Deep Imitative Reinforcement Learning for Temporal Logic Robot Motion Planning with Noisy Semantic Observations

Qitong Gao, Miroslav Pajic, and Michael M. Zavlanos

***Abstract*— In this paper, we propose a Deep Imitative Q-learning (DIQL) method to synthesize control policies for mobile robots that need to satisfy Linear Temporal Logic (LTL) specifications using noisy semantic observations of their surroundings. The robot sensing error is modeled using probabilistic labels defined over the states of a Labeled Transition System (LTS) and the robot mobility is modeled using a Labeled Markov Decision Process (LMDP) with unknown transition probabilities. We use existing product-based model checkers (PMCs) as experts to guide the Q-learning algorithm to convergence. To the best of our knowledge, this is the first approach that models noise in semantic observations using probabilistic labeling functions and employs existing model checkers to provide suboptimal instructions to the Q-learning agent.**

I. Introduction

Formal languages, such as LTL, can be used to describe robot motion planning problems that range beyond point-to-point navigation [1], e.g. complex tasks in environments that can be modeled as transition systems [2]–[6] or MDPs [7]–[11]. In these problems, uncertainty in the robots or the workspace increases the difficulty in synthesizing robot motion plans [12]. Motion planning problems under uncertainty have been studied extensively in [7]–[11], [13]–[25]. Often, uncertainty in the control actions is captured by the transition probabilities in the MDP [7]–[11], [20]–[23], [25]. However, sensing uncertainty typically gives rise to POMDPs [16], [18], [20], that usually require history-dependent controllers and are computationally very challenging to solve.

In this paper, we consider robots that take noisy semantic observations of their surroundings and are responsible for accomplishing complex tasks captured by LTL formulas. Specifically, we model robot motion using Labeled Markov Decision Processes (LMDPs) and sensing uncertainty by defining probabilistic labels in the Labeled Transition System (LTS) that give rise to probabilistic atomic propositions with unknown probabilities. This way we can model noise in semantic observations, e.g., due to the inability of the robot to accurately distinguish between different regions in space such as a master bedroom and a guest room. To synthesize a policy that maximizes the probability of satisfying the LTL specification, we propose a deep imitative Q-learning (DIQL) approach where a Q-learning agent is mentored by product-based model checkers (PMCs). Existing imitation learning methods [26]–[30] require human experts to provide exact and optimal instructions to the learning agent, in contrast to our approach that only requires PMCs to provide suboptimal instructions that the learning agent can refine to synthesize an optimal control policy.

To the best of our knowledge, the most relevant works are [11], [13], [14], [20]–[24], [31]. Although the Q-learning method proposed in [11] can be applied directly to the problem under consideration, it is inefficient because it does not take in to account sensing information and it requires extremely large numbers of sampled data as well as training steps to learn the optimal Q-function due to the low sample efficiency [32], [33]. Similarly, [22]–[24] develop learning based methods to solve MDPs under LTL specifications but do not incorporate sensing information. In [13], [14], only point-to-point navigation tasks are considered, and in [20] only finite horizions are considered and *memory-dependent* controllers are required. In contrast, in our approach the LTL specifications are satisfied by infinite paths and *memoryless* controllers are sufficient. In addition, in [21], [22] the construction of Accepting Maximal End Components (AMECs) are necessary, however, the method we propose can avoid this process, it can handle large-scale environments, and it can still synthesize a policy even if AMECs do not exist.

The paper is organized as follows: In Section II, we review necessary background. In Section III, we formulate the learning problem. In Section IV, we develop the DIQL algorithm. In Section V, numerical experiments are presented. Conclusions are drawn in Section VI.

II. Preliminaries and Problem Definition

In this section, we briefly review preliminaries on LTL, LTSs, LMDPs and deep Q-learning.

A. LTL Specifications

The basic ingredients of LTL are a set of atomic propositions $\mathcal{AP}$, the boolean operators, i.e., conjunction $\wedge$, and negation $\neg$, and two temporal operators, next $\bigcirc$ and until $\mathcal{U}$. LTL formulas over a set $\mathcal{AP}$ can be constructed based on the following grammar: $\phi ::= \text{true} \mid \xi \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid \bigcirc \phi \mid \phi_1 \ \mathcal{U} \ \phi_2$, where $\xi \in \mathcal{AP}$. For the sake of brevity we abstain from presenting the derivations of other Boolean and temporal operators, e.g., *always* $\Box$, *eventually* $\Diamond$, *implication* $\Rightarrow$, which can be found in [34]. Any LTL formula can be translated into a DRA defined as follows [34].

Definition 2.1 (DRA): A DRA over $2^{\mathcal{AP}}$ is a tuple $\mathcal{R}_\phi = (\mathcal{Q}, q^0, \Sigma, \delta, \mathcal{F})$ where $\mathcal{Q}$ is a finite set of states; $q^0 \subseteq \mathcal{Q}$ is the set of initial states; $\Sigma = 2^{\mathcal{AP}}$ is the input alphabet; $\delta : \mathcal{Q} \times \Sigma \to \mathcal{Q}$ is the transition function and $\mathcal{F} =$

Qitong Gao, Miroslav Pajic and Michael M. Zavlanos are with Duke University, Durham, NC, USA. {qitong.gao, miroslav.pajic, michael.zavlanos}@duke.edu. This work is supported in part by AFOSR under award #FA9550-19-1-0169, by ONR under grants #N00014-18-1-2374, #N00014-20-1-2745 and #N00014-17-1-2504, and NSF CNS-1652544 and CNS-1932011 grants.

$\{(\mathcal{G}_1, \mathcal{B}_1), \ldots, (\mathcal{G}_n, \mathcal{B}_n)\}$ is a set of accepting pairs where $\mathcal{G}_i, \mathcal{B}_i \subseteq \mathcal{Q}, i \in \{1, \ldots, n\}$.

A run of $\mathcal{R}_\phi$ over an infinite word $\omega_\Sigma = \omega_\Sigma(1)\omega_\Sigma(2)\omega_\Sigma(3)\cdots \in \Sigma^\omega$ is an infinite sequence $\omega_\mathcal{Q} = \omega_\mathcal{Q}(1)\omega_\mathcal{Q}(2)\omega_\mathcal{Q}(3)\cdots$, where $\omega_\mathcal{Q}(1) \in q_0$ and $\omega_\mathcal{Q}(k+1) \in \delta(\omega_\mathcal{Q}(k), \omega_\Sigma(k))$ for all $k \geq 1$. Let $\text{Inf}(\omega_\mathcal{Q})$ denote the set of states that appear infinitely often in $\omega_\mathcal{Q}$. Then, a run $\omega_\mathcal{Q}$ is accepted by $\mathcal{R}_\phi$ if $\text{Inf}(\omega_\mathcal{Q}) \cap \mathcal{G}_i \neq \emptyset$ and $\text{Inf}(\omega_\mathcal{Q}) \cap \mathcal{B}_i = \emptyset$ for at least one pair $i \in \{1, \ldots, n\}$ of accepting states $i = 1, \ldots, n$ [35].

