Temporal Logic-Based Intent Monitoring for Mobile Robots

Hansol Yoon and Sriram Sankaranarayanan

***Abstract*— We propose a framework that uses temporal logic specifications to predict and monitor the intent of a robotic agent through passive observations of its actions over time. Our approach uses a set of possible hypothesized intents specified as Büchi automata, obtained from translating temporal logic formulae. Based on observing the actions of the robot, we update the probabilities of each hypothesis using Bayes rule. Observations of robot actions provide strong evidence for its "immediate" short-term goals, whereas temporal logic specifications describe behaviors over a "never-ending" infinite time horizon. To bridge this gap, we use a two-level hierarchical monitoring approach. At the lower level, we track the immediate short-term goals of the robot which are modeled as atomic propositions in the temporal logic formalism. We apply our approach to predicting intent of human workers and thus their movements in an indoor space based on the publicly available THÖR dataset. We show how our approach correctly labels each agent with their appropriate intents after relatively few observations while predicting their future actions accurately over longer time horizons.**

I. Introduction

In this paper, we present an approach that predicts and monitors an agent's intent in a given environment through passive observations of their actions. It is useful for many robotics applications. Accurate prediction of intents (goals) leads to predictions of future actions and states of an agent. Likewise, monitoring adherence to the stated intents allows us to detect failures or deviations during deployment.

A large volume of work on intent monitoring focuses on using observations of the robot's action to predict its "immediate" goals. Fig. 1 shows the (x, y) positions of an agent from an actual play of the popular game Overcooked [1], an environment that has been used in the past to study human-robot interaction problems [2]. The figure also shows the stations in the kitchen $p_0 - p_3$ and an oven p_4. From each segment of the trajectory, it is possible to predict that immediate goal is to reach a particular region either by extrapolating its trajectory [3], or by comparing its trajectory against the most efficient path to the region [4], [5]. However, this single immediate goal is part of a larger mission that will dictate the subsequent future goals. For instance, the chef may be executing a mission that involves repeatedly moving between the regions p_1 and p_4. We therefore distinguish between the immediate or "low-level" intent (reaching the oven) and the longer-term "higher-level" intent (the patrolling mission).

In this work, we use temporal logic specifications for high-level intents [6], [7]. Temporal logic has emerged as a very powerful framework for specifying a rich set of reactive behaviors for robots [8]. Efficient algorithms for synthesizing planners and low-level controllers to achieve these specifications are well-known [8], [9], [10], [11]. Temporal logics specify behaviors over an infinite time horizon although finite time horizon temporal logic specifications have also been studied [12]. Infinite time horizon specifications appear unnatural at first glance, since all our data is over a finite time horizon. Nevertheless, temporal logic specifications involving infinite time horizons are a convenient approach to specifying common robot tasks [8]. The time horizons involved in these tasks are long enough to justify idealizing them as infinite/never-ending.

Starting from a given set of temporal logic formulas that express the possible intent hypotheses, we use standard approaches to translate them into *Büchi automata*, which are finite state machines that accept infinite traces [13]. We associate traces of these automata with costs that are incurred by the robot when executing them. In turn, we use these costs with the Boltzmann noisy rationality model of robot actions to assume a distribution over the possible ways a robot could satisfy each hypothesized intent [14], [15]. Our approach then uses a Bayesian inference framework at two levels: at the level of the robot workspace, it observes actions of the robot to predict its immediate goal. At the level of the hypothesized intents, our framework tracks a *belief state* that consists of a probability for each hypothesized intent and each state of the automaton corresponding to the intent. Our approach uses Monte-Carlo simulations using samples from the posterior belief state to infer a tree of possible future actions beyond the immediate goals of the robot.

We evaluate our approach on a real-world human trajectory dataset, THÖR [16] that tracks the positions of nine individuals moving between six pre-defined regions in an environment. These include individuals performing pre-defined tasks that can be specified using temporal logic in addition to "visitors" who may wander around the workspace in an arbitrary manner. We use this dataset to address two questions: (1) Can the framework reliably distinguish between the ground-truth intents for each agent against various "confounders" represented by intents of other agents or just "random" confounding hypotheses?; and (2) Can the framework accurately predict future goals of the agent beyond its immediate goals? The experimental results demonstrate that our approach is effective at predicting three subsequent goals of rational agents, even when there are confounding hypotheses. Additionally, we note that the framework can deduce the correct role of each person with high probability from relatively short observation sequences.

The THÖR trajectory data is rather well-suited to evaluate

Department of Computer Science, University of Colorado Boulder, USA, {firstname.lastname}@colorado.edu

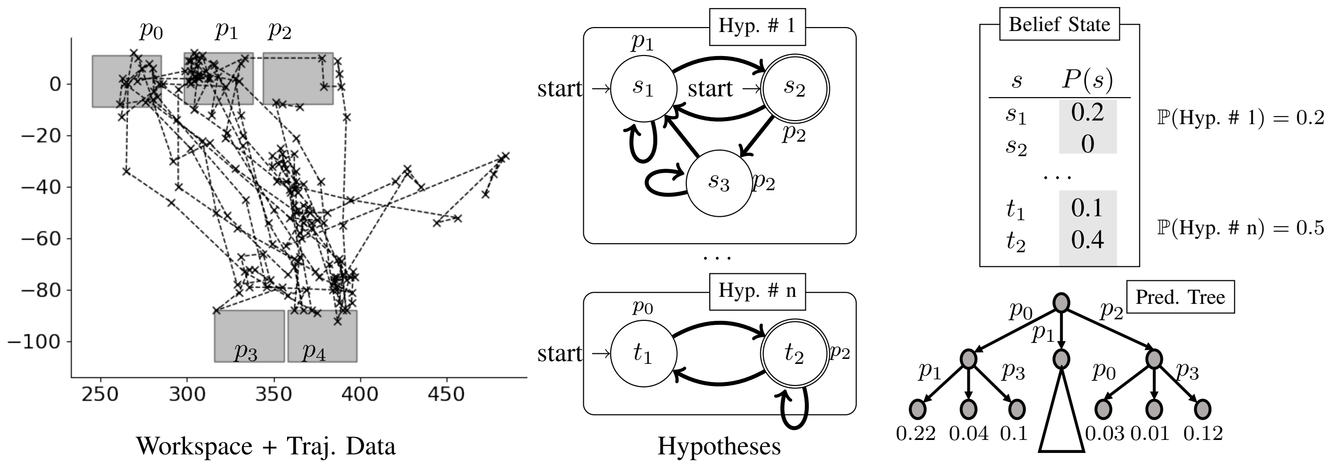


Fig. 1: Components of our approach at a glance. Inputs: trajectory of the agent and workspace regions labeled by atomic propositions and hypothesized intents. Outputs: Belief state and prediction tree for future behaviors.

