Model Elicitation through Direct Questioning

Paper ID – 7970

Abstract

The future will be replete with scenarios where humans are robots will be working together in complex environments. Teammates interact, and the robot's interaction has to be about getting useful information about the human's (teammate's) model. There are many challenges before a robot can interact, such as having a simple methodology for interaction with teammates with varying thought process while working in the same environment, ensuring that the teammate does not feel overwhelmed with these interactions, etc. In this paper, we investigate how a robot can interact by asking eliciting questions from their teammate to localize the human model from a set of models. We show how to generate questions oriented towards different responses from the teammate and ensure meaningful conversation. We evaluate the method in various planning domains for collaboration. The evaluation shows that these questions can be generated offline, and provided the knowledge of the domain, it supports asking about specific parts of the model.

Humans work in complex environments by choosing different actions to reach their goals. For human-aware AI to be useful, the automated agents need to support humans in these complex environments. Essentially, they need to understand a teammate's model working in these complex environments. Ideally, these agents can acquire generalized models for working in the environment by using humans behavioral data such as plan traces (Gil 1994; Stern and Juba 2017; Zhuo et al. 2020; Garrido and Jiménez 2020). These generalized models have to be refined and evaluated with the human teammate. Since the human teammate is working with the agents, it is feasible to interact with them. Thus, in this paper, we look at interaction through directed questions by an automated agent with their teammates, where the simple answers can refine the model.

There are many challenges for asking directed questions, such as questions should be easy enough to understand, or it shouldn't overwhelm the teammate. Furthermore, the lack of knowledge about the way humans think or handle the information makes it difficult to ask questions. In fact, *education* community has spent decades to understand how students conceptualize or interpret knowledge (Ortony and Rumelhart 1977), such as, knowledge can be in the form of concept maps (Novak and Cañas 2006) or in the form of causal rules (Shultz 1982), or the form of *concept images* (Vinner and Hershkowitz 1980). There has also been some work on how we use these knowledge structures by understanding how students construct formal mathematical proofs (Moore 1994). There have been studies to test various hypotheses, but nothing concretely can be said under every condition.

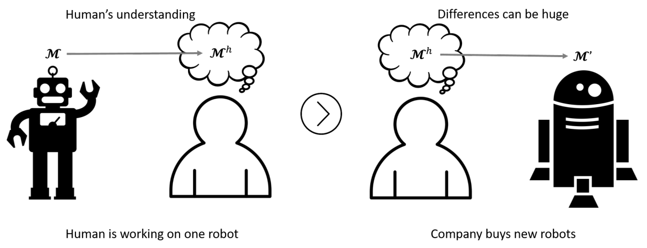

Figure 1: A human teammate is working on one of the robots and has an understanding $\mathcal{M}_H$ of robot's model ($\mathcal{M}$). When the company buys a new robot (model $\mathcal{M}'$) but the human teammate's model $\mathcal{M}_H$ does not change and can cause damage to the robot or effect the collaboration of the team.

On the other hand, an automated agent has structured knowledge with the closed-world assumption to calculate and evaluate their decisions. They can precisely say whether a particular action is possible under the given condition and it's effects. Thus, an agent needs to find a way to question the behavior of the teammate and understand various responses by the user. In this paper, we look at how to generate questions to elicit information about the unstructured model of the teammate while ensuring directed questions and answers. Thus, the major contribution of this paper is to –

(C1) given the unstructured nature of the human thought process, represent uncertainty in the agent's understanding of their teammates model to ask directed questions,

(C2) provides a framework to define questions, and the expected answers to elicit the required information,

(C3) to manage the computational complexity of the questions, design a method to separate different unknown properties,

(C4) despite the uncertainty, discuss the assumptions to ask these questions and obtain relevant information.

Background

In the intelligent tutoring system community, there has been a lot of work to represent the knowledge of a student, usually based on the number of problems they can solve. Sometimes, it can be represented using directed graphs, to depict a dependency among different knowledge concepts called *knowledge spaces* (Falmagne et al. 2006; Doignon and Falmagne 2015). We also use such dependency graphs to represent a robot's knowledge about their teammate's model using STRIPS (Fikes and Nilsson 1971) as the causal dependency can be represented using cause and effect for an action. Robust planning model (Nguyen, Sreedharan, and Kambhampati 2017) is used to represent the uncertainty, as some conditions that may or may not be a part of the human's model of the environment. These probable conditions for preconditions and effects are called annotations. We will now formally describe the annotated model.

The correct human's model is represented as $\mathcal{M}_H$ is one of the concrete models in the set of models $\mathbb{M} = \{\mathcal{M}_0, \mathcal{M}_1, \mathcal{M}_2, ..., \mathcal{M}_n\}$. $\mathbb{M}$ is represented using a superset of predicates and operators of each models as $\mathbb{M} = \langle \mathcal{F}, \mathcal{A} \rangle$. $\mathcal{F}$ are the set of propositional state variables or *facts* and any state $s \in \mathcal{F}$. Any action $a \in \mathcal{A}$ is defined as $a = \langle pre(a), eff^{\pm}(a), \diamond pre(a), \diamond eff^{\pm}(a) \rangle$, where $pre(a), eff^{\pm}(a)$ are certain preconditions and effects and $\diamond pre(a), \diamond eff^{\pm}(a) \subseteq \mathcal{F}$ are the possible preconditions and possible add/delete effects, where $\diamond$ being used to differentiate between the certain and possible predicates. It implies that a precondition and effect may or may not be present in the original model and are used to model the uncertainty in the robot's understanding of the human's model. If their are n uncertain preconditions and effects then $|\mathbb{M}| = 2^n$.