B. Robot Sensing Model

In this paper, we model the robot workspace $\mathcal{W} \subseteq \mathbb{R}^d$, $d = 2, 3$ as an LTS defined as follows.

Definition 2.2 (LTS): An LTS is a tuple $\mathcal{T} = (\Pi, U, \delta^t, \Pi^0, \mathcal{AP}, L_\mathcal{I}, L^*, L_\mathcal{H})$ where $\Pi = \{\pi_i | i \in [1, n]\}$ is the set of regions; $U = \{u_i | i \in [1, m]\}$ is the set of control inputs; $\delta^t : \Pi \times U \to 2^\Pi$ is the transition function; $\Pi^0 \subseteq \Pi$ is the set of initial regions; $\mathcal{AP} = \{ap_i | i \in [1, k]\}$ is the set of atomic propositions; $L_\mathcal{I} : \Pi \to 2^{\mathcal{AP}}$ is a probabilistic labeling function that maps regions $\pi_i \in \Pi$ to labels based on instantaneous sensor observations; $L^* : \Pi \to 2^{\mathcal{AP}}$ is the ground-truth labeling function that maps regions $\pi_i \in \Pi$ to the *ground truth* labels and $L_\mathcal{H} : \Pi \to 2^{\mathcal{AP}}$ is a labeling function that maps regions $\pi_i \in \Pi$ to labels based on a history of sensor observations.

Specifically, we define the instantaneous sensing model as

$$L_\mathcal{I}(\pi_i) = \begin{cases} L^*(\pi_i) & \text{with probability } p_1, \\ 2^{\mathcal{AP}} \backslash L^*(\pi_i) & \text{with probability } 1 - p_1, \end{cases} \quad (1)$$

where $\pi_i \in \Pi$ are the regions and we assume that $p_1 > 0.5$. Moreover, we define sensing buffers $\mathcal{E}^{\pi_i}$ that store the history of observations until the current step t as

$$\mathcal{E}_t^{\pi_i} = \{X_k^{\pi_i}\},\ k \in [1, n_t^{\pi_i}], \quad (2)$$

where $X_k^{\pi_i} = 1$ if $L_\mathcal{I}(\pi_i) = L^*(\pi_i)$, $X_k^{\pi_i} = 0$ if $L_\mathcal{I}(\pi_i) \neq L^*(\pi_i)$ and $n_t^{\pi_i}$ denotes the size of $\mathcal{E}_t^{\pi_i}$. Then, the labeling function $L_\mathcal{H}$ is defined as

$$L_\mathcal{H}(\pi_i) = \begin{cases} L^*(\pi_i) & \text{if } Y_k^{\pi_i} > n_t^{\pi_i}/2, \\ 2^{\mathcal{AP}} \backslash L^*(\pi_i) & \text{if } Y_k^{\pi_i} \leq n_t^{\pi_i}/2, \end{cases} \quad (3)$$

where $Y_k^{\pi_i} = \sum_{i=1}^{n_t^{\pi_i}} X_k^{\pi_i}$. In what follows, we use $L_\mathcal{H}$ in (3) as the robot sensing model. The use of the empirical average in (3) can correct for errors in the instantaneous observations in (1). In the above definitions, $X_k^{\pi_i}$ are *Bernoulli* random variables drawn from a distribution $\mathcal{D}$, while $Y_k^{\pi_i}$ are *binomial* random variables with parameters $(n_t^{\pi_i}, p_1)$. Finally, we make the following assumptions on Π and $L_\mathcal{I}$, respectively.

Assumption 2.3: For all $\pi_i \in \Pi$, there is at most one proposition defined over π_i. As a result, $L_\mathcal{I}(\pi_i)$ contains at most one element.

Assumption 2.4: The sensor $L_\mathcal{I}$ only gathers information from the set of regions $\Pi_{ap} = \{\pi_i \in \Pi | L^*(\pi_i) \neq \emptyset\}$ that have been labeled by some $ap \in \mathcal{AP}$. For the set of regions $\Pi_{\neg ap} = \{\pi_i \in \Pi | L^*(\pi_i) = \emptyset\}$ that have not been labeled by any $ap \in \mathcal{AP}$, the sensor does not gather information from them. Hence $L_\mathcal{I}(\pi_i) \not\equiv \emptyset,\ \forall \pi_i \in \Pi_{ap}$ and $L_\mathcal{I}(\pi_j) \equiv \emptyset,\ \forall \pi_j \in \Pi_{\neg ap}$.

C. Robot Motion Model

Robot mobility in the workspace can be represented by an MDP defined as follows.

Definition 2.5 (MDP): An MDP is a tuple $M = (\mathcal{S}, s_0, \mathcal{A}, P, R, \gamma)$, where S is a finite set of states; s_0 is the initial state; $\mathcal{A}$ is a finite set of actions; P is the transition probability function defined as $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$; $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward function; $\gamma \in [0, 1]$ is the discount factor.

In order to capture the uncertainty both in the robot controller and sensor, we extend Definition 2.5 to include probabilistic labels giving rise to an LMDP.

Definition 2.6 (LMDP): An LMDP is a tuple $\mathcal{M} = (M, L_\mathcal{H}, Z_\mathcal{T})$, where M is an MDP; $L_\mathcal{H} : \mathcal{S} \to 2^{\mathcal{AP}}$ is the labeling function that is inherited from the LTS and $Z_\mathcal{T} : \mathcal{S} \to \Pi$ is the function that maps the set of states $\mathcal{S}$ to the set of regions Π defined in the LTS.

In this paper, we assume that the transition probability function P and the ground truth labeling function L^* is unknown. Finally, we define the policy of an LMDP $\mathcal{M}$ as follows.

Definition 2.7 (Policy of LMDP): A deterministic policy τ of an LMDP $\mathcal{M}$ is a function, $\tau : \mathcal{S} \to \mathcal{A}$, that maps each state $s \in \mathcal{S}$ to an action $a \in \mathcal{A}$.

