Abstract—We study synthesis of control strategies from linear temporal logic (LTL) objectives in unknown environments. We model this problem as a turn-based zero-sum stochastic game between the controller and the environment, where the transition probabilities and the model topology are fully unknown. The winning condition for the controller in this game is the satisfaction of the given LTL specification, which can be captured by the acceptance condition of a deterministic Rabin automaton (DRA) directly derived from the LTL specification. We introduce a model-free reinforcement learning (RL) methodology to find a strategy that maximizes the probability of satisfying a given LTL specification when the Rabin condition of the derived DRA has a single accepting pair. We then generalize this approach to any LTL formulas, for which the Rabin accepting condition may have more than one pairs, providing a lower bound on the satisfaction probability. Finally, we show applicability of our RL method on two planning case studies.

I. INTRODUCTION

Linear temporal logic (LTL) offers a formal language to capture high level requirements of robot planning and control tasks, such as jobs and motion sequencing, obstacle avoidance, and surveillance. Hence, there is a growing interest in using LTL in robotics [1]–[16]. To synthesize a controller strategy from a given LTL task, most of these methods require a model of the environment a-priori, limiting their use in scenarios where the environment is unknown. For such environments, reinforcement learning (RL) is commonly utilized to search for a strategy that performs the task [17].

Recently, control synthesis using RL from LTL specifications is considered for Markov decision processes (MDPs). Model-based techniques (e.g., [18], [19]) are usually based on detection of the MDP end components; yet, with these methods, it is necessary to learn and store the transition probabilities of the MDP, which may result in very large memory requirements. Model-free RL mitigates this problem; however, the given task needs to be represented by a reward function such that a strategy maximizing the discounted cumulative reward satisfies the given LTL task specifications.

Accordingly, the problem of reward shaping from LTL specifications in MDPs has been considered in [20]–[26]. The recent limit-deterministic Büchi automata (LDBAs) based approaches (e.g., [24], [26]) shape rewards in a way that the learned control strategy maximizes the probability of satisfying any given LTL specification. These approaches generally translate the given LTL specification into an LDBA, which is then composed with the initial MDP, and design a reward function based on the acceptance condition of the automaton. State-space augmentation using the LDBA solves the memory requirements of the task, while the Büchi acceptance condition (i.e., repeated reachability) enables the use of simple reward functions.

However, such LDBA-based rewarding approaches are not well-suited for stochastic games [27], since LDBAs and many other nondeterministic automata, in general, cannot be used in solving games [28]. Hence, there are only few studies on learning-based synthesis from temporal objectives for stochastic games. One approach, [29] proposed a model-based probably approximately correct (PAC) learning algorithm for stochastic games with LTL and discounted cumulative reward objectives. Yet, the method requires that (i) the transition graph (i.e., topology) is known a-priori, (ii) the LTL objective must belong to a limited subset of LTL formulas that can be translated into a deterministic Büchi automaton, and (iii) there exists a strategy that almost surely satisfies the LTL objective. This allows pre-computation of the winning regions before learning. Also, [30] introduced a model-based learning method with PAC guarantees for reachability objectives only (i.e., very limited fragment of LTL formulas). The method uses on-the-fly detection of (simple) end components of the games, and careful construction of the confidence intervals on the transition probabilities. However, as model-based methods, both are inefficient in terms of space requirements when the number of possible successors of actions is not small.

Consequently, in this work we introduce a model-free RL approach to synthesize controllers for stochastic games, such that the obtained control policies maximize the (worst-case) probabilities of satisfying the given LTL task objectives. We start by translating the LTL objective into a Deterministic Büchi Automaton (DRA) and introduce a reward and discount function based on the Rabin acceptance condition. We first consider DRAs with a single accepting pair and prove that any model-free RL algorithm using these functions converges to a desired strategy for a sufficiently large discount factor. We then generalize our method to any LTL specification, for which the DRA may have an arbitrary number of accepting pairs; for such specifications, we establish a lower bound on the satisfaction probability. Lastly, we show the applicability of our RL approach on two robot planning case studies.

II. PRELIMINARIES AND PROBLEM STATEMENT

A. Stochastic (Turn-Based Zero-Sum) Two-Player Games

We use turn-based stochastic games to model the interaction between the controller (i.e., Player 1) and unpredictable environment (i.e., Player 2), where actions have probabilistic
outcomes. The controller can only choose actions in certain states; the rest of the states are in control of the environment.

**Definition 1 (Stochastic Games):** A (labeled turn-based) two-player stochastic game is a tuple \( G = (S, (S_\mu, S_\nu), A, P, S_0, AP, L) \) where \( S \) is a finite set of states; \( S_\mu \subseteq S \) is the set of states where the controller chooses actions; \( S_\nu = S \setminus S_\mu \) is the set of states where the environment chooses actions; \( S_0 \) is the initial state; \( A \) is a finite set of actions and \( A(s) \) denotes the set of actions that can be taken in state \( s \in S \); \( P : S \times A \times S \rightarrow [0, 1] \) is a transition probability function such that for all \( s \in S \), \( \sum_{a \in A(s)} P(s, a, s') = 1 \) if \( a \in A(s) \), and 0 otherwise; \( S_0 \) is an initial state; AP is a finite set of atomic propositions; and \( L : S \rightarrow 2^{AP} \) is a labeling function.

A path is an infinite sequence of game states \( s_0, s_1, \ldots \) such that for all \( t \geq 0 \), there exists an action \( a \in A(s_t) \) where \( P(s_t, a, s_{t+1}) > 0 \). We use \( \sigma[i] \), \( \sigma[d] \) and \( \sigma[t] \) to denote \( s_t \), the prefix \( s_0, \ldots, s_t \), and the suffix \( s_t s_{t+1} \ldots \) of the path, respectively. Strategies capture the players’ behaviors, mapping the visited states to the actions available in the current state.

**Definition 2 (Strategies):** For a game \( G \), let \( S^+_{\mu} (S^+_{\nu}) \) denote the set of all finite prefixes \( \sigma^\mu_i (\sigma^\nu_i) \) ending with a state \( s^\mu_i (s^\nu_i) \in S_{\mu} (S_{\nu}) \) of paths in the game. Then, a (pure) control strategy \( \mu \) is a function \( \mu : S^+_{\mu} \rightarrow A \) such that \( \mu(\sigma^\mu_i) \in A(s^\mu_i) \) for all \( \sigma^\mu_i \in S^+_{\mu} \); a (pure) environment strategy \( \nu \) is a function \( \nu : S^+_{\nu} \rightarrow A \) such that \( \nu(\sigma^\nu_i) \in A(s^\nu_i) \) for all \( \sigma^\nu_i \in S^+_{\nu} \); a strategy \( \pi \) is memoryless, if it only depends on the current state, i.e., for any \( \sigma^\mu_i \) and \( \sigma^\nu_i \), \( \pi(\sigma^\mu_i) = \pi(\sigma^\nu_i) \) if \( s^\mu_i = s^\nu_i \), and thus can be defined as \( \pi : S \rightarrow A \).

