2015 IEEE Symposium Series on Computational Intelligence

Learning an Optimal Control Policy for a Markov Decision Process Under Linear Temporal Logic Specifications

Masaki Hiromoto and Toshimitsu Ushio

***Abstract*—In this paper, we consider an optimal control problem under a qualitative specification such as progress and safety properties for a Markov decision process(MDP) with control costs. This control problem is motivated by the optimal path planing and control of mobile robots. We assume the existence of the uncertainty such as the unknown transition probability in the MDP. Then, we apply reinforcement learning(RL) to the generation of the optimal control policy. We consider the case where the qualitative specification is described by a linear temporal logic(LTL) formula. We construct a deterministic Rabin automaton(DRA) by which we check if the controlled behavior of the MDP satisfies the LTL specification with probability 1. We convert the DRA to a weighted automaton whose weight is determined based on the acceptance conditions of the DRA. Thus, the weighted automaton is used to learn a control policy that satisfies the LTL specification. Then, the control problem considered in the paper is reduced to a reinforcement learning problem with multiple rewards that correspond to the control costs and the weights. To generate the optimal control policy, using the MDP and the weighted automaton, we construct a product MDP with sequentially decision making. The proposed controller consists of a sequential decision making of two steps. At the first step, one learns a minimally restrictive set of actions such that if an action selected at the second step generates runs that do not satisfy the LTL specification, it is eliminated from the set. At the second step, one learns the optimal action that minimizes the discounted sum of the control costs in the set. We also consider an illustrative example to show that the proposed RL based controller can learn an optimal control policy.**

I. Introduction

A temporal logic(TL) has been used to verify reactive and concurrent systems [1]. The TL is an extension of a classical logic and describes temporal properties, that is, properties of state transitions. Recently, in many control applications such as the optimal path planing and control of mobile robots, there has been an increased interest in describing a qualitative control specification by the TL [2]–[15]. Several control methods have been proposed. In [2], they considered a discrete-time linear time-invariant system under constraints given by a linear temporal logic(LTL) formula and proposed a receding horizon control(RHC) using a finite abstraction of the continuous state set. In [3], the RHC for finite deterministic systems has been studied. On the other hand, for the logical control of discrete event systems(DESs) where a control specification is given by a language, a supervisory control has been studied [16]. In [5], they consider the case where a control specification is given by an LTL, a computational tree logic(CTL), or a CTL* formula and, for each TL formula, proposed a design method for a supervisor such that all infinite words generated by the supervised DES satisfies the TL formula. They use an ω-autmaton to decide whether the controlled behavior satisfies the TL formula. Such an automaton based approach is often used for the checking the satisfaction of the LTL formula [17]. Safra's method is a well-known method for the conversion from the LTL formula to a deterministic Rabin automaton(DRA) and uses a non-deterministic Büchi automaton as an intermediate automaton [18]. Recently, however, a novel method for direct conversion from the LTL formula to a DRA has been proposed [19]. The DRA constructed by the novel method has acceptance conditions based on not only states but also transitions and the cardinality of its state set is smaller than that of the DRA by Safra's method.

In general, when a system is uncertain, that is, the system's parameters are unknown a priori, the robust control approach is often used when the uncertainty is described in such a way that the unknown parameter is in a given set. In [4], they consider an uncertain Markov decision process(MDP) where each transition probability is in a given uncertain set and proposed a robust controller that satisfies a control specification given by an LTL formula. Recently, learning based control has been proposed as another approach to the design of the controller [20], [21]. A reinforcement learning(RL) based approach is useful for such a situation [22], [23]. The RL is a learning framework that learns a transition probabilities and control cost function of the uncertain MDP and searches an optimal control policy that optimizes an expected discounted sum of the costs through trial and error. In [20], Sadigh et al. considered an uncertain MDP and a control specification given by an LTL formula. They proposed a RL based method for design of a control policy such that the controlled MDP satisfies the LTL formula with probability 1. They constructed a DRA that accepts all words satisfying the formula and computes a product MDP of the MDP and the DRA. Then, to learn the control policy by which the controlled MDP satisfies the LTL formula with probability 1, a reward is assigned to each state of the product MDP in accordance with the acceptance conditions of the DRA. Using the reward, the controller learns an optimal policy such that an expected discounted sum of the rewards on the controlled product MDP is maximized for each state. In [21], Jones et al. considered a control specification described by a signal temporal logic(STL) formula. To the best of our knowledge, a problem to learn a control policy that optimizes the control cost on the MDP under the LTL formula has not been studied. So, we consider an MDP with control cost for each state and formulate an optimal control problem that has the following two control objectives: The first objective is for the controlled MDP to satisfy the LTL formula with probability 1. The second one is to minimize the expected discounted sum of the control costs. Such a control problem is applicable to the optimal path planning of mobile robots with safety and progress properties.

In this paper, first, we construct a DRA that accepts all and only infinite words satisfying the given LTL control specification. Second, we construct a product MDP of the MDP and the DRA to represent a dynamic control policy that satisfies the control specification. Third, we modify the product MDP in order to apply RL to the design of an optimal control policy. The modified product MDP has a set of reward functions that return a reward for each transition on the modified product MDP in accordance with the acceptance conditions of the DRA. The control action of the modified product MDP is a pair of a pattern and an action, where the pattern is a set of actions. Moreover, we introduce a reward that represents both the satisfaction of the control specification and the minimally restrictiveness of the pattern. Finally, we proposed an algorithm for design of an optimal control policy that consists of a sequential decision making of two steps. At the first decision making, we remove actions by which runs that do not satisfy the LTL control specification. At the second one, we select an action from the pattern selected at the first one such that it minimizes the discounted sum of the costs.

This paper is organized as follows: In Section II, we review an LTL formula and a DRA. In Section III, we formulate an optimal control problem of an MDP with control costs under an LTL control specification. In Section IV, we propose a method to design an optimal policy using RL. In Section V, we consider an illustrative example to show that the proposed algorithm can obtain an optimal control policy. We conclude in Section VI.

II. Preliminary

We review Linear temporal logic(LTL) [1]. We use an LTL formula to describe a control specification. An LTL formula is recursively defined over a finite fixed set of atomic propositions AP as follows.

- An atomic proposition $a \in AP$, the symbol tt, and the symbol ff are LTL formulas,
- for given LTL formulas φ and ψ, $\neg\varphi$, $\varphi \wedge \psi$, and $\varphi \vee \psi$ are LTL formulas, and
- for given LTL formulas φ and ψ, $\mathbf{X}\varphi$, $\mathbf{F}\varphi$, $\mathbf{G}\varphi$, and $\varphi\mathbf{U}\psi$ are LTL formulas,

where $\neg$ is a negation, $\wedge$ is a logical conjunction, and $\vee$ is a logical disjunction. The LTL formula φ is qualitatively evaluated by an infinite word $w \in \left(2^{AP}\right)^{\omega}$. The ith letter of ω is denoted by $\omega[i]$, i.e. $\omega = \omega[0]\omega[1]\ldots$. The suffix of ω starting from the ith letter is denoted by $\omega_i = \omega[i]\omega[i+1]\ldots$. We write $w \models \varphi$ iff φ is TRUE for w. We write $w \not\models \varphi$ iff φ is FALSE for w.

Semantics of the formula on the word ω is defined inductively as follows.

$$
\begin{array}{lll}
w \models \text{tt}. & & w \not\models \text{ff}. \\
w \models a & \iff & a \in \omega[0]. \\
w \models \neg a & \iff & a \notin \omega[0]. \\
w \models \varphi \wedge \psi & \iff & w \models \varphi \wedge w \models \psi. \\
w \models \varphi \vee \psi & \iff & w \models \varphi \vee w \models \psi. \\
w \models \mathbf{X}\varphi & \iff & w_1 \models \varphi. \\
w \models \mathbf{F}\varphi & \iff & {}^{\exists}k \geq 0, w_k \models \varphi. \\
w \models \mathbf{G}\varphi & \iff & {}^{\forall}k \geq 0, w_k \models \varphi. \\
w \models \varphi\mathbf{U}\psi & \iff & {}^{\exists}k \geq 0, {}^{\forall}l\ (0 \leq l < k), w_k \models \psi \wedge w_l \models \varphi.
\end{array}
$$

Intuitively, if φ and ψ are atomic propositions, $\mathbf{X}\varphi$ is TRUE when φ is TRUE in the next step, $\mathbf{F}\varphi$ is TRUE when φ is TRUE in some steps, $\mathbf{G}\varphi$ is TRUE when φ is TRUE in all steps, and $\varphi\mathbf{U}\psi$ is TRUE when φ is TRUE until ψ becomes TRUE. Several control specifications such as liveness properties $\mathbf{GF}\varphi$, stability properties $\mathbf{FG}\varphi$ and safety properties $\mathbf{G}\neg\varphi$ are described by the LTL formulas.