Observe that given an initial state s^0 and a policy τ, an *infinite path* u_τ of the LMDP is generated under the policy τ, defined as an infinite sequence $u_\tau = s^0, s^1, \ldots, s^t, \ldots$, where $s^t \in \mathcal{S}$ is the state of the LMDP at the stage t. Note that, given an action a^t due to the policy τ, a transition from $s^t \in \mathcal{S}$ to $s^{t+1} \in \mathcal{S}$ in the LMDP occurs with probability $P(s^t, a^t, s^{t+1})$, and a scalar reward r^t is generated.

D. Deep Q-learning

Deep Q-learning returns an optimal policy τ^* of a LMDP $\mathcal{M}$ by minimizing the following loss function iteratively:

$$\begin{aligned} J(\theta, \theta') =& \mathbb{E}_{(s,a,s',r)\sim\mathcal{B}}[Q(s, a; \theta) - R(s, a, s') \\ & - \gamma Q\left(s', \hat{\tau}^{(k)}(s'); \theta'\right)\Big)^2], \end{aligned} \quad (4)$$

where β is an exploration policy, such as ϵ-greedy, ρ^β is the state visitation distribution over policy β, $Q(\cdot|\cdot; \theta)$ is the state-action value function evaluated from a policy network parameterized by θ, $Q(\cdot|\cdot; \theta')$ is the state-action value function evaluated from a target network parameterized by θ', $\hat{\tau}^{(k)}(s') = \text{argmax}_a Q(s', a; \theta')$ is the target policy at state s' in iteration k and $\mathcal{B} = \{(s, a, s', r) | s, s' \sim \rho^\beta; r \sim R; a \sim \beta\}$ is the empirical training buffer that collects all prior experiences. We refer to [11], [36] for a complete review of deep Q-learning.

E. Problem Definition

Consider a robot operating in a workspace $\mathcal{W} \subseteq \mathbb{R}^d$, $d = 2, 3$ and let the motion of the robot be captured by an LMDP $\mathcal{M} = (M, L_\mathcal{H}, Z_\mathcal{T})$, as defined in Definition 2.6. Moreover, consider an LTS $\mathcal{T} = (\Pi, U, \delta^t, \Pi^0, \mathcal{AP}, L_\mathcal{I}, L^*, L_\mathcal{H})$ that is constructed and maintained using real-time sensing data

Algorithm 1 DIQL Algorithm

Input: $\theta, \mathcal{B}, max\_epi, max\_step, s^0, q^0, \mathcal{M}, \mathcal{T}, \mathcal{R}_\phi$
Begin:
Initialize $\omega^*_\mathcal{P}|_\mathcal{Q} \leftarrow$ null
while $\omega^*_\mathcal{P}|_\mathcal{Q} ==$ null **do**
  Gather sensing information by (3)
  Construct RPA $\mathcal{P} = \mathcal{T} \otimes \mathcal{R}_\phi$
  Synthesize an optimal accepting sequence $\omega_\mathcal{P}$ of $\mathcal{P}$ ▷ [37]
  if an $\omega_\mathcal{P}$ is found **then**
    $\omega^*_\mathcal{P}|_\mathcal{Q} \leftarrow$ Concatenate(`Unique`(Pre($\omega_\mathcal{P}|_\mathcal{Q}$)), `Unique`(Suf($\omega_\mathcal{P}|_\mathcal{Q}$)))
  else
    continue
  end if
end while
for $epi = 0$ to $max\_epi - 1$ **do**
  $s \leftarrow s^0$, $q \leftarrow q^0$, $x \leftarrow (s, q)$
  for $step = 0$ to $max\_step - 1$ **do**
    $a \leftarrow \hat{\beta}(x)$ ▷ *Follow exploration policy*
    Get s' following the dynamics in $\mathcal{M}$ after taking action a
    $q', r \leftarrow$ `CheckRabin`(s, q, s') ▷ [11]
    $x' \leftarrow (s', q')$
    Append tuple (x, a, r, x') to $\mathcal{B}$
    Sample data batch from $\mathcal{B}$ and update θ by minimizing the loss (4) through gradient descent [36]
    $s \leftarrow s'$, $q \leftarrow q'$, $x \leftarrow x'$
    if x is terminal state **then**
      break
    end if
  end for
end for
$\tau(\cdot) \leftarrow \text{argmax}_{a'} Q(\cdot, a'; \theta)$
return π

Algorithm 2 Acquire Instruction from PMC

Input: $\mathcal{T}$, $\mathcal{R}_\phi$, q, s, $\omega^*_\mathcal{P}|_\mathcal{Q}$, $\mathcal{M}$, C
Begin:
Gather sensing information by (3)
Reset Π_0 as $Z_\mathcal{T}(s)$
if q exists in $\omega^*_\mathcal{P}|_\mathcal{Q}$ **then**
  Locate q in $\omega^*_\mathcal{P}|_\mathcal{Q}$ and store the index into idx
  $\hat{q} \leftarrow \omega^*_\mathcal{P}|_\mathcal{Q}[idx + 1]$
  Get set $\mathcal{L} \leftarrow \{l \in 2^{\mathcal{AP}} | \hat{q} \in \delta(q, l)\}$ from $\mathcal{R}_\phi$
  $\phi' \leftarrow \Diamond(\bigvee_{l \in \mathcal{L}} l)$ ▷ *Set new LTL*
  Construct $\mathcal{R}_{\phi'}$ given ϕ'
  $\mathcal{P}' \leftarrow \mathcal{T} \otimes \mathcal{R}_{\phi'}$
  Synthesize an optimal accepting run $\omega_{\mathcal{P}'} = \omega_{\mathcal{P}'}(1)\omega_{\mathcal{P}'}(2)\omega_{\mathcal{P}'}(3)\cdots$ of $\mathcal{P}'$
  Set $\hat{a}$ as the $u \in U$ that satisfies $\omega_{\mathcal{P}'}(2)|_\Pi \in \delta^t(\omega_{\mathcal{P}'}(1)|_\Pi, u)$
else
  $\hat{a} \leftarrow$ random action
end if
return $\hat{a}$

obtained by the robot. Then, the problem that we address in this paper can be summarized as follows.