our approach since the dataset involves participants with specific roles, such as visitor, lab worker or utility worker. These roles can themselves be specified as temporal logic formulas. This is in contrast to applications involving autonomous driving wherein we seek short-term (1-2 seconds) or medium term (3-5 second) predictions for the future positions of agents in the road. Such predictions involve additional contextual information such as pedestrian walk signals and/or markings on the road [17]. Furthermore, predictions of future trajectories can be achieved using machine learning approaches that do not necessarily need to label agents with their presumed intents [18].

II. Related Work

The problem of intent inference has its origins in the AI planning community, wherein many different approaches to plan/intent recognition have been considered [19]. The Bayesian approach which we adopt in this paper arises from the work of Charniak and Goldman who use a Bayesian network called plan recognition network [20]. Interestingly, the need for a hierarchical approach is already evident in their work, wherein the plan recognition network naturally places "high level" plans at the root of the network and the lowest level actions at the leaves which constitute the observations. Key differences include (a) our approach focuses on intent inference for reactive (temporal logic) specifications rather than plans that consist of finitely-many, partial-ordered sequence of actions; and (b) typically plan recognition requires us to construct detailed background theories in first-order logic to bridge from various hypothesized high level goals to possible sequences of actions to carry out these goals. In our approach, such "background knowledge" is implicit in our workspace model, and also in the Boltzmann noisy rationality model used in this paper.

Inverse reinforcement learning (IRL) is yet another related area that focuses on inferring an unknown reward structure for a Markov decision process (MDP) from observations of actions corresponding to some policy [21]. IRL has been used to recognize agent goals by modeling the environment and actions as MDPs and using interpreting how various goals can be specified by different sets of rewards in this MDP [22]. The IRL approach has been very popular with a variety of approaches that seek to apply IRL to various problems of inferring agent goals. Our approach proposed here concerns reactive specifications expressed in temporal logic that are not easy to convert into a set of rewards associated with states/actions of an MDP [23].

Other closely related approaches infer temporal-logic or automata-based specifications directly from observations of a robot's trajectory [24], [25], [26], [27]. The main differences are two-fold: (a) the approaches described above often consider the entire trajectory or a set of trajectories whereas our work considers a set of observations that are part of a trajectory; (b) we adopt a Bayesian approach over a fixed-set of hypothesis whereas the works cited above often consider a large (even potentially infinite) set of possible formulas but infer a best explanation that explains the observed traces.

Whereas we focus on a hierarchical approach, there are numerous works that rely on predicting lower-level (immediate) goals [4], [5], [14]. Best et al. propose a Bayesian inference approach that focuses on recognizing an unknown goal of the robot [4]. A similar approach is taken in our previous work for a slightly more general class of "reach-while-avoid" formulas in temporal logic that represent a subset of possible temporal logic specifications well suited for expressing such "low-level" intents [5]. In this work, we generalize this class of temporal logics considerably but require a new hierarchical approach. Fisac et al. present an interesting combination of intent inference for a human agent inside a shared environment where the intent of the human is used to forecast their immediate future trajectories and guide the robot's planning to avoid the human [14].

III. Preliminaries

In this section, we briefly present models of the workspace, and Büchi automata for representing omega-regular languages that describe the goals of the agent.

A. *Workspace*

We will assume that the robotic agents operate in a *workspace* which is modeled by a workspace graph.

Definition 3.1 (Workspace Graph): A workspace graph M is a tuple : $\langle C, E, A, \ell \rangle$ consisting of

1) A set of cells C, wherein each cell $c_i \in C$ represents a set of possible robot states (e.g., position inside a region and velocity within some bounds).
2) A weighted edge relation E, wherein each edge is of the form (c_1, w, c_2) with $c_1, c_2 \in C$ representing the source and target cells, and $w \in \mathbb{R}_{\geq 0}$ representing the non-negative edge cost. The edge represents a state transition from cell c_1 to neighboring cell c_2 with w representing a cost associated with the move.
3) A set of atomic proposition labels $A : \{p_1, \ldots, p_k\}$ representing special "designations" attached to cells such as work stations, ovens, stoves and so on. We assume that each cell can receive at most one such label.
4) $\ell : C \mapsto A \cup \{\epsilon\}$ a map from each cell to an atomic proposition label, or a special symbol ϵ denoting that the cell is not labeled by any atomic proposition.

Example 1: Consider the Overcooked game-style environment, which is used to study human-AI coordination [28], as illustrated in Fig. 1. It consists of a set of cells along a grid. There are five designated cells labeled by atomic propositions $p_0 - p_4$ corresponding to work stations and an oven. Each cell is connected to at most eight neighboring cells with associated edge cost.

Given a workspace graph M, a robot's trajectory can be described as a finite/infinite sequence of cells $\mathcal{T} : c(0), c(1), c(2), \ldots$, wherein each $c(i) \in C$ and $(c(i), w(i), c(i+1)) \in E$ for each $i \geq 0$ for some cost $w(i) \geq 0$. Furthermore, with each trajectory $\mathcal{T}$, we associate a sequence of atomic proposition labels $\ell(c(0)), \ell(c(1)), \ell(c(2)), \ldots$. The symbol ϵ stands for the empty string in the sequence, and is elided. Also, for convenience we will denote the atomic proposition sequence given by a trajectory $\mathcal{T}$ as $\ell(\mathcal{T})$.

Example 2: When the chef agent is moving from the station p_1 to p_2, one of possible trajectories is a sequence of 4 cells connecting p_1 and p_2 with a running cost of 4.8, and corresponding labels are $p_1, \epsilon, \ldots, \epsilon, p_2$ or simply p_1, p_2 after removing the ϵ symbols.

Our framework relies on two important shortest path notions defined below.

Definition 3.2 (Shortest Paths): Given a cell c and atomic proposition p, we define the shortest path cost $\mathsf{sp}(c, p)$ to be the cost of the shortest path starting from cell c with $\ell(c) = \epsilon$ to any cell labeled p while ensuring that any intermediate cell in the path is not labeled by an atomic proposition.

