Reinforcement Learning of Control Policy for Linear Temporal Logic Specifications Using Limit-Deterministic Generalized Büchi Automata

Ryohei Oura, Ami Sakakibara, and Toshimitsu Ushio

Abstract— **This letter proposes a novel reinforcement learning method for the synthesis of a control policy satisfying a control specification described by a linear temporal logic formula. We assume that the controlled system is modeled by a Markov decision process (MDP). We transform the specification to a limit-deterministic generalized Büchi automaton (LDGBA) with several accepting sets that accepts all infinite sequences satisfying the formula. The LDGBA is augmented so that it explicitly records the previous visits to accepting sets. We take a product of the augmented LDGBA and the MDP, based on which we define a reward function. The agent gets rewards whenever state transitions are in an accepting set that has not been visited for a certain number of steps. Consequently, sparsity of rewards is relaxed and optimal circulations among the accepting sets are learned. We show that the proposed method can learn an optimal policy when the discount factor is sufficiently close to one.**

Index Terms— **Reinforcement Learning, Linear Temporal Logic, Limit-Deterministic Büchi Automata.**

I. Introduction

Reinforcement learning (RL) is a useful approach to learning an optimal policy from sample behaviors of the controlled system [5]. In RL, we use a reward function that assigns a reward to each transition in the behaviors and evaluate a control policy by the return that is an expected (discounted) sum of the rewards along the behaviors. Thus, to apply RL to the synthesis of a control policy for the LTL specification, it is an important issue how to introduce the reward function. Previously, a reward function based on the acceptance condition of a Rabin automaton was proposed in [4]. They converted a given LTL specification to a Rabin automaton. Then, they introduced the reward function that assigns a positive and negative value according to an acceptance pair of the Rabin automaton and made the reward functions for each acceptance pair of the Rabin automaton. The reward based on the acceptance condition of a Rabin automaton was also applied to a deep reinforcement learning methods [6]. They developed the algorithm that can estimate the gradients of an unknown loss function more accurately than the normal stochastic gradient descent methods. Consequently, they achieved a better approximation of optimal state-action value functions for LTL tasks. It is known that the LTL specification is converted into an ω-automaton such as a nondeterministic Büchi automaton and a deterministic Rabin automaton [1], [2]. When constructing a Büchi automaton (BA) from a generalized Büchi automaton (GBA), the order of visits to accepting sets of the GBA is fixed [1]. The construction causes the sparsity of the reward based on the acceptance condition of a BA. It is critical against an RL-based controller synthesis.

Recently, a limit-deterministic Büchi automaton (LDBA) or a generalized one (LDGBA) is paid much attention to as an ω-automaton corresponding to the LTL specification [7]. The RL-based approaches to the synthesis of a control policy using LDBAs or LDGBAs have been proposed in [8]–[11]. To deal with the acceptance condition of an LDGBA that accepts behaviors visiting all accepting sets infinitely often, the accepting frontier function was introduced in [8], [10]. The reward function is defined based on the function. However, the function is memoryless, that is, it does not provide the information of accepting sets that have been visited, which is important to improve learning performance. In this letter, we propose a novel method to augment an LDGBA converted from a given LTL formula. Then, we define a reward function based on the acceptance condition of the product MDP of the augmented LDGBA and the controlled system represented as the MDP. As a result, we relax the sparsity of rewards compared to using LDBAs and a policy satisfying a given LTL specification is more likely to exist than the method in [8].

The rest of the letter is organized as follows. Section II reviews an MDP, LTL, and automata. Section III proposed a novel RL-based method for the synthesis of a control policy. Section IV presents a numerical example for which the previous method cannot learn a control policy but the proposed one can.

Notations: For sets A and B, AB denotes their concatenation. A^ω denotes the infinite concatenation of the set A and A^* denotes the finite one. $\mathbb{N}_0$ is the set of nonnegative integers. $\mathbb{R}_{\geq 0}$ is the set of nonnegative real numbers.

II. Preliminaries

A. Markov Decision Process

Definition 1: A (labeled) Markov decision process (MDP) is a tuple $M = (S, A, P, s_{init}, AP, L)$, where S is a finite set of states, A is a finite set of actions, $P : S \times S \times A \rightarrow [0, 1]$ is a transition probability function, $s_{init} \in S$ is the initial state, AP is a finite set of atomic propositions, and $L : S \times A \times S \rightarrow 2^{AP}$ is a labeling function that assigns a set of atomic propositions to each transition. Let $\mathcal{A}(s) = \{a \in A; \exists s' \in S \text{ s.t. } P(s'|s, a) \neq 0\}$. Note that $\sum_{s' \in S} P(s'|s, a) = 1$ holds for any state $s \in S$ and action $a \in \mathcal{A}(s)$.

This work was partially supported by JST-ERATO HASUO Project Grant Number JPMJER1603, Japan, JST-Mirai Program Grant Number JPMJMI18B4, Japan, and JSPS KAKENHI Grant Number JP19J13487, Japan.

The authors are with the Graduate School of Engineering Science, Osaka University, Toyonaka 560-8531, Japan (e-mail: r-oura, sakakibara@hopf.sys.es.osaka-u.ac.jp; ushio@sys.es.osaka-u.ac.jp).

In the MDP M, an infinite path starting from a state $s_0 \in S$ is defined as a sequence $\rho = s_0 a_0 s_1 \ldots \in S(AS)^\omega$ such that $P(s_{i+1}|s_i, a_i) > 0$ for any $i \in \mathbb{N}_0$. A finite path is a finite sequence in $S(AS)^*$. In addition, we sometimes represent ρ as ρ_{init} to emphasize that ρ starts from $s_0 = s_{init}$. For a path $\rho = s_0 a_0 s_1 \ldots$, we define the corresponding labeled path $L(\rho) = L(s_0, a_0, s_1) L(s_1, a_1, s_2) \ldots \in (2^{AP})^\omega$. $InfPath^M$ (resp., $FinPath^M$) is defined as the set of infinite (resp., finite) paths starting from $s_0 = s_{init}$ in the MDP M. For each finite path ρ, $last(\rho)$ denotes its last state.