We introduce a deterministic Rabin automaton(DRA) [19]. The DRA $\mathcal{R}$ is described by $(Q, q_0, \Sigma, \delta, Acc)$, where Q is a finite and non-empty set of states, $q_0 \in Q$ is an initial state, Σ is a finite set of input alphabets, $\delta : Q \times \Sigma \to Q$ is a transition function, $Acc = \{Acc_i\}_{i=1}^{n_{Acc}}$ is a set of acceptance conditions, $Acc_i = (S_i, T_i)$ for each $i = 1, \ldots, n_{Acc}$, $S_i \subset Q$ is a set of non-acceptance states, $T_i = (G_i, B_i) \subset \Delta \times \Delta$, $\Delta = \{(q, \nu, q') \in Q \times 2^{\Sigma} \times Q \mid \delta(q, \nu) = q'\}$ is a transition relation, G_i is a set of acceptance transitions, and B_i is a set of non-acceptance transitions. We consider two kinds of infinite sequences in the DRA for the input word $\tilde{\omega} \in (\Sigma)^{\omega}$: $r_q \in Q^{\omega}$ and $r_\delta \in \Delta^{\omega}$ defined by $r_q = r_q[0]r_q[1]\ldots$ with $r_q[i] = \delta(r_q[i-1], \tilde{\omega}[i-1])$ $(r_q[0] = q_0)$ and $r_\delta = r_\delta[0]r_\delta[1]\ldots$ with $r_\delta[i] = (r_q[i], \tilde{\omega}[i], r_q[i+1]) \in \Delta$. We call the pair $r = (r_q, r_\delta)$ the run of $\mathcal{R}$ over $\tilde{\omega}$. For each sequence, let $\inf(r_l)$ be a set of elements that appear infinitely often in the sequence.

$$\inf(r_l) = \{\hat{r} \mid \hat{r} = r_l[i] \text{ for infinitely many } i\text{'s}\} \text{ for } l = q, \delta.$$

Note that $\inf(r_q)$ and $\inf(r_\delta)$ are nonempty sets because both Q and Δ are finite sets. The run $r = (r_q, r_\delta)$ is accepted by Acc if there exists $i \in \{1, \ldots, n_{Acc}\}$ such that

$$
\begin{aligned}
&\inf(r_q) \cap S_i = \emptyset \ \wedge \\
&\qquad \inf(r_\delta) \cap G_i \neq \emptyset \ \wedge \ \inf(r_\delta) \cap B_i = \emptyset. \qquad (1)
\end{aligned}
$$

For each LTL formula φ over AP, let $\mathcal{R}_\varphi$ be a DRA with a finite set of input alphabets $\Sigma = 2^{AP}$ that accepts all and only infinite words over AP that satisfy φ.

III. Problem Formulation

We consider a plant modeled by a labeled Markov decision process(MDP) $\mathcal{M} = (\mathcal{S}, A, \mathcal{A}, P, s_0, AP, \mathcal{L}, g)$, where $\mathcal{S}$ is a finite and non-empty set of states of the plant, $A = \{a_1, \ldots, a_{n_A}\}$ is a finite and non-empty set of actions (controls), $\mathcal{A} : \mathcal{S} \to 2^A \setminus \emptyset = \{\pi_1, \ldots, \pi_{2^{n_A}-1}\}$ is a function that represents a set of available actions at each state, $P : \mathcal{S} \times A \times \mathcal{S} \to [0, 1]$ is a transition probability function such that $\sum_{s' \in \mathcal{S}} P(s, a, s') = 1$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$, $s_0 \in \mathcal{S}$ is an initial state, AP is a set of atomic propositions, $\mathcal{L} : \mathcal{S} \to 2^{AP}$ is a labeling function, $g : \mathcal{S} \times A \to \mathbb{R}_{\geq 0}$ is a control cost function, and $\mathbb{R}_{\geq 0}$ denotes the set of non-negative real numbers. Note that the labeling function $\mathcal{L}$ returns atomic propositions that are TRUE at the state. In the following, a subset π of A will be called a pattern.

We also consider a control specification described by an LTL formula φ. Then, we construct a DRA $\mathcal{R}_\varphi = \big(Q, q_0, 2^{AP}, \delta, Acc\big)$ that accepts all and only words satisfying φ. In order to check φ's satisfiability for a run on $\mathcal{M}$, we introduce a product MDP $\mathcal{M}_\varphi = (\mathcal{S}_\times, A, \mathcal{A}_\times, P_\times, s_{0\times}, Acc_\times, g_\times)$ of $\mathcal{M}$ and $\mathcal{R}_\varphi$ with the same action set A as that of $\mathcal{M}$, where:

- $\mathcal{S}_\times = \mathcal{S} \times Q$ is a set of states,
- $\mathcal{A}_\times : \mathcal{S}_\times \to 2^A$ is a function such that $\mathcal{A}_\times(s_\times) = \mathcal{A}([\![s_\times]\!]_s) \subset A$ is a set of actions available at the state $s_\times = (s, q) \in \mathcal{S}_\times$, where $[\![\cdot]\!]_s : \mathcal{S}_\times \to \mathcal{S}$ is a projection function defined by $[\![s_\times]\!]_s = s$,
- $P_\times : \mathcal{S}_\times \times A \times \mathcal{S}_\times \to [0, 1]$ is a probability function defined as:
$$P_\times(s_\times, a, s'_\times) = \begin{cases} P([\![s_\times]\!]_s, a, [\![s'_\times]\!]_s) \\ \quad \text{if } [\![s'_\times]\!]_q = \delta([\![s_\times]\!]_q, \mathcal{L}([\![s'_\times]\!]_s)), \\ 0 \quad \text{otherwise,} \end{cases}$$
where $[\![\cdot]\!]_q : \mathcal{S}_\times \to Q$ is a projection function defined by $[\![s_\times]\!]_q = q$ for $s_\times = (s, q)$,
- $s_{0\times} = (s_0, \delta(q_0, \mathcal{L}(s_0))) \in \mathcal{S}_\times$ is an initial state,
- $Acc_\times = \big\{Acc^i_\times\big\}_{i=1}^{n_{Acc}}$ is a set of acceptance conditions, for each $i = 1, \ldots, n_{Acc}$, $Acc^i_\times = (S^i_\times, T^i_\times)$ and $T^i_\times = (G^i_\times, B^i_\times)$ are defined as:
$$\begin{aligned} S^i_\times &= \{s_\times \mid [\![s_\times]\!]_q \in S_i\}, \\ G^i_\times &= \big\{(s_\times, s'_\times) \mid ([\![s_\times]\!]_q, \mathcal{L}([\![s'_\times]\!]_s), [\![s'_\times]\!]_q) \in G_i\big\}, \\ B^i_\times &= \big\{(s_\times, s'_\times) \mid ([\![s_\times]\!]_q, \mathcal{L}([\![s'_\times]\!]_s), [\![s'_\times]\!]_q) \in B_i\big\}, \end{aligned}$$
and
- $g_\times(s_\times, a) = g([\![s_\times]\!]_s, a)$ is a control cost function.

$P_\times(s_\times, a, s'_\times)$ is a probability such that the transition from $[\![s_\times]\!]_s$ to $[\![s'_\times]\!]_s$ in $\mathcal{M}$ is caused by the occurrence of the action a and the transition from $[\![s_\times]\!]_q$ to $[\![s'_\times]\!]_q$ in $\mathcal{R}_\varphi$ is caused by the occurrence of the set of atomic propositions $\mathcal{L}([\![s'_\times]\!]_s)$. Thus, we define that $[\![s_{0\times}]\!]_q$ is not q_0 but $\delta(q_0, \mathcal{L}(s_0))$.