Likewise, we define $\mathsf{sp}(p, q)$ for two atomic propositions p, q to be the shortest path starting from any cell c labeled with atomic proposition p to any cell labeled with atomic proposition q such that no intermediate cell in the path visits a node labeled with an atomic proposition other than p, q.

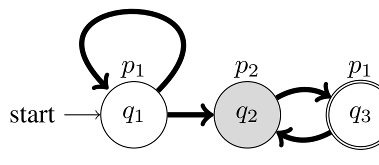


Fig. 2: Büchi automaton for the property that after staying in region p_1 for some finite initial time, the robot repeatedly cycles between p_1 and p_2.

B. *Goal Specifications: Büchi Automata*

We will now recall the use of ω-regular languages to model the temporal goals of a robotic agent.

Definition 3.3 (Büchi Automaton): A Büchi automaton is a finite state machine $\langle Q, A, \delta, \mathsf{label}, Q_0, F \rangle$ wherein, (i) $Q : \{q_1, \ldots, q_m\}$ is a finite set of states, (ii) $A : \{p_1, \ldots, p_k\}$ is a finite set of atomic proposition labels, (iii) δ is an edge relation $\delta \subseteq Q \times Q$ wherein $(q_i, q_j) \in \delta$ signifies an edge from q_i to q_j, (iv) $\mathsf{label} : Q \to A$ is a *labelling* map associating each state with an atomic proposition, (v) $Q_0 \subseteq Q$ is the set of initial states, and (vi) $F \subseteq Q$ is a set of accepting states.

Given an infinite sequence of atomic proposition labels, $\sigma : p_1, p_2, p_3, \ldots$, wherein $p_j \in A$, a run corresponding to such a sequence (if one exists) is an infinite sequence of states $q_0, q_1, q_2, \cdots$ such that (a) $q_0 \in Q_0$ is an initial state, (b) $(q_{i-1}, q_i) \in \delta$ for all $i \geq 1$, and (c) $\mathsf{label}(q_i) = p_i$. Due to the non-deterministic nature of the Büchi automaton, it is possible that for a given sequence of atomic proposition labels, we may have no runs, a single run or multiple runs.

A sequence of atomic propositions σ is *accepted* by a Büchi automaton iff there exists a corresponding run which visits some accepting states in the set F infinitely often. The language $L(\mathcal{A})$ for automaton $\mathcal{A}$ is the set of all sequences of atomic propositions that are accepted by it.

An example of a Büchi automaton corresponding to a mission specification is given in Fig. 2. Büchi automata are a widely used approach to specifying a rich set of robotic agent behaviors [8], [9], [10], [11]. It is well known that temporal logic specifications can be translated into non-deterministic Büchi automata. In this paper, we will use Büchi automata to specify behaviors of robotic agents.

Given a trajectory $\mathcal{T} : c(0), c(1), \ldots, c(t), \ldots$, and a Büchi automaton specification $\mathcal{A}$, we say that $\mathcal{T}$ satisfies $\mathcal{A}$, denoted $\mathcal{T} \models \mathcal{A}$ iff the corresponding labeling $\ell(\mathcal{T})$ is accepted by the automaton $\mathcal{A}$.

IV. Hierarchical Intent Monitoring Framework

In this section, we will describe our monitoring framework in multiple steps: (a) A *cost model* that describes the notion of cost associated with satisfying a Büchi automaton (ω-regular) specification; (b) Translating from costs over runs of the Büchi automata to a probability distribution over them; (c) The "belief state" maintained by the monitor and how this is updated upon various robot moves. By integrating these steps, we will describe our overall monitor implementation that predicts the robot behavior further out into the future.

A. Cost Models and Probabilities over Paths

Now, we consider a question that is fundamental to our approach: *Given a specification for a goal, how would the robot go about satisfying such a goal?* A perfectly efficient robot seeking to optimize costs would seek to achieve its goals using the least possible trajectory cost. However, perfect efficiency relies on having an accurate model of the operating environment including costs and sufficient computational resources. Instead, we will use a convenient assumption of *Boltzmann noisy rationality* [15], [29]. Consider a goal specification such as reaching a given target region in the map that can be achieved by a finite length trajectory. Let $\mathcal{T}$ be such a trajectory achieving this goal. The probability that the agent would choose $\mathcal{T}$ is given by $\mathbb{P}(\mathcal{T}) \propto \exp(-\beta\ \mathsf{cost}(\mathcal{T}))$, wherein $\mathsf{cost}(\mathcal{T})$ is the overall cost of the trajectory and $\beta \geq 0$ is a rationality factor. As $\beta \to 0$, all trajectories that achieve a given goal become equi-probable, whereas as $\beta \to \infty$, the least cost path becomes much more probable than sub-optimal paths.

Next, we define costs over the infinite runs in a Büchi automaton $\mathcal{A}$. These costs will inform the probabilities over these runs through the "rationality" assumption. Recall that the states of a Büchi automaton are labeled with atomic propositions, and these atomic propositions refer to designated cells in the workspace of the robot.

Definition 4.1 (Single-Step Cost): Let q_i, q_j be two states in the automaton connected by an edge, such that $p_i : \mathsf{label}(q_i)$ and $p_j : \mathsf{label}(q_j)$. We define a "single-step cost" $\mathsf{cost}(q_i, q_j)$ as equal to $\mathsf{sp}(p_i, p_j)$ (see Def.3.2). If there is no edge from q_i to q_j, we define $\mathsf{cost}(q_i, q_j) = \infty$.