The induced Markov chain (MC) of game \( G \) under a strategy pair \( (\mu, \nu) \) is tuple \( G_{\mu, \nu} = (S, P_{\mu, \nu}, S_0, AP, L) \), where

\[
P_{\mu, \nu}(s, s') = \begin{cases} P(s, \mu(s), s') & \text{if } s \in S_{\mu} \\ P(s, \nu(s), s') & \text{if } s \in S_{\nu} \end{cases}
\]

We denote \( G^*_{\mu, \nu} \) the MC resulting from changing the initial state from \( S_0 \) to \( s \in S \) in \( G_{\mu, \nu} \), and use \( \sigma \sim G^*_{\mu, \nu} \) to denote a random path sampled from \( G^*_{\mu, \nu} \). Finally, a bottom strongly connected component (BSCC) of the (induced) MC \( G^*_{\mu, \nu} \) is a strongly connected component with no outgoing transitions; we use \( B(G^*_{\mu, \nu}) \) to denote the set of all BSCCs of \( G^*_{\mu, \nu} \).

**B. LTL and Deterministic Rabin Automata**

The desired behavior of a labeled stochastic game \( G \) are captured by LTL specifications, imposing requirements on the label sequences from infinite paths of the game [31]. In addition to the standard Boolean operators, LTL formulas can include two temporal operators, next (\( \delta \)) and until (\( U \)), and any recursive combinations of the operators captured by the syntax \( \varphi := \text{true} | a \land \varphi_1 \land \varphi_2 | \varphi_1 
\varphi_2 | \varphi_1 U \varphi_2 | a \in AP \).

Satisfaction of an LTL formula \( \varphi \) for a path \( \sigma \) of the game \( G \), denoted by \( \sigma \models \varphi \), is defined as follows: \( \sigma \) satisfies an atomic proposition \( a \), if \( a \in L(\sigma[0]) \); \( \sigma \) satisfies \( \varphi \) if \( \sigma[1:] \) satisfies \( \varphi \); and finally, \( \sigma \models \varphi \Uparrow \varphi_2 \), if \( \exists i. \sigma[i] \models \varphi_2 \) and \( \forall j < i. \sigma[j] \models \varphi_1 \). Other temporal operators are derived as:

- (eventually) \( \diamond \varphi := \text{true} \land U \varphi \); and (always) \( \Box \varphi := \neg (\neg \varphi) \).

Any LTL formula can be transformed into a DRA that accepts the language of all paths satisfying the formula [31].

**Definition 3 (Deterministic Rabin Automata):** A DRA is a tuple \( A = (Q, \Sigma, \delta, q_0, \text{Acc}) \) where \( Q \) is a finite set of states; \( \Sigma \) is a finite alphabet; \( \delta : Q \times \Sigma \rightarrow Q \) is the transition function; \( q_0 \in Q \) is an initial state; and \( \text{Acc} \) is a set of \( k \) accepting pairs \( \{(C_i, B_i)\}_{i=1}^{k} \) such that \( C_i, B_i \subseteq Q \).

An infinite path \( \sigma \) induces an execution; the DRA moves one state to another as it consumes the labels of the states on the path. The DRA accepts \( \sigma \) if the induced execution satisfies the **Rabin condition**: there exists a pair \( (C_i, B_i) \) of such that states in \( C_i \) are visited infinitely many times and at least one state in \( B_i \) is visited infinitely often — i.e., \( \exists i : \inf(\sigma) \cap C_i = \varnothing \land \inf(\sigma) \cap B_i \neq \varnothing \), where \( \inf(\sigma) \) denotes the set of states visited infinitely many times during the execution induced by \( \sigma \). The **Rabin index** of an LTL formula is the minimal number of accepting pairs a DRA recognizing the formula can have. Without loss of generality, we assume that the number of accepting pairs \( k \), equals the Rabin index.

**Example 1:** Fig. 1 shows a DRA of the formula \( \varphi = \Box \diamond b \lor \Diamond d \), with Rabin acceptance sets \( B_1 = \{q_1\}, C_2 = \{q_0\} \) and \( B_2 = \{q_2\} \) (i.e., \( \text{Acc} = \{(B_1, C_2), (B_2, C_2)\} \) is the acceptance condition). Any path \( \sigma \) containing infinitely many states labeled with \( b \) induces an execution that visits \( q_1 \) infinitely many times; thereby satisfying the Rabin condition. Other paths satisfying the Rabin condition visit \( q_2 \) infinitely many times but \( q_0 \) only finitely many times, i.e., the paths do not contain a state without the label \( d \) after some point.

**C. Reinforcement Learning for Stochastic Games**

Let \( R : S \rightarrow \mathbb{R} \) be a reward function and \( \gamma \in (0, 1) \) the discount factor for a given two-player zero-sum stochastic game \( G \). The value of a state \( s \) under a strategy pair \( (\mu, \nu) \)

\[
v_{\mu, \nu}(s) = \mathbb{E}_{\sigma \sim \nu \mu} \left[ \sum_{i=0}^{\infty} \gamma^i R(\sigma[t+i]) | \sigma[t] = s \right],
\]

for any fixed \( t \in \mathbb{N} \), such that \( \Pr_{\sigma \sim \nu \mu}[\sigma[t]=s] > 0 \). In the rest of the paper, we simplify our notation, omit the subscript \( \sigma \sim \nu \mu \) from the expectation and use \( \mathbb{E} \) rather than \( \mathbb{E}_{\sigma \sim \nu \mu} \).

The RL objective is to find an optimal control strategy \( \mu^* \) that maximizes the values of every state under the worst environment strategy. A pure and memoryless optimal strategy always exists in two-player turn-based zero-sum stochastic games [27], [32]. The optimal values in these games satisfy \( v_\mu(s) = \max_{\nu} \min_{\nu} v_{\nu \mu}(s) \), where \( \mu \) and \( \nu \) are pure and memoryless control and environment strategies [33]. Also, the optimal values \( v_\mu(s) \) satisfy the Bellman equations

\[
v_\mu(s) = R(s) + \gamma \left[ \max_{\nu} \sum_{s' \in S} P(s, a, s') v_{\nu}(s') \text{ if } s \in S_{\mu}, \right.
\]

\[
\left. \min_{\nu} \sum_{s' \in S} P(s, a, s') v_{\nu}(s') \text{ if } s \in S_{\nu}. \right.
\]

Model-free RL methods aim to learn the optimal values of the stochastic game, when neither the transition probabilities nor the game topology are known, without explicitly constructing a transition model of the game. A popular example is the minimax-Q method that generalizes the standard off-
policy Q-learning algorithm to stochastic games. The mini-
max-Q method can learn the optimal values under any (likely
non-optimal) strategies used during learning as long as all
actions in each state are chosen infinitely often [32], [34].