We consider a control policy $d : \mathcal{S}_\times \to A$ in $\mathcal{M}_\varphi$, which is a dynamic policy in $\mathcal{M}$. In the following, the control policy will be called the policy for short. The controlled $\mathcal{M}_\varphi$ by d is described by a Markov chain(MC) $\mathcal{M}^d_\varphi = \big(\mathcal{S}_\times, s_{0\times}, p, T, Acc_\times, g^d_\times\big)$, where $p : \mathcal{S}_\times \times \mathcal{S}_\times \to \mathbb{R}$ is a transition probability function such that $p(s_\times, \cdot) = P_\times(s_\times, d(s_\times), \cdot)$ for all $s_\times \in \mathcal{S}_\times$, $T = \{(s_\times, s'_\times) \mid p(s_\times, s'_\times) > 0\}$ is a set of transitions, and $g^d_\times(s_\times) = g_\times(s_\times, d(s_\times))$ is a control cost function of the policy d. We define a run $r^d_\times = (r^d_{q\times}, r^d_{\delta\times})$ on $\mathcal{M}^d_\varphi$ by $r^d_{q\times} = r^d_{q\times}[0] r^d_{q\times}[1] \ldots \in (\mathcal{S}_\times)^\omega$ that satisfies $P_\times(r_{q\times}[i], d(r_{q\times}[i]), r_{q\times}[i+1]) > 0$ for each i and $r^d_{q\times}[0] = s_{0\times}$ and $r^d_{\delta\times} = r^d_{\delta\times}[0] r^d_{\delta\times}[1] \ldots$ with $r^d_{\delta\times}[i] = (r^d_{q\times}[i], r^d_{q\times}[i+1])$.

Based on (1), we introduce the following definition.

Definition 1: Let $\mathcal{M}^d_\varphi$ be an MC. We say that $\mathcal{M}^d_\varphi$ satisfies φ with probability 1 if $Pr(\{(r^d_{q\times}, r^d_{\delta\times}) : {}^\exists Acc^i_\times \text{ s.t. } \inf(r^d_{q\times}) \cap S^i_\times = \emptyset \ \wedge \ \inf(r^d_{\delta\times}) \cap G^i_\times \neq \emptyset \ \wedge \ \inf(r^d_{\delta\times}) \cap B^i_\times = \emptyset\}) = 1$. Let D be a set of policies d by which $\mathcal{M}^d_\varphi$ satisfies φ with probability 1. We define a discounted sum of the control costs under the policy d after a state $r^d_{q\times}[t]$ as follows.

$$R^d_t = \sum_{i=0}^{\infty} \gamma^i g^d_\times(r^d_{q\times}[t+i+1]),$$

where $\gamma \in [0, 1)$ is a discount rate. We consider a state value function $V^d : \mathcal{S}_\times \to \mathbb{R}_{\geq 0}$ for $\mathcal{M}^d_\varphi$ given by

$$\begin{aligned} V^d(s_\times) &= \mathbb{E}_d\{R^d_t \mid r^d_{q\times}[t] = s_\times\} \\ &= \sum_{s'_\times \in \mathcal{S}_\times} p(s_\times, s'_\times) \mathbb{E}_d\{g^d_\times(r^d_{q\times}[t+1]) + \\ &\qquad \gamma R^d_{t+1} \mid r^d_{q\times}[t] = s_\times, r^d_{q\times}[t+1] = s'_\times\} \\ &= \sum_{s'_\times \in \mathcal{S}_\times} p(s_\times, s'_\times)[g^d_\times(s'_\times) + \gamma V^d(s'_\times)]. \end{aligned}$$

Our objective is to design an optimal policy $d^* \in D$ such that for all $s_\times \in \mathcal{S}_\times$, $V^{d^*}(s_\times) = \min_{d \in D} V^d(s_\times)$. In other words, for a given LTL control specification φ, the optimal control policy d^* achieves the minimization of the discounted sum of the control costs under the requirement that the controlled plant $\mathcal{M}^{d^*}_\varphi$ satisfies φ with probability 1.

IV. RL BASED DESIGN OF OPTIMAL POLICY

We consider a design problem of the optimal control policy under the existence of uncertainty in the plant $\mathcal{M}$, that is, the transition probability function P of the plant is unknown. We apply reinforcement learning(RL) to learn the optimal control policy on-line. Recall that the considered optimal control problem has two requirements, that is, the minimization of the discounted sum of the control costs given by the control cost function g and the satisfaction of the LTL control specification φ with probability 1. The former requirement is reduced to the standard RL problem for MDP where the reward corresponds to the control cost. For the latter requirement, D. Sadigh *et al.* introduced rewards that evaluate the acceptance conditions in a DRA $\mathcal{R}_\varphi$ for the LTL specification φ [20]. Thus, we formulated the control problem as a reinforcement learning problem with multiple rewards.

We consider a learning problem of a set of actions, that is, a pattern satisfying the latter requirement for each state of the product MDP $\mathcal{M}_\varphi$ by which we decide the satisfaction of the LTL formula. Thus, a control policy is described by a mapping from the state set of the product MDP to a pair of a pattern π and an action a, where $a \in \pi$. The control objective is to learn the optimal pair (π^*, a^*). The optimal pattern π^* is a minimally restrictive set of actions such that an action by which the LTL control specification is satisfied with the probability less than 1 is removed from the set. The optimal action a^* achieves the optimization of the discounted sum of the control costs in π^* while the optimal action a^* satisfies the

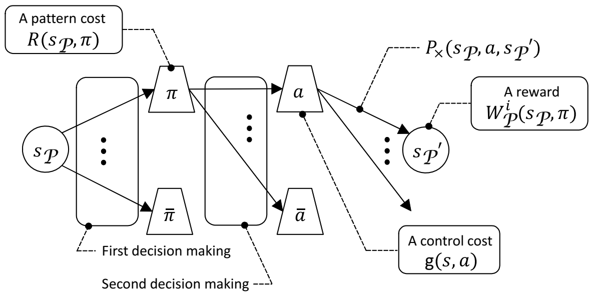


Fig. 1: A sequential decision making in the modified MDP $\mathcal{P}$ with Acc^i.

LTL specification in π^*. Thus, the optimal control is reduced to RL based optimal control consisting of sequentially decision making as shown in Fig. 1. At the first decision, we use rewards corresponding to the acceptance conditions of the DRA and learn a pattern that includes actions satisfying the LTL specification as many as possible so as to restrict the search of the optimal action in the pattern. At the second decision, we use the control costs and learn the optimal action in the pattern selected at the first decision. In the following, we formulate the RL based optimal control problem with the sequentially decision making.

First, we define a modified product MDP $\mathcal{P}$ from $\mathcal{M}_\varphi$ as follows.