Definition 2: A policy on an MDP M is defined as a mapping $\pi : FinPath^M \times \mathcal{A}(last(\rho)) \to [0, 1]$. A policy π is a *positional* policy if for any $\rho \in FinPath^M$ and any $a \in \mathcal{A}(last(\rho))$, it holds that $\pi(\rho, a) = \pi(last(\rho), a)$ and there exists $a' \in \mathcal{A}(last(\rho))$ such that $\pi(\rho, a) = 1$ if $a = a'$, otherwise $\pi(\rho, a) = 0$.

Let $InfPath^M_\pi$ (resp., $FinPath^M_\pi$) be the set of infinite (resp., finite) paths starting from $s_0 = s_{init}$ in the MDP M under a policy π. The behavior of an MDP M under a policy π is defined on a probability space $(InfPath^M_\pi, \mathcal{F}_{InfPath^M_\pi}, Pr^M_\pi)$.

A Markov chain induced by an MDP M with a positional policy π is a tuple $MC_\pi = (S_\pi, P_\pi, s_{init}, AP, L)$, where $S_\pi = S$, $P_\pi(s'|s) = P(s'|s, a)$ for $s, s' \in S$ and $a \in \mathcal{A}(s)$ such that $\pi(s, a) = 1$. The state set S_π of MC_π can be represented as a disjoint union of a set of transient states T_π and closed irreducible sets of recurrent states R^j_π with $j \in \{1, \ldots, h\}$, i.e., $S_\pi = T_\pi \cup R^1_\pi \cup \ldots \cup R^h_\pi$ [12]. In the following, we say a "recurrent class" instead of a "closed irreducible set of recurrent states" for simplicity.

In an MDP M, we define a reward function $R : S \times A \times S \to \mathbb{R}_{\geq 0}$. The function denotes the immediate reward received after the agent performs an action a at a state s and reaches a next state s' as a result.

Definition 3: For a policy π on an MDP M, any state $s \in S$, and a reward function R, we define the expected discounted reward as

$$V^\pi(s) = \mathbb{E}^\pi[\sum_{n=0}^{\infty} \gamma^n R(S_n, A_n, S_{n+1}) | S_0 = s],$$

where $\mathbb{E}^\pi$ denotes the expected value given that the agent follows the policy π from the state s and $\gamma \in [0, 1)$ is a discount factor. The function $V^\pi(s)$ is often referred to as a state-value function under the policy π. For any state-action pair $(s, a) \in S \times A$, we define an action-value function $Q^\pi(s, a)$ under the policy π as follows.

$$Q^\pi(s, a) = \mathbb{E}^\pi[\sum_{n=0}^{\infty} \gamma^n R(S_n, A_n, S_{n+1}) | S_0 = s, A_0 = a].$$

Definition 4: For any state $s \in S$, a policy π^* is optimal if

$$\pi^* \in \arg\max_{\pi \in \Pi^{pos}} V^\pi(s),$$

where Π^{pos} is the set of positional policies over the state set S.

B. Linear Temporal Logic and Automata

In our proposed method, we use linear temporal logic (LTL) formulas to describe various constraints or properties and to systematically assign corresponding rewards. LTL formulas are constructed from a set of atomic propositions, Boolean operators, and temporal operators. We use the standard notations for the Boolean operators: $\top$ (true), $\neg$ (negation), and $\wedge$ (conjunction). LTL formulas over a set of atomic propositions AP are defined as

$$\varphi ::= \top \mid \alpha \in AP \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \mathbf{X}\varphi \mid \varphi_1 \mathbf{U} \varphi_2,$$

where φ, φ_1, and φ_2 are LTL formulas. Additional Boolean operators are defined as $\perp := \neg\top$, $\varphi_1 \vee \varphi_2 := \neg(\neg\varphi_1 \wedge \neg\varphi)$, and $\varphi_1 \Rightarrow \varphi_2 := \neg\varphi_1 \vee \varphi_2$. The operators **X** and **U** are called "next" and "until", respectively. Using the operator **U**, we define two temporal operators: (1) *eventually*, $\mathbf{F}\varphi := \top \mathbf{U} \varphi$ and (2) *always*, $\mathbf{G}\varphi := \neg\mathbf{F}\neg\varphi$.

Let M be an MDP. For an infinite path $\rho = s_0 a_0 s_1 \ldots$ of M with $s_0 \in S$, let $\rho[i]$ be the i-th state of ρ i.e., $\rho[i] = s_i$ and let $\rho[i:]$ be the i-th suffix $\rho[i:] = s_i a_i s_{i+1} \ldots$.

Definition 5: For an LTL formula φ, an MDP M, and an infinite path $\rho = s_0 a_0 s_1 \ldots$ of M with $s_0 \in S$, the satisfaction relation $M, \rho \models \varphi$ is recursively defined as follows. $M, \rho \models \top$; $M, \rho \models \alpha \in AP$ iff $\alpha \in L(s_0, a_0, s_1)$; $M, \rho \models \varphi_1 \wedge \varphi_2$ iff $M, \rho \models \varphi_1 \wedge M, \rho \models \varphi_2$; $M, \rho \models \neg\varphi$ iff $M, \rho \not\models \varphi$; $M, \rho \models \mathbf{X}\varphi$ iff $M, \rho[1:] \models \varphi$; and $M, \rho \models \varphi_1 \mathbf{U} \varphi_2$ iff $\exists j \geq 0,\ M, \rho[j:] \models \varphi_2 \wedge \forall i, 0 \leq i < j,\ M, \rho[i:] \models \varphi_1$. The next operator **X** requires that φ is satisfied by the next state suffix of ρ. The until operator **U** requires that φ_1 holds true until φ_2 becomes true over the path ρ. In the following, we write $\rho \models \varphi$ for simplicity without referring to MDP M.

For any policy π, the probability of all paths starting from s_{init} on the MDP M that satisfy an LTL formula φ under the policy π, or the satisfaction probability under π is defined as

$$Pr^M_\pi(s_{init} \models \varphi) := Pr^M_\pi(\{\rho_{init} \in InfPath^M_\pi; \rho_{init} \models \varphi\}).$$

We say that an LTL formula φ is satisfied by a positional policy π if

$$Pr^M_\pi(s_{init} \models \varphi) > 0.$$

Any LTL formula φ can be converted into various automata, namely finite state machines that recognize all words satisfying φ. We define a generalized Büchi automaton at the beginning, and then introduce a limit-deterministic generalized Büchi automaton [10].

