

Model-Free Reinforcement Learning for Stochastic Games with Linear Temporal Logic Objectives

Alper Kamil Bozkurt, Yu Wang, Michael Zavlanos, and Miroslav Pajic

Abstract—We study the problem of synthesizing control strategies for 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 LTL formulas for which the Rabin condition has a larger number of accepting pairs, providing a lower bound on the satisfaction probability. Finally, we illustrate applicability of our RL method on two motion planning case studies.

I. INTRODUCTION

Reinforcement learning (RL) provides methods for finding solutions to sequential decision-making problems where the objective is to maximize the discounted rewards in unknown environments [1]. Recent RL developments, such as the use of deep learning and Monte Carlo tree search, have led to successful applications in a range of domains including (super) human-level board and Atari game playing [2], [3]. Yet, despite these advances, there is a need to provide RL methods with robustness and safety guarantees to allow their use in real-world systems, particularly in cyber-physical [4], [5], [6], [7], [8] and robotic systems [9], [10], [11]. To achieve this, a major challenge is to design reward signals such that a strategy optimizing the discounted reward achieves the desired task [12].

Linear temporal logic (LTL) allows formal capturing desired temporal properties of a control task or system (e.g., robot planning). Using LTL to specify the task objective can prevent unintended consequences of an optimal strategy for a shaped reward function. LTL specifications can be directly extracted from high-level requirements in robot planning and control [13], [14], [15]. Thus, synthesizing controllers from LTL objectives for Markov decision processes (MDPs) using RL has attracted significant attention [16], [17], [18]. These methods generally translate the LTL specification into a limit-deterministic Büchi automaton (LDBA), which is

then composed with the initial MDP, and design a reward function based on the acceptance condition of the automaton. Augmentation of the state space using the states of the LDBA solves the memory requirements of the task. In addition, the Büchi acceptance condition, which is repeated reachability, enables the use of simple reward functions.

Such LDBA-based rewarding approaches are not well-suited for stochastic games because LDBAs and many other nondeterministic automata, in general, cannot be used in solving games [19]. Hence, there are few studies on learning-based synthesis from temporal objectives for stochastic games. One approach is to translate the LTL specifications to deterministic automata with more complicated acceptance conditions than LDBAs. For example, [20] proposed a probably approximately correct (PAC) learning algorithm for stochastic games with LTL and discounted sums of rewards objective. However, the approach assumes that the transition graph (i.e., topology) is known a-priori, the LTL objective must belong to a very limited subset of LTL formulas that can be translated into a deterministic Büchi automaton (DBA), and there exists a strategy that almost surely satisfies the LTL objective. These assumptions allow the pre-computation of the winning regions before the learning.

Recently, [21] introduced a model-based learning method that yields PAC guarantees for reachability objectives. The method uses on-the-fly detection of (simple) end components of stochastic games, and careful construction of the confidence intervals on the transition probabilities. However, only a small fragment of LTL formulas can be expressed by the reachability objectives. Also, as a model-based method, it is not efficient in terms of space requirements when the number of possible successors of actions is not small.

To address these limitations, 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. To achieve this, we translate the LTL specification into a DRA and introduce a reward and a discount (or termination) function based on the Rabin acceptance condition. We first consider DRAs with a single accepting pair and prove any model-free RL algorithm using these functions converges 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 motion planning case studies.

This work is sponsored in part by the ONR under agreements N00014-17-1-2504, N00014-20-1-2745 and N00014-18-1-2374, AFOSR award number FA9550-19-1-0169, and the NSF CNS-1652544 and CNS-1932011 grants.

Alper Kamil Bozkurt, Yu Wang, Michael Zavlanos, and Miroslav Pajic are with Duke University, Durham, NC 27708, USA, {alper.bozkurt, yu.wang094, michael.zavlanos, miroslav.pajic}@duke.edu

II. PRELIMINARIES AND PROBLEM STATEMENT

A. Stochastic (Turn-Based) 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 $\mathcal{G} = (S, (S_\mu, S_\nu), A, P, s_0, \text{AP}, L)$, 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 at which the environment chooses actions; A is a finite set of actions whereas $A(s)$ denotes the set of actions that can be taken in state $s \in S$; $P : S \times A \times S \rightarrow [0, 1]$ is the transition probability function such that for all $s \in S$, $\sum_{s' \in S} P(s, a, s') = 1$ if $a \in A(s)$, and 0 otherwise; AP is a finite set of atomic propositions; and $L : S \rightarrow 2^{\text{AP}}$ is a labeling function.

A *path* in a stochastic game \mathcal{G} is an infinite sequence of states $\sigma = s_0 s_1, \dots$ 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 write $\sigma[t]$, $\sigma[:t]$ and $\sigma[t+1:]$ to denote the state s_t , the prefix $s_0 s_1 \dots s_t$ and the suffix $s_{t+1} s_{t+2} \dots$ of the path, respectively.

The behavior of the players in stochastic games is described by strategies, which maps the previously visited states to the available actions in the current state.

Definition 2 (Strategies): For a game \mathcal{G} , let $S_\mu^+(S_\nu^+)$ denote the set of all **finite** prefixes $\sigma_f^{s_\mu} (\sigma_f^{s_\nu})$ ending with a state $s_\mu \in S_\mu (s_\nu \in S_\nu)$, respectively) of paths in the game. Then,

- a (pure) **control strategy** μ is a function $\mu : S_\mu^+ \rightarrow A$ such that $\mu(\sigma_f^{s_\mu}) \in A(s_\mu)$ for all $\sigma_f^{s_\mu} \in S_\mu^+$,
- a (pure) **environment strategy** ν is a function $\nu : S_\nu^+ \rightarrow A$ such that $\nu(\sigma_f^{s_\nu}) \in A(s_\nu)$ for all $\sigma_f^{s_\nu} \in S_\nu^+$,
- a strategy π is **memoryless**, if it only depends on the current state, i.e., $\pi(\sigma_f^s) = \pi(\sigma_f^{s'})$ if $s = s'$ for any σ_f^s and $\sigma_f^{s'}$, and thus can be defined as $\pi : S \rightarrow A$.

The induced Markov chain (MC) of a game \mathcal{G} under a strategy pair (μ, ν) is tuple $\mathcal{G}_{\mu, \nu} = (S, P_{\mu, \nu}, s_0, \text{AP}, L)$, 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 by $\mathcal{G}_{\mu, \nu}^s$ the MC resulting from changing the initial state from s_0 to $s \in S$ in $\mathcal{G}_{\mu, \nu}$, and use $\sigma \sim \mathcal{G}_{\mu, \nu}^s$ to denote a random path sampled from $\mathcal{G}_{\mu, \nu}^s$. Finally, a **bottom strongly connected component** (BSCC) of the (induced) MC $\mathcal{G}_{\mu, \nu}$ is a strongly connected component with no outgoing transitions; we use $\mathcal{B}(\mathcal{G}_{\mu, \nu})$ to denote the set of all BSCCs of $\mathcal{G}_{\mu, \nu}$.