Definition 2: Let $\mathcal{M}_\varphi$ be a product MDP. The modified product MDP is described by $\mathcal{P} = (\mathcal{S}_{\mathcal{P}}, A, \mathcal{A}_{\mathcal{P}}, P_{\mathcal{P}}, s_{\mathcal{P}0}, Acc_{\mathcal{P}}, W_{\mathcal{P}}, g_{\mathcal{P}})$, where $\mathcal{S}_{\mathcal{P}} = \mathcal{S}_\times$ is a set of states, $\mathcal{A}_{\mathcal{P}} : \mathcal{S}_{\mathcal{P}} \to 2^A \setminus \emptyset \times A$ is a function such that $\mathcal{A}_{\mathcal{P}}(s_{\mathcal{P}}) = \{(\pi, a) \mid \pi \in 2^{A_\times(s_{\mathcal{P}})} \setminus \emptyset, a \in \pi\}$, $P_{\mathcal{P}} : \mathcal{S}_{\mathcal{P}} \times \mathcal{A}_{\mathcal{P}} \times \mathcal{S}_{\mathcal{P}} \to [0,1]$ is a probability function defined as :

$$P_{\mathcal{P}}(s_{\mathcal{P}}, a_{\mathcal{P}}, s'_{\mathcal{P}}) = P_\times(s_{\mathcal{P}}, a, s'_{\mathcal{P}}) \quad (a_{\mathcal{P}} = (\pi, a)),$$

$s_{\mathcal{P}0} = s_{0\times}$ is an initial state, $Acc_{\mathcal{P}} = \left\{Acc^i_{\mathcal{P}}\right\}_{i=1}^{n_{Acc}} = \{(\mathfrak{G}_i, \mathfrak{B}_i)\}_{i=1}^{n_{Acc}}$ is a set of acceptance conditions defined as : $\mathfrak{G}_i = G^i_\times$ and $\mathfrak{B}_i = B^i_\times \cup \text{Influx}(S^i_\times)$,

$$\begin{aligned}\text{Influx}(S^i_\times) &= \{(s_{\mathcal{P}}, s'_{\mathcal{P}}) \mid {}^\exists a \text{ s.t.} \\ &\qquad P_\times(s_{\mathcal{P}}, a, s_{\mathcal{P}}) > 0 \ \wedge \ s'_{\mathcal{P}} \in S^i_\times\},\end{aligned}$$

$W_{\mathcal{P}} = \{W^i_{\mathcal{P}}\}_{i=1}^{n_{Acc}}$ is a set of reward functions for $Acc_{\mathcal{P}}$, $W^i_{\mathcal{P}} : \mathcal{S}_{\mathcal{P}} \times \mathcal{S}_{\mathcal{P}} \to \mathbb{R}$ is defined as:

$$W^i_{\mathcal{P}}(s_{\mathcal{P}}, s'_{\mathcal{P}}) = \begin{cases} w_G(>0) & \text{if } (s_{\mathcal{P}}, s'_{\mathcal{P}}) \in \mathfrak{G}_i, \\ w_B(<0) & \text{if } (s_{\mathcal{P}}, s'_{\mathcal{P}}) \in \mathfrak{B}_i, \\ 0 & \text{otherwise,} \end{cases}$$

and $g_{\mathcal{P}}(s_{\mathcal{P}}, a_{\mathcal{P}}) = g_\times(s_{\mathcal{P}}, a)$ is a control cost function.

Note that the set of non-acceptance states $S^i_\times$ is merged into the set of non-acceptance transitions $B^i_\times$ and the probability measure of the set of all sequences that satisfy φ is unchanged because of (1). Then, if a sequence satisfies φ, the agent obtains a positive reward w_G infinitely often and a negative reward w_B finitely often. We define a policy $\mu : \mathcal{S}_{\mathcal{P}} \to \mathcal{A}_{\mathcal{P}}$ defined as $\mu(s_{\mathcal{P}}) = (\mu_1(s_{\mathcal{P}}), \mu_2(s_{\mathcal{P}})) = (\pi, a) \in \mathcal{A}_{\mathcal{P}}(s_{\mathcal{P}})$.

Next, we introduce two kinds of optimal value functions $Q^{\mu^*}_{1i}$ $(i = 1, 2, \ldots, n_{ACC})$ and $Q^{\mu^*}_2$ that are used for the first and the second decision, respectively.

The function $Q^{\mu^*}_{1i}$ is an optimal value function about the LTL control specification for each acceptance condition $Acc^i_{\mathcal{P}}$, that is, $Q^{\mu^*}_{1i}(s_{\mathcal{P}}, \pi)$ is the expected discounted sum of the rewards when transitions depend on an optimal policy μ^* and a pattern π at the first decision making. Thus, the following optimal Bellman equation holds.

$$\begin{aligned} Q^{\mu^*}_{1i}(s_{\mathcal{P}}, \pi) = &\sum_{t \in \mathcal{S}_{\mathcal{P}}} Pro\,(s_{\mathcal{P}}, \pi, t) \cdot \\ &\left\{ R_{1i}\,(s_{\mathcal{P}}, \pi, t) + \gamma_1 \max_{\bar{\pi} \in 2^{\mathcal{A}([\![t]\!]_s)} \setminus \emptyset} Q^{\mu^*}_{1i}(t, \bar{\pi}) \right\}, \end{aligned}$$

where $Pro(s_{\mathcal{P}}, \pi, t)$ is a probability of a transition from the state $s_{\mathcal{P}}$ to the state t caused by the pattern π, $R_{1i}\,(s_{\mathcal{P}}, \pi, t)$ is a reward obtained by the transition from the state $s_{\mathcal{P}}$ to the state t caused by the pattern π for the acceptance condition $Acc^i_{\mathcal{P}}$, and $\gamma_1 \in (0, 1)$ is the discount rate for the evaluation of the LTL control specification. $Pro(s_{\mathcal{P}}, \pi, t)$ is reduced to the following equation.

$$Pro(s_{\mathcal{P}}, \pi, t) = \sum_{\sigma \in \pi} P_1(s_{\mathcal{P}}, \pi, \sigma) P_2(s_{\mathcal{P}}, \sigma, t),$$

where $P_1(s_{\mathcal{P}}, \pi, \sigma)$ is the decision probability of the action σ at the state $s_{\mathcal{P}}$ and the pattern π, and $P_2(s_{\mathcal{P}}, \sigma, t)$ is the transition probability from the state $s_{\mathcal{P}}$ to the state t by the occurrence of the action σ.

The function $Q^{\mu^*}_2$ is an optimal value function about control costs, that is, $Q^{\mu^*}_2(s_{\mathcal{P}}, \pi)$ is the expected discounted sum of the control costs when the pattern π is selected at the first decision making under the optimal policy μ^*. Thus, the following equation holds.

$$\begin{aligned} Q^{\mu^*}_2(s_{\mathcal{P}}, \pi) = &\min_{\sigma \in \pi} \sum_{t \in \mathcal{S}_{\mathcal{P}}} P_2(s_{\mathcal{P}}, \sigma, t) \cdot \\ &\left\{ g([\![s_{\mathcal{P}}]\!]_s, \sigma) + \gamma_2 \min_{\bar{\pi} \in 2^{\mathcal{A}([\![t]\!]_s)} \setminus \emptyset} Q^{\mu^*}_2(t, \bar{\pi}) \right\}, \end{aligned}$$

where $g([\![s_{\mathcal{P}}]\!]_s, \sigma)$ is the control cost for the state $[\![s_{\mathcal{P}}]\!]_s$ and $\gamma_2 \in [0, 1)$ is the discount rate for the cost evaluation. Note that the value function about control costs is defined independently for each acceptance condition. We make the following two assumptions [24], [25].

Assumption 1: $R_{1i}\,(s_{\mathcal{P}}, \pi, t)$ is given by

$$R_{1i}\,(s_{\mathcal{P}}, \pi, t) = R\,(s_{\mathcal{P}}, \pi) + W^i_{\mathcal{P}}\,(s_{\mathcal{P}}, t)\,,$$

where $R\,(s_{\mathcal{P}}, \pi)$ is a pattern cost that evaluates the restriction of selectable actions by the pattern π at the state $s_{\mathcal{P}}$ defined by

$$R\,(s_{\mathcal{P}}, \pi) = \left\{ \begin{array}{ll} R_C \frac{|\mathcal{A}([\![s_{\mathcal{P}}]\!]_s)| - |\pi|}{|\mathcal{A}([\![s_{\mathcal{P}}]\!]_s)| - 1} & \text{if } |\mathcal{A}([\![s_{\mathcal{P}}]\!]_s)| \geq 2, \\ 0 & \text{if } |\mathcal{A}([\![s_{\mathcal{P}}]\!]_s)| = 1, \end{array} \right.$$

and $R_C \in \mathbb{R}_{\leq 0}$ is a non-positive parameter that indicates the importance of less restrictiveness of the patterns.

If R_C equals to 0 and $g_{\mathcal{P}}(s_{\mathcal{P}}, a_{\mathcal{P}}) = 1$ for all $s_{\mathcal{P}}$ and $a_{\mathcal{P}}$, the optimal action is same as an optimal action defined in [20]. Therefore, the method in [20] is a special case of our proposed method. As R_C decreases, the agent learns a policy that makes the discounted sum of control costs smaller and may select a pattern including an action by which the controlled MDP satisfies φ with probability less than 1.