A planning problem in the domain is given by $\mathcal{P} = \langle \mathbb{M}, \mathcal{I}, \mathcal{G} \rangle$, where $\mathcal{I}$ is the initial state and $\mathcal{G}$ is the goal. $\delta_{\mathcal{M}}$ is the transition function $\delta_{\mathcal{M}} : S \times \mathcal{A} \rightarrow S$. It can not be executed in a state $s \not\models pre(a)$; else $\delta_{\mathcal{M}}(s, a) \models s \cup eff^{+}(a) / eff^{-}(a)$. There are two different transition functions for the condition when a specific action in a plan can't be executed (due to $\diamond pre(a)$ being true and not planned for), then the plan either goes to a fail state (pessimistic) or the next action can still be executed (optimistic) in the plan (Nguyen, Sreedharan, and Kambhampati 2017). An agent assumes an optimistic approach as it is concerned about the interaction and the information about the feasibility of the action. A plan π for the model $\mathcal{M}$ is a sequence of actions $\pi = \langle a_0, a_1, ..., a_n \rangle$. A plan is a valid plan if $\delta_{\mathcal{M}}(\mathcal{I}, \pi) \models \mathcal{G}$. A set of plans Π are $\Pi = \{\pi | \forall \pi \delta_{\mathcal{M}}(\mathcal{I}, \pi \models \mathcal{G}\}$

Motivating example Figure 1 shows the situation, where a human teammate working with one robot has to migrate to a new one. We now describe the *move* action for the new robot compared to the earlier model. For a robot (such as fetch[1]) it is essential to ensure that the robot's torso is NOT in the stretched state and the arm is tucked close to the body, else it can lead to a fatal accident due to its tendency to topple while turning. The previous model of the robot had no such requirements. The domain for the new robot with uncertainty about the preconditions of a move action, i.e. *hand_tucked* (hand not stretched) and *is_crouch* (torso not stretched). The robot's uncertain model of the human is –

```
(:action tuck
:parameter                ()
:precondition             ()
:possible-precondition    ()
:effect                   (and    (is_crouch)
                                  (hand_tucked))
:possible-effect          ()
)
(:action crouch
:parameter                ()
:precondition             ()
:possible-precondition    ()
:effect                   (and    (is_crouch)
:possible-effect          ()
)
(:action move
:parameter                (?from ?to - location)
:precondition             (robot-at ?from)
:possible-precondition    (and    (is_crouch)
                                  (hand_tucked))
:effect                   (and    (robot-at ?to)
                                  (not (robot-at
                                  ?from)))
:possible-effect          ())
```

There are two unknown preconditions hence there are four possible models. If the objects that are defined in the model are – *roomA* and *roomB*. Then a question Q to ask human's understanding would be defined as $Q = \langle \mathcal{I}, \mathcal{G} \rangle$, i.e. a planning task, where $\mathcal{I} = \{$*robot-at(roomA)*$\}$, and $\mathcal{G} = \{$*(robot-at(roomB), (not (robot-at(roomA))*$\}$. There are three possible plans that give us the complete information about the preconditions. $\pi_1 = \langle$*move(roomA, roomB)*$\rangle$, $\pi_2 = \langle$*crouch, move(roomA, roomB)*$\rangle$, and $\pi_3 = \langle$*tuck, move(roomA, roomB)*$\rangle$. π_1 shows that there are no preconditions in the human's model, π_2 says that human has the precondition of is_crouch, and π_3 shows that human has both is_crouch and hand_tucked preconditions.

An important point to note is that there might be a correct model for the environment that might be known to the automated agent. However, the agent is only trying to learn the exact human model $\mathcal{M}_H$. The correct model might be different from the human model and may or may not be part of the set of models $\mathbb{M}$.

Problem Formulation

After describing the robot's model to represent the uncertainty in the human's model, we are now ready to formally define the questions to elicit the behavior of the teammate.

Definition 1. *Question is a tuple, $\langle \mathcal{I}, \mathcal{G} \rangle$, i.e. initial and the goal states. The solution to a question are plans π such that $\delta(\mathcal{I}, \pi) \models \mathcal{G}$. A sequence of questions as $Q = \langle \langle \mathcal{I}_0, \mathcal{G}_0 \rangle, \langle \mathcal{I}_1, \mathcal{G}_1 \rangle, ..., \langle \mathcal{I}_n, \mathcal{G}_n \rangle \rangle$, the tuple of sequential planning tasks where predicates and actions are based on the agent's uncertain model.*

Thus, interaction with the user is through structured questions, and the response is in the form of a plan. For now, we assume that the user is responding with a plan, but it can

[1]https://fetchrobotics.com/robotics-platforms/fetch-mobile-manipulator/

be computationally taxing for them to construct it. Thus, we look for different responses in the next section, when we define a distinguishing query. Now we formally define the problem of asking questions to the user.

Question Framing Problem Given the tuple $\langle \mathbb{M}, \mathcal{I}_e \rangle$, find the sequence of questions Q to learn the correct model with $|Q| \leq n$, where n is the number of possible predicates.

$\mathbb{M}$ represents the annotated model of the robot and $\mathcal{I}_e$ represents the fully-specified input state of the environment in which we are interacting with the human-in-the-loop. Since, the automated agent has to construct questions where models will differ, defining the initial state of the environment is useful. To support interaction, initial state $\mathcal{I}_q$ for any query q can be specified as a set difference from the fully-specified initial state of the environment $\mathcal{I}_e$.

Constrained and Relaxed models. As we described earlier, that for any n possible pre-conditions and effects, hence, there is an exponential set of possible models (2^n). It is impossible to analyze all of them to generate questions. Thus, for our analysis we look at two extreme models in the robot's model – (1) most constrained model $\mathcal{M}_{con}$ and (2) most relaxed model $\mathcal{M}_{rel}$. $\mathcal{M}_{con}$ is the model where only $pre(a) = pre(a) \cup \diamond pre$, and $eff^-(a) = eff^-(a) \cup \diamond eff^-$. $\mathcal{M}_{rel}$ is the model where $eff^+ = eff^+ \cup \diamond eff^+$. Since, the cost of actions is same across all the concrete models in the robot's model (as one of them represents the model of the same hilp), $C^*_{\mathcal{M}_{con}}$ is the cost of the optimal plan in $\mathcal{M}_{con}$, and similarly $C^*_{\mathcal{M}_{rel}}$ for $\mathcal{M}_{rel}$. The corollary follows from the construction of relaxed and constrained models (Sreedharan and Kambhampati 2018).