B. LTL and Deterministic Rabin Automata

We capture the desired behaviors of a labeled stochastic game by LTL specifications, which impose requirements on the label sequences corresponding to the infinite paths of the game. LTL offers a formal language that can be used to specify desired temporal characteristics or tasks of a controller [22]. In addition to the standard Boolean operators:

negation (\neg) and conjunction (\wedge), LTL formulas can include two temporal operators, namely next (\bigcirc) and until (\mathbf{U}), and any recursive combinations of the operators. The formal syntax of LTL is defined as [22]

$$\varphi := \text{true} \mid a \mid \varphi_1 \wedge \varphi_2 \mid \neg \varphi \mid \bigcirc \varphi \mid \varphi_1 \mathbf{U} \varphi_2, \quad a \in \text{AP}, \quad (1)$$

where AP is a set of atomic propositions.

For a stochastic game \mathcal{G} with a labeling function L , the LTL semantics is defined over the paths of the game. A path σ satisfies an LTL formula φ , denoted by $\sigma \models \varphi$ if:

- $\varphi = a$ and $a \in L(\sigma[0])$,
- $\varphi = \varphi_1 \wedge \varphi_2$, $\sigma \models \varphi_1$ and $\sigma \models \varphi_2$,
- $\varphi = \neg \varphi'$ and $\sigma \not\models \varphi'$,
- $\varphi = \bigcirc \varphi'$ and $\sigma[1:] \models \varphi'$,
- $\sigma \models \varphi_1 \mathbf{U} \varphi_2$, $\exists i. \sigma[i] \models \varphi_2$ and $\forall j < i. \sigma[j] \models \varphi_1$.

Other operators can be easily derived: $\varphi_1 \vee \varphi_2 := \neg(\neg \varphi_1 \wedge \neg \varphi_2)$, $\varphi_1 \rightarrow \varphi_2 := \neg \varphi_1 \vee \varphi_2$, (eventually) $\diamond \varphi := \text{true} \mathbf{U} \varphi$; and (always) $\square \varphi := \neg(\diamond \neg \varphi)$ [22].

Any LTL formula can be systematically transformed into a DRA that accepts the language of all paths satisfying the formula [22]. DRAs are similar to deterministic finite automata except for the acceptance criteria, which is defined based on infinite visits of some states.

Definition 3 (Deterministic Rabin Automata): A DRA is a tuple $\mathcal{A} = (Q, \Sigma, \delta, q_0, \text{Acc})$ where Q is a finite set of states; Σ is a finite alphabet; $\delta : Q \times \Sigma \rightarrow Q$ is the transition function; $q_0 \in Q$ is an initial state; and 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 σ is accepted by the DRA if it satisfies the **Rabin condition** – i.e., there exists a pair $(C_i, B_i) \in \text{Acc}$ such that the states in C_i are visited finitely many times and at least one state in B_i visited is infinitely often, namely,

$$\exists i : \text{inf}(\sigma) \cap C_i = \emptyset \wedge \text{inf}(\sigma) \cap B_i \neq \emptyset, \quad (2)$$

where $\text{inf}(\sigma)$ denotes the set of states visited by the path σ for infinitely many times. The **Rabin index** of an LTL formula is the minimum number of accepting pairs a DRA recognizing the formula can have. Without loss of generality, we assume that the number of accepting pairs, k , is equal to the Rabin index of the accepted language.

Example 1: Figure 1 illustrates a DRA derived from the LTL formula $\varphi = \square \diamond b \vee \diamond \square d$, with Rabin acceptance sets $B_1 = \{q_1\}$, $C_2 = \{q_0\}$ and $B_2 = \{q_2\}$ (i.e., acceptance condition $\text{Acc} = \{(\emptyset, B_1), (C_2, B_2)\}$). Note that consuming a label having b in it, leads to a transition to the state q_1 from any state. Thus, any path σ containing infinitely many states labeled with b induces an execution that visits q_1 infinitely many times; thereby satisfying the Rabin condition. Other executions satisfying the Rabin conditions are the ones visiting q_2 infinitely many times but q_0 only finitely many times. Those can be only produced by the paths that after some point, do not contain a state whose label is not d .

C. Reinforcement Learning for Stochastic Games

Let $R : S \rightarrow \mathbb{R}$ be a *reward function* and $\gamma \in (0, 1)$ be the discount factor for a given two-player zero-sum stochastic

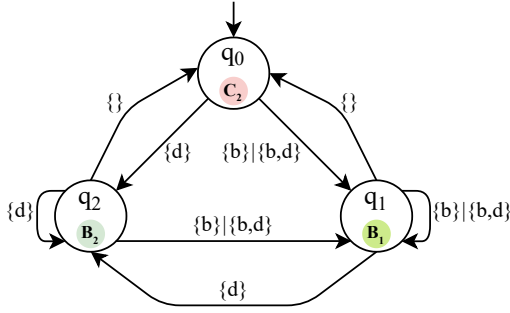


Fig. 1: A DRA derived from the LTL formula $\varphi = \square \diamond b \vee \square \diamond d$. Here, $B_1 = \{q_1\}$, $C_2 = \{q_0\}$ and $B_2 = \{q_2\}$ are the states in the Rabin acceptance condition $\text{Acc} = \{(\emptyset, B_1), (C_2, B_2)\}$.

game \mathcal{G} . The *value* of a state s under the strategy pair (μ, ν) is the expected sum of the discounted reward

$$v_{\mu, \nu}(s) = \mathbb{E}_{\sigma \sim \mathcal{G}_{\mu, \nu}} \left[\sum_{i=0}^{\infty} \gamma^i R(\sigma[t+i]) \mid \sigma[t] = s \right], \quad (3)$$

for any fixed $t \in \mathbb{N}$, such that $Pr_{\sigma \sim \mathcal{G}_{\mu, \nu}}[\sigma[t]=s] > 0$. We omit the subscript $\sigma \sim \mathcal{G}_{\mu, \nu}$ from the expectation and write \mathbb{E} rather than $\mathbb{E}_{\sigma \sim \mathcal{G}_{\mu, \nu}}$.

The RL objective is to find an optimal *control strategy* μ_* to maximize 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 [23], [24]. The optimal values in these games satisfy [25]

$$v_*(s) = \max_{\mu} \min_{\nu} v_{\mu, \nu}(s), \quad (4)$$

where μ and ν are pure and memoryless control and environment strategies, respectively. In addition, the *optimal values* $v_*(s)$ need to satisfy the Bellman equations

$$v_*(s) = R(s) + \gamma \begin{cases} \max_{a \in A(s)} \sum_{s' \in S} P(s, a, s') v_*(s') & \text{if } s \in S_{\mu}, \\ \min_{a \in A(s)} \sum_{s' \in S} P(s, a, s') v_*(s') & \text{if } s \in S_{\nu}. \end{cases}$$

Model-free RL methods aim to learn the optimal values of the stochastic game, in which neither the transition probabilities nor the game topology are known, without explicitly constructing a transition-Q 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 minimax-Q method can learn the optimal values from any (likely non-optimal) strategies used during learning as long as all the actions in each state are chosen infinitely often [24], [26].