In general, an optimal action is not unique. So, for simplicity, we introduce a priority for the set of actions in such a way that the smaller the index of the action is, the higher its priority of the action is. We say that the action σ_i has a higher priority than the action σ_j if $i < j$.

Assumption 2: $P_1(s_{\mathcal{P}}, \pi, \sigma)$ is given by

$$P_1(s_{\mathcal{P}}, \pi, \sigma) = \begin{cases} 1 & \text{if } \sigma = \text{opt\_action}^*(s_{\mathcal{P}}, \pi), \\ 0 & \text{otherwise,} \end{cases}$$

where

$$\text{opt\_action}^*(s_{\mathcal{P}}, \pi) = \sigma_{j^*}$$

and

$$\begin{aligned} j^* \;=\; & \min \arg \min_{j \in \{j \mid \sigma_j \in \pi\}} \sum_{t \in \mathcal{S}_{\mathcal{P}}} P_2(s_{\mathcal{P}}, \sigma_j, t) \cdot \\ & \left\{ g([\![s_{\mathcal{P}}]\!]_s, \sigma_j) + \gamma_2 \min_{\bar{\pi} \in 2^{\mathcal{A}([\![t]\!]_s)} \setminus \emptyset} Q_2^{\mu^*}(t, \bar{\pi}), \right\}. \end{aligned}$$

Assumption 2 means that the action σ is selected with probability 1 at the state $s_{\mathcal{P}}$ from the pattern π in such a way that it is the highest priority among actions achieving the minimum expected discounted sum of the costs. This assumption is made for simplicity. Note that the probability Pro in the optimal Bellman equation depends on the expected discounted sum of the control costs.

Using Assumptions 1 and 2, we rewrite $Q_{1i}^{\mu^*}(s_{\mathcal{P}}, \pi)$ as follows.

$$\begin{aligned} Q_{1i}^{\mu^*}(s_{\mathcal{P}}, \pi) &= \sum_{t \in \mathcal{S}_{\mathcal{P}}} Pro\,(s_{\mathcal{P}}, \pi, t) \{ R_{1i}(s_{\mathcal{P}}, \pi, t) + \\ &\quad \gamma_1 \max_{\bar{\pi} \in 2^{\mathcal{A}([\![t]\!]_s)} \setminus \emptyset} Q_{1i}^{\mu^*}(t, \bar{\pi}) \} \\ &= \sum_{t \in \mathcal{S}_{\mathcal{P}}} [\sum_{\sigma \in \pi} P_1(s_{\mathcal{P}}, \pi, \sigma) P_2(s_{\mathcal{P}}, \sigma, t) \cdot \\ &\quad \{ R(s_{\mathcal{P}}, \pi) + W_{\mathcal{P}}^i(s_{\mathcal{P}}, t) + \\ &\quad \gamma_1 \max_{\bar{\pi} \in 2^{\mathcal{A}([\![t]\!]_s)} \setminus \emptyset} Q_{1i}^{\mu^*}(t, \bar{\pi}) \}] \\ &= R(s_{\mathcal{P}}, \pi) + \\ &\quad \sum_{\sigma \in \pi} P_1(s_{\mathcal{P}}, \pi, \sigma) T_i^{\mu^*}(s_{\mathcal{P}}, \sigma), \end{aligned} \tag{2}$$

where

$$\begin{aligned} T_i^{\mu^*}(s_{\mathcal{P}}, \sigma) &= \sum_{\sigma \in \pi} P_a(s_{\mathcal{P}}, \pi, \sigma) \sum_{t \in \mathcal{S}_{\mathcal{P}}} P_2(s_{\mathcal{P}}, \sigma, t) \cdot \\ &\quad \left\{ W_{\mathcal{P}}^i(s_{\mathcal{P}}, t) + \gamma_1 \max_{\bar{\pi} \in 2^{\mathcal{A}([\![t]\!]_s)} \setminus \emptyset} Q_{1i}^{\mu^*}(t, \bar{\pi}) \right\}. \end{aligned}$$

We learn P_1 and $T_i^{\mu^*}$ in order to calculate $Q_{1i}^{\mu^*}$ from (2). Note that R is given before learning.

From the above discussion, the controller makes two decisions sequentially in the modified MDP $\mathcal{P}$ with Acc^i as shown Fig.1. Such sequential decision making is different from that in normal MDPs. For the two kinds of value functions $Q_{1i}^{\mu^*}$ and $Q_2^{\mu^*}$, we learn an optimal policy as follows. An optimal pattern at the state $s_{\mathcal{P}}$ for the acceptance condition $Acc_{\mathcal{P}}^i$ is given as follows.

$$\mu_1^*(s_{\mathcal{P}}) = \pi_{j^*},$$

where

$$j^* = \min \arg \max_{j \in \{j \mid \pi_j \in 2^{\mathcal{A}([\![s_{\mathcal{P}}]\!]_s)} \setminus \emptyset\}} \max_i Q_{1i}^{\mu^*}(s_{\mathcal{P}}, \pi_j).$$

An optimal action at the state $s_{\mathcal{P}}$ for the acceptance condition $Acc_{\mathcal{P}}^i$ under the optimal pattern $\mu_1(s_{\mathcal{P}})$ is given as follows.

$$\mu_2^*(s_{\mathcal{P}}) = \text{opt\_action}^*(s_{\mathcal{P}}, \mu_1(s_{\mathcal{P}})).$$

Therefore, we obtain the following optimal control policy $\mu^*(s_{\mathcal{P}})$ at each state $s_{\mathcal{P}}$.

$$\mu^*(s_{\mathcal{P}}) = (\mu_1^*(s_{\mathcal{P}}), \mu_2^*(s_{\mathcal{P}})).$$

Note that, even if the pattern $\mu_1^*(s_{\mathcal{P}})$ contains an action that leads to the non-acceptance transition or transitions to a livelock state, the action dose not tend to be selected as the action $\mu_2^*(s_{\mathcal{P}})$ at the second decision making because the expected discounted sum of the control costs for such an action is not minimum in the pattern $\mu_1^*(s_{\mathcal{P}})$. We learn an optimal policy by optimizing these value functions.

Finally, we introduce a method for checking whether an MC $\mathcal{P}_\mu$ generated by a policy μ and a product MDP $\mathcal{P}$ satisfies φ with probability 1. The set $\mathcal{S}_{\mathcal{P}}$ of states of $\mathcal{P}_\mu$ is partitioned into a disjoint union of transient states Tra_μ and irreducible sets of recurrent classes Rec_μ^j as follows [26].

$$\mathcal{S}_{\mathcal{P}} = Tra_\mu \sqcup Rec_\mu^1 \sqcup \cdots \sqcup Rec_\mu^n.$$

We check whether $\mathcal{P}_\mu$ satisfies φ with probability 1 by computing strongly connected components of $\mathcal{P}_\mu$ [1].

Proposition 1: Consider the MDP $\mathcal{M}$ and the LTL formula φ. Let $\mathcal{P}$ be the modified product of $\mathcal{M}$ and $\mathcal{R}_\varphi$, and μ be a policy. $\mathcal{M}_\mu$ satisfies φ with probability 1 iff for all $j \in \{1, \ldots, n\}$, there exists $(\mathfrak{G}_i, \mathfrak{B}_i) \in Acc_{\mathcal{P}}$ such that $\mathfrak{B}_i \cup Tran_{\mu,j}^{rec} = \emptyset$ and $\mathfrak{G}_i \cup Tran_{\mu,j}^{rec} \neq \emptyset$, where

$$\begin{aligned} Tran_{\mu,j}^{rec} &= \{ (s_{\mathcal{P}}, s'_{\mathcal{P}}) \mid {}^\exists s'_{\mathcal{P}} \text{ s.t. } (s_{\mathcal{P}} \in Rec_\mu^j, \\ &\quad s'_{\mathcal{P}} \in Rec_\mu^j) \wedge P_{\mathcal{P}}(s_{\mathcal{P}}, \mu(s_{\mathcal{P}}), s'_{\mathcal{P}}) > 0 \}. \end{aligned}$$

The proof of Proposition 1 is shown in Appendix A. We check whether the policy satisfies the LTL formula based on Proposition 1.