Corollary 1. $C^*_{\mathcal{M}_{rel}} \leq C^*_{\mathcal{M}_{con}}$.

Equality in the corollary exists when $\mathcal{M}_{rel}$ is the same as $\mathcal{M}_{con}$, i.e., there is no uncertainty in the model. Now we try to extend these properties to any model in the set of models.

Human Model. We assume that the human model is one of the models in the set of robot's model $\mathcal{M}_H \in \mathbb{M}$. Thus when presented with a question q, let $\Pi^q_{\mathcal{M}_H}$ represent the possible set of solutions in the human's model and $C^*_{\mathcal{M}_H}$, is the cost of the optimal plan in the human's model. Given this current setup, we can say a few things about the solution provided by the user to any question.

Corollary 2. *Assuming user to be an optimal planner then for a specific question,* $C^*_{\mathcal{M}_{rel}} \leq C^*_{\mathcal{M}_H} \leq C^*_{\mathcal{M}_{con}}$.

The corollary explains that the optimal solution constructed by hilp follows these bounds. They can be verified by assuming each inequality and realizing that the human model for that particular inequality is the most constrained or relaxed model for a subset of possible predicates. For comparison between $\mathcal{M}_H$ and $\mathcal{M}_{rel}$, the $\mathcal{M}_H$ is the most constrained version of $\mathcal{M}_{rel}$ model for a subset of constraints. Similarly, while comparing $\mathcal{M}_H$ and $\mathcal{M}_{con}$, we can see that $\mathcal{M}_H$ is the most relaxed version of the $\mathcal{M}_{con}$ model on the subset of original constraints. Showing the bounds for the possible human solution to any question doesn't take the actual human model into account. Now we analyze the construction of initial and goal state to elicit different properties. Let π_H be the solution provided by hilp with cost C_H.

Distinguishing Query

Due to the combinatorial explosion, the agent can not search through all possible initial and goal states, and the agent needs to generate a query from the bounds on the human model. Thus, to analyze the specific initial and the goal state, we look at the properties of the plans which should be discussed and then find the initial and goal state that will support their execution. We start our analysis with the simplest case, how to distinguish models with a single probable predicate (i.e., a single unknown predicate in an in the model), and then generate a directed question when there are many possible predicates in the model.

The solution provided by the user depends on the question posed by the agent, and the query depends on the annotated constraint. If p is the predicate possibly (for now assuming it's a pre-condition) present in action a (we should be writing $\mathcal{M}^{a\diamond p}$, but for simplicity, we write $\mathcal{M}^{\diamond p}$). Thus there can be four different models which the agent needs to consider –

* $\mathcal{M}^{-p}_{con}$ – Constrained model with p is not part of it.
* $\mathcal{M}^{+p}_{rel}$ – Relaxed model with p is part of the model.
* $\mathcal{M}^{-p}_{H}$ – Human model with p is not part of the model.
* $\mathcal{M}^{+p}_{H}$ – Human model with p is part of the model.

Please note, a predicate p being part of the model means that it is present in the specific action a, for which it was a possible predicate. It does *not* mean that a predicate was abstracted from the model. Also, note that only one of the models $\mathcal{M}^{-p}_{H}$ and $\mathcal{M}^{+p}_{H}$ is the true model. Given, the solution π_H, and assuming human is an optimal planner, there can be a few possibilities –

1. $C^*_H < C^*_{\mathcal{M}^{+p}_{rel}}$, would mean that $\pi_H \notin \Pi_{\mathcal{M}^{+p}_{rel}}$ as the solution plan cost provided by hilp is less than the optimal plan cost in $\mathcal{M}^{+p}_{rel}$. Thus, $\mathcal{M}^{-p}_{H}$ is the real human model, i.e. the constraint is not part of the human model.
2. $C^*_H \geq C^*_{\mathcal{M}^{-p}_{con}}$, would mean that $\pi_H \notin \Pi_{\mathcal{M}^{-p}_{con}}$ as it is not a valid plan in $\mathcal{M}^{-p}_{con}$. Thus, $\mathcal{M}^{+p}_{H}$ is the real human model, i.e. the constraint is part of the human model.

In principle, the plan for the question posed by the agent to determine a constraint p that will be harder to execute in $\mathcal{M}^{+p}_{H}$ and easier in $\mathcal{M}^{-p}_{H}$ and help us distinguish the models. It can be achieved by tweaking the initial state of the query. For example, if p is not part of the initial state, then human needs to achieve this pre-condition to execute the action, only if the pre-condition is part of the human model. Similarly, we can extend this idea to any possible effect e, where a plan involving the action would be costlier to generate using $\mathcal{M}^{-e}_{H}$, as compared to $\mathcal{M}^{+e}_{H}$ by changing the goal state (in different scenarios, i.e. with or without the effect e). We call such queries *distinguishing query*.

Definition 2. *A question Q_p is a* distinguishing query *if the plan for it can distinguish between models $\mathcal{M}^{+p}_{H}$ and $\mathcal{M}^{-p}_{H}$.*

Until now, we were looking at different possible models, and responses were analyzed based on the cost of the optimal plan in each model. In the next subsection, we look at the properties of the plans responded by the human π_H.

Properties

We have outlined the central idea where the action with unknown predicate has to be part of the plan for a distinguishing query. Now, we will formally describe these properties and how it can be used to generate query. For this we introduce a topic from the planning called *landmarks* to formally define the property we explained.

Landmark. A *landmark* L is a logical formula, where, $\forall\pi : \delta(\mathcal{I}, \pi) \models \mathcal{G}$ and for some prefix of plan $\pi_{pre} = \langle a_0, a_1, ..., a_i \rangle, i < n$, $\delta(\mathcal{I}, \pi_{pre}) \models L$. An *action* landmark is for any action a,$\forall\pi : \delta(\mathcal{I}, \pi) \models \mathcal{G}, a \in \pi$ (Keyder, Richter, and Helmert 2010). For example, the $\mathcal{G}$ is a trivial landmark and if there exists only one action a that reaches a particular fact $f \in \mathcal{G}$, then the action is an action landmark. Finding a landmark is a PSPACE-complete problem (Hoffmann, Porteous, and Sebastia 2004).