D. Problem Formulation

We assume that the considered game \mathcal{G} is fully observable for both players; i.e., both the controller and environment are aware of the current game state. A control synthesis problem can be roughly described as finding a strategy for the controller in a stochastic game such that all the

paths produced under the strategy satisfy the given LTL specification regardless of the behavior of the environment. If such a strategy does not exist, the objective becomes to find a strategy that maximizes the probability that a produced path satisfies the specification in the worst case.

To simplify our notation, we use $Pr_{\mu, \nu}^{\mathcal{G}}(s \models \varphi)$ to denote the probability of the paths starting from the state s that satisfy the formula φ under the strategy pair (μ, ν) – i.e.,

$$Pr_{\mu, \nu}^{\mathcal{G}}(s \models \varphi) := Pr_{\sigma \sim \mathcal{G}_{\mu, \nu}^s}(\sigma \models \varphi); \quad (5)$$

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

Problem 1: Given a labeled turn-based stochastic game \mathcal{G} , where the transition probabilities are fully unknown, and an LTL specification φ , design a model-free RL algorithm that finds a *pure finite-memory* controller strategy μ_* such that

$$Pr_{\mu_*, \nu}(\mathcal{G} \models \varphi) \geq Pr_*(\mathcal{G} \models \varphi) \quad (6)$$

for any environment strategy ν .

III. LEARNING FOR STOCHASTIC RABIN GAMES

In this section, we describe our model-free RL approach to derive control strategies that maximize the (worst-case) probability of satisfying a given LTL formula. 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 objective φ 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 an arbitrary ($k > 1$) number of accepting pairs; thereby allowing the use of our method for all possible LTL specifications. Specifically, we construct a game from k copies of the original game, where only a single Rabin pair needs to be satisfied; an optimal solution to this game guarantees a lower bound on the satisfaction probabilities.

A. Product Game Construction

Our main idea is that by forming an augmented state space, Problem 1 can be reduced into finding a memoryless control strategy. Specifically, we compose the states of the game \mathcal{G} with the states of the DRA \mathcal{A} derived from the LTL specification φ . Then, the goal in this space is to satisfy the Rabin acceptance condition, for which memoryless control strategies suffice [27].

Definition 4 (Product Game): A product game $\mathcal{G}^{\times} = (S^{\times}, (S_{\mu}^{\times}, S_{\nu}^{\times}), A^{\times}, P^{\times}, s_0^{\times}, \text{Acc}^{\times})$ of a labeled turn-based stochastic game $\mathcal{G} = (S, (S_{\mu}, S_{\nu}), A, P, s_0, \text{AP}, L)$ and a DRA $\mathcal{A} = (Q, 2^{\text{AP}}, \delta, q_0, B)$ is defined as follows:

- $S^{\times} = S \times Q$ is the set of augmented states, where the initial state s_0^{\times} is $\langle s_0, q_0 \rangle$,
- $S_{\mu}^{\times} = S_{\mu} \times Q$ is the set of augmented controller states,

- $S_\nu^\times = S_\nu \times Q$ is the set of augmented environment states,
- $A^\times = A$ is the set of actions,
- $P^\times : S^\times \times A^\times \times S^\times \rightarrow [0, 1]$ is the transition function:

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

- Acc^\times is a set of k accepting pairs $\{(C_i^\times, B_i^\times)\}_{i=1}^k$ where $C_i^\times = C_i \times Q$ and $B_i^\times = B_i \times Q$.

Similarly to (2), a path σ^\times of the product game \mathcal{G}^\times satisfies the Rabin condition if there exists i , such that $\text{inf}(\sigma^\times) \cap C_i^\times = \emptyset \wedge \text{inf}(\sigma^\times) \cap B_i^\times \neq \emptyset$. 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 game and the original game. Similarly, a strategy for the product game induces a strategy in the original game and vice versa. Note that, 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. On the other hand, the probability of satisfying the Rabin condition under any strategy pair in the product game is equivalent 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 \times and use $\mathcal{G} = (S, (S_\mu, S_\nu), A, P, s_0, \text{Acc})$ and $s \in S$ instead of $\mathcal{G}^\times = (S^\times, (S_\mu^\times, S_\nu^\times)A^\times, P^\times s_0^\times, \text{Acc}^\times)$ and $\langle s, q \rangle \in S^\times$.

B. Rabin(1) Condition to Discounted Rewards

We first consider the case where the LTL formula φ has one accepting pair in the Rabin acceptance condition. In stochastic Rabin(1) games, where $\text{Acc} = \{(C, B)\}$, the objective of the controller is to repeatedly visit some states in B and visit the states in C for only finitely many times. On the other hand, the environment's goal is to prevent this from happening, which can be also expressed as a Rabin condition with two accepting pairs $\text{Acc}' = \{(\emptyset, C), (B, S)\}$. Thus, pure and memoryless strategies suffice for both players on the considered product game [27].

To find a control strategy that satisfies (6) for stochastic Rabin(1) games using RL, 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 proposition summarizes our key results.

Theorem 1: Consider a given turn-based stochastic Rabin(1) product game \mathcal{G} and the return of any path σ defined as

$$G_t(\sigma) := \sum_{i=0}^{\infty} R_B(\sigma[t+i]) \cdot \prod_{j=0}^{i-1} \Gamma_{B,C}(\sigma[t+j]), \quad (7)$$

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

$$R_B(s) := \begin{cases} 1 - \gamma_B & \text{if } s \in B \\ 0 & \text{if } s \notin B \end{cases}, \quad (8)$$

$$\Gamma_{B,C}(s) := \begin{cases} \gamma_B & \text{if } s \in B \\ \gamma_C & \text{if } s \in C \\ \gamma & \text{if } s \in S \setminus (B \cup C) \end{cases}. \quad (9)$$

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

$$\lim_{\gamma \rightarrow 1^-} \frac{1 - \gamma}{1 - \gamma_B(\gamma)} = \lim_{\gamma \rightarrow 1^-} \frac{1 - \gamma_B(\gamma)}{1 - \gamma_C(\gamma)} = 0. \quad (10)$$

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

$$\lim_{\gamma \rightarrow 1^-} v_{\mu,\nu}^\gamma(s) = Pr_{\mu,\nu}^{\mathcal{G}}(s \models \varphi_{B,C}); \quad (11)$$

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

Before proving Theorem 1, we show Lemma 1, later used to establish bounds on the states' values. Intuitively, Lemma 1 shows 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 σ and a fixed $t \geq 0$, in a stochastic game with the path return defined as in (7), it holds that

$$\gamma_C G_{t+1}(\sigma) \leq \gamma G_{t+1}(\sigma) \leq G_t(\sigma) \leq 1 - \gamma_B + \gamma_B G_{t+1}(\sigma), \quad (12)$$

$$0 \leq G_t(\sigma) \leq 1. \quad (13)$$

Proof: It holds that $0 \leq \gamma_C G_{t+1}(\sigma) \leq \gamma G_{t+1}(\sigma)$ since the rewards are nonnegative and $\gamma > \gamma_C$. Now, let us assume that we do not discount in the states that do not belong to B , i.e., we replace γ and γ_C with 1 in $G_t(\sigma)$ as

$$\Gamma'_B(s) = \begin{cases} \gamma_B & \text{if } s \in B \\ 1 & \text{if } s \notin B \end{cases}.$$

Then the return

$$G'_t(\sigma) = \sum_{i=0}^{\infty} R_B(\sigma[t+i]) \prod_{j=0}^{i-1} \Gamma'_B(\sigma[t+j])$$

is evidently larger than $G_t(\sigma)$. Furthermore, it holds that $G_t(\sigma) \leq G'_t(\sigma) \leq 1 - \gamma_B(b+1) \leq 1$, where b is the number of times a B state is visited; thus, the return of any path is bounded by 1 from above.