Problem 1: Given an LMDP $\mathcal{M} = (M, L_\mathcal{H}, Z_\mathcal{T})$ with unknown transition probabilities P, an unknown set of ground truth atomic propositions $\mathcal{AP}$ and an LTL specification ϕ, find a policy τ that maximizes the probability that the resulting trajectory $tr(\tau) = s^0, s^1, s^2, ..., s^k, ...$ satisfies ϕ, where $s^k \in \mathcal{S}\ \forall\ k \geq 0$, $s^{k+1} \sim P(s^k, \tau(s^k), \cdot)\ \forall\ k \geq 0$.

III. Deep Imitative Q-Learning Algorithm

In this section, we propose a DIQL algorithm to solve Problem 1. Specifically, we design PMCs which take real-time sensing information as input and provide suboptimal instructions to the Q-learning agent during the exploration step within the deep Q-learning framework proposed in [36] to design optimal control policies that maximize the probability of satisfying an LTL specification ϕ.

To construct a PMC, we first define a Rabin product automaton (RPA) as follows [35].

Definition 3.1 (RPA): A RPA is defined by $\mathcal{P} = \mathcal{T} \otimes \mathcal{R}_\phi = (S_\mathcal{P}, s^0_\mathcal{P}, \delta_\mathcal{P}, \Sigma, \mathcal{F}_\mathcal{P})$ where $S_\mathcal{P} = \Pi \times \mathcal{Q}$ is the set of states; $s^0_\mathcal{P} = \Pi^0 \times q^0$ is the initial state; $\delta_\mathcal{P} : S_\mathcal{P} \times \Sigma \to S_\mathcal{P}$ is the transition function where $\delta_\mathcal{P}((\pi_i, q_j), \sigma) = \{(\pi'_i, q'_j) \in S_\mathcal{P} | q'_j \in \delta(q_j, L_\mathcal{H}(\pi_i)), \exists\ u \in U, \pi'_i \in \delta^t(\pi_i, u)\}$; $\Sigma = 2^{\mathcal{AP}}$ is the input alphabet and $\mathcal{F}_\mathcal{P} = (\Pi \times \mathcal{G}_1, \Pi \times \mathcal{B}_1), \ldots, (\Pi \times \mathcal{G}_n, \Pi \times \mathcal{B}_n)$ is the Rabin acceptance condition.

Since the RPA is still a DRA, the accepting run of $\mathcal{P}$ over an infinite word $\omega_\Sigma = \omega_\Sigma(1)\omega_\Sigma(2)\omega_\Sigma(3)\cdots \in \Sigma^\omega$ is also a infinite sequence $\omega_\mathcal{P} = \omega_\mathcal{P}(1)\omega_\mathcal{P}(2)\omega_\mathcal{P}(3)\cdots$, where $\omega_\mathcal{P}(1) \in s^0_\mathcal{P}$ and $\omega_\mathcal{P}(k+1) \in \delta_\mathcal{P}(\omega_\mathcal{P}(k), \omega_\Sigma(k))$. The sequence $\omega_\mathcal{P}$ can be projected into a sequence of $\mathcal{T}$ denoted by $\omega_\mathcal{P}|_\Pi = \omega_\mathcal{P}(1)|_\Pi \omega_\mathcal{P}(2)|_\Pi \omega_\mathcal{P}(3)|_\Pi \cdots$, and a sequence of $\mathcal{R}_\phi$ denoted by $\omega_\mathcal{P}|_\mathcal{Q} = \omega_\mathcal{P}(1)|_\mathcal{Q} \omega_\mathcal{P}(2)|_\mathcal{Q} \omega_\mathcal{P}(3)|_\mathcal{Q} \cdots$. A run $\omega_\mathcal{P}$ is accepted by $\mathcal{P}$ if $\text{Inf}(\omega_\mathcal{P}) \cap (\Pi \times \mathcal{G}_i) \neq \emptyset$ and $\text{Inf}(\omega_\mathcal{P}) \cap (\Pi \times \mathcal{B}_i) = \emptyset$ for at least one pair $i \in \{1, \ldots, n\}$ of accepting states $i = 1, \ldots, n$.

The algorithm contains two phases: PMC initialization and imitative learning, which are introduced in the following subsections respectively.

A. PMC Initialization Phase

We initialize the PMC with an optimal rabin sequence $\omega^*_\mathcal{P}|_\mathcal{Q}$ that is essential to synthesize control instructions during the learning phase that follows; see, e.g., [Alg.1, line 16] using the exploration policy, which is introduced with details in the next subsection as in (5). The initialization phase starts by initializing the optimal Rabin sequence $\omega^*_\mathcal{P}|_\mathcal{Q}$ [Alg. 1, lines 1-11]. Specifically, we set $\omega^*_\mathcal{P}|_\mathcal{Q}$ as null at the beginning and perform the following process repeatedly until $\omega^*_\mathcal{P}|_\mathcal{Q}$ is found: first, the robot sensor acquires sensing information using the the noisy robot sensing model (3) and the outputs of $L_\mathcal{H}$ are updated accordingly; second, we

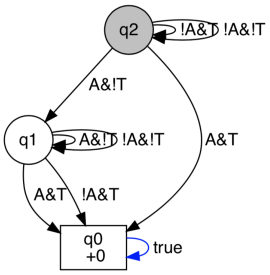


Fig. 1. The DRA $\mathcal{R}_\phi$ corresponding to $\phi = \diamondsuit(A \wedge \diamondsuit T)$, where q_2 is the initial state and the set of accepting pairs $\mathcal{F} = \{(\{q_0\}, \emptyset)\}$.

construct a RPA $\mathcal{P}$ by taking the product between the LTS $\mathcal{T}$ and the DRA $\mathcal{R}_\phi$; third, we synthesize an optimal accepting run $\omega_\mathcal{P}$ of the RPA $\mathcal{P}$ by Alg.3 proposed in [37]; last, if $\omega_\mathcal{P}$ is found, we concatenate one run of the projected sequence suffix with unique DRA states $\texttt{Unique}(\text{Suf}(\omega_\mathcal{P}|_\mathcal{Q}))$ to the end of the projected sequence prefix with unique DRA states $\texttt{Unique}(\text{Pre}(\omega_\mathcal{P}|_\mathcal{Q}))$, as defined in Alg. 3, to ensure that the suffix runs are correctly learned.

Remark 3.2: Note that the robot has an inaccurate controller which gives rise to the unknown transition probabilities P in the LMDP $\mathcal{M}$. However, in Alg. 1, the RPA $\mathcal{P}$ used by PMC does not take into account this uncertainty and hence its instructions are considered *suboptimal*.