Definition 6: A transition-based generalized Büchi automaton (tGBA) is a tuple $B = (X,\ x_{init},\ \Sigma,\ \delta,\ \mathcal{F})$, where X is a finite set of states, $x_{init} \in X$ is the initial state, Σ is an input alphabet including ε, $\delta \subset X \times \Sigma \times X$ is a set of transitions, and $\mathcal{F} = \{F_1, \ldots, F_n\}$ is an acceptance condition, where for each $j \in \{1, \ldots, n\}$, $F_j \subset \delta$ is a set of accepting transitions and called an accepting set. We refer to a tGBA with one accepting set as a tBA.

Let Σ^ω be the set of all infinite words over Σ and let an infinite run be an infinite sequence $r = x_0 \sigma_0 x_1 \ldots \in$

$X(\Sigma X)^{\omega}$ where $(x_i, \sigma_i, x_{i+1}) \in \delta$ for any $i \in \mathbb{N}_0$. An infinite word $w = \sigma_0\sigma_1 \ldots \in \Sigma^{\omega}$ is accepted by B_{φ} if and only if there exists an infinite run $r = x_0\sigma_0x_1 \ldots$ starting from $x_0 = x_{init}$ such that $inf(r) \cap F_j \neq \emptyset$ for each $F_j \in \mathcal{F}$, where $inf(r)$ is the set of transitions that occur infinitely often in the run r.

Definition 7: A transition-based limit-deterministic generalized Büchi automaton (tLDGBA) is a tGBA $B = (X, x_{init}, \Sigma, \delta, \mathcal{F})$ such that X is partitioned into two disjoint sets $X_{initial}$ and X_{final} such that

- $F_j \subset X_{final} \times \Sigma \times X_{final}$, $\forall j \in \{1, ..., n\}$,
- $|\{(x, \sigma, x') \in \delta; x' \in X_{final}\}| \leq 1, \forall x \in X_{final}, \forall \sigma \in \Sigma$,
- $|\{(x, \sigma, x') \in \delta; x' \in X_{initial}\}| = 0, \forall x \in X_{final}, \forall \sigma \in \Sigma$.
- there are ε-transitions into X_{final} from $X_{initial}$.

An ε-transition enables the tLDGBA to change its state with no input. Then, the ε-transitions reflect the single "guess" from $X_{initial}$ to X_{final}. Note that by the construction in [7], the transitions in each part are deterministic except for ε-transitions into X_{final} from X_{intial}. It is known that, for any LTL formula φ, there exists a tLDGBA that accepts all words satisfying φ [7]. We refer to a tLDGBA with one accepting set as a tLDBA. In particular, we represent a tLDGBA recognizing an LTL formula φ as B_{φ}, whose input alphabet is given by $\Sigma = 2^{AP} \cup \{\varepsilon\}$.

III. Reinforcement-Learning-Based Synthesis of Control Policy

We introduce an automaton augmented with binary-valued vectors. The automaton can explicitly represent whether transitions in each accepting set occur at least once, and ensure transitions in each accepting set occur infinitely often.

Let $V = \{(v_1, \ldots, v_n)^T \ ; \ v_i \in \{0, 1\}, \ i \in \{1, \ldots, n\}\}$ be a set of binary-valued vectors, and let $\mathbf{1}$ and $\mathbf{0}$ be the n-dimentional vectors with all elements 1 and 0, respectively. In order to augment a tLDGBA B_{φ}, we introduce three functions $visitf : \delta \to V$, $reset : V \to V$, and $Max : V \times V \to V$ as follows. For any $e \in \delta$, $visitf(e) = (v_1, \ldots, v_n)^T$, where $v_i = 1$ if $e \in F_i$ and $v_i = 0$ otherwise. For any $v \in V$, $reset(v) = \mathbf{0}$ if $v = \mathbf{1}$ and $reset(v) = v$ otherwise. For any $v, u \in V$, $Max(v, u) = (l_1, \ldots, l_n)^T$, where $l_i = max\{v_i, u_i\}$ for any $i \in \{1, \ldots, n\}$.

Intuitively, each vector v represents which accepting sets have been visited. The function $visitf$ returns a binary vector whose i-th element is 1 if and only if a transition in the accepting set F_i occurs. The function $reset$ returns the zero vector $\mathbf{0}$ if at least one transition in each accepting set has occurred after the latest reset. Otherwise, it returns the input vector without change.

Definition 8: For a tLDGBA $B_{\varphi} = (X, x_{init}, \Sigma, \delta, \mathcal{F})$, its augmented automaton is a tLDGBA $\bar{B}_{\varphi} = (\bar{X}, \bar{x}_{init}, \bar{\Sigma}, \bar{\delta}, \bar{\mathcal{F}})$, where $\bar{X} = X \times V$, $\bar{x}_{init} = (x_{init}, \mathbf{0})$, $\bar{\Sigma} = \Sigma$, $\bar{\delta}$ is defined as $\bar{\delta} = \{((x, v), \bar{\sigma}, (x', v')) \in \bar{X} \times \bar{\Sigma} \times \bar{X} \ ; \ (x, \bar{\sigma}, x') \in \delta, \ v' = reset(Max(v, visitf((x, \bar{\sigma}, x'))))\}$, and $\bar{\mathcal{F}} = \{\bar{F}_1, \ldots, \bar{F}_n\}$ is defined as $\bar{F}_j = \{((x, v), \bar{\sigma}, (x', v')) \in \bar{\delta} \ ; \ (x, \bar{\sigma}, x') \in F_j, \ v_j = 0\}$ for each $j \in \{1, ..., n\}$.