From the path return definition (7), it holds that

$$G_t(\sigma) = \begin{cases} 1 - \gamma_B + \gamma_B G_{t+1}(\sigma) & \text{if } \sigma[t] \in B \\ \gamma_C G_{t+1}(\sigma) & \text{if } \sigma[t] \in C \\ \gamma G_{t+1}(\sigma) & \text{otherwise} \end{cases}. \quad (14)$$

Using $\gamma_B < \gamma$ and $G_{t+1}(\sigma) \leq 1$, we obtain that

$$1 - \gamma_B + \gamma_B G_{t+1}(\sigma) \geq 1 - \gamma + \gamma G_{t+1}(\sigma) \geq \gamma G_{t+1}(\sigma), \quad (15)$$

which along with (14) concludes the proof of (12), (13). ■

Under a strategy pair (μ, ν) , it is straightforward to check the probability that a Rabin condition is satisfied in a game \mathcal{G} (i.e., MC $\mathcal{G}_{\mu, \nu}$). All paths in the induced MC $\mathcal{G}_{\mu, \nu}$ eventually reach a BSCC $T \in \mathcal{B}(\mathcal{G}_{\mu, \nu})$ and visit its states infinitely many times. A path reaching a state in a BSCC that does not contain any state in B or C , does not satisfy the Rabin condition. We denote the set of all such states by $U_{\overline{BC}}$. Similarly, if a path reaches a state in a BSCC without any state in C but with a state in B , it satisfies the Rabin condition; finally, if it reaches a state in a BSCC that does contain a state from C , it does not satisfy the Rabin condition. We write U_B and U_C to denote the set of these states, respectively (formally defined in Lemma 2). This reasoning reduces finding the probability of satisfying the Rabin condition to finding the probability of reaching a state in U_B , which allows us to focus on the reachability objective $\varphi_{U_B} := \diamond U_B$ instead of $\varphi_{B,C}$ defined in Theorem 1.

We now show that the expected values of the returns (7) (i.e., the state values) reflect the Rabin acceptance condition.

Lemma 2: For any stochastic Rabin game \mathcal{G} with $\text{Acc} = \{(C, B)\}$ under a strategy pair (μ, ν) , it holds that:

$$\lim_{\gamma \rightarrow 1^-} v_{\mu, \nu}^\gamma(s) = 0 \text{ if } s \in U_{\overline{BC}}, \quad (16)$$

$$\lim_{\gamma \rightarrow 1^-} v_{\mu, \nu}^\gamma(s) = 1 \text{ if } s \in U_B, \quad (17)$$

$$\lim_{\gamma \rightarrow 1^-} v_{\mu, \nu}^\gamma(s) = 0 \text{ if } s \in U_C, \quad (18)$$

where the sets $U_{\overline{BC}}$, U_B and U_C are defined as:

$$\begin{aligned} U_{\overline{BC}} &:= \{s_{\overline{BC}} \mid s_{\overline{BC}} \in T, T \in \mathcal{B}(\mathcal{G}_{\mu, \nu}), T \cap B = \emptyset, T \cap C = \emptyset\}, \\ U_B &:= \{s_B \mid \exists T \in \mathcal{B}(\mathcal{G}_{\mu, \nu}), s_B \in T \cap B, T \cap C = \emptyset\}, \\ U_C &:= \{s_C \mid \exists T \in \mathcal{B}(\mathcal{G}_{\mu, \nu}), s_C \in T \cap C\}. \end{aligned} \quad (19)$$

Proof: To simplify our notation, in this proof and the proof of Theorem 1, we use v^γ , Γ and Pr instead of $v_{\mu, \nu}^\gamma$, $\Gamma_{B,C}$ and $Pr_{\mu, \nu}^\mathcal{G}$. We define a return for a finite path as:

$$G_{t:t+k}(\sigma) := \sum_{i=0}^k R_B(\sigma[t+i]) \prod_{j=0}^{i-1} \Gamma_{B,C}(\sigma[t+j]) \quad (20)$$

Case I: Once a state $s_{\overline{BC}} \in U_{\overline{BC}}$ is reached, it is impossible to later visit a B state and receive a nonzero reward; thus, the values of all states in $U_{\overline{BC}}$ are zero and (16) holds.

Case II: Once a state $s_B \in U_B$ is reached, it will be visited infinitely often as it belongs to a BSCC. Let N_t be the time to the next visit to s_B after visiting it at t – i.e.,

$$N_t = \min\{\tau \mid \sigma[t+\tau] = s_B, \tau > 0\}. \quad (21)$$

From the definition of the return (7), it holds that

$$\begin{aligned} v^\gamma(s_B) &= 1 - \gamma_B + \gamma_B \mathbb{E}[G_{t+1}(\sigma) \mid \sigma[t] = s_B] = \\ &= 1 - \gamma_B + \gamma_B \mathbb{E}[G_{t+1:t+N_t-1}(\sigma) \\ &+ \left(\prod_{i=1}^{N_t-1} \Gamma(\sigma[t+i]) \right) \cdot G_{t+N_t}(\sigma) \mid \sigma[t] = s_B]. \end{aligned} \quad (22)$$

We can ignore the return of the prefix, $G_{t+1:t+N_t-1}(\sigma)$ and obtain an upper bound. Using $G_t(\sigma) \geq \gamma G_{t+1}(\sigma)$, we have

$$\begin{aligned} v^\gamma(s_B) &\geq 1 - \gamma_B + \gamma_B \mathbb{E}[\gamma^{N_t-1} G_{t+N_t}(\sigma) \mid \sigma[t] = s_B] \\ &\stackrel{\textcircled{1}}{\geq} 1 - \gamma_B + \gamma_B \mathbb{E}[\gamma^{N_t-1} \mid \sigma[t] = s_B] v^\gamma(s_B) \\ &\stackrel{\textcircled{2}}{\geq} 1 - \gamma_B + \gamma_B \gamma^{\mathbb{E}[N_t-1 \mid \sigma[t] = s_B]} v^\gamma(s_B) \\ &\geq 1 - \gamma_B + \gamma_B \gamma^n v^\gamma(s_B) \end{aligned} \quad (23)$$

where $\textcircled{1}$ holds by the Markov property, $\textcircled{2}$ holds from the Jensen's inequality, and $n \geq 1$ is a constant. From (23),

$$\begin{aligned} v^\gamma(s_B) &\geq \frac{1 - \gamma_B}{1 - \gamma_B \gamma^n} \stackrel{\textcircled{3}}{\geq} \frac{1 - \gamma_B}{1 - \gamma_B (1 - n(1 - \gamma))} = \\ &= \frac{1}{1 + n \frac{1 - \gamma}{1 - \gamma_B} - n(1 - \gamma)}. \end{aligned} \quad (24)$$

where $\textcircled{3}$ holds as $(1 - (1 - \gamma))^n \geq 1 - n(1 - \gamma)$ for $\gamma \in (0, 1)$. Finally, as $v^\gamma(s_B) \leq 1$ from Lemma 1, letting $\gamma, \gamma_B \rightarrow 1^-$ under the condition (10), concludes the proof of (17).