Proposition 1. *Necessary condition for a question Q_p to distinguish between models $\mathcal{M}^{+p}$ and $\mathcal{M}^{-p}$ is $a_p \in L(\mathcal{M}^{-p})$, where a_p is the action with proposition p is a landmark in model $\mathcal{M}^{-p}$ for the query.*

Proof. Proof is divided into two parts. First, we show that being a landmark is necessary and then we show that landmark has to be of model $\mathcal{M}^{-p}$. For the first part, let's assume if there is a plan π for a distinguishing problem Q_p and such that $a_p \notin \pi$. Then $\delta_{\mathcal{M}^{+p}}(\mathcal{I}_{Q_p}, \pi) \models \mathcal{G}_{Q_p}$ and $\delta_{\mathcal{M}^{-p}}(\mathcal{I}_{Q_p}, \pi) \models \mathcal{G}_{Q_p}$, i.e. the plan can be executed in both the models. Which refutes the assumption that Q_p is a distinguishing problem.

For the second part, observe that $\Pi_{Q_p}^{\mathcal{M}^{+p}} \subseteq \Pi_{Q_p}^{\mathcal{M}^{-p}}$, i.e. every plan possible in more constrained model ($\mathcal{M}^{+p}$) is also a plan in the less constrained model ($\mathcal{M}^{-p}$). Again, if we assume that the action a_p is a landmark in $\mathcal{M}^{-p}$, then there are some solutions $\pi \in \Pi_{Q_p}^{\mathcal{M}^{-p}} \setminus \Pi_{Q_p}^{\mathcal{M}^{+p}}$, where a_p is not a landmark. Following the previous proof, we can conclude that all plans have the property $a_p \in L(\mathcal{M}^{-p})$. □

Due to page constraint, the proof is presented in the supplementary material. An important point to note here is that we do not distinguish between fact p being a pre-condition or effect for action because this condition is necessary for either of them. The agent has to ensure that the action with a possible predicate is a landmark action. One of the methods is to use the *add effects* as the goal of the question. In theory, we can use this method to ensure that the action a is a landmark action in the human's model, but in practice, we need to assume that human is an optimal planner, and thus the action is an optimal landmark. Removing this assumption is out of scope for this work and will be part of our future work. Proposition 1 can be extended, that by ensuring that action a is a landmark in $\mathcal{M}^{-p}$, it is harder to achieve all pre-condition for the action a in the constrained model $\mathcal{M}^{+p}$. The proposition directly leads to two corollaries that can establish the distinguishing property.

Corollary 3. *The distinguishing problem Q_p should have atleast one solution in $\mathcal{M}^{-p}$.*

Corollary 4. *The distinguishing problem Q_p should not be solvable in $\mathcal{M}^{+p}$.*

Both the above corollaries follow from the way constrained and relaxed models are constructed, where distinguishing problem Q_p should have atleast one solution in $\mathcal{M}^{-p}$, and no solutions in $\mathcal{M}^{+p}$. Thus, if the problem is solved (by executing the plan in the environment or asking the human for the plan in their model) then the fact is fictitious else the fact is real. The answer to the distinguishing query is a simple yes/no, facilitating the interaction with the human-in-the-loop.

Difference between Pre-conditions and Effects. The analysis stands true for any possible predicate, be it precondition or an effect, and results in different ways of constructing models $\mathcal{M}^{-p}$ and $\mathcal{M}^{+p}$. The basis for constructing these models is that $\mathcal{M}^{+p}$ is the constrained model, and $\mathcal{M}^{-p}$ is the relaxed one. Thus for any possible pre-condition and a delete effect – $\mathcal{M}^{+p}$ is where the predicate is true, and $\mathcal{M}^{-p}$ is where the predicate is not part of the action in the model. Conversely, for add effects $\mathcal{M}^{+p}$ – the predicate is not part of the action in the model, and $\mathcal{M}^{-p}$ – predicate is part of the action in the model.

Proposition Isolation Principle (PIP)

In this section, we discuss how to ask questions to ensure that uncertainty about a predicate does not affect the interaction about another predicate. This method can be seen as a brute force method to ask questions by isolating the predicate, i.e., there will be n questions for n possible unknowns. The PIP method involves constructing the two models –

- $\mathcal{M}^{+p}$ is the most constrained model $\mathcal{M}_{con}$.
- $\mathcal{M}^{-p}$ is $\mathcal{M}_{con}^{-p}$, i.e. most constrained model where the predicate is considered not present in case of preconditions and delete effects, and considered part of the model in case of add-effects.

It naturally follows that the models differ by a single predicate. Using $\mathcal{M}_{con}$ to pursue a landmark essentially helps isolate the specific predicate p by supporting both the presence and absence of other predicates. For example, an action in the plan which may have possible predicate as a precondition, then that particular pre-condition can be provided through the initial state of the query.

Proposition 2. *Any question Q_p, which distinguishes the model $\mathcal{M}^{+p}$ and $\mathcal{M}^{-p}$, is isolated by $\mathcal{M}_{con}$ and $\mathcal{M}_{con}^{-p}$.*

To prove the statement, we need to understand that the models differ due to the predicate p, thus all the plans will only differ due to the absence (presence, in case of add effect) of the predicate in $\mathcal{M}_{con}$. Since, the plans are being constructed for $\mathcal{M}_{con}^{-p}$, thus the agent is able to ensure that rest of possible or true constraints (predicates in actions) are either satisfied through the initial state or are easily achievable through the actions in the plan. In other words, the plans are possible in all the other models, and the failure is due to the constraint (predicate) p. This completes the sketch.