Shown in Algorithm 1 is a modified temporal difference learning algorithm. Shown in Algorithm 2 is an update algorithm used in Algorithm 1. It is assumed that the transition probabilities and the control cost function of $\mathcal{M}$ are unknown a priori. Let α_1 and $\alpha_2 \in [0, 1]$ be the rates of the update of

Algorithm 1: A proposed algorithm based on RL for $\mathcal{P}$

```
Input : $s'_{\mathcal{P}}$          // A current state on $\mathcal{P}$
Output: $a'$        // A selected action at $s'_{\mathcal{P}}$
Persistent variables:
• Functions $Q_2, T_i, Q_{1i}, R_{2i}, R_{COST}, Pro_1, Pro_2$ are initialized as 0.
• A frequency counter table of a pair of a state $[\![s_{\mathcal{P}}]\!]_s$ and an action $a$ $N_{sa}([\![s_{\mathcal{P}}]\!]_s, a)$ is initialized as the empty table.
• A frequency counter table of a pair of $([\![s_{\mathcal{P}}]\!]_s, a)$ and $[\![s_{\mathcal{P}}]\!]'_s$ $N_{s'|sa}([\![s_{\mathcal{P}}]\!]_s, a, [\![s_{\mathcal{P}}]\!]'_s)$ is initialized as the empty table.
• An Estimated optimal policy $\mu_{op} = (\mu_{1op}, \mu_{2op})$ is initialized.
• $s_{\mathcal{P}}$, $\pi$ and $a$ are initialized as NULL.
1  if ResetConditionMet() returns TRUE then
2      $s'_{\mathcal{P}}$ =ResetRabinState($s'_{\mathcal{P}}$);
3  else if $s_{\mathcal{P}} \neq NULL$ then
4      UpdateFunctions($s_{\mathcal{P}}$, $a$, $s'_{\mathcal{P}}$);
                       // See algo. 2
5      $\mu_{1op}(s_{\mathcal{P}})$ = opt_pattern $(s_{\mathcal{P}})$;
6      $\mu_{2op}(s_{\mathcal{P}}) = \sigma$ s.t. $Pro_1(s_{\mathcal{P}}, \pi_{op}(s_{\mathcal{P}}), \sigma) = 1$;
7  if $s'_{\mathcal{P}}$ has already visited before then
8      $\pi'$ is updated from $\mu_{1op}(s'_{\mathcal{P}})$ using the $\epsilon$-greedy method and $a'$ is updated s.t. $Pro_1(s'_{\mathcal{P}}, \pi', a') = 1$.;
9  else
10     $\pi'$ is updated at random and $a'$ is determined from $\pi'$ at random.;
11 $s_{\mathcal{P}} = s'_{\mathcal{P}}, \pi = \pi', a = a'$;
```

Q_{1i} and Q_2, respectively. The function ResetConditionMet() at line 1 in Algorithm 1 returns TRUE if the behavior is livelock in a sufficiently long specified time interval. The function ResetRabinState($s'_{\mathcal{P}}$) at line 2 in Algorithm 1 initializes the current state of $\mathcal{R}_{\varphi}$. Using these functions, the agent avoids visiting states that need not explore. These functions are valid when φ is constructed from $\mathbf{FG}\varphi'$, $\mathbf{GF}\varphi'$, $\mathbf{F}\varphi'$, or $\mathbf{G}\varphi$ because the updates of estimation functions are independent of the initial state for these formulas. The function opt_action used at line 12 in Algorithm 2 is defined as follows.

$$\text{opt\_action}\,(s_{\mathcal{P}}, \pi) = \sigma_{j^*}$$

where

$$j^* \;=\; \min\arg \min_{j \in \{j \;|\; \sigma_j \in \pi\}} \sum_{t \in \mathcal{P}} Pro_2\,(s_{\mathcal{P}}, \sigma_j, t) \cdot \left\{ R_{COST}\,([\![s_{\mathcal{P}}]\!]_s, \sigma_j) + \gamma_2 \min_{\bar{\pi} \in 2^{\mathcal{A}([\![t]\!]_s)} \setminus \emptyset} Q_2(t, \bar{\pi}) \right\}.$$

The function opt_pattern used at line 5 in Algorithm 1 is defined as follows.

$$\text{opt\_pattern}\,(s_{\mathcal{P}}) = \pi_{j^*}$$

Algorithm 2: An update function of N_{sa}, $N_{s'|sa}$, Pro_2, R_{COST}, Q_2, Pro_1, T_i and Q_{1i} : UpdateFunctions($s_{\mathcal{P}}$, a, $s'_{\mathcal{P}}$)

```
Input  : $s_{\mathcal{P}}$, $a$ and $s'_{\mathcal{P}}$
Output: NULL
Persistent variables:
• Functions $Q_2, T_i, Q_{1i}, R_{2i}, R_{COST}, Pro_1, Pro_2$ are common to ones in Algorithm 1.
• Tables $N_{sa}$ and $N_{s'|sa}$ are common to ones in Algorithm 1.
1  $N_{sa}([\![s_{\mathcal{P}}]\!]_s, a) \leftarrow N_{sa}([\![s_{\mathcal{P}}]\!]_s, a) + 1$;
2  $N_{s'|sa}([\![s_{\mathcal{P}}]\!]_s, a, [\![s'_{\mathcal{P}}]\!]_s) \leftarrow N_{s'|sa}([\![s_{\mathcal{P}}]\!]_s, a, [\![s'_{\mathcal{P}}]\!]_s) + 1$;
3  for all $t$ s.t. $N_{s'|sa}([\![s_{\mathcal{P}}]\!]_s, a, [\![t]\!]_s) \neq 0 \;\wedge\; [\![t]\!]_q = \delta([\![s_{\mathcal{P}}]\!]_q, \mathcal{L}([\![t]\!]_s))$ do
4      $Pro_2(s_{\mathcal{P}}, a, t) \leftarrow N_{s'|sa}([\![s_{\mathcal{P}}]\!]_s, a, [\![t]\!]_s) / N_{sa}([\![s_{\mathcal{P}}]\!]_s, a)$;
5  $R_{COST}([\![s_{\mathcal{P}}]\!]_s, a) \leftarrow g([\![s_{\mathcal{P}}]\!]_s, a)$;
6  if Updating $Q_2(s_{\mathcal{P}}, \pi)$ is the first time. then
7      $Q_2(s_{\mathcal{P}}, \pi) \leftarrow R_{COST}\,([\![s_{\mathcal{P}}]\!]_s, a)$;
8  else
9      $Q_2(s_{\mathcal{P}}, \pi) \leftarrow Q_2(s_{\mathcal{P}}, \pi) + \alpha_2[\min_{\sigma \in \pi} \sum_{t \in \mathcal{S}_{\mathcal{P}}} Pro_2(s_{\mathcal{P}}, \sigma, t)(R_{COST}([\![s_{\mathcal{P}}]\!]_s, \sigma) + \gamma_2 \min_{\bar{\pi} \in 2^{\mathcal{A}([\![t]\!]_s)} \setminus \emptyset} Q_2(t, \bar{\pi})) - Q_2(s_{\mathcal{P}}, \pi)]$;
10 for all $t \in \mathcal{S}_{\mathcal{P}}, \bar{\pi} \in 2^{\mathcal{A}([\![t]\!]_s)} \setminus \emptyset$ s.t. $([\![t]\!]_s = [\![s_{\mathcal{P}}]\!]_s \wedge a \in \bar{\pi}) \vee (^{\exists}\sigma \in \bar{\pi}$ s.t. $Pro_2(t, \sigma, s_{\mathcal{P}}) \neq 0)$ do
11     for all $\sigma \in \bar{\pi}$ do
12         if $\sigma = opt\_action(t, \bar{\pi})$ then
13             $Pro_1(t, \bar{\pi}, \sigma) = 1$;
14         else
15             $Pro_1(t, \bar{\pi}, \sigma) = 0$;
16 for all $i \in \{1, \ldots, n_{Acc}\}$ do
17     $R_{2i}(s_{\mathcal{P}}, s'_{\mathcal{P}}) \leftarrow W^i_{\mathcal{P}}(s_{\mathcal{P}}, s'_{\mathcal{P}})$;
18     if Updating $T_i(s_{\mathcal{P}}, a)$ is the first time then
19         $T_i(s_{\mathcal{P}}, a) \leftarrow R_{2i}(s_{\mathcal{P}}, s'_{\mathcal{P}})$;
20     else
21         $T_i(s_{\mathcal{P}}, a) \leftarrow T_i(s_{\mathcal{P}}, a) + \alpha_1[R_{2i}(s_{\mathcal{P}}, s'_{\mathcal{P}}) + \gamma_1 \max_{\bar{\pi} \in 2^{\mathcal{A}([\![s'_{\mathcal{P}}]\!]_s)} \setminus \emptyset} Q_{1i}(s'_{\mathcal{P}}, \bar{\pi}) - T_i(s_{\mathcal{P}}, a)]$;
22     for all $\forall t \in \mathcal{S}_{\mathcal{P}}, \bar{\pi} \in 2^{\mathcal{A}([\![t]\!]_s)} \setminus \emptyset$ s.t. $(t = s_{\mathcal{P}} \wedge a \in \bar{\pi}) \vee (^{\exists}\sigma \in \bar{\pi}$ s.t. $Pro_2(t, \sigma, s_{\mathcal{P}}) \neq 0)$ do
23         $Q_{1i}(t, \bar{\pi}) \leftarrow R(t, \bar{\pi}) + \sum_{\sigma \in \bar{\pi}} Pro_1\,(t, \bar{\pi}, \sigma)\, T_i\,(t, \sigma)$/;
```