The next component of the cost model concerns the "cost-to-satisfy" starting from a given state of the automaton. Suppose the robot has just visited region p_j and this corresponds to a state q_j in the automaton $\mathcal{A}$, such that $\mathsf{label}(q_j) = p_j$. We would like to associate a cost with an infinite accepting run for the Büchi automaton starting from q_j (recall that such a run must visit some accepting state infinitely often). There is no single unique way to define such a cost. For instance, taking an infinite sum of all individual costs will lead to a divergent summation. Therefore, we have to "weight down" or discount future costs in some manner.

a) Costs for Accepting Runs: First, let $\sigma : q_k \to q_{k+1} \to q_{k+2} \to \cdots$ represent some infinite accepting run of the Büchi automaton starting from the state q_k. There are many options for how to associate a cost with such a run. We discuss a few such options below:

Discounted Cost: For discount factor $0 < \lambda < 1$, we define

$$\mathsf{cost}(\sigma) : \sum_{j=1}^{\infty} \lambda^j \mathsf{cost}(q_{k+j-1}, q_{k+j})\,.$$

Cost-to-Epoch: Let $\mathsf{epoch}(\sigma)$ represent the index $j+k$ of the first state in the run that is accepting:

$$\mathsf{epoch}(\sigma) = \min_{j=0}^{\infty}\{j \mid q_{j+k} \text{ is accepting}\}\,.$$

We define $\mathsf{cost}(\sigma) = \sum_{j=1}^{\mathsf{epoch}(\sigma)} \mathsf{cost}(q_{k+j-1}, q_{k+j})$, as simply the cost to the first accepting state.

Blended Cost: We combine these cost metrics by adding the cost to epoch until the first accepting state with a discounted cost moving forward from that state onwards.

$$\mathsf{cost}(\sigma) : \left\{ \begin{array}{l} \sum_{j=1}^{\mathsf{epoch}(\sigma)} \mathsf{cost}(q_{k+j-1}, q_{k+j}) + \\ \sum_{j=1+\mathsf{epoch}(\sigma)}^{\infty} \lambda^j \mathsf{cost}(q_{k+j-1}, q_{k+j}) \end{array} \right. .$$

b) Cost-to-Satisfy: We will now define the overall cost-to-satisfy starting from an state q_k $\mathsf{costToSat}(q_k)$ as

$$\inf\ \{\mathsf{cost}(\sigma) \mid \sigma : q_k \to q_{k+1} \to q_{k+2} \to \cdots \text{ is accepting.}\}\ .$$

We will assume that the $\mathsf{costToSat}(q)$ for each state has the property that $\mathsf{costToSat}(q) < \infty$ if and only if there are accepting runs starting from the state q. Specifically, if a state q cannot lead to any accepting runs, then $\mathsf{costToSat}(q) = \infty$. Likewise, for states q with at least one accepting run starting from them, $\mathsf{costToSat}(q) < \infty$.

Example 3: Consider the Büchi automaton from Figure 2 with 3 states and 2 atomic propositions $\{p_1, p_2\}$ with $\mathsf{label}(q_1) = \mathsf{label}(q_3) = p_1$ and $\mathsf{label}(q_2) = p_2$. Furthermore, assume $\mathsf{sp}(p_1, p_2) = \mathsf{sp}(p_2, p_1) = 10$.

Suppose we use a discounted cost model with discount factor $\lambda = 0.5$, the cost from state q_1, $\mathsf{costToSat}(q_1) = 0$ since we note that by staying in q_1 for a long enough initial prefix, the expensive moves from p_1 to p_2 and back can be pushed arbitrarily far into the future.

The epoch cost measures the shortest path to the final state without discounting. Under this model $\mathsf{costToSat}(q_1) = 20$ since the shortest path requires us to go from p_1 to p_2 and back to reach the accepting state q_3. ■

Computing the $\mathsf{costToSat}$ for each state, given an automaton can be performed quite efficiently, depending on the kind of cost-metric chosen. The discounted cost can be computed as the solution to a linear programming problem by treating the automaton as a Markov-Decision Process (MDP). Non-accepting runs can be excluded simply by removing any state which does not reach a strongly connected component involving the accepting states of the automaton. The cost-to-epoch can be computed using a simple shortest path algorithm that treats accepting states as destination. By computing both costs, we can also compute the blended cost.

c) From Costs to Probabilities: We can now use the Boltzmann rationality model to designate probabilities associated with state transitions. Let us consider adjacent states q_i, q_j. We define $\mathbb{P}(q_j|q_i)$, the probability of visiting state q_j next having just visited q_i as in (1) below. The expression

$$\mathbb{P}(q_j|q_i) \ \propto\ \exp\left\{-\beta\left(\mathsf{cost}(q_i, q_j) + \mathsf{costToSat}(q_j)\right)\right\} \ =\ \frac{\exp\left(-\beta\left(\mathsf{cost}(q_i, q_j) + \mathsf{costToSat}(q_j)\right)\right)}{\sum_{q_k \in \mathsf{adj}(q_i)} \exp\left\{-\beta\left(\mathsf{cost}(q_i, q_k) + \mathsf{costToSat}(q_k)\right)\right\}} \qquad (1)$$

on the denominator normalizes the probability over all states q_k adjacent to q_i (denoted $q_k \in \mathsf{adj}(q_i)$).

The probability of a finite path $q_1 \to \cdots \to q_n$ is given by the product $\mathbb{P}(q_2|q_1) \times \cdots \times \mathbb{P}(q_n|q_{n-1})$. In other words, the choice of subsequent state is considered independent of the actual path taken to reach the previous state.

Example 4: Continuing with the automaton in Ex. 3, let us assume the cost-to-epoch model. Therefore, $\mathsf{costToSat}(q_1) = 20$, $\mathsf{costToSat}(q_2) = 10$ and $\mathsf{costToSat}(q_3) = 0$. Thus, using Eq. (1), we obtain $\mathbb{P}(q_1|q_1) = \frac{\exp(-\beta(0+20))}{\exp(-\beta(0+20))+\exp(-\beta(10+10))} = 0.5$ for all values of β. Similarly, we can verify that $\mathbb{P}(q_2|q_1) = 0.5$ and $\mathbb{P}(q_3|q_2) = \mathbb{P}(q_2|q_3) = 1$. ■

B. "High-Level" Intent Monitor

Now we will discuss the overall monitoring algorithm. We input automata $\mathcal{A}_1, \ldots, \mathcal{A}_n$. Let each $\mathcal{A}_i$ be denoted by the set of states Q_i, edges δ_i and initial states $Q_{i,0}$. $\mathcal{A}_1, \ldots, \mathcal{A}_n$ share the same set of atomic proposition labels.

Assumption: We will assume that the given hypotheses $\mathcal{A}_1, \ldots, \mathcal{A}_n$ are *exhaustive*. I.e., the union of the languages: $L(\mathcal{A}_1) \cup \cdots \cup L(\mathcal{A}_n) = A^\omega$ covers all possible sequences of atomic propositions. To ensure this, we let $\mathcal{A}_n$ be a "default" automaton that permits all possible behaviors, in order to represent the fact that we are unable to explain the robot's behavior using the remaining hypotheses.