In the following. we call a binary-valued vector v a memory vector. It is obvious by Definition 8 that a tLDGBA and its augmented automaton accept the same language. The augmented tLDGBA $\bar{B}_{\varphi}$ keeps track of previous visits to the accepting sets of B_{φ} as a state. Intuitively, for an input word w, a memory vector v is reset to $\mathbf{0}$ when the corresponding run on a tLDGBA B_{φ} visits all accepting sets of B_{φ}. For example, shown in Figs. 1 and 2 are a tLDGBA and its augmented automaton, respectively, associated with the following LTL formula.

$$\varphi = \mathbf{GF}a \wedge \mathbf{GF}b \wedge \mathbf{G}\neg c. \tag{1}$$

The acceptance condition $\mathcal{F}$ of the tLDGBA is given by $\mathcal{F} = \{F_1, F_2\}$, where $F_1 = \{(x_0, \{a\}, x_0), \ (x_0, \{a, b\}, x_0)\}$ and $F_2 = \{(x_0, \{b\}, x_0), \ (x_0, \{a, b\}, x_0)\}$.

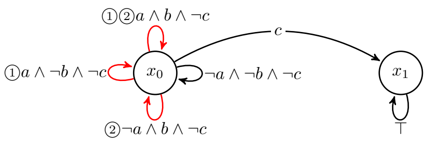
①②$a \wedge b \wedge \neg c$
c
①$a \wedge \neg b \wedge \neg c$
x_0
$\neg a \wedge \neg b \wedge \neg c$
x_1
②$\neg a \wedge b \wedge \neg c$
$\top$

Fig. 1. The tLDGBA recognizing the LTL formula $\mathbf{GF}a \wedge \mathbf{GF}b \wedge \mathbf{G}\neg c$, where the initial state is x_0. Red arcs are accepting transitions that are numbered in accordance with the accepting sets they belong to.

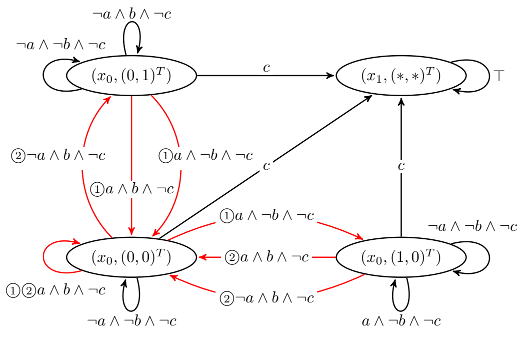
$\neg a \wedge b \wedge \neg c$
$\neg a \wedge \neg b \wedge \neg c$
$(x_0, (0,1)^T)$
c
$(x_1, (*,*)^T)$
$\top$
②$\neg a \wedge b \wedge \neg c$
①$a \wedge \neg b \wedge \neg c$
①$a \wedge b \wedge \neg c$
c
c
①$a \wedge \neg b \wedge \neg c$
$\neg a \wedge \neg b \wedge \neg c$
$(x_0, (0,0)^T)$
②$a \wedge b \wedge \neg c$
$(x_0, (1,0)^T)$
①②$a \wedge b \wedge \neg c$
②$\neg a \wedge b \wedge \neg c$
$\neg a \wedge \neg b \wedge \neg c$
$a \wedge \neg b \wedge \neg c$

Fig. 2. The augmented automaton for the tLDGBA in Fig. 1 recognizing the LTL formula $\mathbf{GF}a \wedge \mathbf{GF}b \wedge \mathbf{G}\neg c$, where the initial state is $(x_0, (0,0)^T)$. Red arcs are accepting transitions that are numbered in accordance with the accepting sets they belong to. All states corresponding to x_1 are merged into $(x_1, (*,*)^T)$.

We modify the standard definition of a product MDP to deal with ε-transitions in the augmented automaton.

Definition 9: Given an augmented tLDGBA $\bar{B}_{\varphi}$ and an MDP M, a tuple $M \otimes \bar{B}_{\varphi} = M^{\otimes} = (S^{\otimes}, A^{\otimes}, s^{\otimes}_{init}, P^{\otimes}, \delta^{\otimes}, \mathcal{F}^{\otimes})$ is a product MDP, where $S^{\otimes} = S \times \bar{X}$ is the finite set of states, $A^{\otimes}$ is the finite set of actions such that $A^{\otimes} = A \cup \{\varepsilon_{\bar{x}'}; \exists \bar{x}' \in X \text{ s.t. } (\bar{x}, \varepsilon, \bar{x}') \in \bar{\delta}\}$ where $\varepsilon_{\bar{x}'}$ is the action for the ε-transition to the state $\bar{x}' \in \bar{X}$, $s^{\otimes}_{init} = (s_{init}, \bar{x}_{init})$ is the initial states, $P^{\otimes} : S^{\otimes} \times S^{\otimes} \times A^{\otimes} \to [0, 1]$ is the transition probability defined as $P^{\otimes}(s^{\otimes\prime}|s^{\otimes}, a) = P(s'|s, a)$ if $(\bar{x}, L((s, a, s')), \bar{x}') \in \bar{\delta}$ and $a \in \mathcal{A}(s)$, $P^{\otimes}(s^{\otimes\prime}|s^{\otimes}, a) = 1$ if $s = s', (\bar{x}, \varepsilon, \bar{x}') \in \bar{\delta}$, and $a = \varepsilon_{x'}$, otherwise $P^{\otimes}(s^{\otimes\prime}|s^{\otimes}, a) = 0$ where $s^{\otimes} = (s, (x, v))$ and $s^{\otimes\prime} = (s', (x', v'))$, $\delta^{\otimes} = \{(s^{\otimes}, a, s^{\otimes\prime}) \in$

$S^{\otimes} \times A^{\otimes} \times S^{\otimes}; P^{\otimes}(s^{\otimes\prime}|s^{\otimes}, a) > 0\}$ is the set of transitions, and $\mathcal{F}^{\otimes} = \{\bar{F}_1^{\otimes}, \ldots, \bar{F}_n^{\otimes}\}$ is the acceptance condition, where $\bar{F}_i^{\otimes} = \{((s, \bar{x}), a, (s', \bar{x}')) \in \delta^{\otimes} \ ; \ (\bar{x}, L(s, a, s'), \bar{x}') \in \bar{F}_i\}$ for each $i \in \{1, \ldots, n\}$.