where

$$j^* = \min\arg \max_{j \in \{j \;|\; \pi_j \in 2^{\mathcal{A}([\![s_{\mathcal{P}}]\!]_s)} \setminus \emptyset\}} \max_i Q_{1i}(s_{\mathcal{P}}, \bar{\pi}_j).$$

V. Illustrative example

In this section, as an example, we consider a 5×5 grid world, which is given in [20], where an agent is required to visit cells labeled A and B infinitely often, while avoiding

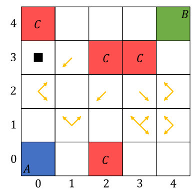


Fig. 2: A grid world example with arrows implies control costs 1000. An initial state's location is $(0, 3)$ denoted by a solid square.

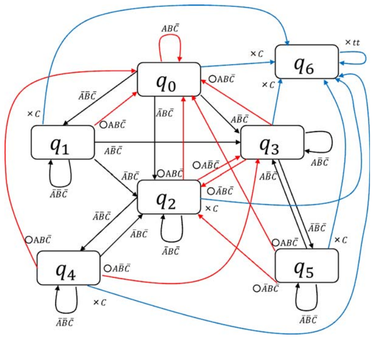


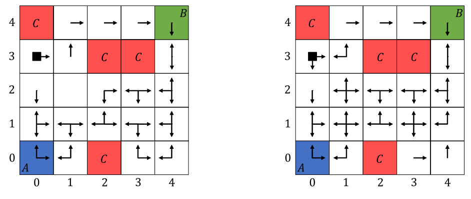
Fig. 3: A DRA constructed from φ. There is only one valid acceptance condition. Blue, red and black arrows represent non-acceptance, acceptance, and neutral transitions, respectively.

cell labeled C. This requirement is described by the following LTL formula φ.

$$\varphi = \mathbf{GF}A \ \wedge \ \mathbf{GF}B \ \wedge \ \mathbf{G}\neg C.$$

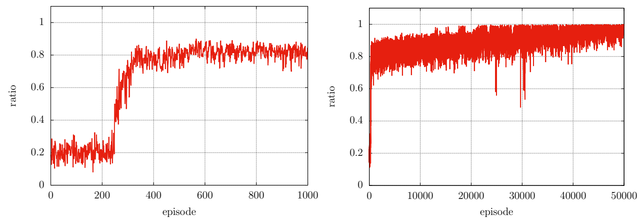
Fig. 3 shows the DRA constructed from φ. The agent selects an action from 4 actions in each cell: upper right, lower right, lower left, and upper left. The action "Upper right" causes the agent to move rightward, upward, and remain stationary with probability (w.p.) 0.4, 0.4, and 0.2, respectively. The agent does not move upper-leftward because a probability such that the agent moves to it in one step is 0. If a wall is located in the agent's right cell, the agent moves upward w.p. 0.8 and remains stationary w.p. 0.2. If it is located above, the agent moves rightward w.p. 0.8 and remains stationary w.p. 0.2. If the agent is in the upper right corner, it remains stationary w.p. 1. The probabilities for the other actions are determined in the same way. The yellow arrows in the Fig. 2 represent control costs of 1000. When the agent selects an action corresponding to the arrow, it obtains the control cost. When the agent selects the others, the obtained control cost is 0.

Simulation parameters are set as follows. For both cases (a) and (b), the number of the steps in each episode is 1000, the function ResetConditionMet() returns TRUE if the agent is in a livelock state by 4 steps, the positive reward ω_G is 100, the negative reward ω_B is -1000, the constant R_C is -6, the discount rates are $\gamma_1 = \gamma_2 = 0.99$ and $\epsilon = 1$. For the case (a), the maximum values of the learning rates α_1 and α_2 are 0.1, their minimum values are 0.01. For the case (b), the maximum values of the learning rates α_1 and α_2 are 0.1, their



(a) episodes=1000 (b) episodes=50000

Fig. 4: Simulation results of the projection of the run for 5000 steps by the estimated optimal policy.



(a) episodes=1000 (b) episodes=50000

Fig. 5: Simulation results of concordance ratio between a obtained optimal action and a calculated optimal action by DP.

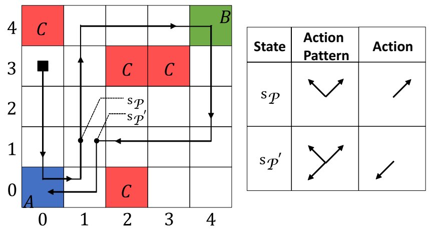


Fig. 6: A sample path for the generated policy of the case (b).

minimum values are 0.0001. We linearly decreased the rates while learning the policy.

Shown in Fig. 4 are the results of the learning algorithm. The arrows represent direction for which the agent moved for 5000 steps by the obtained optimal policy from the initial state. We confirmed that both policies satisfy the LTL formula φ with probability 1. Shown in Fig. 5 is a ratio η defined by the following equation for each episode ep.

$$\eta(ep) = \frac{X_1(ep)}{X_{21}(ep) - X_{22}(ep)},$$

where $X_1(ep)$ is the number of times that matched between a obtained optimal action and an optimal action calculated by dynamic programming(DP) at the non-livelock state in the episode ep, the livelock state is a state $s_{\mathcal{P}}$ such that $[\![s_{\mathcal{P}}]\!]_q = q_6$ in this example, $X_{21}(ep)$ is the total number of steps in the episode ep, that is, $X_{21}(ep)$ equals to 1000, $X_{22}(ep)$ is in the episode ep the number of times that the agent is in the livelock state. Shown in Fig. 6 is a sample path for the obtained policy of the case (b). As you can see on cell $(1, 1)$, the optimal policy depend on a past behavior of the agent.

VI. Conclusion

We considered an uncertain MDP with control cost and an LTL control specification. We have proposed a method for designing a policy based on RL such that the controlled modified product MDP satisfies the control specification with probability 1 and the policy minimizes the expected discounted sum of the costs. In the method, we transformed the problem into the optimization of the value functions and we applied temporal difference learning algorithm. It is future work to prove a condition for which the proposed algorithm obtains an optimal policy. It is also future work to prove the convergence of the value function about control costs.

Acknowledgment

This research was supported in part by JSPS KAKENHI Grant Number 24360164.

References

[1] C. Baier and J. Katoen, *Principles of Model Checking*, The MIT Press, 2008.