The key task of the monitor is to (a) maintain a belief state and (b) update belief state whenever a workspace "event" occurs: i.e., a robot reaches a region labeled by an atomic proposition. We will first describe the belief state of the robot, followed by how it is updated.

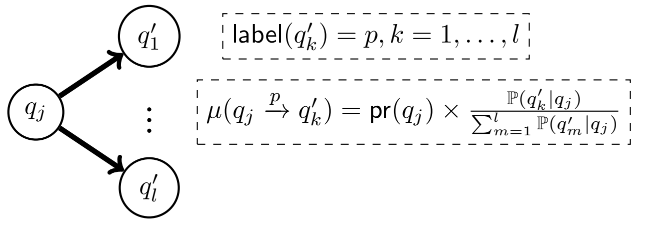
a) Belief State: The overall belief state is a vector of probabilities: for each automaton $\mathcal{A}_i$ and each state $q_k \in Q_i$ of $\mathcal{A}_i$, we maintain $\mathsf{pr}(q_k)$, the probability that we associate with $\mathcal{A}_i$ being the intent *and* being in the state q_k of $\mathcal{A}_i$.

The *initial belief state* is defined as $\mathsf{pr}(q_j) = \frac{1}{nK_i}$, for $q_j \in Q_{i,0}$, an initial state of automaton $\mathcal{A}_i$, and n being the number of automata. Otherwise, $\mathsf{pr}(q_j) = 0$ for non-initial states. Here $K_i = |Q_{i,0}|$. We will ensure the following normalization property of our monitor's belief state at all times: $\sum_{\mathcal{A}_i, i=1}^{n} \sum_{q_k \in Q_i} \mathsf{pr}(q_k) = 1$.

For each automaton $\mathcal{A}_i$, we calculate the overall probability that $\mathcal{A}_i$ specifies the robot's intent as: $\mathsf{pr}(\mathcal{A}_i) = \sum_{q_k \in Q_i} \mathsf{pr}(q_k)$. An intent is said to be *ruled out* if $\mathsf{pr}(\mathcal{A}_i) = 0$: which would indicate that prior actions of the robot have directly violated the specification. Our approach can be easily extended to handle the possibility that the robot may change its intent dynamically by allowing an "implicit" transition from every state to the starting states of every automaton with a low probability that represents the probability of an intent change at each step.

b) Belief State Update: We describe the belief state update for the robot when the robot visits one of the cells designated by the atomic proposition p. The idea is that we will consider the next states for each automaton $\mathcal{A}_i$.

Consider a state q_j belonging to automaton $\mathcal{A}_i$. Suppose an atomic proposition p is encountered, let us consider the states $q'_1, \ldots, q'_l$ that are adjacent to q_j, and labeled with atomic proposition p:


q'_1
q_j
q'_l
$\mathsf{label}(q'_k) = p, k = 1, \ldots, l$
$\mu(q_j \xrightarrow{p} q'_k) = \mathsf{pr}(q_j) \times \frac{\mathbb{P}(q'_k|q_j)}{\sum_{m=1}^{l} \mathbb{P}(q'_m|q_j)}$

We define the "contribution" to next state q'_k from current state q_j upon obtaining the atomic proposition p to be: $\mu(q_j \xrightarrow{p} q'_k)$. If $\mathsf{label}(q'_k)$ is the atomic proposition p, the contribution is as defined above. Otherwise, it is defined to be 0. Note that $\mu(q_j \xrightarrow{p} q'_k)$ distributes the current belief $\mathsf{pr}(q_j)$ in a manner proportional to the probability $\mathbb{P}(q'_k|q_j)$ defined in Eq. (1). We define the normalizing factor N as the sum of all contributions $\sum_{(q,q') \in \delta_j} \mu(q \xrightarrow{p} q')$ over the edges (q, q') of all the automata $\mathcal{A}_j$ for $j = 1, \ldots, n$.

The next belief state probability for a state q in automaton $\mathcal{A}_i$ is given by $\mathsf{pr}'(q) : \frac{1}{N} \sum_{(\hat{q},q) \in \delta_i} \mu(\hat{q} \xrightarrow{p} q)$. The division by the normalizing factor ensures that the resulting belief state is normalized as well.

Thus, we transition from the current belief states $\mathsf{pr}(q_j)$ for each state q_j in each automaton, to the next belief state given by $\mathsf{pr}'(q_j)$ for each state q_j. Let $\mathcal{A}_1, \ldots, \mathcal{A}_n$ be a given set of hypotheses and $\sigma_N : p_0, \ldots, p_N$ be the set of atomic propositions encountered thus far. Our monitoring approach guarantees the following key properties:

Theorem 4.1: The following are true of the belief state of the monitor after encountering σ_N:

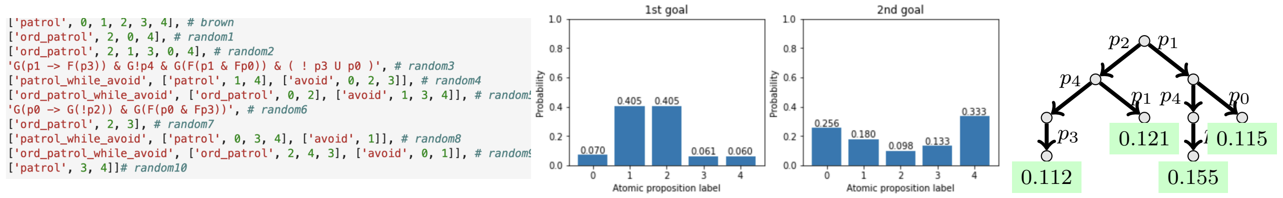
1) The belief state is normalized;
2) For each automaton $\mathcal{A}_i$, if there exists a prefix of an accepting run $q_0 \to \cdots \to q_N$ such that $\mathsf{label}(q_i) = p_i$, then $\mathsf{pr}(q_N) > 0$. I.e., the belief state represents all possible automaton states reached by the finite prefix encountered thus far.
3) $\mathsf{pr}(\mathcal{A}_i) > 0$ iff there exists some infinite suffix σ_w such that $\sigma_N \circ \sigma_w$ is accepted by $\mathcal{A}_i$.

Proofs are provided in the extended version.