B. Imitative Learning Phase

The PMC can be used to provide suboptimal instructions to the Q-learning agent, as discussed in Alg.2. Specifically, in the learning phase we train a Q-learning agent that can refine these suboptimal instructions into an optimal policy. The learning phase starts by setting the current LMDP state s as the initial LMDP state s^0, the current DRA state q as the initial DRA state q^0 and by forming an s-q tuple $x = (s, q)$ that is fed into the deep Q-network $Q(\cdot, \cdot; \theta)$ with neural network (NN) parameters θ and stored in the experience replay buffer $\mathcal{B}$ [Alg.1, line 13]. Then an action a is sampled from the exploration policy $\hat{\beta}$ [Alg.1, line 15] defined as follows:

$$\hat{\beta}(x) = \begin{cases} \text{random action} & \text{with probability } p_r, \\ \text{follow PMC (Alg.2)} & \text{with probability } p_p, \\ \text{argmax}_a Q(x, a; \theta) & \text{with probability } p_q, \end{cases} \tag{5}$$

where the PMC instructions are obtain by executing Alg. 2 which is discussed in detail later. After performing action a, the next LMDP state s' is obtained by following the dynamics in $\mathcal{M}$ [Alg.1, line 16]. The next DRA state q' and reward r is obtained by executing the `CheckRabin` function defined in [11], which basically obtains the next state q' and the reward r given an action a, and the next s-a tuple $x' = (s', q')$ is formed [Alg.1, lines 17-18]. Using x', an experience tuple (x, a, r, x') is formed and stored into the replay buffer and the parameters of the Q-network, θ, are updated by performing gradient descent steps with the data batch sampled from $\mathcal{B}$ [36]. Next, s, q, x are set to s', q', x', respectively [Alg. 1, line 21]. If x is a terminal state, the current episode is completed [Alg. 1, lines 22-25] and the next episode starts. Note that we terminate every episode when $q \in \mathcal{G}_i$, since the goal of the robot is to learn a policy so that DRA states $q \in \mathcal{G}_i$ are visited infinitely often, as this satisfies the accepting condition of the DRA. At last, the optimal control policy τ^* is constructed as [Alg. 1, line 27]:

$$\tau^*(x) = \text{argmax}_{a'} Q(x, a'; \theta), \tag{6}$$

where $x = (s, q)$, $s \in S$, $q \in Q$.

Algorithm 3 `Unique`(ω)

Input: A sequence ω
Begin:
1: $\hat{\omega} \leftarrow [\omega[0]]$
2: **for** i = 1 : `len`(ω) $-$ 1 **do**
3: **if** $\omega[i]$! = $\hat{\omega}[-1]$ **then**
4: $\hat{\omega}$.`append`($\omega[i]$)
5: **end if**
6: **end for**
7: **return** $\hat{\omega}$

In the rest of this section, we discuss how the PMC generates instructions for the exploration policy $\hat{\beta}$ using Alg.2. Specifically, first the robot uses the current history of observations and updates the labels $L_\mathcal{H}$ using (3). After that, the initial region Π_0 of the LTS $\mathcal{T}$ is set to be the region that corresponds to the current s [Alg.2, line 2]. The procedure that follows is split into two cases depending on whether the current DRA state q exists in the optimal Rabin sequence $\omega^*_\mathcal{P}|_\mathcal{Q}$. If q exists in $\omega^*_\mathcal{P}|_\mathcal{Q}$, then we identify the DRA state $\hat{q}$ that follows q in $\omega^*_\mathcal{P}|_\mathcal{Q}$ [Alg.2, lines 4-5] and form a *partial* LTL ϕ' by taking disjunctions of all the boolean expressions that transition q to $\hat{q}$ [Alg.2, lines 6-7], see Example 3.3. Next, we construct a DRA $\mathcal{R}_{\phi'}$ that corresponds to ϕ' and take the product between $\mathcal{R}_{\phi'}$ and $\mathcal{T}$ to form a *partial* RPA automaton $\mathcal{P}'$ that is used to synthesize *partial* plans that allow the current DRA state q to transition to $\hat{q}$ [Alg.2, lines 8-9]. If q does not exist in $\omega^*_\mathcal{P}|_\mathcal{Q}$, then a state $\hat{q}$ that is necessary for the PMC to give instructions cannot be identified by [Alg.2, line 5] and as a result it is trivial to execute the procedures defined in [Alg.2, lines 6-11]. In this case, the PMC suggest a random action [Alg.2, line 13].

Example 3.3: In this example we illustrate [Alg. 2, lines 6-7]. Consider the LTL $\phi = \diamondsuit(A \wedge \diamondsuit T)$ whose corresponding DRA $\mathcal{R}_\phi$ is shown in Figure 1. Suppose that the robot is currently in state q_1 and needs to transition to q_0. According to [Alg.2, line 6], $\mathcal{L} = \{\neg A \wedge T, A \wedge T\}$ because $q_0 \in \delta(q_1, \neg A \wedge T)$ and $q_0 \in \delta(q_1, A \wedge T)$. Then in [Alg.2, line 7], the *local* ϕ' is constructed as $\phi' = \diamondsuit(\bigvee_{l \in \mathcal{L}} l) = \diamondsuit((\neg A \wedge T) \vee (A \wedge T))$.

C. Correctness

In this section, we provide theoretical results showing probabilistic completeness of Alg.1. We start with a lemma

showing that the robot sensing model $L_{\mathcal{H}}$ defined in (3) can reconstruct ground-truth labels for all $\pi \in \Pi_{ap}$ eventually.

Lemma 3.4: Under Assumptions 2.3 and 2.4, for all labeled regions $\pi \in \Pi_{ap}$,

$$\mathrm{P}(Y_k^\pi \leq n_t^\pi/2) \leq e^{-n_t^\pi p_1 (1-\frac{1}{2p_1})^2/2}. \tag{7}$$

Proof: Using the Chernoff Bound [38], [39] we have that

$$\mathrm{P}(Y_k^\pi \leq (1-\delta)\mu) \leq e^{-\frac{1}{2}\mu\delta^2}. \tag{8}$$

Then, using fact that for a binomial distribution $\mu = n_t^\pi p_1$ and set $\delta = 1 - \frac{1}{2p_1}$, (8) becomes

$$\mathrm{P}(Y_k^\pi \leq \frac{1}{2p_1} n_t^\pi p_1) \leq e^{-n_t^\pi p_1 (1-\frac{1}{2p_1})^2/2}, \tag{9}$$

which completes the proof. ∎

Lemma 3.4 shows that the probability of reconstructing an *incorrect* label using the sensing model (3) converges to 0 as $n_k^\pi \to \infty$. Equivalently, $L_{\mathcal{H}}(\pi) \to L^*(\pi)$ as $n_k^\pi \to \infty$, meaning that the sensing model (3) approximates the ground truth model L^* when a sufficiently large history of observations is considered. The completeness of Alg.1 is shown by first showing the completeness of the PMC initialization phase and then of the imitative learning phase.