[2] W. Tichakorn, T. Ufuk, and M. M. Richard, “Receding Horizon Temporal Logic Planning,” *IEEE Trans. Automat. Contr.*, vol. 57, no. 11, pp. 2817–2830, 2012.

[3] X. Ding, M. Lazar, and C. Belta, “LTL receding horizon control for finite deterministic systems,” *Automatica*, vol. 50, no. 2, pp. 399–408, 2014.

[4] E. M. Wolff, U. Topcu, and R. M. Murray, “Robust control of uncertain Markov Decision Processes with temporal logic specifications,” in *Proc. 51th IEEE Conference on Decision and Control*, pp. 3372–3379, 2012.

[5] S. Jiang and R. Kumar, “Supervisory Control of Discrete Event Systems with CTL* Temporal Logic Specifications,” *SIAM Journal on Control and Optimization*, vol. 44, no. 6, pp. 2079–2103, 2006.

[6] S. Karaman and E. Frazzoli, “Sampling-based Motion Planning with Deterministic μ-Calculus Specifications,” in *Proc. 48th IEEE Conference on Decision and Control*, pp. 2222–2229, 2009.

[7] E. F. Gerogios, G. Antoine, K. Hadas, and J. P. George, “Temporal logic motion planning for dynamic robots,” *Automatica*, vol. 45, no. 2, pp. 343–352, 2009.

[8] E. F. Gerogios and J. P. George, “Robustness of Temporal Logic Specifications for Continuous-Time Signals,” *Theoretical Computer Science*, vol. 410, no. 42, pp. 4262–4291, 2009.

[9] M. Kloetzer and C. Belta, “A Fully Automated Framework for Control of Linear Systems from Temporal Logic Specifications,” *IEEE Trans. Automat. Contr.*, vol. 53, no. 1, pp. 287–297, 2008.

[10] I. Filippidis, D. V. Dimarogonas, and K. J. Kyriakopoulos, “Decentralized multi-agent control from local LTL specifications,” in *Proc. 51th IEEE Conference on Decision and Control*, pp. 6235–6240, 2012.

[11] M. Lahijanian, S. B. Andersson, and C. Belta, “A Probabilistic Approach for Control of a Stochastic System from LTL Specifications,” in *Proc. 48th IEEE Conference on Decision and Control*, pp. 2236–2241, 2009.

[12] M. Svorenova, I. Cerna, and C. Belta, “Optimal Temporal Logic Control for Deterministic Transition Systems With Probabilistic Penalties,” *IEEE Trans. Automat. Contr.*, vol. 60, no. 6, pp. 1528–1541, 2015.

[13] A. Sakakibara, S. Pruekprasert, and T. Ushio, “Optimal Directed Control of Discrete Event Systems with Linear Temporal Logic Constraints,” to appear in *Proc. ETFA*, 2015.

[14] X. Ding, S. L. Smith, C. Belta, and D. Rus, “Optimal Control of Markov Decision Processes With Linear Temporal Logic Constraints,” *IEEE Trans. Automat. Contr.*, vol. 59, no. 5, pp. 1244–1257, 2014.

[15] A. Nilim and L. El Ghaoui, “Robust Control of Markov Decision Processes with Uncertain Transition Matrices,” *Oper. Res.*, vol. 53, no. 5, pp. 780–798, 2005.

[16] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*, 2nd ed. Springer, 2006.

[17] M. Y. Vardi, “An Automata-Theoretic Approach to Linear Temporal Logic,” *Banff Higher Order Workshop 1995*, pp. 238–266, 1995.

[18] S. Safra, “On the complexity of omega -automata,” in *Proc. 29th IEEE Annual Symposium on Foundations of Computer Science*, pp. 319–327, 1988.

[19] J. Esparza and J. Kretínský, “From LTL to Deterministic Automata: A Safraless Compositional Approach,” *CoRR*, vol. abs/1402.3388, 2014.

[20] D. Sadigh, E. S. Kim, S. Coogan, S. S. Sastry, and S. A. Seshia, “A Learning Based Approach to Control Synthesis of Markov Decision Processes for Linear Temporal Logic Specifications,” in *Proc. 53rd IEEE Conference on Decision and Control*, pp. 1091–1096, 2014.

[21] A. Jones, D. Aksaray, Z. Kong, M. Schwager, and C. Belta, “Enforcing temporal logic specifications via reinforcement learning,” in *Proc. 18th International Conference on Hybrid Systems : Computation and Control*, pp. 279–280, 2015.

[22] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, vol. 1, 3rd ed. Athena Scientific, 2005.

[23] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, A Bradford Book, 1998.

[24] T. Ushio and T. Yamasaki, “Supervisory control of partially observed discrete event systems based on a reinforcement learning,” in *Proc. International Conference on Systems, Man, and Cybernetics*, pp. 2956–2961, 2003.

[25] T. Ushio and T. Yamasaki, “Reinforcement Learning of Optimal Supervisor Based on Laguage Measure,” in *Proc. 44th IEEE Conference on Decision and Control and European Control Conference*, pp. 126–131, 2005.

[26] D. Richard, *Essentials of stochastic processes*, 2nd ed. Springer, 2012.

Appendix A
Proof of Proposition 1

From Definition 1, we obtain the following fact:

Fact 1: If

$$Pr(\{r_\delta^\mu : {}^\exists Acc_\times^i \text{ s.t. } \inf(r_\delta^\mu) \cap \mathfrak{G}_i \neq \emptyset \ \wedge \\ \inf(r_\delta^\mu) \cap \mathfrak{B}_i = \emptyset\}) = 1,$$

then controlled $\mathcal{M}_\varphi$ satisfies φ with probability 1, where r_δ^μ is a sequence of transitions of $\mathcal{P}_\mu$.

($\Leftarrow$) : Suppose that for all $j \in \{1, \ldots, n\}$, there exists $(\mathfrak{G}_i, \mathfrak{B}_i) \in Acc_\mathcal{P}$ such that $\mathfrak{B}_i \cap Tran_{\mu,j}^{rec} = \emptyset \ \wedge \ \mathfrak{G}_i \cap Tran_{\mu,j}^{rec} \neq \emptyset$, we have $Pr(\{r_\delta^\mu : {}^\exists Acc_\times^i \text{ s.t. } \inf(r_\delta^\mu) \cap \mathfrak{G}_i \neq \emptyset \ \wedge \ \inf(r_\delta^\mu) \cap \mathfrak{B}_i = \emptyset\}) = 1$. By Fact 1, we have the conclusion.

($\Rightarrow$) : we prove by contradiction. Suppose that $\mathcal{M}_\mu$ satisfies φ with probability 1 and there exists $j \in \{1, \ldots, n\}$ such that for all $(\mathfrak{G}_i, \mathfrak{B}_i) \in Acc_\mathcal{P}$, $\mathfrak{B}_i \cap Tran_{\mu,j}^{rec} \neq \emptyset$ or $\mathfrak{G}_i \cap Tran_{\mu,j}^{rec} = \emptyset$. We consider the case where $\mathfrak{B}_i \cap Tran_{\mu,j}^{rec} \neq \emptyset$ holds. Then,

$$\begin{aligned} & Pr(\{r_\delta^\mu : {}^\exists Acc_\times^i \text{ s.t. } \inf(r_\delta^\mu) \cap \mathfrak{G}_i \neq \emptyset \ \wedge \\ & \qquad\qquad \inf(r_\delta^\mu) \cap \mathfrak{B}_i = \emptyset\}) \\ \leq \ & Pr(\{r_\delta^\mu : {}^\exists Acc_\times^i \text{ s.t. } \inf(r_\delta^\mu) \cap \mathfrak{B}_i = \emptyset\}) \\ < \ & 1 \quad (\because \mathfrak{B}_i \cap Tran_{\mu,j}^{rec} \neq \emptyset) \end{aligned} \tag{3}$$

From Fact 1 and Eq. (3), we have a contradiction. We consider the case where $\mathfrak{G}_i \cup Tran_{\mu,j}^{rec} = \emptyset$ holds. Then, by the similar discussion, we have a contradiction.

Therefore, Proposition 1 holds.