The proposition 2 explains that by using the PIP method, the agent can ask questions about every predicate. It works on the idea of providing all the pre-conditions that affect the action a_p. If the user might fail to execute a plan involving a_p, then it is only due to the constraint p. Thus, we have discussed the assumptions using which it is possible to create a plan, but now we discuss the scenarios (or edge-cases) where the agent may not be able to localize the teammate's model in a set of models.

Sufficiency Condition. The agent uses a $\mathcal{M}_{con}$ model to find the solution with the landmark. However, it doesn't prove that the same or similar plan involving the landmark action would be optimal in the human model. Due to fewer constraints compared to $\mathcal{M}_{con}$, there could be many other actions that could be used by the agent and thus might have a different plan which might not involve the action. Thus, the agent needs to find this scenario and prevent it by increasing the cost of executing other actions.

To understand these scenarios, the agent can convert action to a landmark by adding it's $eff^{+}(a_p)$ as the goal. But it is always possible to have two or more actions that provide a subset of the goal predicates (whose union is the goal set). In the teammate's model, the plan with these actions could be smaller. Thus, the agent needs to ensure that these actions are harder to execute in any less constrained model. Another scenario could be when an action that provides the precondition p for action a_p also satisfies another pre-condition in it. Thus, when the teammate's plan includes the action which provides the possible pre-condition, the agent can't be sure the action is for pre-condition p or not. It could be impossible to check all the action combinations for every possible condition, but once the plan for the query is known in con, the number of actions in the plan are finite, and thus evaluating these conditions is feasible.

Solving v\s Validating. We have outlined a method that will be discussed in the solution section, but there are still conditions that might ensure the teammate's plan might not include the a_p. An agent can continue to constrain the set of possible plans by removing actions from the model or some other condition. However, these extra additions can overwhelm the teammate as the communication keeps getting complex and can burden the teammate in finding a plan. The agent can ask the query as the initial, the goal state, and the plan. The teammate can respond to the validity of the plan in the model. In this scenario, the agent needs to ensure that the plan can be executed only in the $\mathcal{M}^{-p}_{con}$. Where solving a problem would have given more information, but currently, our framework handles the human as an optimal agent. Thus, the agent can still ask the plan constructed with the initial and goal state in $\mathcal{M}^{-p}_{con}$ for validation.

Decreasing Questions

In the previous section, we have shown how the agent can interact with the teammate, and ensure that every interaction can be useful. However, if the agent wants to decrease the number of questions, it has to question more than one predicate, and thus, there could be multiple reasons for the infeasibility of the query. There are conditions when both $\mathcal{M}^{+p}$ and $\mathcal{M}^{-p}$ have a feasible different solution. Thus, using PIP and these conditions, every subset of the models is bound to have an optimal plan. Thus, the agent uses the difference in the plan to infer the teammate's model. The analysis is based on the optimality assumption and positive action cost for the model. We present the step-by-step construction of such questions that we call *templates*, as they can be merged to construct a query for more than one predicate, where every subset of the model has a valid and optimal solution.

Pre-conditions. For a proposition p which is a possible pre-condition of the action a_p. For the action to executed $pre(a_p)$ can be provided by – (1) initial state, or (2) executing another action a' where $p \in eff(a')$. When the precondition comes from the initial state, then the distinguishing question can constructed using isolated proposition. Now, we discuss how such a template exists and how a query can ensure different plans in both the models ($\mathcal{M}^{+p}$ and $\mathcal{M}^{-p}$).

Proposition 3. *Distinguishing question for $\diamond p$ of an action a_p has a distinct valid plan in models $\mathcal{M}^{+p}$ and $\mathcal{M}^{-p}$ when for another action a', $p \in eff^{+}(a')$.*

Proof. Assume for a given distinguishing problem Q_p actions $a \in L(\mathcal{M}^{-p})$ and a' can be executed i.e. $\mathcal{I}_{Q_p} = \{pre(a) \cup pre(a')\} \setminus \{p\}$ and $\mathcal{G}_{Q_p} = \{eff^{+}(a)\} \setminus \{p\}$. For, model $\mathcal{M}^{+p}$ the plan is $\pi = \langle a', a \rangle$. This plan is also a valid plan in $\mathcal{M}^{-p}$. But due to optimality and non-zero action costs, the plan π is not an optimal plan in $\mathcal{M}^{-p}$ as pre-condition provided by a' is not required in the model to execute action a. Thus, the optimal plan will be $\pi' = \langle a \rangle$ which is distinct. □

Add Effects. For possible add effects $p = \diamond eff^{c}(a)$ an action a' such that $p \in pre(a')$. If there is another action a'' where $p \in eff^{+}(a'')$. Now we will discuss the template in some more detail.

Proposition 4. *Given three actions, a, a', a'' where $p = \diamond eff^{c}(a)$ $\diamond eff^{c}(a) \in pre(a')$ and $\diamond eff^{c}(a) \in eff^{c}(a'')$ will have distinct plans in models $\mathcal{M}^{+p}$ and $\mathcal{M}^{-p}$.*

Proof. We will again use proof by construction. Consider a distinguishing proble Q_p, where $\mathcal{I}_{Q_p} = \{pre(a) \cup pre(a') \cup pre(a'')\} \setminus \{p\}$ and $\mathcal{G}_{Q_p} = \{eff^{+}(a) \cup eff^{+}(a')\} \setminus \{p\}$. This will ensure actions a and a' are landmarks and a'' can be executed. Now in the case of $\mathcal{M}^{-p}$ (remember constructive effect are part of less constrained model, follows from proposition 2), $\pi = \langle a, a' \rangle$, and for model $\mathcal{M}^{+p}$, $\pi = \langle a, a'', a' \rangle$, which completes the construction. □

The limitation of asking about binary interaction, is that the agent is not using the model structure. However using the templates checks whether a particular causal dependency exists in the teammate's model. If it is not part of the true model then an optimal plan will not use the specific action in the plan that provides a relationship. In this case as well we need to be sure there aren't other causal dependency among that actions and in that case the agent can't be certain. The PIP principle holds true for asking questions like this and the agent has to isolate the predicates that it wants to use for asking questions. Extra care has to be taken to ensure that the predicates that are used as template – (1) do not have more