D. Problem Formulation

We assume that the considered game $G$ is fully observable
for both players; i.e., both are aware of the current game
state. The considered control synthesis problem is to find a
strategy for the controller that maximizes the probability that a
produced path satisfies the specification in the worst case.

To simplify our notation, we use $Pr^G_{\mu,\nu}(s \models \varphi)$
to denote the probability of the paths that start from state $s$,
and satisfy the formula $\varphi$ under the strategy pair $(\mu, \nu)$ — i.e.,

$$Pr^G_{\mu,\nu}(s \models \varphi) := Pr_{\sigma \sim G^\mu_{\varphi,\nu}}(\sigma \models \varphi);$$

we write $Pr^G_{\nu}(\varphi) := Pr^G_{\mu,\nu}(s_0 \models \varphi)$ and use $Pr_\varphi$ to
denote the maximin probability $\max_{\mu} \min_{\nu} Pr^G_{\mu,\nu}$. We can
now formally define the considered problem as follows.

**Problem 1:** Given a labeled turn-based stochastic game $G$,
where the transition probabilities are fully known, and an
LTL specification $\varphi$, design a model-free RL algorithm that
finds a pure finite-memory controller strategy $\mu$, such that

$$Pr^{G\mu}_{\mu,\nu}(G \models \varphi) \geq Pr_\varphi(G \models \varphi)$$

for any environment strategy $\nu$.

III. LEARNING FOR STOCHASTIC RABIN GAMES

In this section, we introduce our model-free RL approach
to solve Problem 1. First, we describe the product game
construction, a key step in reducing the problem of satisfying
an LTL specification into the problem of satisfying a Rabin
condition. We then consider the case where the DRA derived
from the LTL specification $\varphi$ has a single Rabin pair, and
introduce our rewarding and discounting mechanisms based
on it. We show that maximization of the discounted reward
maximizes the minimal probability of satisfying the single
pair Rabin condition, and thus the initial LTL objective.

Finally, we provide a generalization to Rabin conditions with
multiple pairs $(k>1)$ with a lower bound on the satisfaction
probabilities; thereby allowing the use of our method for all
possible LTL specifications.

A. Product Game Construction

By forming an augmented state space, Problem 1 can be
reduced into finding a memoryless control strategy. Specifi-
cally, we compose the states of the game $G$ with the states
of the DRA $A$ derived from the LTL specification $\varphi$. Then, the
goal in this space is to satisfy the Rabin acceptance condition,
for which memoryless control strategies suffice [35].

**Definition 4 (Product Game):** A product game $G^\times = (S^\times, (S^\times, S^\times), A^\times, P^\times, s^\times_0, Acc^\times)$ of a labeled turn-based
stochastic game $G = (S, (S_\mu, S_\nu), A, P, s_0, AP, L)$ and a DRA
$A = (Q, 2^AP, \delta, q_0, Acc)$ is defined as follows: $S^\times = S \times Q$ is
the set of augmented states, the initial state $s^\times_0 = (s_0, q_0)$,
$S^\times = S \times Q$ and $S^\times = S \times Q$ are the sets of augmented
controller and environment states, respectively; $A^\times = A$ is the
set of actions; $P^\times : S^\times \times A^\times \times S^\times \rightarrow [0,1]$ is the transition
function such that

$$P^\times((s, q), a, (s', q')) = \begin{cases} P(s, a, s') & \text{if } q' = \delta(q, L(s)) \\ 0 & \text{otherwise}; \end{cases}$$

and $Acc^\times$ is a set of $k$ accepting pairs $\{(C^\times, B^\times)^{k}\}$ where $C^\times_i = C_i \times Q$ and $B^\times_i = B_i \times Q$.

Similarly to DRAs, a path $\sigma^\times$ of the product game $G^\times$
satisfies the Rabin condition if there exists $i$, such that
$inf(\sigma^\times) \cap C^\times_i = \emptyset$ and $inf(\sigma^\times) \cap B^\times_i \neq \emptyset$. Finally, we refer to a product game with $k$ accepting pairs as a Rabin($k$) game.

There is a one-to-one correspondence between the paths
in the product and original game. Similarly, a strategy for
the product game induces a strategy in the original game and vice versa. However, the corresponding strategies in the original
game require additional memory described by the DRA; i.e.,
the strategy in the original game may not be memoryless. Yet,
the probability of satisfying the Rabin condition under any
strategy pair in the product game is equal to the probability
of satisfying the LTL formula in the original game under
the corresponding strategy pair. Hence, in the rest of the
section we focus on the product games, i.e., stochastic Rabin
games; to simplify our notation, we omit the superscript $x$
and use $G = (S, (S_\mu, S_\nu), A, P, s_0, Acc)$ and $s \in S$ instead of
$G^\times = (S^\times, (S^\times, S^\times), A^\times, P^\times s^\times_0, Acc^\times)$ and $(s, q) \in S^\times$.

B. Rabin(1) Condition to Discounted Rewards

We start with the case where the LTL formula $\varphi$ has one
accepting pair in the Rabin acceptance condition. In stochastic
Rabin(1) games, where $Acc =\{(C, B), (B, C)\}$, the controller
objective is to repeatedly visit some states in $B$ and visit
the states in $C$ only finitely many times. The environment’s
goal is to prevent this from happening, which can be also
expressed as a Rabin condition $Acc^\times =\{(\emptyset, C), (B, S)\}$ with
two accepting pairs. Thus, pure and memoryless strategies
suffice for both players on the considered product game [35].

To solve Problem 1 for stochastic Rabin(1) games, our key
idea is to assign small rewards to the states in $B$ to encourage
visiting $B$ states as often as possible; but discount more
compared to the other states to eliminate the importance of
the frequency of visits. In addition, we discount even more in
the states in $C$ without giving any rewards, which diminishes
the worth of the rewards to be obtained by visiting the states
in $B$. The following theorem summarizes our key results.