Case III: Similarly to the previous case, we define a stopping time for the number of time steps between two consecutive visits to a state $s_C \in U_C$ – i.e., for a fixed $t \in \mathbb{N}$

$$M_t = \min\{\tau \mid \sigma[t+\tau] = s_C, \tau > 0\}. \quad (25)$$

Now, we can split the value of s_C into two expectations

$$\begin{aligned} v^\gamma(s_C) &= \gamma_C \mathbb{E}[G_{t+1:t+M_t-1}(\sigma) \mid \sigma[t] = s_C] + \\ &\gamma_C \mathbb{E}\left[\left(\prod_{i=1}^{M_t-1} \Gamma(\sigma[t+i])\right) G_{t+M_t}(\sigma) \mid \sigma[t] = s_C\right]. \end{aligned} \quad (26)$$

Using the inequalities in Lemma 1, it holds that

$$\begin{aligned} v^\gamma(s_C) &\leq \gamma_C \mathbb{E}[1 - \gamma_B^{M_t}] + \gamma_C \mathbb{E}[G_{t+M_t}(\sigma) \mid \sigma[t] = s_C] \\ &\stackrel{\textcircled{1}}{\leq} \gamma_C (1 - \gamma_B^m) + \gamma_C \mathbb{E}[G_{t+M_t}(\sigma) \mid \sigma[t] = s_C] \\ &\leq 1 - \gamma_B^m + \gamma_C v_{\mu, \nu}^\gamma(s_C). \end{aligned} \quad (27)$$

where $\textcircled{1}$ holds by Jensen's inequality and $m \geq 0$ is a constant. Now, from (27) it holds that

$$v^\gamma(s_C) \leq \frac{1 - \gamma_B^m}{1 - \gamma_C} \leq \frac{m(1 - \gamma_B)}{1 - \gamma_C}. \quad (28)$$

Thus, from (10), since $v^\gamma(s_C)$ is nonnegative, (18) holds. ■

We now provide the proof of Theorem 1.

Proof: [Proof of Theorem 1] We divide the expected return of a random path σ visiting a state $s \in S$ depending on whether it satisfies $\varphi_{U_B} := \diamond U_B$ or not – i.e.,

$$\begin{aligned} v^\gamma(s) &= \mathbb{E}[G_t(\sigma) \mid \sigma[t] = s, \sigma \models \diamond U_B] Pr(\sigma \models \diamond U_B) \\ &+ \mathbb{E}[G_t(\sigma) \mid \sigma[t] = s, \sigma \not\models \diamond U_B] Pr(\sigma \not\models \diamond U_B) \end{aligned} \quad (29)$$

for some fixed $t \in \mathbb{N}$. Notice that $\sigma \not\models \diamond U_B$ implies $\sigma[t:] \not\models \diamond U_B$, and $\sigma \models \diamond U_B$ implies $\sigma[t:] \models \diamond U_B$ almost surely. Hence, $Pr(s \models \diamond U_B)$ and $Pr(s \not\models \diamond U_B)$ can be replaced with $Pr(\sigma \models \diamond U_B)$ and $Pr(\sigma \not\models \diamond U_B)$, 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 (19) – i.e.,

$$L_t = \min\{\tau \mid \sigma[t+\tau] \in U_B, \tau > 0\}. \quad (30)$$

Then, by Lemma 1, it holds that

$$\begin{aligned}
v^\gamma(s) &\geq \mathbb{E}[G_t(\sigma) \mid \sigma[t]=s, \sigma \models \Diamond U_B] Pr(s \models \Diamond U_B) \\
&\geq \mathbb{E}[\gamma^{L_t} G_{t+L_t}(\sigma) \mid \sigma[t]=s, \sigma \models \Diamond U_B] Pr(s \models \Diamond U_B) \\
&\stackrel{\textcircled{1}}{\geq} \mathbb{E}[\gamma^{L_t} \mid \sigma[t]=s, \sigma \models \Diamond U_B] \underline{v}^\gamma(U_B) Pr(s \models \Diamond U_B) \\
&\stackrel{\textcircled{2}}{\geq} \gamma^{\mathbb{E}[L_t \mid \sigma[t]=s, \sigma \models \Diamond U_B]} \underline{v}^\gamma(U_B) Pr(s \models \Diamond U_B) = \\
&= \gamma^l \underline{v}^\gamma(U_B) Pr(s \models \Diamond U_B); \tag{31}
\end{aligned}$$

here, $\underline{v}^\gamma(U_B) = \min_{s_B \in U_B} v^\gamma(s_B)$, l is constant, and $\textcircled{1}$ and $\textcircled{2}$ 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 $U_{\overline{BC}} \cup U_C$ is reached – i.e.,

$$L'_t = \min \{ \tau \mid \sigma[t+\tau] \in U_{\overline{BC}} \cup U_C, \tau > 0 \}. \tag{32}$$

Then, using Lemma 1 and the Markov property, it holds that

$$\begin{aligned}
v^\gamma(s) &\leq \mathbb{E}[G_t(\sigma) \mid \sigma[t]=s, \sigma \not\models \Diamond U_B] Pr(s \not\models \Diamond U_B) \\
&\quad + Pr(s \models \Diamond U_B) \\
&\leq \mathbb{E}[1 - \gamma^{L'_t} \mid \sigma[t]=s, \sigma \not\models \Diamond U_B] Pr(s \not\models \Diamond U_B) \\
&\quad + Pr(s \models \Diamond U_B) \\
&\leq 1 - \gamma_B^{\mathbb{E}[L'_t \mid \sigma[t]=s, \sigma \not\models \Diamond U_B]} Pr(s \not\models \Diamond U_B) \\
&\quad + Pr(s \models \Diamond U_B) \\
&= (1 - \gamma_B^{l'}) Pr(s \not\models \Diamond U_B) + Pr(s \models \Diamond U_B), \tag{33}
\end{aligned}$$

where l' is some constant. The upper bound (33) and the lower bound (31) (due to (17)) approach the probability $Pr(s \models \Diamond U_B)$ as $\gamma \rightarrow 1^-$, thereby concluding the proof. ■

C. Reduction to Stochastic Rabin(1) Games

We now provide a generalization of our approach from Section III-B to the Rabin conditions with k pairs. The idea is to construct k different stochastic Rabin(1) games and connect them with ε 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 $\{1, 2, \dots, n\}$ for a positive integer n . For a given stochastic Rabin(k) game $\mathcal{G} = (S, (S_\mu, S_\nu), A, P, s_0, \text{Acc})$, with $\text{Acc} = \{(C_i, B_i)\}_{i=0}^k$, a k -copy game $\mathcal{G}^* = (S^*, (S_\mu^*, S_\nu^*), A^*, P^*, s_0^*, \text{Acc}^*)$ is a stochastic Rabin(1) game defined by:

- $S^* = (S_\mu \times [2k]) \cup (S_\nu \times [k])$ is the augmented state set with $S_\mu^* = S_\mu \times [k]$ the controller and $S_\nu^* = S \setminus S_\mu$ the environment states, and $s_0^* = \langle s_0, 1 \rangle$ is the initial state;
- $A^* = A \cup \{\varepsilon_i \mid i \in [k]\} \cup \{\varepsilon'\}$ is the set of actions;
- $P^* : S^* \times A^* \times S^* \rightarrow [0, 1]$ is the transition function defined as $P^*(\langle s, i \rangle, a, \langle s', i' \rangle)$

$$= \begin{cases} P(s, a, s') & \text{if } a \in A, i = i', \\ 1 & \text{if } s \in S_\mu, s = s', a = \varepsilon_i, i' = k + i, \\ 1 & \text{if } s \in S_\nu, s = s', a = \varepsilon', i' = i - k, \\ 0, & \text{otherwise;} \end{cases}$$

- $\text{Acc}^* = \{(C^*, B^*)\}$ is the Rabin accepting set where $C^* := \{\langle s, i \rangle \mid s \in C_i, i \in [k] \text{ or } s \in S_\mu, i \in [2k] \setminus [k]\}$, $B^* := \{\langle s, i \rangle \mid s \in B_i, i \in [k]\}$.

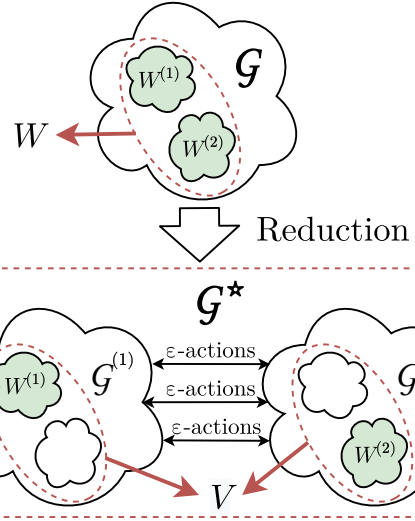


Fig. 2: A k -copy game obtained from a stochastic Rabin(2) game. $W^{(1)}$ and $W^{(2)}$ denote the winning sets for the first and the second Rabin pair respectively.

Intuitively, the k -copy game \mathcal{G}^* consists of k exact copies of the original game \mathcal{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 $i \in [k]$ (Fig. 2). The controller can choose an ε_j in a state $\langle s, i \rangle$ and makes a transition to the dummy environment state $\langle s, j+k \rangle$ where the environment can only take the action ε' , 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 ε -actions so that in any state, the controller can jump to the j -th copy via an $\varepsilon_j \rightarrow$ a-dummy-state $\rightarrow \varepsilon'$ sequence. All the dummy states belong to C^* , prohibiting the ε -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 $\mathcal{G}^{(j)}$ be the stochastic Rabin(1) game obtained from a Rabin(k) game \mathcal{G} by replacing Acc with $\{(C_j, B_j)\}$, and $W^{(j)}$ be the set of winning states such that for any $s \in W^{(j)}$, $Pr_{\mathcal{G}^{(j)}}(s \models \varphi_{B_j, C_j}) = 1$. Then, for any $\langle s, i \rangle \in S^*$, it holds that

$$Pr_{\mathcal{G}^*}(\langle s, i \rangle \models \varphi_{B^*, C^*}) = Pr_{\mathcal{G}^*}(\langle s, i \rangle \models \Diamond V), \tag{34}$$

where $V = \{\langle s', i' \rangle \in S^* \mid s' \in \bigcup_{j=0}^k W^{(j)}\}$.

Proof: We prove (34) in two directions.

\geq : If a state $\langle s, i \rangle \in V$, then, by definition, there must be some j such that $s \in W^{(j)}$. The controller can make a transition from $\langle s, i \rangle$ to $\langle s, j \rangle$ via the ε -actions and satisfy φ_{B^*, C^*} by satisfying φ_{B_j, C_j} . Thus, the control strategy that maximizes the reachability probabilities in the worst case also guarantees that the satisfaction probabilities are at least the maximin reachability probabilities, i.e., $Pr_{\mathcal{G}^*}(\langle s, i \rangle \models \varphi_{B^*, C^*}) \geq Pr_{\mathcal{G}^*}(\langle s, i \rangle \models \Diamond V)$.

\leq : All the transitions via the ε -actions pass through a state in

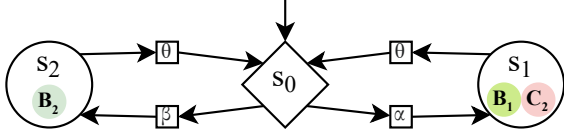


Fig. 3: A stochastic Rabin game with two accepting pairs: $(C_1, B_1) = (\emptyset, \{s_1\})$ and $(C_2, B_2) = (\{s_1\}, \{s_2\})$. The initial state s_0 is controlled by the environment; s_1 and s_2 are the controller states; α , β and θ are actions; and all transition probabilities are 1.

C^* . Under any strategy pair, the BSCCs having ε -transitions of the induced MC are rejecting. Since without some ε -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 maximizing 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 $Pr_{*}^{\mathcal{G}^*}(\langle s, i \rangle \models \varphi_{B^*, C^*}) \leq Pr_{*}^{\mathcal{G}^*}(\langle s, i \rangle \models \Diamond V)$. ■

Any control strategy μ^* in \mathcal{G}^* has a corresponding finite-memory strategy μ in the Rabin(k) game \mathcal{G} , which can be captured by a deterministic finite automaton (DFA) with k states. In state s , the state of the DFA changes from state $i \in [k]$ to $j \in [k]$, if $\mu^*(\langle s, i \rangle) = \varepsilon_j$; the DFA state stays the same and the control strategy μ chooses action $a \in A$ if $\mu^*(\langle s, i \rangle) = a$. If μ^* is a maximin strategy for \mathcal{G}^* then under the induced strategy μ , the controller satisfies the acceptance condition with probability that is not lower than the probability of reaching a winning state of an accepting pair.

Corollary 1: A maximin control strategy for \mathcal{G}^* of a stochastic Rabin game \mathcal{G} with k accepting pairs induces a control strategy μ for \mathcal{G} such that

$$Pr_{\mu, \nu}(\mathcal{G} \models \varphi_{\text{Acc}}) \geq Pr_{*}(\mathcal{G} \models \Diamond W) \quad (35)$$

for any environment strategy ν , where

$$\varphi_{\text{Acc}} := \bigvee_{(B_i, C_i) \in \text{Acc}} (\Box \Diamond B_i \wedge \neg \Box \Diamond C_i), \quad W := \bigcup_{i=1}^k W^{(i)}$$

with $W^{(i)}$ defined as in Theorem 2.