In what follows, we assume a long enough history of observations so that the sensing model (3) returns ground truth labels, by Lemma 3.4.

Proposition 3.5: Let Assumptions 2.3 and 2.4 hold, and assume a large enough history of observations in (2) for all $\pi \in \Pi$. Then the PMC initialization phase [Alg.1, lines 1-11] will eventually find the optimal Rabin sequence $\omega_{\mathcal{P}}^*|_{\mathcal{Q}}$ with probability 1.

Proof: According to [Alg.1, line 5], a sufficient condition for finding $\omega_{\mathcal{P}}^*|_{\mathcal{Q}}$ is that an optimal accepting run exists in $\mathcal{P}$ and this requires that the set of labels $\mathcal{L}_1 = \{l = L^*(\pi)|\ \forall\ \pi \in \Pi_{ap}\}$ all appear in the set $\mathcal{L}_2 = \{l = L_{\mathcal{H}}(\pi)|\ \forall\ \pi \in \Pi_{ap}\}$. This means that $L_{\mathcal{H}}$ returns all the ground truth labels in $2^{\mathcal{AP}}$ for all $\pi \in \Pi_{ap}$, but not necessarily at the correct locations. In what follows, we show that $\mathrm{P}(\mathcal{L}_1 = \mathcal{L}_2)$ is positive. Specifically, we have that

$$\mathrm{P}(\mathcal{L}_1 = \mathcal{L}_2) > \prod_{\pi \in \Pi_{ap}} \mathrm{P}(L_{\mathcal{H}}(\pi) = L^*(\pi)) \tag{10}$$

$$= \prod_{\pi \in \Pi_{ap}} \mathrm{P}(Y_k^\pi > n_t^\pi/2) > 0, \tag{11}$$

where the first inequality follows from the fact that the probability that all the labels in $2^{\mathcal{AP}}$ are detected at any locations is larger than the probability that they are detected at the correct locations, and the second inequality follows from Lemma 3.4 and the fact that n_k^π is large enough. Since $\mathrm{P}(\mathcal{L}_1 = \mathcal{L}_2) > 0$ and by Alg.1 all locations will be visited infinitely many times, we get that eventually $\mathcal{L}_1 = \mathcal{L}_2$ with probability 1. ∎

Proposition 3.6: After the optimal Rabin sequence $\omega_{\mathcal{P}}^*|_{\mathcal{Q}}$ has been detected by [Alg.1, lines 1-11], it remains constant during the entire learning process.

Proof: The optimal Rabin sequence $\omega_{\mathcal{P}}^*|_{\mathcal{Q}}$ only depends on the transition function of the DRA $\mathcal{R}_\phi$ which remains unchanged during the entire learning process, as a result $\omega_{\mathcal{P}}^*|_{\mathcal{Q}}$ is constant. ∎

Note that Propositions 3.5 and 3.6 show completeness of the PMC initialization phase [Alg.1, lines 1-11]. The next proposition shows the completeness of the imitative learning phase [Alg.1, lines 12-28].

Proposition 3.7: Let Assumptions 2.3 and 2.4 hold, and assume a large enough history of observations in (2) for all $\pi \in \Pi$. The imitative learning phase [Alg.1, lines 12-28] is probabilistically complete, meaning that if a policy exists that maximizes the probability of satisfying the LTL formula ϕ, then [Alg.1, lines 12-28] will find it with probability 1.

Proof: The difference between standard Q-learning and Alg.1 is that here we modify the exploration policy with PMC instructions, cf. [Alg.1, line 15], returned by Alg.2 which utilizes the optimal Rabin sequence calculated from [Alg.1, lines 1-11]. To show this result we need to show that the proposed modification to the exploration policy does not affect the convergence of standard Q-learning. For this, we first need to show that the sequence $\omega_{\mathcal{P}}^*|_{\mathcal{Q}}$ returned by [Alg.1, lines 1-11] is optimal and does not change during training. This sequence is an input to the Q-learning algorithm that follows [Alg.1, lines 12-28] and by remaining constant we ensure that it does not affect convergence of the algorithm. Second, we need to show that the exploration policy $\hat{\beta}$ returned by Alg.2 ensures that the Q-values $Q(s,a)$ for all state-action pairs are updated infinitely many times as the number of episodes goes to infinity, as required in Q-learning for sufficient exploration and as a result the learned policy can be considered as *optimal* [40], [41].

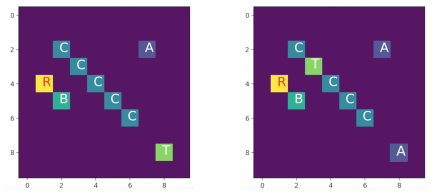
To show that $\omega_{\mathcal{P}}^*|_{\mathcal{Q}}$ returned by [Alg.1, lines 2-12] is constant during training, note that by By Proposition 3.5 the sequence $\omega_{\mathcal{P}}^*|_{\mathcal{Q}}$ will be found with probability 1, while by Proposition 3.6 it will remain constant across training steps. As a result even though the labels reconstructed using sensing information, and the optimal product sequence $\omega_{\mathcal{P}}$ can vary at different training steps, $\omega_{\mathcal{P}}^*|_{\mathcal{Q}}$ always remains optimal and constant. On the other hand, to show that the exploration policy $\hat{\beta}$ returned by Alg.2 still allows for sufficient exploration, note that according to (5), the robot performs random actions with probability $p_r > 0$ and hence $\hat{\beta}$ is considered ϵ-*soft*, see [40]. Therefore, $\hat{\beta}$ satisfies the assumptions required from an exploration policy in Q-learning and the resulting policy τ [Alg.1 line 28] is optimal. ∎

Combining Propositions 3.5, 3.6 and 3.7, we obtain that Alg. 1 is probabilistically complete. Specifically, we have the following result.

Theorem 3.8: Let Assumptions 2.3 and 2.4 hold, and assume a large enough history of observations in (2) for all $\pi \in \Pi$. Then, Alg.1 is probabilistically complete.