Note that our approach treats non-deterministic transitions of the Büchi automaton as probabilistic choices made by the agent using our cost-based model described above. We justify this by noting that various non-deterministic paths in a Büchi automaton represent different "options" available to the robot to satisfy the property represented by the automaton. Therefore, our assumption here states that the choice is made by the robot by taking the costs of these options into account.

V. Hierarchical Predictor

We will describe a hierarchical predictor that constructs a tree of possible future goals of the robot with associated probabilities starting with the next goal. We will first consider an "immediate-goal" monitor that predicts the probability that the next atomic proposition will be p_j, given the history of cells visited by the robot. Using this information, and



```
['patrol', 0, 1, 2, 3, 4], # brown
['ord_patrol', 2, 0, 4], # random1
['ord_patrol', 2, 1, 3, 0, 4], # random2
'G(p1 -> F(p3)) & G!p4 & G(F(p1 & Fp0)) & ( ! p3 U p0 )', # random3
['patrol_while_avoid', ['patrol', 1, 4], ['avoid', 0, 2, 3]], # random4
['ord_patrol_while_avoid', ['ord_patrol', 0, 2], ['avoid', 1, 3, 4]], # random5
'G(p0 -> G(!p2)) & G(F(p0 & Fp3))', # random6
['ord_patrol', 2, 3], # random7
['patrol_while_avoid', ['patrol', 0, 3, 4], ['avoid', 1]], # random8
['ord_patrol_while_avoid', ['ord_patrol', 2, 4, 3], ['avoid', 0, 1]], # random9
['patrol', 3, 4]]# random10
```

Fig. 3: Results on the overcooked data. **(Left)** Specification of hypotheses; **(Middle)** Probabilities of two subsequent goals; and **(Right)** (part of) prediction tree obtained with probability along each branch shown.

the current belief state, we will construct a prediction tree up to some given depth D, wherein each branch provides the probability that the next $D > 0$ atomic propositions are represented by $p^{(1)} \cdots p^{(D)}$. The prediction tree will integrate information from the hypothesized intents and the immediate-goal predictor.

A. Immediate Goal Predictor

The immediate goal predictor uses an approach described in our previous work [5], which was in-turn inspired by that of Best et al [4]. We will elide the details of this approach and describe it at a high level. Given the current cell $c(t)$ visited by the robot in its workspace, our approach updates a probability distribution $\mathbb{P}(p_1|t), \ldots, \mathbb{P}(p_m|t)$ for the atomic proposition corresponding to the region that will be visited next by the robot. If $c(t)$ is already designated by the atomic proposition p_j then we set its probability to 1 and the probability of all other atomic propositions to 0. Otherwise, we use the same Boltzmann rationality assumptions as the high level monitor. Recall the definition of $\mathsf{sp}(c(t), p_j)$ (see Def. 3.2). Let $\mathbb{P}(p_j|t)$, for $p_j \in A$ be the current next-goal probabilities associated with the atomic proposition p at time t. Suppose the robot moves from the cell $c(t)$ to a neighboring cell $c(t+1)$, we update the new probabilities using Bayes' rule :

$$\mathbb{P}(p|t+1) \propto \mathbb{P}(p|t) \times \exp\left\{ -\beta \left(\begin{array}{c} \mathsf{cost}(c(t), c(t+1)) + \\ \mathsf{sp}(c(t+1), p) \end{array} \right) \right\} .$$

We use a recursive formulation wherein we assume $\mathbb{P}(p|t)$ is a prior, and compute $\mathbb{P}(c(t+1)|c(t), p)$ using the equation above. The equation above omits the normalization factor, which can be easily computed.

B. Prediction Tree

We will briefly describe our approach to building a prediction tree of depth up to D where we assume $D \geq 2$ by integrating information from two sources (a) the next goal probabilities $\mathbb{P}(p|t)$ of encountering atomic proposition p, given the history up to time t; (b) the belief state of the high level monitor. Each branch of the prediction tree is a sequence of D atomic propositions $p^{(1)} \ldots p^{(D)}$. We wish to associate a probability for each branch of the tree starting from the current history. We will describe an approach to sampling branches of the prediction tree:

1) Sample $p^{(1)}$ the first atomic proposition according to the next-goal probabilities: $p^{(1)} \sim \mathbb{P}(p|t)$.
2) Compute the belief state of the high level monitor *as if* the atomic proposition $p^{(1)}$ were encountered.
3) Sample a state $q_j \in Q_i$ from automaton $\mathcal{A}_i$ according to the belief-state probability distribution $\mathsf{pr}(q_j)$.
4) Simulate $D-1$ steps in the automaton $\mathcal{A}_i$ starting from q_j wherein next q' is obtained from a previous state q according to probability $\mathbb{P}(q'|q)$ (see Eq. (1)).
5) Record the sequence of $D-1$ atomic proposition labels: $p^{(2)}, \ldots, p^{(D)}$ corresponding to the states encountered in the previous step.

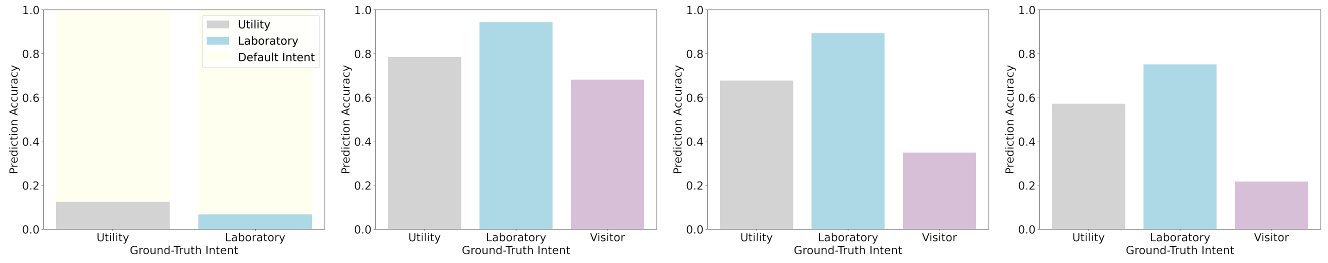
The sampling process described above can be repeated a sufficient number of times to provide unbiased estimates for the probabilities of various branches in the prediction tree. The other branches are assigned 0 (or negligible) probability. Alternatively, we can perform an exhaustive exploration through all possible next goal atomic propositions and all possible sequences of length $D-1$ starting from each automata state with non-zero probability. This can be used to construct the exact probabilities although the resulting complexity will be exponential in D.