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

The induced control strategy μ guarantees a satisfaction probability that is larger than or equal to $Pr_{*}(\mathcal{G} \models \Diamond W)$. However, for some stochastic games, there exist some control strategies with improved lower bounds. This is due to the fact that there could be other winning states not belonging to W . One such game is illustrated in Fig. 3. In this game, if the environment chooses the action β all the time,

the generated path does not satisfy the first accepting pair; similarly, if it chooses α , the path does not satisfy the second accepting pair. Thus, neither s_1 nor s_2 belong to W of \mathcal{G} ; yet, they are winning states since independently from the used environment strategy, an accepting pair is always satisfied.

D. Controller Synthesis via Reinforcement Learning

We now state the main result of our approach.

Theorem 3: For a given stochastic Rabin(k) game \mathcal{G} , there exists a γ' such that for all γ , such that $\gamma' < \gamma < 1$, a minimax-Q learning algorithm using the reward and the discount functions in Theorem 1 is guaranteed to converge to a strategy μ satisfying (35).

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 [27]. ■

For a given stochastic game and an LTL specification, we can reduce the control synthesis problem to finding a maximin controller strategy in a stochastic Rabin(k) game \mathcal{G} using the standard automata-based approach described in Section III-A. This can be 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 \mathcal{G}^* using the method from Section III-C. In Section III-B, we provide an approach that transforms the objective of satisfying a Rabin pair to a discounted reward maximization objective, allowing the use of RL to synthesize controllers.

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

Computing the winning states in stochastic Rabin games is NP-Complete in the number accepting pairs [27]. 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. Instead, we provide a simple yet powerful approach combining all the accepting pairs that only requires a linear increase in the number of states.

IV. EXPERIMENTAL RESULTS

We implemented a software tool [28] in *Python* that takes a description of a labeled stochastic game and an LTL specification of a task, and outputs the desired control strategy using RL. Our tool uses *Rabinizer 4* [29] to translate the given LTL formula into a DRA then constructs the product game of the given game and the DRA. It performs minimax-Q learning using the presented reward and discount functions.

Algorithm 1 Model-free RL for control synthesis in stochastic games from LTL specifications.

Input: LTL formula φ , stochastic game \mathcal{G}
 Translate φ to a DRA \mathcal{A}_φ
 Construct the product \mathcal{G}^\times of \mathcal{G} and \mathcal{A}_φ
 Reduce \mathcal{G}^\times to \mathcal{G}^*
 Initialize $Q(s, a)$ on \mathcal{G}^*
for $t = 0, 1, \dots, T$ **do**
 Derive an ϵ -greedy strategy pair (μ^*, ν^*) 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}
 $Q(s_t, a_t) \leftarrow (1 - \alpha)Q(s_t, a_t) + \alpha R(s_t)$
 $+ \alpha \Gamma(s_t) \cdot \begin{cases} \max_{a'} Q(s_{t+1}, a'), & s_{t+1} \in S_\mu^* \\ \min_{a'} Q(s_{t+1}, a'), & s_{t+1} \in S_\nu^* \end{cases}$
end for
 Get a greedy control strategy μ_*^* from Q
return μ_*^*

During learning, ϵ -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 ϵ and the learning rate α 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 on two different scenarios in two-dimensional (5×5) grid worlds. The controller navigates a robot which occupies one cell and can move to adjacent four cells in a single time step using four actions: *North*, *South*, *East* and *West*. There are three types of cells: empty cells, cells with an obstacle and absorbing cells. 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. Each cell is labeled with a set of atomic propositions, depicted as encircled letters in the lower part of the cells.

A. Robust Control Design

In our first case study, the robot can unpredictably move in a direction that is orthogonal to the intended direction after taking an action. 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. The controller, in this case, tries to come up with a robust and conservative strategy that maximizes the probability of achieving the task in the worst case.

The environment can reactively choose one of the following four actions: *None*, *Both*, *Right* and *Left* (Fig. 4). For *None*, the robot moves in the intended direction without any disturbance. If *Both* is chosen, it can go sideways with a probability of 0.2 (i.e., 0.1 for each direction). If *Right* (*Left*) is chosen, the robot moves as intended with probability of 0.9 and in right (left) direction with probability of 0.1.

The objective is to visit a state labeled with b and a state

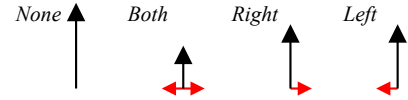
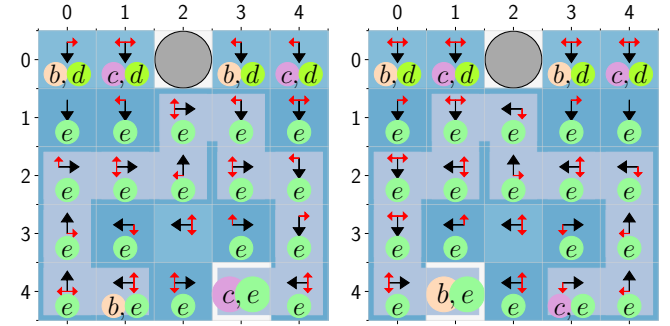


Fig. 4: Illustration of the actions of the environment for the control action *North*. The arrow lengths are loosely proportional to the probabilities of the movement directions.



(a) Strategy from b to c

(b) Strategy from c to b

Fig. 5: The objective strategy and the estimated maximal probabilities of satisfying φ_1 from (36). The most likely path is highlighted in a lighter blue.

labeled with c infinitely often, and the safe states, labeled with either d or e , should not be left after a certain point of time. The task is formally described by the LTL formula

$$\varphi_1 = \square \diamond b \wedge \square \diamond c \wedge (\diamond \square d \vee \diamond \square e), \quad (36)$$

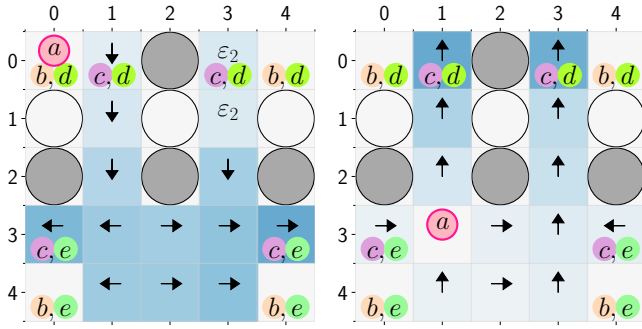
which we translated into a DRA with two accepting pairs.

Fig. 5 depicts the grid world we used and the strategy obtained for it after 128K episodes. The objective cannot be satisfied by visiting the states (labeled with b and c) at the top-left or the top-right corner of the grid because the environment can force the robot to leave the safe states. The only possible 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. The strategy in Fig. 5 ensures that the robot stays below the second row once it reaches the area, and does not visit the unsafe state in the middle regardless of the environment's actions. Also, under the strategies in Fig. 5a, 5b, the robot eventually reaches the states labeled with b and c , respectively.

B. Avoiding Adversary

In this case study, the robot movement is not affected by the environment actions. Instead, we model the imprecise movement by a fixed probability distribution – 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 is controlled by an adversary (the environment), which can take the same four actions as the controller, with the same probability distribution.