Algorithm 1: Query Generation Algorithm (QGA)

```
Input  : $\mathcal{I}_e, \mathcal{M}^{-p}_{con}, a', a_p$
Output: $\langle \mathcal{I}_q, \mathcal{G}_q, \pi_q \rangle$
1  begin
2      $\mathcal{G} \leftarrow pre(a') \cup pre(a_p)$;
3      $\pi \leftarrow Solve(\langle \mathcal{M}^{-p}_{con}, \mathcal{I}_e, \mathcal{G} \rangle)$;
4      $\pi_q \leftarrow \langle \pi, a_p \rangle$;
5      $\mathcal{I}_q, \mathcal{I}_{temp} \leftarrow$ Project($\mathcal{I}_e, \pi$);
6      $\mathcal{G}_q, \mathcal{G}_{temp} \leftarrow eff^+(a_p)$ ;
7      for $a_x \in \{a | a \in \pi_q \& eff^+(a) \in \mathcal{G}_q\}$ do
8          $f \leftarrow \{f | f \in eff^+(a_x) \& f \notin \mathcal{G}_q\}$;
9          $\mathcal{G}_{temp} = \mathcal{G}_{temp} \cup \tilde{f}$;
10     end
11     for $a_x \in \{a | a \in \pi_q, p \in eff(a), pre(a_p) \setminus p \in$
         $eff(a)\}$ do
12         $f \leftarrow \{f | f \in pre(a_p), f \in eff(a_x), f \neq p\}$;
13         $I_{temp} = I_{temp} \cup f$;
14     end
15     $\pi \leftarrow Solve(\langle \mathcal{M}^{-p}_{con}, \mathcal{I}_{temp}, \mathcal{G}_{temp} \rangle)$;
16     if $a_p \in \pi$ then
17         return $\langle \mathcal{I}_{temp}, \mathcal{G}_{temp}, \pi \rangle$;
18     else
19         return $\langle \mathcal{I}_q, \mathcal{G}_q, \pi_q \rangle$;
20     end
21 end
```

than one causal dependency, and (2) do not have mutually exclusive relationships in the plans.

These templates can be merged as every sub-model (with or without the predicates) has a valid plan. Thus, if there are negative interaction between pair-wise actions in the plan, merging the plans won't be a problem. Since, it's not possible to check whether all the cases are possible we construct a planning problem from the initial state of the environment. In the next section, we will discuss the solution in detail.

Given the assumption of no destructive interactions the proof is easy to follow as plan for each sub-space is union of distinct plans in the template. Thus, we can merge the questions and even ask the teammate whether which specific plan is valid in the model.

Proposed Solution

As we described in the earlier sections, a query is an initial and goal state in which the action a_p is an optimal landmark in the model $\mathcal{M}^{-p}_{con}$. The easiest way to achieve this is to define the goal state as the add-effects of the action a_p. Finding an initial state is difficult because we need to ensure that given any possible pre-conditions in effect, the interaction should provide information. Thus, in this section, we describe two algorithms – (1) to iterate over each unknown predicate and decide whether a particular template can be merged with another, and (2) for generating the query.

The parent routine is to iterate over each unknown in an order decided by $\mathcal{M}_{join}$, which assumes that all the unknown predicates are true. This model can't be used for analysis, as it is neither most constrained nor most relaxed, and it can't be shown that the plans generated in this model are part of other models too, such as in the case of $\mathcal{M}_{con}$. Then we construct a relaxed planning graph with pair-wise mutexes called graph-plan planning graph (Kambhampati, Parker, and Lambrecht 1997). It also merges the templates generated by checking whether plans for the template can be executed sequentially or not, by checking if the actions in two plans are mutexes.

Algorithm 1, generated queries for specific p, a_p pairs. First, it solve a planning problem, where the plan is to reach the preconditions of the action a_p in the model $\mathcal{M}^{-p}_{con}$. If the query is for the template, then preconditions of other actions is used as well (follows from proposition 3 and 4). The solution of the planning problem and then executing action a_p as the goal is $eff^+(a_p)$. The project function finds the subset of the initial state was required for constructing the plan, to ensure that other plans for the query are not feasible in the teammate's model.

Then we look for the sufficiency conditions for the plan using $\mathcal{M}_{join}$. From lines 7-10, we handle every effect in a_p, provided by other actions. The negation of the predicate from these actions to the initial and goal state. From lines 11-13, the algorithm checks if an action in the plan threatens p. The threat is handled by removing the action from the plan and adding its constraints to the initial state. Finally, it reevaluates whether the update initial and goal state construct the plan in the constrained model or not. If the plan still contains the action a_p then the query is to validate the plan π_q, instead of asking them to generate the plan. This solution follows the complete analysis to ensure a sufficient and minimal response from the teammate.

These queries have no sequence as through PIP and sufficiency conditions, we have ensured that other constraints do not affect the query for any predicate. But while asking these queries, the sequence was generated based on the graph-plan planning graph for $\mathcal{M}_{con}$, where the order was based on action closer to the initial state.

Empirical Evaluation

We have theoretically discussed the process of generating questions. The question generation method uses APDDL parser (Nguyen, Kambhampati, and Do 2013) based on PDDLPy[2], and an optimal planner Fast downward (Helmert 2006) to solve the planning problems. The results reported are from experiments run on a 12 core Intel(R) Xeon(R) CPU with an E5-2643 v3 @3.40GHz processor and a 64G RAM. The experiments were performed on – rover, blocksworld, satellite, and zenotravel[3]. The IPC domains were assumed to be the correct human model, and some predicates were randomly assumed to be possible predicates in the action and an equal number of predicates were randomly added to the actions based on the parameters for the action. Special care was needed to ensure that the extra predicate did not make the action impossible by adding a mutex to already available pre-conditions. Please note, that the