**Theorem 1:** Consider a given turn-based stochastic
Rabin(1) product game $G$ and the return of any path $\sigma$

$$G_t(\sigma) := \sum_{i=0}^{\infty} R_B(\sigma[t+i]) \cdot \prod_{j=0}^{i-1} \Gamma_{B,C}(\sigma[t+j]),$$

where $\prod_{j=0}^{i-1} := 1$, $R_B : S \rightarrow [0,1]$ and $\Gamma_{B,C} : S \rightarrow (0,1)$
are the reward and the terminal functions defined as

$$R_B(s) := \begin{cases} 1-\gamma_B, & \text{if } s \in B \\ 0, & \text{if } s \notin B, \ & \Gamma_{B,C}(s) := \begin{cases} \gamma_B, & \text{if } s \in B \\ \gamma_C, & \text{if } s \in C \\ \gamma, & \text{otherwise}. \end{cases} \end{cases}$$

Here, $\gamma_B$ and $\gamma_C$ are functions of $\gamma$ such that $0<\gamma_C(\gamma)<\gamma_B(\gamma)<1$, and $lim_{\gamma \rightarrow 1} \gamma_B = lim_{\gamma \rightarrow 1} \gamma_C = 1$, as well as

$$lim_{\gamma \rightarrow 1} 1 - \gamma_B(\gamma) = lim_{\gamma \rightarrow 1} 1 - \gamma_C(\gamma) = 0.$$ 

Then, the value of the game $V^\times_{\mu,\nu}$, i.e., the expected return $E[G_t(\sigma)]$ for the strategy pair $(\mu, \nu)$ and the discount factor
$\gamma$ satisfies that for all states $s \in S$ it holds that

$$lim_{\gamma \rightarrow 1} V^\times_{\mu,\nu}(s) = Pr^G_{\mu,\nu}(s \models \varphi_{B,C});$$

here, $\varphi_{B,C} := \Box B \wedge \neg \Box C$ is the Rabin condition of the
DRA derived from the LTL objective $\varphi$. 

Before proving Theorem 1, we use Lemma 1 to establish bounds on the state values. We show that if we replace a state on a path with a state in $B$, we obtain a larger or equal return; if we replace it with a state in $S \setminus B$, we obtain a smaller or equal return; and the return is always between 0 and 1. 

**Lemma 1:** For any path $\sigma$ and a fixed $t \geq 0$, in a stochastic game with the path return defined as in (4), it holds that
\[
\gamma CG_{t+1}(\sigma) \leq \gamma G_{t+1}(\sigma) \leq G_t(\sigma) \leq 1 - \gamma B + \gamma B G_{t+1}(\sigma),
\]
respectively (formally defined in Lemma 2). This allows us to focus on the reachability objective $\varphi_U := \emptyset B$ instead of $\varphi_{U,C \setminus C}$ defined in Theorem 1.

We now show that the expected values of the returns (4) (i.e., the state values) reflect the Rabin acceptance condition.

**Lemma 2:** For any stochastic Rabin game $G$ with $\text{Acc} = \{(C, B)\}$ under a strategy pair $(\mu, \nu)$, it holds that:
\[
\lim_{t \to \infty} v^*_{\mu,\nu}(s) = 0 \text{ if } s \in U_{BCC}, \\
\lim_{t \to \infty} v^*_{\mu,\nu}(s) = 1 \text{ if } s \in U_B, \\
\lim_{t \to \infty} v^*_{\mu,\nu}(s) = 0 \text{ if } s \in U_C,
\]
where the sets $U_{BCC}, U_B$ and $U_C$ are defined as:
\[
U_{BCC} := \{ s \in E | \exists t \in \mathbb{N}, s \in B(G_{\mu,\nu}), s \in T \cap C = \emptyset \}, \\
U_B := \{ s \in B | \forall t \in \mathbb{N}, s \in B(G_{\mu,\nu}), s \in T \cap C = \emptyset \}, \\
U_C := \{ s \in C | \forall t \in \mathbb{N}, s \in B(G_{\mu,\nu}), s \in T \cap C \}
\]

**Proof:** Due to space constraint, the proof is in [36].

We now provide the proof of Theorem 1.

**Proof:** (Theorem 1) We divide the expected return of a random path $\sigma$ visiting a state $s \in S$ depending on whether it satisfies $\varphi_U := \emptyset B$ or not – i.e.,
\[
v^*(s) = E[G_t(\sigma) | s[t] = s, s[\leq t] \in U_B] Pr(s[\leq t] \in U_B) \\
+ E[G_t(\sigma) | s[t] = s, s[\leq t] \notin U_B] Pr(s[\leq t] \notin U_B) \]
for some fixed $t \in \mathbb{N}$. Notice that $s[\leq t] \notin U_B$ implies $s[t] \notin U_B$, and $s[\leq t] \in U_B$ implies $s[t] \in U_B$ almost surely. Hence, $Pr(s[\leq t] \in U_B)$ and $Pr(s[\leq t] \notin U_B)$ can be replaced with $Pr(s[\leq t] \in U_B)$ and $Pr(s[\leq t] \notin U_B)$, respectively.

After visiting the state $s$ at time $t$, let $L_t$ be the number of time steps until the first visit to a state in $U_B$ in (12) – i.e.,
\[
L_t = \min \{ \tau | s[t+\tau] \in U_B, \tau > 0 \}.
\]
Then, by Lemma 1, it holds that
\[
v^*(s) \geq E[G_t(\sigma) | s[t] = s, s[\leq t] \in U_B] Pr(s[\leq t] \in U_B) \\
\geq E\left[ \gamma^{|t-L_t|} G_{t+L_t}(\sigma) | s[t] = s, s[\leq t] \in U_B \right] Pr(s[\leq t] \in U_B) \\
\geq E\left[ \gamma^{|L_t|} | s[t] = s, s[\leq t] \in U_B \right] v^*(U_B) Pr(s[\leq t] \in U_B) = \\
= \gamma^{|t-L_t|} U_B Pr(s[\leq t] \in U_B); \\
\]
and
\[
v^*(s) \leq E[G_t(\sigma) | s[t] = s, s[\leq t] \notin U_B] Pr(s[\leq t] \notin U_B) \]
\[
\leq E\left[ \gamma^{|t-L_t|} G_{t+L_t}(\sigma) | s[t] = s, s[\leq t] \notin U_B \right] Pr(s[\leq t] \notin U_B) \\
\leq E\left[ \gamma^{|L_t|} | s[t] = s, s[\leq t] \notin U_B \right] v^*(U_B) Pr(s[\leq t] \notin U_B) = \\
= \gamma^{|t-L_t|} U_B Pr(s[\leq t] \notin U_B), \\
\]
where $v^*(U_B) = \min_{s \in U_B} v^*(s_B)$, $l$ is constant, and 0 and 0 hold from the Markov property and Jensen’s inequality.