The size of the state space here is $(5 \times 5) \times (5 \times 5) = 625$ since there are two independent agents. The labeling function



(a) Adversary is at (0, 0) and $i=1$ (b) Adversary is at (3, 1) and $i=2$

Fig. 6: The control strategies obtained for φ_2 from (38). The values of the states are represented by the shades of blue (the darker, the higher value), which are the estimation of how likely the controller satisfies the objective.

is based on the position of the first agent as

$$L(\langle s_1, s_2 \rangle) := \begin{cases} L(s_1), & s_1 \neq s_2 \\ L(s_1) \cup \{a\}, & s_1 = s_2 \end{cases} \quad (37)$$

The label a represents the state where both agents are in the same position (adversary ‘catches’ the robot). The robot objective is the same as in the first scenario (36), except that it additionally needs to avoid the adversary at all costs – i.e.,

$$\varphi_2 = \varphi_1 \wedge \Box \neg a. \quad (38)$$

Fig. 6 shows the control strategy obtained after 512K episodes. There are four safe zones in this grid world: one at the top-left, another 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 strategy for the controller is not to switch zones unless the adversary is in the same zone. For example, in Fig. 6a, 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 she 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 ε_2 and make the robot stay in the same zone. However, in Fig. 6b, the robot cannot stay in the bottom part because otherwise the adversary will eventually catch her.

V. CONCLUSIONS

In this paper, we introduced an RL-based approach for synthesis of controllers from LTL specifications in stochastic games. We first provided a reduction from this synthesis problem to the problem of finding a control strategy in a stochastic Rabin game with a single accepting pair. We introduced a rewarding and discounting mechanism that transforms the objective of maximizing the (minimal/worst-case) probability of satisfying the Rabin condition into the objective of maximizing the discounted reward, and introduced an RL algorithm to find such a policy. We then

provided a reduction allowing us to generalize our approach to any LTL specification, with the Rabin condition having $k > 1$ accepting pairs, with a lower bound on the satisfaction probabilities. Finally, we showed the applicability of our approach on two case path planning case-studies.

REFERENCES

- [1] Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, USA, 2nd edition, 2018.
- [2] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.
- [3] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.
- [4] Tommaso Dreossi, Alexandre Donzé, and Sanjit A. Seshia. Compositional Falsification of Cyber-Physical Systems with Machine Learning Components. In Clark Barrett, Misty Davies, and Temesghen Kahsai, editors, *NASA Formal Methods*, Lecture Notes in Computer Science, pages 357–372. Springer International Publishing, 2017.
- [5] Xiaowu Sun, Haitham Khedr, and Yasser Shoukry. Formal Verification of Neural Network Controlled Autonomous Systems. In *Proceedings of the 22Nd ACM International Conference on Hybrid Systems: Computation and Control*, HSCC ’19, pages 147–156, New York, NY, USA, 2019. ACM.
- [6] Hoang-Dung Tran, Feiyang Cai, Manzan Lopez Diego, Patrick Musau, Taylor T. Johnson, and Xenofon Koutsoukos. Safety Verification of Cyber-Physical Systems with Reinforcement Learning Control. *ACM Transactions on Embedded Computing Systems*, 18(5s):1–22, October 2019.
- [7] Shakiba Yaghoubi and Georgios Fainekos. Gray-box Adversarial Testing for Control Systems with Machine Learning Components. In *Proceedings of the 22Nd ACM International Conference on Hybrid Systems: Computation and Control*, HSCC ’19, pages 179–184, New York, NY, USA, 2019. ACM.
- [8] Mojtaba Zarei, Yu Wang, and Miroslav Pajic. Statistical verification of learning-based cyber-physical systems. In *ACM International Conference on Hybrid Systems: Computation and Control (HSCC)*, pages 1–7, Sydney, Australia, October 2020.
- [9] Richard Cheng, Gábor Orosz, Richard M. Murray, and Joel W. Burdick. End-to-End Safe Reinforcement Learning through Barrier Functions for Safety-Critical Continuous Control Tasks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):3387–3395, July 2019.
- [10] Andrew Taylor, Andrew Singletary, Yisong Yue, and Aaron Ames. Learning for Safety-Critical Control with Control Barrier Functions. In *Learning for Dynamics and Control*, pages 708–717. PMLR, July 2020.
- [11] Yu Wang and Miroslav Pajic. Hyperproperties for robotics: Motion planning via HyperLTL. In *IEEE International Conference on Robotics and Automation (ICRA)*, page accepted, Paris, France, 2020.
- [12] Andrew G Barto. Reinforcement learning: Connections, surprises, challenges. *AI Magazine*, 40(1), 2019.
- [13] Hadas Kress-Gazit, Georgios E Fainekos, and George J Pappas. Translating structured English to robot controllers. *Advanced Robotics*, 22(12):1343–1359, 2008.
- [14] 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.
- [15] Tichakorn Wongpiromsarn, Ufuk Topcu, and Richard M Murray. Receding horizon temporal logic planning. *IEEE Transactions on Automatic Control*, 57(11):2817–2830, 2012.
- [16] Alper Kamil Bozkurt, Yu Wang, Michael M Zavlanos, and Miroslav Pajic. Control synthesis from linear temporal logic specifications using model-free reinforcement learning, 2019. arXiv:1909.07299 [cs.LG].
- [17] Mohammadhossein Hasanbeig, Yiannis Kantaros, Alessandro Abate, Daniel Kroening, George J Pappas, and Insup Lee. Reinforcement learning for temporal logic control synthesis with probabilistic satisfaction guarantees, 2019. arXiv:1909.05304 [cs.LG].

- [18] Ernst Moritz Hahn, Mateo Perez, Sven Schewe, Fabio Somenzi, Ashutosh Trivedi, and Dominik Wojtczak. Omega-regular objectives in model-free reinforcement learning. In Tomáš Vojnar and Lijun Zhang, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 395–412, Cham, 2019. Springer International Publishing.
- [19] Thomas A Henzinger and Nir Piterman. Solving games without determinization. In *International Workshop on Computer Science Logic*, pages 395–410. Springer, 2006.
- [20] 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.
- [21] Pranav Ashok, Jan Křetínský, 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.
- [22] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. MIT Press, Cambridge, MA, USA, 2008.
- [23] Lloyd S Shapley. Stochastic games. *Proceedings of the national academy of sciences*, 39(10):1095–1100, 1953.
- [24] Michael L Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, pages 157–163. Elsevier, 1994.
- [25] Junling Hu and Michael P Wellman. Nash q-learning for general-sum stochastic games. *Journal of machine learning research*, 4(Nov):1039–1069, 2003.
- [26] 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.
- [27] Krishnendu Chatterjee and Thomas A. Henzinger. A survey of stochastic ω -regular games. *Journal of Computer and System Sciences*, 78(2):394 – 413, 2012. Games in Verification.
- [28] A. K. Bozkurt, Y. Wang, M. M. Zavlanos, and M. Pajic. CSRL, 2020. <https://gitlab.oit.duke.edu/cpsl/csrl>.
- [29] Jan Křetínský, Tobias Meggendorfer, Salomon Sickert, and Christopher Ziegler. Rabinizer 4: from LTL to your favourite deterministic automaton. In *International Conference on Computer Aided Verification*, pages 567–577. Springer, 2018.