IV. Numerical Experiments

In this section, we illustrate the DIQL algorithm on a planning task for a single robot. We start with examining



(a) Ground-Truth Environment.

(b) Sample environment with labels reconstructed by the sensing model (3).

Fig. 2. Simulation Environments.

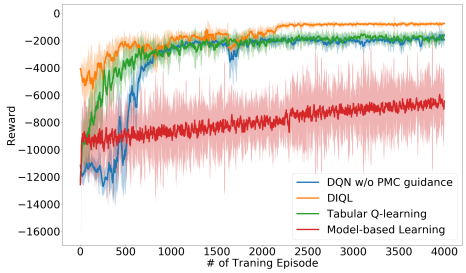


Fig. 3. Comparison of the DIQL approach with the pure DQN approach proposed by [11], the model-based learning approach proposed by [24] and the tabular Q-learning approach.

the performance of Alg.1 for a 10×10 discrete grid world, where each discrete point is associated with a state of an LMDP that models the robot; see Figure 2(a). The ground truth set of atomic propositions $\mathcal{AP}$ is defined as $\mathcal{AP} = \{A, B, C, T, \varnothing\}$, where the atomic propositions A, B, T are observed in the respective regions shown in Figure 2(a), while $\varnothing$ denotes that nothing is observed.

The robot can take 5 actions: *UP, RIGHT, DOWN, LEFT, NONE*, where action *NONE* means that the robot chooses to remain idle. The robot has a noisy controller, which can only execute the desired action with probability 0.8 and random action otherwise. The initial location of the robot is at $(4, 1)$. According to Assumption 2.4, the instantaneous sensing model $L_{\mathcal{I}}$ only gathers sensing information from the regions $\pi \in \Pi$ if and only if $L^*(\pi) \neq \emptyset$, i.e., states $(2, 7), (5, 2), (8, 8), (2, 2), (3, 3), (4, 4), (5, 5), (6, 6)$, and we assume that the probability of $L_{\mathcal{I}}$ collecting ground truth labels is 0.8, i.e., $p_1 = 0.8$ in (1). An example of the environment reconstructed by the robot sening model (3) is shown in Figure 2(b). We consider the following LTL task that the robot needs to satisfy

$$\phi = \Diamond(A \wedge \Diamond(B \wedge \Diamond T)) \wedge \Box\Diamond A \wedge \Box\Diamond B \wedge \Box\neg C. \quad (12)$$

In words, the LTL formula in (12) requires the robot to first visit A, B, and T in this order and visit infinitely often A and B while always avoiding C. The DRA that corresponds to ϕ has 16 states and 241 edges. Figure 3 shows a comparison between DIQL, the pure Deep Q-Network (DQN) approach proposed in [11], the model based method developed in [24] and the tabular Q-learning method [40], where the specific neural networks architectures and reward functions are defined as the same as in [11]. It can be seen that DIQL converges faster and synthesizes a policy that achieves

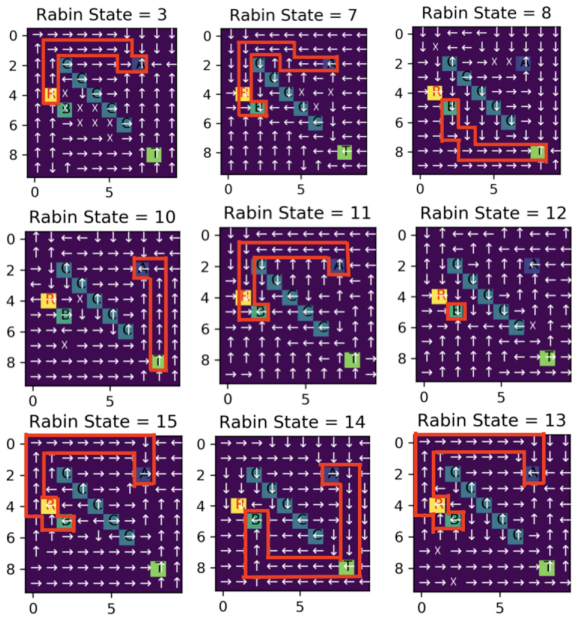


Fig. 4. Policy synthesized by DIQL.

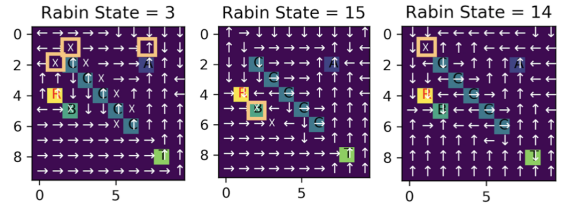


Fig. 5. Failures from policy synthesized by pure DQN [11].

a higher cumulative reward than the others. The resulting policy of DIQL is shown in Figure 4. Specifically, each subfigure corresponds to the optimal policy of Rabin states $q \in Q$, where in this case $Q = \{3, 7, 8, 10, 11, 12, 15, 14, 13\}$ and the optimal trajectory induced from the policy is highlighted with red boundaries. Note that in the Rabin states 14 and 13, the robot learns to maintain a safe distance from C instead of travelling along the shortest path as suggested by the PMC. The reason is that these two Rabin states are associated with the suffix run that is executed indefinitely, i.e., to visit A and B infinitely often, and Alg.1 determines that the optimal way to maximize the probability of satisfying the LTL is to remain away from C because of the noisy controller. Figure 5 shows that the policy learned by the pure DQN method [11] contains several false actions that prevent the robot from reaching its goal. Thus, the resulting trajectory does not satisfy the LTL specification. These actions are highlighted with brown boundaries.

V. Conclusion

In this paper, we proposed a DIQL method to synthesize control policies for mobile robots that need to satisfy Linear Temporal Logic (LTL) specifications using noisy semantic observations of their surroundings. We used suboptimal instructions from PMCs to guide the Q-learning algorithm and showed that our method outperforms existing methods in the literature in terms of convergence speed and quality of the resulting control policy.

REFERENCES

[1] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[2] M. Guo and D. V. Dimarogonas, "Multi-agent plan reconfiguration under local LTL specifications," *The International Journal of Robotics Research*, vol. 34, no. 2, pp. 218–235, 2015.

[3] Y. Kantaros and M. M. Zavlanos, "Sampling-based optimal control synthesis for multi-robot systems under global temporal tasks," *IEEE Transactions on Automatic Control*, vol. PP, no. 99, 7 2018.