Similarly, after leaving $s$ at $t$, let $L'_t$ be the number of time steps until the first time a state in $U_{\overline{BCC}} \cup U_C$ is reached – i.e.,
\[
L'_t = \min \{ \tau | s[t+\tau] \in U_{\overline{BCC}} \cup U_C, \tau > 0 \}.
\]

Then, using Lemma 1 and the Markov property, it holds that
\[
v^*(s) \leq E[G_t(\sigma) | s[t] = s, s[\leq t] \notin U_B] Pr(s[\leq t] \notin U_B) + Pr(s[\leq t] \in U_B) \\
\leq E\left[ \gamma^{|t-L'_t|} G_{t+L'_t}(\sigma) | s[t] = s, s[\leq t] \notin U_B \right] Pr(s[\leq t] \notin U_B) + Pr(s[\leq t] \in U_B) \\
\leq E\left[ \gamma^{|L'_t|} | s[t] = s, s[\leq t] \notin U_B \right] v^*(U_B) Pr(s[\leq t] \notin U_B) + Pr(s[\leq t] \in U_B), \\
\]
where $l'$ is some constant. The upper bound (18) and the lower bound (16) (due to (10)) approach the probability $Pr(s[\leq t] \in U_B)$ as $\gamma \to 1^-$, thereby concluding the proof.

**C. Reduction to Stochastic Rabin(1) Games**

To generalize our approach to Rabin conditions with $k$ pairs, we construct $k$ different stochastic Rabin(1) games and connect them with $\varepsilon$ actions so that the controller is able to switch between the Rabin pairs it aims to satisfy.

**Definition 5 (k-copy Game):** Let $[n]$ denote the set $\{1, 2, \ldots, n\}$ for a positive integer $n$. For a given stochastic Rabin($k$) game $G = (S, \{S_\mu\}, A, P, s_0, \text{Acc})$, with $\text{Acc}=\{(C_i, B_i)\}$, a $k$-copy game $G^* = (S^*, \{S^*_\mu\}, A^*, P^*, s^*_0, \text{Acc}^*)$ is a stochastic Rabin(1) game defined by:
- $S^* = \{S^*_\mu \times \{i\} \} \cup \{S^*_i \times \{k\}\}$ is the augmented state set with $S^*_\mu = S^*_i \times \{k\}$ the controller and $S^*_i \times \{k\}$ is the environment states, and $s^*_0 = (s_0, 1)$ is the initial state;
- $A^* = A \cup \{s_i | i \in [k]\}$ is the set of actions;
- $P^* : S^* \times A^* \times S^* \to [0, 1]$ is the transition function defined as
\[
P^*(\langle(s, i), a, (s', i')\rangle) = \\
\begin{cases} 
1 & \text{if } a \in A, i = i', \\
0 & \text{if } a \notin A, i = i', \\
\end{cases}
\]
\[
\text{if } s \in S^*_\mu, s = s', a = \varepsilon_i, i' = k + i, \\
\text{if } s \in S^*_\mu, s = s', a = \varepsilon', i' = i - k,
\]
otherwise;
- $\text{Acc}^* = \{(C^*, B^*)\}$ is the Rabin accepting set where
\[
C^* := \{\langle s, i \rangle | s \in C_i, i \in [k] \text{ or } s \in S^*_\mu, i \in [2k] \setminus [k]\}, \\
B^* := \{\langle s, i \rangle | s \in B_i, i \in [k]\}.
\]

Intuitively, the $k$-copy game $G^*$ consists of $k$ exact copies of the game $G$ for each accepting pair, and a dummy state $\langle s, i+k \rangle$ for every controller state $s \in S^*_\mu$ for each copy $i \in [k]$. The controller can choose an $\varepsilon_i$ in a state $\langle s, i \rangle$ and makes a transition to the dummy environment state $\langle s, i+k \rangle$ where the environment can only take the action $\varepsilon'$, which makes a transition to the controller state $\langle s, j \rangle$. The idea here is to connect the $k$ copies of the original game using these $\varepsilon$ actions so that in any state, the controller can jump to the $j$-th copy via an $\varepsilon_j$-a-dummy-state-$\varepsilon'$ sequence. All the
dummy states belong to $C^*$, prohibiting the $\varepsilon$-actions from being visited infinitely many times. Also, the only states belonging to $C^*$ and $B^*$ in the $i$-th copy are the ones belonging to $C_i$ and $B_i$, respectively. This allows each accepting pair to be independently satisfied in its corresponding copy as stated in the following theorem.

**Theorem 2:** Let $G^{(j)}$ be the stochastic Rabin($1$) game obtained from a Rabin($k$) game $G$ by replacing $\text{Acc}$ with $\{(C_j, B_j)\}$, and $W^{(j)}$ be the set of winning states such that for any $s \in W^{(j)}$, $P_{\mu^*}^{G^{(j)}}(s) = \varphi_{B,C}$, then, for any $(s,i) \in S^*$, it holds that

$$P_{\mu^*}^{G^{(j)}}((s,i) \models \varphi_{B^*,C}) = P_{\mu^*}^{G^{(j)}}(s,i) \models \Diamond \psi,$$

where $V = \{(s',i') \in S^* | s' \in \bigcup_{j=0}^\infty W^{(j)}\}$.

**Proof:** We prove (19) in two directions.

$\geq$: If a state $(s,i) \in V$, then, by definition, there exists $j$ such that $s \in W^{(j)}$. The controller can make a transition from $(s,i)$ to $(s,j)$ via the $\varepsilon$-actions and satisfy $\varphi_{B^*,C^*}$ by satisfying $\varphi_{B,C}$. Thus, the controller strategy maximizing the reachability probabilities in the worst case also guarantees the satisfaction probabilities of at least the maximin reachability probabilities i.e. $P_{\mu^*}^{G^{(j)}}((s,i) \models \varphi_{B^*,C^*}) \geq P_{\mu^*}^{G^{(j)}}((s,i) \models \Diamond \psi)$.

$\leq$: All the transitions via the $\varepsilon$-actions pass through a state in $C^*$. Under any strategy pair, the BSCCs having $\varepsilon$-transitions of the induced MC are rejecting. Since without some $\varepsilon$-transitions, it is not possible for a BSCC to contain states from two different accepting pairs, an accepting BSCC must satisfy only a single pair. In addition, in the worst case, the satisfaction probability can be maximized by being rejected by the probability of reaching a state that belongs to an accepting BSCC for any environment strategy. Thus, such states must be a winning state for some accepting pair, which implies that $P_{\mu^*}^{G^{(j)}}((s,i) \models \varphi_{B^*,C^*}) \leq P_{\mu^*}^{G^{(j)}}((s,i) \models \Diamond \psi)$.