Definition 10: The reward function $\mathcal{R} : S^{\otimes} \times A^{\otimes} \times S^{\otimes} \to \mathbb{R}_{\geq 0}$ is defined as

$$\mathcal{R}(s^{\otimes}, a, s^{\otimes\prime}) = \begin{cases} r_p \text{ if } \exists i \in \{1, \ldots, n\}, \ (s^{\otimes}, a, s^{\otimes\prime}) \in \bar{F}_i^{\otimes}, \\ 0 \ \text{ otherwise,} \end{cases}$$

where r_p is a positive value.

Remark 1: When constructing a tBA from a tGBA, the order of visits to accepting sets of the tGBA is fixed. Consequently, the rewards based on the acceptance condition of the tBA tends to be sparse and the sparsity is critical against RL-based controller synthesis problems. The augmentation of tGBA relaxes the sparsity since the augmented tGBA has all of the order of visits to all accepting sets of the original tGBA. For the acceptance condition $\mathcal{F}$ of a tGBA, the size of the state space of the augmented tGBA is about $\frac{2^{|\mathcal{F}|}-1}{|\mathcal{F}|}$ times larger than a tBA constructed from the tGBA. However, the ratio of the number of accepting transitions to the number of all transitions in the augmented tGBA is much greater than the tBA. Therefore, our proposed method is expected to be better sample efficient than using tLDBA.

Under the product MDP $M^{\otimes}$ and the reward function $\mathcal{R}$, which is based on the acceptance condition of $M^{\otimes}$, we show that if there exists a positional policy π satisfying the LTL specification φ on $M^{\otimes}$, maximizing the expected discounted reward with γ sufficiently close to 1 produces a positional policy satisfying φ on $M^{\otimes}$.

For a Markov chain $MC_{\pi}^{\otimes}$ induced by a product MDP $M^{\otimes}$ with a positional policy π, let $S_{\pi}^{\otimes} = T_{\pi}^{\otimes} \cup R_{\pi}^{\otimes 1} \cup \ldots \cup R_{\pi}^{\otimes h}$ be the set of states in $MC_{\pi}^{\otimes}$, where $T_{\pi}^{\otimes}$ is the set of transient states and $R_{\pi}^{\otimes i}$ is the recurrent class for each $i \in \{1, \ldots, h\}$, and let $R(MC_{\pi}^{\otimes})$ be the set of all recurrent states in $MC_{\pi}^{\otimes}$. Let $\delta_{\pi}^{\otimes i}$ be the set of transitions in a recurrent class $R_{\pi}^{\otimes i}$, namely $\delta_{\pi}^{\otimes i} = \{(s^{\otimes}, a, s^{\otimes\prime}) \in \delta^{\otimes}; s^{\otimes} \in R_{\pi}^{\otimes i}, \ P^{\otimes}(s^{\otimes\prime}|s^{\otimes}, a) > 0\}$, and let $P_{\pi}^{\otimes} : S_{\pi}^{\otimes} \times S_{\pi}^{\otimes} \to [0, 1]$ be the transition probability under π.

Lemma 1: For any policy π and any recurrent class $R_{\pi}^{\otimes i}$ in the Markov chain $MC_{\pi}^{\otimes}$, $MC_{\pi}^{\otimes}$ satisfies one of the following conditions.

1) $\delta_{\pi}^{\otimes i} \cap \bar{F}_j^{\otimes} \neq \emptyset$, $\forall j \in \{1, \ldots, n\}$,
2) $\delta_{\pi}^{\otimes i} \cap \bar{F}_j^{\otimes} = \emptyset$, $\forall j \in \{1, \ldots, n\}$.

Proof: Suppose that $MC_{\pi}^{\otimes}$ satisfies neither conditions 1 nor 2. Then, there exist a policy π, $i \in \{1, \ldots, h\}$, and j_1, $j_2 \in \{1, \ldots, n\}$ such that $\delta_{\pi}^{\otimes i} \cap \bar{F}_{j_1}^{\otimes} = \emptyset$ and $\delta_{\pi}^{\otimes i} \cap \bar{F}_{j_2}^{\otimes} \neq \emptyset$. In other words, there exists a nonempty and proper subset $J \in 2^{\{1,\ldots,n\}} \setminus \{\{1, \ldots, n\}, \emptyset\}$ such that $\delta_{\pi}^{\otimes i} \cap \bar{F}_j^{\otimes} \neq \emptyset$ for any $j \in J$. For any transition $(s, a, s') \in \delta_{\pi}^{\otimes i} \cap \bar{F}_j^{\otimes}$, the following equation holds by the properties of the recurrent states in $MC_{\pi}^{\otimes}$ [12].

$$\sum_{k=0}^{\infty} p^k((s, a, s'), (s, a, s')) = \infty, \tag{2}$$

where $p^k((s, a, s'), (s, a, s'))$ is the probability that the transition (s, a, s') reoccurs after it occurs in k time steps. Eq. (2) means that all transition in $R_{\pi}^{\otimes i}$ occurs infinitely often. However, the memory state v is never reset in $R_{\pi}^{\otimes i}$ by the assumption. This directly contradicts Eq. (2). ■

Lemma 1 implies that for an LTL formula φ if a path ρ under a policy π does not satisfy φ, then the agent obtains no reward in recurrent classes; otherwise there exists at least one recurrent class where the agent obtains rewards infinitely often.

Theorem 1: For a product MDP $M^{\otimes}$ of an MDP M and an augmented tLDBA $\bar{B}_{\varphi}$ corresponding to a given LTL formula φ and the reward function given by Def. 10, if there exists a positional policy satisfying φ on $M^{\otimes}$, then there exists a discount factor γ^* such that any algorithm that maximizes the expected discounted reward with $\gamma > \gamma^*$ will find a positional policy satisfying φ on $M^{\otimes}$.