C. Running Example

Figure 3 shows our approach on the running example. Our implementation supports easy specification of patterns using linear temporal logic formulas as well as the specification patterns formulated by Menghi et al [8]. In addition to these hypotheses, our approach inputs the trajectory data and the workspace description in order to output the prediction tree as well as the probabilities of the subsequent goals.

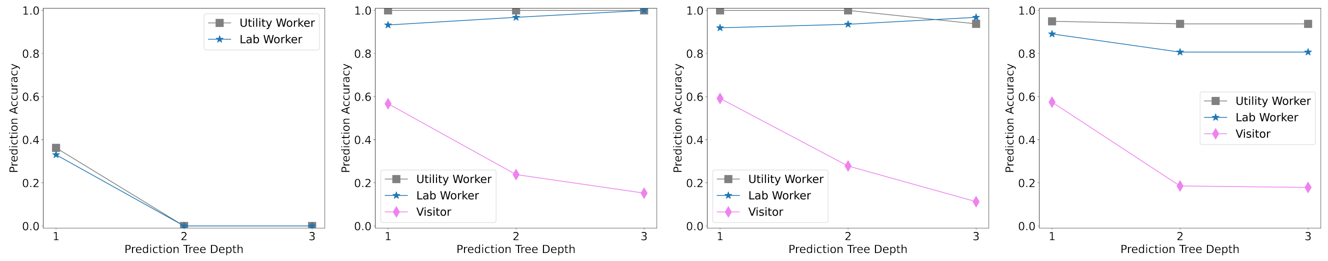
VI. Experimental Evaluation

In this section, we evaluate the proposed framework using an indoor human trajectory data set, THÖR, which contains over 600 human trajectories. The data was collected in a room of 8.4×18.8 m and provides (x, y) position of participants. We discretized the room and all trajectory data into a workspace with 100×50 cells. The workspace contains six designated regions, and we labeled them with atomic proposition labels $p_0, p_1, \ldots, p_5$. Nine participants repeatedly patrol those regions according to pre-defined roles: *visitors*, *laboratory worker*, and *utility worker*. Six of the nine participants are visitors, who keep strolling between regions designated p_0, p_1, p_2, and p_3, in an unspecified order. Two participants designated as workers patrol assigned regions; the laboratory worker patrols p_0, p_2, and p_4, following the specific order, and the utility worker patrols p_1 and p_4, in



(a) One incorrect intent (b) Three ground-truth intents (c) (b) + three random intents (d) (b) + ten random intents

Fig. 4: Prediction accuracy for identifying the role of an observed trajectory with a given ground-truth role. The y-axis measures how often the most likely intent coincides with the ground-truth for that role. We report this for varying number of additional confounding hypotheses.



(a) One incorrect intent (b) Three ground-truth intents (c) (b) + three random intents (d) (b) + ten random intents

Fig. 5: Performance of the prediction tree forecasting three next target regions of a robot. The x-axis represents predictions for the immediate goal, and the two subsequent goals labeled 1, 2, and 3 respectively. The y-axis represents the fraction of time the most likely prediction coincided with the actual ground-truth.

an unspecified order. Every participant avoids regions that are not part of their targets. We generated temporal logic specifications for each of the three roles, which are labeled as "ground-truths" for these roles.

We evaluate our framework with four test cases distinguished by the type and number of hypothesized intents. (a) A single hypothesis that does not correctly describe any role; (b) Three of the ground-truth intents for the visitor, lab worker, and utility worker; (c) Three randomly generated "confounding" intents in addition to the three ground-truth intents; (d) Ten randomly generated "confounding" intents in addition to the three ground-truth intents. The randomly generated "confounding" intents follow the same overall pattern as the ground-truth intent: i.e., involving patrolling some regions while avoiding others. However, the regions to patrol and avoid are chosen randomly. Thus, there may be some overlap between the confounding intents and the actual ground-truths. In all cases, we add a "default" intent represented by a Büchi automaton accepting all possible sequences.

Code for the experiment and data can be found in `https://github.com/cuplv/HIMO_intent/`. The details of hypothesized intents used for each test case are available as part of the repository above.

Identifying Correct Intent: Fig. 4 illustrates our framework's performance in recognizing the correct intent while rejecting other intents. When an incorrect intent is provided, the framework infers that the given intent is unable to explain the observed trajectory. As a result, the default intent receives a high probability, as shown in Fig. 4a. In contrast, the framework effectively detects the real intent of each participant when the ground-truth is a part of the hypothesized intents. We note that the performance degrades as we add more confounding intents, as expected. However, the ground-truth intents are still the most probable among all hypothesized intents in all cases. The probability assigned to the ground-truth for visitor is markedly lower than the other two roles, because visitor trajectories are quite sub-optimal in terms of cost when compared to the two worker roles.

Goal Prediction: Fig. 5 shows the prediction accuracy in forecasting three subsequent goals for the robot. We consider a prediction attempt a "success" when the most probable atomic proposition predicted by the framework matches the actual ground-truth. As illustrated in Fig. 5a, accurate prediction is impossible over random hypotheses. However, the immediate goal prediction is unaffected. Fig. 5b-5d illustrates that the framework is capable of predicting long-term goals with high probability for laboratory and utility workers. Since their tasks consist of patrolling specific regions in a fixed order, the framework can predict future goals unless they change their intents during the missions. On the contrary, the prediction accuracy for visitors is lower than that of others due to their sub-optimal behaviors, as noted.

All computations were performed on a MacBook Pro with 2.6 GHz Intel Core i7 and 16GB RAM. The computation time for monitoring, including generating a prediction tree, took less than one second, on average.

VII. Conclusion

Thus, we have proposed a logic-based hierarchical intent monitoring framework for mobile robots that predicts both high-level and low-level intents. Future work will consider extensions to more interesting scenarios combining robot actions with changes in the environment. We are also investigating extensions to cases where the robot is non-holonomic and in the presence of constraints on its motions.

ACKNOWLEDGMENT

We thank the anonymous reviewers for their insightful comments. This work was funded by the US National Science Foundation (NSF) under award numbers 1815983 and 1836900 and the NSF/IUCRC Center for Autonomous Air Mobility and Sensing (CAAMS).