Any control strategy $\mu^*$ in $G^*$ has a corresponding finite-memory strategy $\mu$ in the Rabin($k$) game $G$, which can be captured by a deterministic finite automaton (DFA) with $k$ states. In state $i$, the state of the DFA changes from state $i \in [k]$ to $j \in [k]$, if $\mu^*(s,i) = \varepsilon_j$; the DFA state stays the same and the control strategy chooses action $a \in A$ if $\mu^*(s,i) = a$. If $\mu^*$ is a maximin strategy for $G^*$ then under the induced strategy $\mu$, the controller satisfies the acceptance condition with probability that is no lower than that of the probability of reaching a winning state of an accepting pair.

**Corollary 1:** A maximin control strategy for $G^*$ of a stochastic Rabin($k$) game $G$ induces a control strategy $\mu$ for $G$ such that, for any environment strategy $\nu$,

$$P_{\mu,\nu}(G \models \varphi_{\text{Acc}}) \geq P_{\mu}(G \models \Diamond \psi),$$

where $\varphi_{\text{Acc}} := \bigvee_{(B_j,C_j) \in \text{Acc}} (\Box \Diamond B_j \land \neg \Box B_j)$, and $W := \bigcup_{i=1}^\infty W^{(i)}$, with $W^{(i)}$ defined as in Theorem 2.

**Proof:** For any environment strategy $\nu$ in $G$ we can construct a corresponding environment strategy $\nu^*$ in $G^*$ such that $\nu^*(s,i) = \nu(s)$ for all $i \in [k]$ and $\nu^*(s,i) = \varepsilon'$ for all $[2k] \in [k]$. Note that the strategy pairs $(\mu, \nu)$ and $(\mu^*, \nu^*)$ induce the same MCs. Since satisfying $(C_j, B_j)$ satisfies $\varphi_{\text{Acc}}$, we have $P_{\mu,\nu}(G \models \varphi_{\text{Acc}}) \geq P_{\mu^*,\nu^*}(G^* \models \varphi_{B^*,C^*})$, which combined with Theorem 2 concludes the proof.

The induced control strategy $\mu$ guarantees a satisfaction probability that is larger than or equal to $P_{\mu}(G \models \Diamond \psi)$.

Note that computing the winning states in stochastic Rabin games is NP-Complete in the number accepting pairs [35]. Thus, it is unlikely to construct a stochastic Rabin($1$) game from any given stochastic Rabin($k$) game without an exponential blowup in the number of states.

### D. Controller Synthesis via Reinforcement Learning

We now state the main result of our approach.

**Theorem 3:** For a given stochastic Rabin($k$) game, there exists a $\gamma'$ such that for any $\gamma \in (\gamma', 1)$, the minimax-Q using the reward and the discount functions in Theorem 1 is guaranteed to converge to a strategy $\mu$ satisfying (20).

**Proof:** The claim directly follows from Theorem 1, Corollary 1, and the fact that pure and memoryless strategies are finite and sufficient for both the controller and the environment in stochastic Rabin($1$) games [35].

For a given stochastic game and an LTL specification, we reduce the control synthesis problem to finding a maximin controller strategy in a stochastic Rabin($k$) game $G$ using the automata-based approach from Sec. III-A. This is further reduced to finding a strategy that maximizes the probability of satisfying a single Rabin pair in the worst case, in a stochastic Rabin($1$) game $G^*$ using the method from Sec. III-B. Finally, we transform the objective of satisfying a Rabin pair to a discounted reward maximization objective (Sec. III-B), allowing the use of RL to synthesize controllers.

Algorithm 1 summarizes the steps of our approach. Here, $\alpha$ is the learning rate and the bold $s$ character denotes a state vector consisting of the state of the original game, the DFA state, and the index of Rabin pair. After the construction of $G^*$, the algorithm performs minimax-Q learning. In each iteration of learning, the algorithm derives an $\epsilon$-greedy strategy pair, which means that under these strategies, the controller and the environment randomly choose their actions with probability of $\epsilon$, as well choose their best action with probability of $1-\epsilon$. After the convergence, the algorithm returns a maximin control strategy $\mu^*$ for $G^*$, which induces a finite-memory strategy for the original game, which guarantees the lower bound provided in Corollary 3.

### IV. Experimental Results

Our RL-based control synthesis framework was implemented as a software tool [37] in Python; we used Rabinizer 4 [38] to translate LTL formulas into DRAs, and minimax-Q learning using the presented reward and discount functions.

During learning, $\epsilon$-greedy strategies are followed by both players after starting in a random state, and the episodes are terminated after 1K steps. We set the parameter $\epsilon$ and the learning rate $\alpha$ to 0.5 and gradually decreased them to 0.05 during learning; we used the discount factors of $\gamma_C=1-(0.01)$, $\gamma_B=1-(0.01)^2$ and $\gamma=1-(0.01)^3$.

We considered robot planning tasks in 2-D grid worlds. A mobile robot can move to adjacent four cells in a single step using actions: North, South, East and West. When the robot tries to move to a cell with an obstacle, it hits the obstacle and stays in its previous position; once it moves to an absorbing cell it cannot leave it. In the figures, obstacles and absorbing cells are represented by filled and empty circles. Finally, each cell is labeled with a set of atomic propositions.
Algorithm 1: Model-free RL for control synthesis in stochastic games from LTL specifications.

**Input:** LTL formula $\phi$, stochastic game $G$

Translate $\phi$ to a DRA $A_{\phi}$

Construct the product $G^* \times G$ and $A_{\phi}$

Reduce $G^* \times G$; Initialize $Q(s, a)$ on $G^*$

for $t = 0, 1, \ldots, T$ do

- Derive an $\epsilon$-greedy strategy pair $(\mu^*, \nu^*)$ from $Q$

- Take the action $a_t \leftarrow \begin{cases} \mu^*(s_t), & s_t \in S_\mu^* \\ \nu^*(s_t), & s_t \in S_\nu^* \end{cases}$

- Observe the next state $s_{t+1}$

- Update the policy with $Q(s_t, a_t) \leftarrow (1 - \alpha)Q(s_t, a_t) + \alpha R(s_t) + \alpha \Gamma(s_t) \cdot \left( \min_{a'} \max_{s'} Q(s_{t+1}, a'), s_{t+1} \in S_\mu^* \right) \cdot \left( \max_{s'} \min_{a'} Q(s_{t+1}, a'), s_{t+1} \in S_\nu^* \right)$

end for

return a greedy control strategy $\mu^*_\epsilon$ from $Q$

---

Fig. 2: The environment actions for control action North. The arrow lengths are proportional to the movement direction probabilities.