Proof: Suppose that π^* is an optimal policy but does not satisfy the LTL formula φ. Then, for any recurrent class $R_{\pi^*}^{\otimes i}$ in the Markov chain $MC_{\pi^*}^{\otimes}$ and any accepting set $\bar{F}_j^{\otimes}$ of the product MDP $M^{\otimes}$, $\delta_{\pi^*}^{\otimes i} \cap \bar{F}_j^{\otimes} = \emptyset$ holds by Lemma 1. Thus, the agent under the policy π^* can obtain rewards only in the set of transient states. We consider the best scenario in the assumption. Let $p^k(s, s')$ be the probability of going to a state s' in k time steps after leaving the state s, and let $Post(T_{\pi}^{\otimes})$ be the set of states in recurrent classes that can be transitioned from states in $T_{\pi}^{\otimes}$ by one action. For the initial state s_{init} in the set of transient states, it holds that

$$\begin{aligned} V^{\pi^*}(s_{init}) = & \sum_{k=0}^{\infty} \sum_{s \in T_{\pi^*}^{\otimes}} \gamma^k p^k(s_{init}, s) \\ & \sum_{s' \in T_{\pi^*}^{\otimes} \cup Post(T_{\pi^*}^{\otimes})} P_{\pi^*}^{\otimes}(s'|s) \mathcal{R}(s, a, s') \\ \leq & \ r_p \sum_{k=0}^{\infty} \sum_{s \in T_{\pi^*}^{\otimes}} \gamma^k p^k(s_{init}, s), \end{aligned}$$

where the action a is selected by π^*. By the property of the transient states, for any state $s^{\otimes}$ in $T_{\pi^*}^{\otimes}$, there exists a bounded positive value m such that $\sum_{k=0}^{\infty} \gamma^k p^k(s_{init}, s) \leq \sum_{k=0}^{\infty} p^k(s_{init}, s) < m$ [12]. Therefore, there exists a bounded positive value $\bar{m}$ such that $V^{\pi^*}(s_{init}) < \bar{m}$. Let $\bar{\pi}$ be a positional policy satisfying φ. We consider the following two cases.

1) Assume that the initial state s_{init} is in a recurrent class $R_{\bar{\pi}}^{\otimes i}$ for some $i \in \{1, \ldots, h\}$. For any accepting set $\bar{F}_j^{\otimes}$, $\delta_{\bar{\pi}}^{\otimes i} \cap \bar{F}_j^{\otimes} \neq \emptyset$ holds by the definition of $\bar{\pi}$. The expected discounted reward for s_{init} is given by

$$\begin{aligned} V^{\bar{\pi}}(s_{init}) = & \sum_{k=0}^{\infty} \sum_{s \in R_{\bar{\pi}}^{\otimes i}} \gamma^k p^k(s_{init}, s) \\ & \sum_{s' \in R_{\bar{\pi}}^{\otimes i}} P_{\bar{\pi}}^{\otimes}(s' \mid s) \mathcal{R}(s, a, s'), \end{aligned}$$

where the action a is selected by $\bar{\pi}$. Since s_{init} is in $R_{\bar{\pi}}^{\otimes i}$, there exists a positive number $\bar{k} = \min\{k \ ; \ k \geq$

$n, p^k(s_{init}, s_{init}) > 0\}$ [12]. We consider the worst scenario in this case. It holds that

$$\begin{aligned} V^{\bar{\pi}}(s_{init}) \geq & \sum_{k=n}^{\infty} p^k(s_{init}, s_{init}) \sum_{i=1}^{n} \gamma^{k-i} r_p \\ \geq & \sum_{k=1}^{\infty} p^{k\bar{k}}(s_{init}, s_{init}) \sum_{i=0}^{n-1} \gamma^{k\bar{k}-i} r_p \\ > & r_p \sum_{k=1}^{\infty} \gamma^{k\bar{k}} p^{k\bar{k}}(s_{init}, s_{init}), \end{aligned}$$

whereas all states in $R(MC^{\otimes}_{\bar{\pi}})$ are positive recurrent because $|S^{\otimes}| < \infty$ [13]. Obviously, $p^{k\bar{k}}(s_{init}, s_{init}) \geq (p^{\bar{k}}(s_{init}, s_{init}))^k > 0$ holds for any $k \in (0, \infty)$ by the Chapman-Kolmogorov equation [12]. Furthermore, we have $\lim_{k\to\infty} p^{k\bar{k}}(s_{init}, s_{init}) > 0$ by the property of irreducibility and positive recurrence [14]. Hence, there exists $\bar{p}$ such that $0 < \bar{p} < p^{k\bar{k}}(s_{init}, s_{init})$ for any $k \in (0, \infty]$ and we have

$$V^{\bar{\pi}}(s_{init}) > r_p \bar{p} \frac{\gamma^{\bar{k}}}{1-\gamma^{\bar{k}}}.$$

Therefore, for any $\bar{m} \in (V^{\pi^*}(s_{init}), \infty)$ and any $r_p < \infty$, there exists $\gamma^* < 1$ such that $\gamma > \gamma^*$ implies $V^{\bar{\pi}}(s_{init}) > r_p \bar{p} \frac{\gamma^{\bar{k}}}{1-\gamma^{\bar{k}}} > \bar{m}$.

2) Assume that the initial state s_{init} is in the set of transient states $T^{\otimes}_{\bar{\pi}}$. $P^{M^{\otimes}}_{\bar{\pi}}(s_{init} \models \varphi) > 0$ holds by the definition of $\bar{\pi}$. For a recurrent class $R^{\otimes i}_{\bar{\pi}}$ such that $\delta^{\otimes i}_{\bar{\pi}} \cap \bar{F}^{\otimes}_j \neq \emptyset$ for each accepting set $\bar{F}^{\otimes}_j$, there exist a number $\bar{l} > 0$, a state $\hat{s}$ in $Post(T^{\otimes}_{\bar{\pi}}) \cap R^{\otimes i}_{\bar{\pi}}$, and a subset of transient states $\{s_1, \ldots, s_{\bar{l}-1}\} \subset T^{\otimes}_{\bar{\pi}}$ such that $p(s_{init}, s_1) > 0$, $p(s_i, s_{i+1}) > 0$ for $i \in \{1, ..., \bar{l}-2\}$, and $p(s_{\bar{l}-1}, \hat{s}) > 0$ by the property of transient states. Hence, it holds that $p^{\bar{l}}(s_{init}, \hat{s}) > 0$ for the state $\hat{s}$. Thus, by ignoring rewards in $T^{\otimes}_{\bar{\pi}}$, we have