[4] ——, "Sampling-based control synthesis for multi-robot systems under global temporal specifications," in *2017 ACM/IEEE 8th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 2017, pp. 3–14.

[5] ——, "Distributed optimal control synthesis for multi-robot systems under global temporal tasks," in *Proceedings of the 9th ACM/IEEE International Conference on Cyber-Physical Systems*. IEEE Press, 2018, pp. 162–173.

[6] Y. Kantaros and M. Zavlanos, "STyLuS*: A temporal logic optimal control synthesis algorithm for large-scale multi-robot systems," *arXiv preprint arXiv:1809.08345*, 06 2019.

[7] A. Ulusoy, T. Wongpiromsarn, and C. Belta, "Incremental controller synthesis in probabilistic environments with temporal logic constraints," *The International Journal of Robotics Research*, vol. 33, no. 8, pp. 1130–1144, 2014.

[8] X. Ding, S. L. Smith, C. Belta, and D. Rus, "Optimal control of markov decision processes with linear temporal logic constraints," *IEEE Transactions on Automatic Control*, vol. 59, no. 5, pp. 1244–1257, 2014.

[9] ——, "Mdp optimal control under temporal logic constraints," in *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*. IEEE, 2011, pp. 532–538.

[10] M. Guo and M. M. Zavlanos, "Probabilistic motion planning under temporal tasks and soft constraints," *IEEE Transactions on Automatic Control*, vol. PP, no. 99, p. 11, 2018. [Online]. Available: DOI: 10.1109/TAC.2018.2799561

[11] Q. Gao, D. Hajinezhad, Y. Zhang, Y. Kantaros, and M. Zavlanos, "Reduced variance deep reinforcement learning with temporal logic specifications," in *10th ACM/IEEE International Conference on Cyber-Physical Systems (with CPS-IoT Week 2019)*. ACM, 2019.

[12] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005.

[13] K. Chatterjee, M. Chmelík, R. Gupta, and A. Kanodia, "Qualitative analysis of pomdps with temporal logic specifications for robotics applications," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 325–330.

[14] M. Svoreňová, M. Chmelík, K. Leahy, H. F. Eniser, K. Chatterjee, I. Černá, and C. Belta, "Temporal logic motion planning using pomdps with parity objectives: case study paper," in *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*. ACM, 2015, pp. 233–238.

[15] J. Van Den Berg, D. Wilkie, S. J. Guy, M. Niethammer, and D. Manocha, "Lqg-obstacles: Feedback control with collision avoidance for mobile robots with motion and sensing uncertainty," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 346–353.

[16] J. Van Den Berg, S. Patil, and R. Alterovitz, "Motion planning under uncertainty using differential dynamic programming in belief space," in *Robotics Research*. Springer, 2017, pp. 473–490.

[17] S. Patil, J. Van Den Berg, and R. Alterovitz, "Estimating probability of collision for safe motion planning under gaussian motion and sensing uncertainty," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 3238–3244.

[18] J. Van Den Berg, S. Patil, and R. Alterovitz, "Motion planning under uncertainty using iterative local optimization in belief space," *The International Journal of Robotics Research*, vol. 31, no. 11, pp. 1263–1278, 2012.

[19] B. Burns and O. Brock, "Sampling-based motion planning with sensing uncertainty," in *Robotics and Automation, 2007 IEEE International Conference on*. IEEE, 2007, pp. 3313–3318.

[20] X. Zhang, B. Wu, and H. Lin, "Supervisor synthesis of pomdp based on automata learning," *arXiv preprint arXiv:1703.08262*, 2017.

[21] J. Wang, X. Ding, M. Lahijanian, I. C. Paschalidis, and C. Belta, "Temporal logic motion control using actor–critic methods," *The International Journal of Robotics Research*, vol. 34, no. 10, pp. 1329–1344, 2015.

[22] J. Fu and U. Topcu, "Probably approximately correct mdp learning and control with temporal logic constraints," in *Proceedings of Robotics: Science and Systems*, Berkeley, USA, July 2014.

[23] M. Wen, R. Ehlers, and U. Topcu, "Correct-by-synthesis reinforcement learning with temporal logic constraints," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 4983–4990.

[24] D. Sadigh, E. S. Kim, S. Coogan, S. S. Sastry, and S. A. Seshia, "A learning based approach to control synthesis of markov decision processes for linear temporal logic specifications," in *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*. IEEE, 2014, pp. 1091–1096.

[25] D. Aksaray, A. Jones, Z. Kong, M. Schwager, and C. Belta, "Q-learning for robust satisfaction of signal temporal logic specifications," in *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE, 2016, pp. 6565–6570.

[26] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband, *et al.*, "Deep q-learning from demonstrations," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[27] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 627–635.

[28] K. Muelling, "Modeling and learning of complex motor tasks: A case study with robot table tennis," Ph.D. dissertation, Technische Universität, 2013.

[29] M. Wen, I. Papusha, and U. Topcu, "Learning from demonstrations with high-level side information," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, 2017.

[30] B. Araki, K. Vodrahalli, T. Leech, C.-I. Vasile, M. D. Donahue, and D. L. Rus, "Learning to plan with logical automata," 2019.

[31] A. Bozkurt, Y. Wang, M. Zavlanos, and M. Pajic, "Control synthesis from linear temporal logic specifications using model-free reinforcement learning," in *2020 International Conference on Robotics and Automation (ICRA)*, 2020.

[32] S. Y. Lee, S. Choi, and S.-Y. Chung, "Sample-efficient deep reinforcement learning via episodic backward update," *arXiv preprint arXiv:1805.12375*, 2018.

[33] B. Spector and S. Belongie, "Sample-efficient reinforcement learning through transfer and architectural priors," *arXiv preprint arXiv:1801.02268*, 2018.

[34] C. Baier, J.-P. Katoen, and K. G. Larsen, *Principles of model checking*. MIT press, 2008.

[35] C. Belta, B. Yordanov, and E. A. Gol, *Formal Methods for Discrete-Time Dynamical Systems*. Springer, 2017, vol. 89.

[36] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[37] M. Guo, "Cooperative motion and task planning under temporal tasks," Ph.D. dissertation, KTH Royal Institute of Technology, 2014.

[38] H. Chernoff, "A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations," 1952.

[39] M. Mitzenmacher and E. Upfal, *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. New York, NY, USA: Cambridge University Press, 2005.

[40] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[41] C. J. C. H. Watkins, "Learning from delayed rewards," Ph.D. dissertation, King's College, Cambridge, 1989.