References

[1] GhostTown Games Inc., "Overcooked on Steam," [Online; accessed 1. Mar. 2023]. [Online]. Available: https://store.steampowered.com/app/448510/Overcooked

[2] M. Fontaine, Y.-C. Hsu, Y. Zhang, B. Tjanaka, and S. Nikolaidis, "On the Importance of Environments in Human-Robot Coordination," in *Proceedings of Robotics: Science and Systems*, Virtual, July 2021.

[3] Y. Chou, H. Yoon, and S. Sankaranarayanan, "Predictive runtime monitoring of vehicle models using bayesian estimation and reachability analysis," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 2111–2118.

[4] G. Best and R. Fitch, "Bayesian intention inference for trajectory prediction with an unknown goal destination," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 5817–5823.

[5] H. Yoon and S. Sankaranarayanan, "Predictive runtime monitoring for mobile robots using logic-based bayesian intent inference," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 8565–8571.

[6] Z. Manna and A. Pnueli, *Temporal Verification of Reactive Systems: Specification*. New York: Springer, 1992.

[7] E. A. Emerson, *Handbook of Theoretical Computer Science: Formal Models and Semantics*. North-Holland Publishing Company and MIT Press, 1990, vol. B, ch. Temporal and Modal Logic, pp. 995–1072.

[8] C. Menghi, C. Tsigkanos, P. Pelliccione, C. Ghezzi, and T. Berger, "Specification patterns for robotic missions," *IEEE Transactions on Software Engineering*, vol. 47, no. 10, pp. 2208–2224, 2021.

[9] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Temporal-logic-based reactive mission and motion planning," *IEEE Transactions on Robotics*, vol. 25, no. 6, pp. 1370–1381, 2009.

[10] G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas, "Temporal logic motion planning for dynamic robots," *Automatica*, vol. 45, no. 2, pp. 343–352, 2009.

[11] A. Ulusoy, S. L. Smith, X. C. Ding, C. Belta, and D. Rus, "Optimality and robustness in multi-robot path planning with temporal logic constraints," *The International Journal of Robotics Research*, vol. 32, no. 8, pp. 889–911, 2013.

[12] G. De Giacomo and M. Y. Vardi, "Linear temporal logic and linear dynamic logic on finite traces," in *IJCAI '13: Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*. AAAI Press, Aug. 2013, pp. 854–860.

[13] C. Baier and J.-P. Katoen, *Principles of Model Checking*. MIT Press, 2008.

[14] J. F. Fisac, A. Bajcsy, S. L. Herbert, D. Fridovich-Keil, S. Wang, C. Tomlin, and A. D. Dragan, "Probabilistically safe robot planning with confidence-based human predictions," in *Proceedings of Robotics: Science and Systems*, 2018.

[15] B. D. Ziebart, A. L. Maas, J. A. Bagnell, A. K. Dey *et al.*, "Maximum entropy inverse reinforcement learning." in *Aaai*, vol. 8. Chicago, IL, USA, 2008, pp. 1433–1438.

[16] A. Rudenko, T. P. Kucner, C. S. Swaminathan, R. T. Chadalavada, K. O. Arras, and A. J. Lilienthal, "Thör: Human-robot navigation data collection and accurate motion trajectories dataset," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 676–682, 2020.

[17] M. Gulzar, Y. Muhammad, and N. Muhammad, "A Survey on Motion Prediction of Pedestrians and Vehicles for Autonomous Driving," *IEEE Access*, vol. 9, pp. 137 957–137 969, Oct. 2021.

[18] R. Chandra, T. Guan, S. Panuganti, T. Mittal, U. Bhattacharya, A. Bera, and D. Manocha, "Forecasting Trajectory and Behavior of Road-Agents Using Spectral Clustering in Graph-LSTMs," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4882–4890, Jun. 2020.

[19] G. Sukthankar, C. Geib, H. Bui, D. Pynadath, and R. P. Goldman, *Plan, Activity, and Intent Recognition: Theory and Practice*. Mogan-Kauffman, 2014.

[20] E. Charniak and R. Goldman, "A bayesian model of plan recognition," *Artificial Intelligence*, vol. 64, pp. 53–79, 11 1993.

[21] A. Y. Ng and S. Russell, "Algorithms for inverse reinforcement learning," in *Proc. International Conf. on Machine Learning (ICML)*. Morgan Kaufmann, 2000, pp. 663–670.

[22] C. L. Baker, R. Saxe, and J. B. Tenenbaum, "Action understanding as inverse planning," *Cognition*, vol. 113, pp. 329–349, 2009.

[23] E. M. Hahn, M. Perez, S. Schewe, F. Somenzi, A. Trivedi, and D. Wojtczak, "Omega-regular objectives in model-free reinforcement learning," in *Tools and Algorithms for the Construction and Analysis of Systems*. Cham: Springer International Publishing, 2019, pp. 395–412.

[24] S. Jha, A. Tiwari, S. A. Seshia, T. Sahai, and N. Shankar, "Telex: learning signal temporal logic from positive examples using tightness metric," *Formal Methods Syst. Des.*, vol. 54, no. 3, pp. 364–387, 2019. [Online]. Available: https://doi.org/10.1007/s10703-019-00332-1

[25] M. Vazquez-Chanlatte and S. A. Seshia, "Maximum causal entropy specification inference from demonstrations," in *Computer Aided Verification*. Cham: Springer International Publishing, 2020, pp. 255–278.

[26] M. Vazquez-Chanlatte, A. Shah, G. Lederman, and S. A. Seshia, "Demonstration informed specification search," *CoRR*, vol. abs/2112.10807, 2021. [Online]. Available: https://arxiv.org/abs/2112.10807

[27] K. Watanabe, N. Renninger, S. Sankaranarayanan, and M. Lahijanian, "Probabilistic specification learning for planning with safety constraints," in *Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 6558–6565.

[28] M. Carroll, R. Shah, M. K. Ho, T. Griffiths, S. Seshia, P. Abbeel, and A. Dragan, "On the utility of learning about humans for human-AI coordination," *Advances in neural information processing systems*, vol. 32, 2019.

[29] C. L. Baker, J. B. Tenenbaum, and R. R. Saxe, "Goal inference as inverse planning," in *Proceedings of the Annual Meeting of the Cognitive Science Society*, vol. 29, no. 29, 2007.