Fig. 3: The derived strategy: (a) from $b$ to $c$, and (b) from $c$ to $b$, under which $\phi_1$ from (21) is almost surely satisfied, independently from the environment. The most likely paths are in a lighter blue.

A. Robust Control Design

In our first case study, the robot could unpredictably move in a direction orthogonal to the intended direction. We model this source of nondeterminism as the environment, observing the actions of the controller and acting to minimize the probability that the controller achieves the given task. Specifically, the environment can reactively choose one of the actions: None, Both, Right and Left (Fig. 2). For None, the robot moves in the intended direction without any disturbance. With Both, it can go sideways with probability 0.2 (0.1 for each direction); with Right (Left), it moves as intended with probability of 0.9 and right (left) with probability of 0.1.

The control objective is to visit a state labeled with $b$ and a state labeled with $c$ repeatedly; and the safe states, labeled with either $d$ or $e$, should not be left after a certain time, i.e.,

$$\phi_1 = \square(b \land c) \land (\boxdot d \lor \boxdot e),$$  (21)

which we translated into a DRA with two accepting pairs.

Fig. 3 shows the grid world and the derived strategy after 128K episodes. The objective cannot be satisfied by visiting the states (with $b$ and $c$) at the top-left or top-right corner, since the environment could force the robot to leave the safe states. The only way to achieve the task is going from the state labeled with $b$ to the state $c$ and vice versa without leaving the safe states. With the strategy in Fig. 3, the robot does not move to the top row once leaving it, and does not visit the unsafe state in the middle regardless of the environment actions. With the strategies in Fig. 3(a),(b), the robot eventually reaches the states with $b$ and $c$, respectively.

B. Avoiding Adversary

In this study, robot movement is not affected by the environment. Instead, the robot moves as intended with probability of 0.8 and goes to the right or the left side of the intended direction with probabilities of 0.1. Another agent, controlled by an adversary, can take the same four actions as the controller, with the same probability distribution.

The size of the state space here is $(5 \times 5) \times (5 \times 5) = 625$, as there are two independent agents. We define the labeling function as $L(s_1, s_2) := \begin{cases} L(s_1), & s_1 \neq s_2, \\ L(s_1) \cup \{a\}, & s_1 = s_2, \end{cases}$ where the label $a$ represents the state when both agents are in the same position (adversary ‘catches’ the robot). The robot objective is the same as in (21), except that it additionally needs to avoid the adversary at all costs, i.e., $\phi_2 = \phi_1 \land \boxdot \neg a$.

Fig. 4 shows the control strategy obtained after 512K episodes. There are four safe zones: at the top-left, at the top-right, and two at the bottom part of the grid. The robot or the adversary can get trapped in a sink state with probability $p \geq 0.2$ while traveling between the top and the bottom parts of the grid. Thus, the optimal controller strategy is not to switch zones unless the adversary is in the same zone. For example, in Fig. 4a, if the robot is in the bottom part, the controller should not try to move the robot to the top-right part, a farther safe zone, because there is a chance ($p \geq 0.2$) that the adversary ends up with a sink state if (s)he tries to move to the bottom part. If the robot is in the top-right part, the controller should switch to the second Rabin pair via $\varepsilon_2$ and make the robot stay in the same zone. However, in Fig. 4b, the robot cannot stay in the bottom part because otherwise the adversary will eventually catch her or him.

V. Conclusions

In this paper, we introduced an RL-based approach for synthesis of controllers from LTL specifications in stochastic games. We reduced this problem to finding a control strategy in a stochastic Rabin game with a single accepting pair. We introduced a rewarding mechanism that transforms the objective of maximizing the (worst-case) probability of satisfying the Rabin condition into maximizing the discounted reward, and presented an RL algorithm to find such a strategy. We also generalized our approach to any LTL specification, with the Rabin condition having $k > 1$ accepting pairs, providing a lower bound on the satisfaction probabilities.
Yasser Shoukry, Pierluigi Nuzzo, Ayca Balkan, Indranil Saha, Al-

Tomasz Brzázil, Krishnendu Chatterjee, Martin Chmelík, Vojtech Forejt, Jan I

REFERENCES

[1] Fahiem Bacchus and Froduald Kabanza. Using temporal logics to express search control knowledge for planning. Artificial intelligence, 116(1-2):123–191, 2000.
[2] Georgios E. Fainekos, Hadas Kress-Gazit, and George J. Pappas. Temporal logic motion planning for mobile robots. In Proceedings of the 2005 IEEE International Conference on Robotics and Automation, pages 2020–2025. IEEE, 2005.
[3] Hadas Kress-Gazit, Georgios E. Fainekos, and George J. Pappas. Where’s Waldo? sensor-based temporal logic motion planning. In Proceedings 2007 IEEE International Conference on Robotics and Automation, pages 3116–3121. IEEE, 2007.
[4] Hadas Kress-Gazit, Georgios E Fainekos, and George J Pappas. Translating structured English to robot controllers. Advanced Robotics, 22(12):1343–1359, 2008.
[5] Georgios E. Fainekos, Antoine Girard, Hadas Kress-Gazit, and George J. Pappas. Temporal logic motion planning for dynamic robots. Automatica, 45(2):343–352, 2009.
[6] Hadas Kress-Gazit, Georgios E. Fainekos, and George J. Pappas. Temporal-logic-based reactive mission and motion planning. IEEE Transactions on Robotics, 25(6):1370–1381, 2009.
[7] Amit Bhatia, Lydia E. Kavraki, and Moshe Y. Vardi. Sampling-based motion planning with temporal goals. In 2010 IEEE International Conference on Robotics and Automation, pages 2689–2696. IEEE, 2010.
[8] Morteza Lahijanian, Joseph Wasniewski, Sean B. Andersson, and Calvin Bela. Motion planning and control from temporal logic specifications with probabilistic satisfaction guarantees. In 2010 IEEE International Conference on Robotics and Automation, pages 3227–3232. IEEE, 2010.
[9] Stephen L Smith, Jana Tumova, Calvin Bela, and Daniela Rus. Optimal path planning for surveillance with temporal-logic constraints. The International Journal of Robotics Research, 30(14):1695–1708, 2011.
[10] Tichkorn Wongpiromsarn, Ufuk Topcu, and Richard M Murray. Receding horizon temporal logic planning. IEEE Transactions on Automatic Control, 57(1):2817–2830, 2012.
[11] Meng Guo, Karl H. Johansson, and Dimos V Dimarogonas. Revising motion planning under linear temporal logic specifications in partially known workspaces. In 2013 IEEE International Conference on Robotics and Automation, pages 5025–5032. IEEE, 2013.
[12] Alphan Ulusoy, Stephen L. Smith, Xu Chu Ding, Calvin Bela, and Daniela Rus. Optimality and robustness in multi-robot path planning with temporal logic constraints. The International Journal of Robotics Research, 32(8):889–911, 2013.
[13] Indranil Saha, Rattanachai Ramaithima, Vijay Kumar, George J. Pappas, and Sanjit A. Seshia. Automated composition of motion primitives for multi-robot systems from safe LTL specifications. In 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 1525–1532. IEEE, 2014.
[14] Keliang He, Morteza Lahijanian, Lydia E. Kavraki, and Moshe Y. Vardi. Towards manipulation planning with temporal logic specifications. In 2015 IEEE International Conference on Robotics and Automation (ICRA), pages 346–352. IEEE, 2015.
[15] Jana Tumova and Dimos V Dimarogonas. Multi-agent planning under local LTL specifications and event-based synchronization. Automatica, 70:259–248, 2016.
[16] Yasser Shoukry, Pierluigi Nuzzo, Ayca Balkan, Indranil Saha, Albert L. Sangiovanni-Vincentelli, Sanjit A. Seshia, George J. Pappas, and Paulo Tabuada. Linear temporal logic motion planning for teams of underactuated robots using satisfiability modulo convex programming. In 2017 IEEE 56th Annual Conference on Decision and Control (CDC), pages 1132–1137. IEEE, 2017.
[17] Andrew G Barto. Reinforcement learning: Connections, surprises, challenges. AI Magazine, 40(1), 2019.
[18] Jie Fu and Ufuk Topcu. Probably approximately correct MDP learning and control with temporal logic constraints, 2014. arXiv:1404.7073 [cs.SY].
[19] Tomas Brzázil, Krishnendu Chatterjee, Martin Chmelík, Vojtech Forejt, Jan I