[2]https://pypi.org/project/pddlpy/

[3]https://github.com/potassco/pddl-instances

	$\lvert \diamond p \rvert$	$\lvert \mathcal{Q} \rvert$	Val	Plan	Templ	Time
Blocks	4	3.7	1.9	0.7	1.1	1.79
	6	5.4	3.1	0.8	1.5	5.62
	8	7.4	3.8	1.1	2.5	12.24
Rover	4	3.8	2.0	0.8	1.0	2.21
	6	5.6	3.1	0.8	1.5	7.89
	8	7.3	4.0	1.5	1.8	13.24
	10	9.4	4.8	2.4	2.2	29.53
Satellite	4	3.7	1.8	0.8	1.1	2.11
	6	5.5	2.9	1.2	1.4	6.48
	8	7.4	3.9	1.8	1.7	13.69
	10	9.3	4.8	2.5	2.0	25.55
ZenoTravel	4	3.7	1.8	0.9	1.0	2.05
	6	5.4	3.0	1.0	1.4	5.93
	8	7.4	3.9	1.5	2.0	12.97

Table 1: Comparison of different types of Queries

Objects	$\lvert \mathcal{Q} \rvert$	Val	Plan	Templ	Time
8	5.6	3.1	0.8	1.5	7.89
15	5.7	3.1	1.0	1.4	8.02
22	5.6	3.0	1.3	1.3	8.45
29	5.5	3.2	1.1	1.3	9.11

Table 2: Effect of $\mathcal{I}_e$ on the Queries for Rover domain

predicates were randomly removed from the lifted domain, and asking a question about any grounded action will localize the human model in the lifted domain.

Table 1 shows the evaluation for the different number of unknowns and the time taken to find the questions for the domains. The number of questions generated and the time are averaged over 10 different runs. The decrease in the questions is because of some predicates were merged using the templates. Except for one case in Rover (with 8 unknown predicates), where we were able to find two different merging templates (thus total questions became 6), we usually had roughly 1 question decrease in the problems. The average number of queries are presented in the table for each domain. The algorithm needs to solve multiple planning problems, but due to PIP query generation can be executed offline. All the queries were first constructed and then validated with the human model (correct IPC domain). The table shows that roughly half of the queries were through validation that was higher than our expectations.

Table 2 shows varying the initial state condition on question generation. We constructed new initial states by adding objects to the environment. We randomly removed 3 and added 3 different predicates in the lifted domain and generated questions using different initial states. Time and number of questions are again averaged over 10 different random selections. We expected to have an effect on time due to extra objects for the time taken to solve multiple planning problem. But, since queries were constructed using Graphplan Planning graph, we did not see any change in time, just a very small increase. This also shows that the initial state does not have much effect on the queries, but the causal structure of the domain does.

Related Work

Our work of asking directed questions for model localization, and understanding what every response from the teammate could mean has been motivated from Intelligent Tutoring System community. But the idea of learning model from data points with specific queries or plan traces has been applied in active learning as well as learning models from plan traces (behavior) for the environment.

Intelligent Tutoring System as a community is working towards maximizing the learning of the students for procedural knowledge. Their central goal is to provide a teacher to every student, and the biggest challenge for them is to understand the model of the student from the work they do and provide feedback or new questions to them. The process of generating questions for students has been used in the past (Zhang and VanLehn 2016), and they have also used structured knowledge bases to generate more meaningful questions for concepts like photosynthesis (Zhang and VanLehn 2017). The modeling scheme in ITS is shallow where they represent the knowledge of any concept as a hidden variable using HMM (Corbett and Anderson 1993). Parameters learning using sequential data of student's interaction for HMM (Grover, Wetzel, and VanLehn 2018) or deep neural networks (Piech et al. 2015). Based on the learned models, they have also tried dynamic policies to present questions to students using multi-armed bandits (Clement et al. 2014). Our work differs from the ITS community as we are learning detailed human model for collaboration. We can see this as the first step towards having informative interaction with the user to improve collaboration with them.

Learning planning model using traces. There has been some work to learn the planning models using the behavior in the environment using state predicate differences (Gil 1994; Stern and Juba 2017), weighted max-sat (Yang, Wu, and Jiang 2007) and finite state machines (Cresswell, McCluskey, and West 2009; Cresswell and Gregory 2011). Author's (Zhuo et al. 2020) used deep neural networks to learn shallow machine learning model and use it to predict behavior on the test set. There have been other structured formulations such as Linear Temporal Logic (LTL) to represent the knowledge and learning the model using behavior trajectories and an oracle to validate the model (Camacho and McIlraith 2019). In (Bryce, Benton, and Boldt 2016), authors use a questioning strategy to decrease the number of particles to find how the model has changed from the original behavior. The response to the query is in the form of – model provided by the user, labeled valid plan, or some specific predicate that is part of the model now. Recently, there has been some work to infer the model by asking specific queries in the form of the initial state and the plan (Verma and Srivastava 2019). There are two main differences from our work first, they assume detailed responses of up to which step the plan could be executed in their model, and second, the interaction is involved (like an interrogation), which can overwhelm the human teammate.

Active Learning has an oracle to question classes of specific data points (Settles 2009). Similarity with the community of active learning querying an all knowing oracle we want to learn the underlying model. The difference with the field is that the data of plan traces is not readily available to the agent, such as in scenarios of stream-based selective sampling (Cohn 1994). Stream-based active learning ideas (Dagan and Engelson 1995) are useful for the continuous space of probability distribution, but can't be used directly in our discrete space of questions.

As we can see, the idea of questioning the user (or an oracle) for relevant information is not new. In this paper, we have looked at how it is useful for human-robot teaming. The essential part is to understand how to formally define interaction in the form of question and answer and generate useful queries to localize the model. Where the idea might not be new, the application to the novel area comes with its different challenges.

Conclusion and Future Work

Through this paper we have shown how to construct queries with uncertainty in the model. The analyze how to distinguish between different models through behavioral plans and using bounds on the models. The proposed solution ensures that the conditions for the interaction to be meaningful to elicit specific properties of the model. Evaluation show that these questions can be constructed with higher degree of unknowns in the domains as well as under different initial conditions of the environment. The responses are simple enough such as is the plan valid or not, to get minimal information from the user. The next step for us is to incorporate any response from the user for plan construction and understand how it elicits model information when teammate is not an optimal planner.

