamental principles and 10 grand challenges', *Statistics Surveys*, **16**, 1–85, (2022).

[38] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra, 'Habitat: A platform for embodied ai research', in *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9338–9346, (2019).

[39] Herbert A. Simon, 'Rational Choice and the Structure of the Environment', *Psychological Review*, **63**, 129–138, (1956).

[40] Mohan Sridharan, 'Integrated Knowledge-based Reasoning and Data-driven Learning for Explainable Agency in Robotics', in *Explainable Agency in Artificial Intelligence: Research and Practice*, CRC Press, (2024).

[41] Mohan Sridharan, 'Back to the Future of Integrated Robot Systems', in *AAAI Conference on Artificial Intelligence*, Philadelphia, US, (February 25-March 4, 2025).

[42] Mohan Sridharan, Michael Gelfond, Shiqi Zhang, and Jeremy Wyatt, 'REBA: A Refinement-Based Architecture for Knowledge Representation and Reasoning in Robotics', *Journal of Artificial Intelligence Research*, **65**, 87–180, (May 2019).

[43] Mohan Sridharan and Ben Meadows, 'Knowledge Representation and Interactive Learning of Domain Knowledge for Human-Robot Collaboration', *Advances in Cognitive Systems*, **7**, 77–96, (December 2018).

[44] Mohan Sridharan and Ben Meadows, 'Towards a theory of explanations for human–robot collaboration', *KI - Künstliche Intelligenz*, **33**, 1–12, (09 2019).

[45] Peter Stone, Gal Kaminka, Sarit Kraus, and Jeffrey Rosenschein, 'Ad Hoc Autonomous Agent Teams: Collaboration without Pre-Coordination', in *AAAI Conference on Artificial Intelligence*, pp. 1504–1509, (2010).

[46] Penghui Xu, Yu Zhang, Le Hao, and Qilin Yan, 'DETEAMSK: A Model-Based Reinforcement Learning Approach to Intelligent Top-Level Planning and Decisions for Multi-Drone Ad Hoc Teamwork by Decoupling the Identification of Teammate and Task', *Aerospace*, **12**(7), 1–32, (2025).

[47] Luisa Zintgraf, Sam Devlin, Kamil Ciosek, Shimon Whiteson, and Katja Hofmann, 'Deep interactive bayesian reinforcement learning via meta-learning', in *International Conference on Autonomous Agents and Multiagent Systems*, (May 2021).