[20] Dona Sadigh, Eric S. Kim, Samuel Coogan, S. Shankar Sastry, and Sanjit A. Seshia. A learning based approach to control synthesis of Markov decision processes for linear temporal logic specifications. In 53rd IEEE Conference on Decision and Control, pages 1091–1096. IEEE, 2014.
[21] Xiao Li, Cristian-Ioan Vasile, and Calin Bela. Reinforcement learning with temporal logic rewards. In 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 3834–3839. IEEE, 2017.
[22] Rodrigo Toro Icarte, Torny Klassen, Richard Valenzano, and Sheila McIlraith. Using reward machines for high-level task specification and decomposition in reinforcement learning. In International Conference on Machine Learning, pages 2107–2116, 2018.
[23] Alberto Camacho and Sheila A. McIlraith. Learning interpretable models expressed in linear temporal logic. In Proceedings of the International Conference on Automated Planning and Scheduling, volume 29, pages 621–630, 2019.
[24] Ernst Moritz Hahn, Mateo Perez, Sven Schewe, Fabio Somenzi, Ashutosh Trivedi, and Dominik Wojtczak. Omega-regular objectives in model-free reinforcement learning. In International Conference on Tools and Algorithms for the Construction and Analysis of Systems, pages 395–412. Springer, 2019.
[25] Mohammadseif Hassanbeh, Yannis Kantaros, Alessandro Abate, Daniel Kroening, George J Pappas, and Insup Lee. Reinforcement learning for temporal logic control synthesis with probabilistic satisfaction guarantees. In 2019 IEEE 58th Conference on Decision and Control (CDC), pages 5338–5343. IEEE, 2019.
[26] Alper Kamil Bozkurt, Yu Wang, Michael M. Zavlanos, and Miroslav Pajic. Control synthesis from linear temporal logic specifications using model-free reinforcement learning. In 2020 IEEE International Conference on Robotics and Automation (ICRA), pages 10349–10355. IEEE, 2020.
[27] Lloyd S. Shapley. Stochastic games. Proceedings of the national academy of sciences, 39(10):1109–1105, 1953.
[28] Thomas A. Henzinger and Nir Piterman. Solving games without determinization. In International Workshop on Computer Science Logic, pages 395–410. Springer, 2006.
[29] Min Wen and Ufuk Topcu. Probably approximately correct learning in stochastic games with temporal logic specifications. In Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI’16, page 3630–3636. AAAI Press, 2016.
[30] Pranav Ashok, Jan Krétsinský, and Maximilian Weininger. PAC statistical model checking for Markov decision processes and stochastic games. In International Conference on Computer Aided Verification, pages 497–519. Springer, 2019.
[31] Christel Baier and Joost-Pieter Katoen. Principles of Model Checking. MIT Press, Cambridge, MA, USA, 2008.
[32] Michael L. Littman. Markov games as a framework for multi-agent reinforcement learning. In Machine Learning Proceedings 1994, pages 157–163. Elsevier, 1994.
[33] Junling Hu and Michael P Wellman. Nash q-learning for general-sum stochastic games. Journal of Machine Learning Research, 4(Nov):1039–1069, 2003.
[34] Michael Bowling and Manuela Veloso. An analysis of stochastic game theory for multiagent reinforcement learning. Technical report, Carnegie-Mellon Univ Pittsburgh Pa School of Computer Science, 2000.
[35] Krishnendu Chatterjee and Thomas A. Henzinger. A survey of statistical model checking of Markov decision processes and stochastic games. In International Conference on Computer Aided Verification, pages 497–519. Springer, 2019.
[36] Alper Kamil Bozkurt, Yu Wang, Michael M. Zavlanos, and Miroslav Pajic. Model-free reinforcement learning for stochastic games with linear temporal logic objectives, 2020. arXiv:2010.01050 [cs.RO].
[37] CPSL@Duke. CSRL, 2020.
[38] Alper Kamil Bozkurt, Yu Wang, Michael M. Zavlanos, and Miroslav Pajic. Model-free reinforcement learning for stochastic games with linear temporal logic objectives, 2020. arXiv:2010.01050 [cs.RO].
[39] CPSL@Duke. CSRL, 2020. https://gitlab.oit.duke.edu/cpsl/csr1.
[40] Jan Krétsinský, Tobias Meggendorfer, Salomon Sickert, and Christoph Ziegler. Rabinizer 4: from LTL to your favourite deterministic automaton. In International Conference on Computer Aided Verification, pages 567–577. Springer, 2018.