$$\begin{aligned} V^{\bar{\pi}}(s_{init}) \geq & \gamma^{\bar{l}} p^{\bar{l}}(s_{init}, \hat{s}) \sum_{k=0}^{\infty} \sum_{s' \in R^{\otimes i}_{\bar{\pi}}} \gamma^k p^k(\hat{s}, s') \\ & \sum_{s'' \in R^{\otimes i}_{\bar{\pi}}} P^{\otimes}_{\bar{\pi}}(s''|s') \mathcal{R}(s', a, s'') \\ > & \gamma^{\bar{l}} p^{\bar{l}}(s_{init}, \hat{s}) r_p \bar{p} \frac{\gamma^{\bar{k}'}}{1-\gamma^{\bar{k}'}}, \end{aligned}$$

where $\bar{k}' \geq n$ is a constant and $0 < \bar{p} < p^{k\bar{k}'}(\hat{s}, \hat{s})$ for any $k \in (0, \infty]$. Therefore, for any $r_p < \infty$ and any $\bar{m} \in (V^{\pi^*}(s_{init}), \infty)$, there exists $\gamma^* < 1$ such that $\gamma > \gamma^*$ implies $V^{\bar{\pi}}(s_{init}) > \gamma^{\bar{l}} p^{\bar{l}}(s_{init}, \hat{s}) \frac{r_p \bar{p} \gamma^{\bar{k}'}}{1-\gamma^{\bar{k}'}} > \bar{m}$.

The results contradict the optimality assumption of π^*. ■

Theorem 1 implies that for the product MDP $M^{\otimes}$ of an MDP M and an augmented tLDGBA corresponding to a given LTL formula φ, we can obtain a feasible positional policy satisfying φ on $M^{\otimes}$ by an algorithm maximizing the expected discounted reward with a discount factor sufficiently close to 1 if there exists a positional policy on $M^{\otimes}$ satisfying φ.

IV. Example

In this section, we compare our proposed method with the method in [8], [10] and the method using tLDBA. In the method using tLDBA, the reward function is defined like Def. 10, namely it is based on the one accepting set of the corresponding product MDP. We consider a path planning problem of a robot in an environment consisting of eight rooms and one corridor as shown in Fig. 3. The state s_7 is the initial state and the action space is specified with $\mathcal{A}(s) = \{Right, Left, Up, Down\}$ for any state $s \neq s_4$ and $\mathcal{A}(s_4) = \{to\_s_0, to\_s_1, to\_s_2, to\_s_3, to\_s_5, to\_s_6, to\_s_7, to\_s_8\}$, where to_s_i means attempting to go to the state s_i for $i \in \{0, 1, 2, 3, 5, 6, 7, 8\}$. The robot moves in the intended direction with probability 0.9 and it stays in the same state with probability 0.1 if it is in the state s_4. In the states other than s_4, it moves in the intended direction with probability 0.9 and it moves in the opposite direction to that it intended to go with probability 0.1. If the robot tries to go to outside the environment, it stays in the same state. The labeling function is as follows.

$$L((s, act, s')) = \begin{cases} \{c\} & \text{if } s' = s_i,\ i \in \{2, 3, 5, 6\}, \\ \{a\} & \text{if } (s, act, s') = (s_4, to\_s_0, s_0), \\ \{b\} & \text{if } (s, act, s') = (s_4, to\_s_8, s_8), \\ \emptyset & \text{otherwise.} \end{cases}$$

In the example, the robot tries to take two transitions that we want to occur infinitely often, represented by arcs labeled by $\{a\}$ and $\{b\}$, while avoiding unsafe transitions represented by the arcs labeled by $\{c\}$. This is formally specified by the LTL formula given by (1). The LTL formula requires the robot to keep on entering the two rooms s_0 and s_8 from the corridor s_4 regardless of the order of entries, while avoiding entering the four rooms s_2, s_3, s_5, and s_6. They have two accepting sets. The tLDGBA $B_{\varphi} = (X, x_{init}, \Sigma, \delta, \mathcal{F})$ and its augmented automaton $\bar{B}_{\varphi} = (\bar{X}, \bar{x}_{init}, \bar{\Sigma}, \bar{\delta}, \bar{\mathcal{F}})$ are shown in Figs. 1 and 2, respectively.

For the three methods, we use Q-learning[1] with ε-greedy policy and gradually reduce ε to 0 to learn an optimal policy asymptotically. We set the positive reward $r_p = 2$, the epsilon greedy parameter $\varepsilon = \frac{0.95}{n_t(s^{\otimes})}$, where $n_t(s^{\otimes})$ is the number of

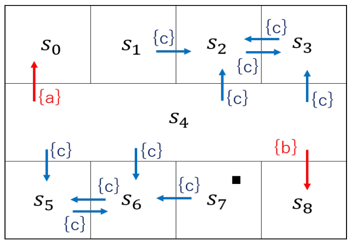

Fig. 3. The environment consisting of eight rooms and one corridor. Red arcs are the transitions that we want to occur infinitely often, while blue arcs are the transitions that we never want to occur. s_7 is the initial state.

[1]We employ Q-learning here but any algorithm that maximizes the discounted expected reward can be applied to our proposed method.

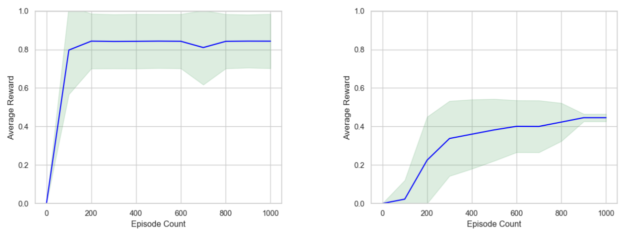

Fig. 4. The mean of average reward in each episode for 20 learning sessions obtained from our proposed method (left) and the method using tLDBA (right). They are plotted per 100 episodes and the green areas represent the range of standard deviations.

visits to state $s^{\otimes}$ within t time steps [15], and the discount factor $\gamma = 0.95$. The learning rate α varies in accordance with *the Robbins-Monro condition*. We train the agent in 10000 iterations and 1000 episodes for 20 learning sessions.