Ethical Impact

The work is motivated by collaborating with human teammates. But we as authors believe that any automated agent becomes self-aware when it has its own goals. Here the goal of the agent is to learn the user's model for collaboration and provide support for achieving the team's goals. These goals are of the human teammate, and thus, ethical scenarios and societal impact of this work are minimized.

References

Bryce, D.; Benton, J.; and Boldt, M. W. 2016. Maintaining evolving domain models. In *Proceedings of the twenty-fifth international joint conference on artificial intelligence*, 3053–3059.

Camacho, A., and McIlraith, S. A. 2019. Learning interpretable models expressed in linear temporal logic. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 29, 621–630.

Clement, B.; Roy, D.; Oudeyer, P.-Y.; and Lopes, M. 2014. Online optimization of teaching sequences with multi-armed bandits.

Cohn, D. A. 1994. Neural network exploration using optimal experiment design. In *Advances in neural information processing systems*, 679–686.

Corbett, A. T., and Anderson, J. R. 1993. Student modeling in an intelligent programming tutor. In Lemut, E.; du Boulay, B.; and Dettori, G., eds., *Cognitive Models and Intelligent Environments for Learning Programming*, 135–144. Berlin, Heidelberg: Springer Berlin Heidelberg.

Cresswell, S., and Gregory, P. 2011. Generalised domain model acquisition from action traces. In *Twenty-First International Conference on Automated Planning and Scheduling*. Citeseer.

Cresswell, S.; McCluskey, T. L.; and West, M. M. 2009. Acquisition of object-centred domain models from planning examples. AAAI press.

Dagan, I., and Engelson, S. P. 1995. Committee-based sampling for training probabilistic classifiers. In *Machine Learning Proceedings 1995*. Elsevier. 150–157.

Doignon, J.-P., and Falmagne, J.-C. 2015. Knowledge spaces and learning spaces. *arXiv preprint arXiv:1511.06757*.

Falmagne, J.-C.; Cosyn, E.; Doignon, J.-P.; and Thiéry, N. 2006. The assessment of knowledge, in theory and in practice. In *Formal concept analysis*. Springer. 61–79.

Fikes, R. E., and Nilsson, N. J. 1971. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence* 2(3-4):189–208.

Garrido, A., and Jiménez, S. 2020. Learning temporal action models via constraint programming.

Gil, Y. 1994. Learning by experimentation: Incremental refinement of incomplete planning domains. In *Machine Learning Proceedings 1994*. Elsevier. 87–95.

Grover, S.; Wetzel, J.; and VanLehn, K. 2018. How should knowledge composed of schemas be represented in order to optimize student model accuracy? In *International Conference on Artificial Intelligence in Education*, 127–139. Springer.

Helmert, M. 2006. The fast downward planning system. *Journal of Artificial Intelligence Research* 26:191–246.

Hoffmann, J.; Porteous, J.; and Sebastia, L. 2004. Ordered landmarks in planning. *Journal of Artificial Intelligence Research* 22:215–278.

Kambhampati, S.; Parker, E.; and Lambrecht, E. 1997. Understanding and extending graphplan. In *European Conference on Planning*, 260–272. Springer.

Keyder, E.; Richter, S.; and Helmert, M. 2010. Sound and complete landmarks for and/or graphs. In *ECAI*, volume 215, 335–340.

Moore, R. C. 1994. Making the transition to formal proof. *Educational Studies in mathematics* 27(3):249–266.

Nguyen, T. A.; Kambhampati, S.; and Do, M. 2013. Synthesizing robust plans under incomplete domain models. In *Advances in Neural Information Processing Systems*, 2472–2480.

Nguyen, T.; Sreedharan, S.; and Kambhampati, S. 2017. Robust planning with incomplete domain models. *Artificial Intelligence* 245:134–161.

Novak, J. D., and Cañas, A. J. 2006. The theory underlying concept maps and how to construct them. *Florida Institute for Human and Machine Cognition* 1(1):1–31.

Ortony, A., and Rumelhart, D. E. 1977. The representation of knowledge in memory. *Schooling and the acquisition of knowledge* 99–135.

Piech, C.; Bassen, J.; Huang, J.; Ganguli, S.; Sahami, M.; Guibas, L. J.; and Sohl-Dickstein, J. 2015. Deep knowledge tracing. In *Advances in neural information processing systems*, 505–513.

Settles, B. 2009. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences.

Shultz, T. R. 1982. Rules of causal attribution. *Monographs of the society for research in child development* 1–51.

Sreedharan, S., and Kambhampati, S. 2018. Handling model uncertainty and multiplicity in explanations via model reconciliation. In *Twenty-Eighth International Conference on Automated Planning and Scheduling*.

Stern, R., and Juba, B. 2017. Efficient, safe, and probably approximately complete learning of action models. *arXiv preprint arXiv:1705.08961*.

Verma, P., and Srivastava, S. 2019. Learning generalized models by interrogating black-box autonomous agents. *arXiv preprint arXiv:1912.12613*.

Vinner, S., and Hershkowitz, R. 1980. Concept images and common cognitive paths in the development of some simple geometrical concepts. In *Proceedings of the fourth international conference for the psychology of mathematics education*, volume 1, 177–184.

Yang, Q.; Wu, K.; and Jiang, Y. 2007. Learning action models from plan examples using weighted max-sat. *Artificial Intelligence* 171(2-3):107–143.

Zhang, L., and VanLehn, K. 2016. How do machine-generated questions compare to human-generated questions? *Research and practice in technology enhanced learning* 11(1):7.

Zhang, L., and VanLehn, K. 2017. Adaptively selecting biology questions generated from a semantic network. *Interactive Learning Environments* 25(7):828–846.

Zhuo, H. H.; Zha, Y.; Kambhampati, S.; and Tian, X. 2020. Discovering underlying plans based on shallow models. *ACM Transactions on Intelligent Systems and Technology (TIST)* 11(2):1–30.