Fig. 4 shows the average rewards obtained by our proposed method and the method using a tLDBA B'_{φ} converted from φ, respectively. Both of methods eventually acquire an optimal policy satisfying φ. As shown in Fig. 4, however, our proposed method converges faster. This is because the number of accepting transitions to all transitions in $\bar{B}_{\varphi}$ is greater than B'_{φ}. That is, using $\bar{B}_{\varphi}$ lessens the sparsity of rewards than using B'_{φ}.

In [8], [10], the accepting frontier function is proposed. In this example, we use the function for the tLDGBA $Acc : \delta \times 2^{\delta} \rightarrow 2^{\delta}$. Initializing a set of transitions $\mathbb{F}$ with the set of the all accepting transitions in B_{φ}, the function receives the transition (x, σ, x') that occurs and the set $\mathbb{F}$. If (x, σ, x') is in $\mathbb{F}$, then Acc removes the accepting sets containing (x, σ, x') from $\mathbb{F}$. For the product MDP of the MDP M and the tLDGBA B_{φ}, the reward function is based on the removed states of B_{φ}.

Fig. 5 shows the optimal policies obtained by our proposed method and the method in [8], [10][2], respectively. The result for the method in [8], [10] is because it is impossible the transitions labeled with $\{a\}$ and $\{b\}$ occur from s_4 infinitely often by any positional policy with B_{φ} shown in Fig. 1. In detail, the state of the B_{φ} is always x_0 while the agent does not move to bad states s_2, s_3, s_5, and s_6. Thus, unless the bad states are visited, the product state is always (s_4, x_0) while the agent is in s_4. Thus, the agent cannot visit both of s_0 and s_8 by a deterministic action selection at s_4. While our proposed method can recognize the previous visits as a state. Thus, our proposed method can synthesize a positional policy satisfying φ on the product MDP, while the method in [8], [10] cannot. That is, in order to obtain a positional policy satisfying a given LTL formula φ with the method in [8], [10], we may have to make a redundant LDGBA associated with φ heuristically. However, the heuristic redundancy can cause a sparsity of rewards and a search space larger than necessary, and so on.

[2]We obtain the same result even with a state-based LDGBA.

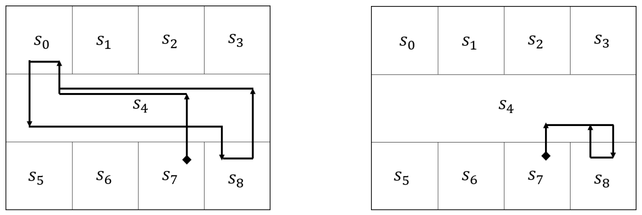

Fig. 5. The optimal policy obtained from our proposed method (left) and the method in [8], [10] (right).

V. Conclusions

The letter proposed a novel RL-based method for the synthesis of a control policy for an LTL specification using a limit-deterministic generalized Büchi automaton. The proposed method improved the learning performance compared to existing methods. It is future work to investigate a method that maximizes the satisfaction probability.

References

[1] C. Baier and J.-P. Katoen, *Principles of Model Checking*. MIT Press, 2008.
[2] C. Belta, B. Yordanov, and E. A. Gol, *Formal Methods for Discrete-Time Dynamical Systems*. Springer, 2017.
[3] M. L. Puterman, *Markov Decison Processes, Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1994.
in *Proc. 51st IEEE Conference on Decision and Control*, 2012, pp. 3372–3379.
[4] D. Sadigh, E. S. Kim, A. Coogan, S. S. Sastry, and S. Seshia, "A learning based approach to control synthesis of Markov decision processes for linear temporal logic specifications," *in Proc. 53rd IEEE Conference on Decision and Control*, pp. 1091-1096, 2014.
[5] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd Edition. MIT Press, 2018.
[6] Q. Gao, D. Hajinezhad, Y. Zhang, Y. Kantaros, and M. M. Zavlanos, "Reduced variance deep reinforcement learning with temporal logic specifications," *Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems*, pp. 237-248, 2019, ACM.
[7] S. Sickert, J. Esparaza, S. Jaax, and J. Křetı̀nský, "Limit-deterministic Büchi automata for linear temporal logic," in *International Conference on Computer Aided Verification*, 2016, pp. 312-332.
[8] M. Hasanbeig, A. Abate, and D. Kroening, "Logically-constrained reinforcement learning," *arXiv:1801.08099v8*, Feb. 2019.
[9] E. M. Hahn, M. Perez, S. Schewe, F. Somenzi, A. Triverdi, and D. Wojtczak, "Omega-regular objective in model-free reinforcement learning," *Lecture Notes in Computer Science*, no. 11427, pp. 395–412, 2019.
[10] M. Hasanbeig, Y. Kantaros, A. Abate, D. Kroening, G. J. Pappas, and I. Lee, "Reinforcement learning for temporal logic control synthesis with probabilistic satisfaction guarantee," *arXiv:1909.05304v1*, 2019.
[11] A. K. Bozkurt, Y. Wang, M. Zavlanos, and M. Pajic, "Control synthesis from linear temporal logic specifications using model-free reinforcement learning," *arXiv:1909.07299*, 2019.
[12] R. Durrett, *Essentials of Stochastic Processes*, 2nd Edition. ser. Springer texts in statistics. New York; London; Springer, 2012.
[13] L. Breuer, "Introduction to Stochastic Processes," [Online]. Available: https://www.kent.ac.uk/smsas/personal/lb209/files/sp07.pdf
[14] S.M. Ross, *Stochastic Processes*, 2nd Edition. University of California, Wiley, 1995.
[15] S. Singh, T. Jaakkola, M. L. Littman, and C. Szepesv́ari, "Convergence results for single-step on-policy reinforcement learning algorithms" *Machine Learning*, vol. 38, no. 3, pp, 287–308, 1998.
[16] J. Kretínský, T. Meggendorfer, S. Sickert, and C. Ziegler. "Rabinizer 4: From ltl to your favourite deterministic automaton," *In CAV*, 2018. To appear.