特　別　研　究　報　告

題　目

This Is

My Bachelor Thesis

指 導 教 員

Professor Jane Doe

Associate Professor Joe Smith

報 告 者

John Doe

平成 28 年 2 月吉日

大阪大学基礎工学部システム科学科
知能システム学コース

Abstract

This is abstruct.

Contents

Chapter 1

Introduction

Temporal logic has been developed in computer engineering as a useful formalism of formal specifications [1, 2]. A merit of temporal logics is its resemblance to natural languages and it has been widely used in several other areas of engineering. Especially, a complicated mission or task in computer-controlled systems such as robots can be described by a temporal logic specification precisely and many synthesis algorithms of a controller or a planner that satisfies the specification have been proposed [3, 4, 5, 6]. Linear temporal logic (LTL) is often used as a specification language because of its rich expressiveness. It can explain many important ω-regular properties such as liveness, safety, and persistence [1]. 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, 7]. Previously, in the synthesis of a control policy for the LTL specification, we model a controlled system by a transition system that abstracts its dynamics, construct a product automaton of the transition system and the ω-automaton corresponding to the LTL specification, and compute a winning strategy of a game over the product automaton [7].

Because of inherent stochasticity of controlled systems, we often use a Markov decision process (MDP) as a finite-state abstraction of the controlled systems [8]. In the case where the probabilities are unknown a priori, we have two approaches to the synthesis of the control policy. One is robust control where we assume that state transition probabilities are in uncertainty sets [9] while the other is learning using samples [10].

Reinforcement learning (RL) is a useful approach to learning an optimal policy from sample behaviors of the controlled system [11]. 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, which depends on the acceptance condition of an ω-automaton converted from the LTL specification. A reward function based on the acceptance condition of a Rabin automaton was proposed in [10]. It was applied to a control problem where the controller optimizes a given control cost under the LTL constraint [12].

Recently, a limit-deterministic Büchi automaton (LDBA) or generalized one (LDGBA) is paid much attention to as an ω-automaton corresponding to the LTL specification [13]. The RL-based approaches to the synthesis of a control policy using LDBAs have been proposed in [14, 15, 16, 17]. In [15, 17], they use a LDBA. However, when constructing a Büchi automaton (BA) from a generalized Büchi automaton (GBA), the order of visits to accepting sets of the BA is fixed. The construction causes the sparsity of the reward based on the acceptance condition of a BA. On the other hand, 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 [14, 16]. 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. As a result, we can improve the sparsity of rewards and expand the class of policies that satisfy the LTL specification compared to [14].

Chapter 2

Preliminaries

2.1 Markov Decision Processes

We define a controlled system as a labeled Markov decision process.

Definition 2.1 (Labeled Markov Decision Process) A (labeled) Markov decision process (MDP) is a tuple $M = (S, A, \mathcal{A}, P, s_{init}, AP, L)$, where S is a finite set of states, A is a finite set of actions, $\mathcal{A} : S \to 2^A$ is a mapping that maps each state to the set of possible actions at the state, $P : S \times S \times A \to [0,1]$ is a transition probability such that $\sum_{s' \in S} P(s'|s,a) = 1$ for any state $s \in S$ and any action $a \in \mathcal{A}(s)$, $s_{init} \in S$ is the initial state, AP is a finite set of atomic propositions, and $L : S \times A \times S \to 2^{AP}$ is a labeling function that assigns a set of atomic propositions to each transition $(s, a, s') \in S \times A \times S$.

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 $i \in \mathbb{N}_0$, where $\mathbb{N}_0$ is the set of natural numbers including zero. 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 2.2 (Policy) 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

$$\pi(\rho, a) = \begin{cases} 1 & \text{if } a = a', \\ 0 & \text{otherwise.} \end{cases}$$

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 $(InfPath^M_\pi, \mathcal{F}_{InfPath^M_\pi}, Pr^M_\pi)$.

Definition 2.3 (Markov chain) A Markov chain induced by an MDP M with a positional policy π is a tuple $MC_\pi = (S_\pi, P_\pi, s_0, AP, L)$, 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 $j \in \{1, \ldots, h\}$, as $S_\pi = T_\pi \sqcup R^1_\pi \sqcup \ldots \sqcup R^h_\pi$ [18]. 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 $\mathcal{R} : S \times A \times S \to \mathbb{R}$, where $\mathbb{R}$ is the set of real numbers. The function denotes the immediate scalar bounded reward received after the agent performs an action a at a state s and reaches a next state s' as a result.

2.2 Reinforcement Learning

Reinforcement learning is the theoretical framework to find a policy maximizing or minimizing an objective function through the iterative interactions between the learner referred to the agent and the controlled system referred to the environment. The interaction is that the agent takes an action on the environment and the environment returns a observation such as a immediate reward or next state. In this section, since we use model-free method in this thesis, we refer the model-free reinforcement learning, which find a policy maximizing or minimizing an objective function without inference the environment implicitly.

2.2.1 Objective functions and an Optimal policy

Definition 2.4 (Expected discounted reward for MDPs) For a policy π on an MDP M, any state $s \in S$, and a reward function $\mathcal{R}$, we define the expected discounted reward as

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

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. Intuitively, the magnitude of the discount factor γ determines how much we consider rewards received in the future. 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.

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

We have the following recursively equation for the state-value function and the action-value function.

$$\begin{aligned}
V^\pi(s) =& \mathbb{E}^\pi[\sum_{n=0}^{\infty} \gamma^n \mathcal{R}(S_n, A_n, S_{n+1}) | S_0 = s] \\
=& \sum_{a \in \mathcal{A}(s)} \pi(s,a) \sum_{s' \in S} P(s'|s,a) \mathbb{E}^\pi[\sum_{n=0}^{\infty} \gamma^n \mathcal{R}(S_n, A_n, S_{n+1}) | S_0 = s, A_0 = a, S_1 = s'] \\
=& \sum_{a \in \mathcal{A}(s)} \pi(s,a) \sum_{s' \in S} P(s'|s,a) \{\mathcal{R}(s,a,s') + \gamma \mathbb{E}^\pi[\sum_{n=0}^{\infty} \gamma^n \mathcal{R}(S_n, A_n, S_{n+1}) | S_1 = s']\} \\
=& \sum_{a \in \mathcal{A}(s)} \pi(s,a) \sum_{s' \in S} P(s'|s,a) \{\mathcal{R}(s,a,s') + \gamma V^\pi(s')\}, \qquad (2.1)
\end{aligned}$$

by the definition of the action-value function, it holds that

$$\begin{aligned}
Q^\pi(s,a) &= \max_{a \in \mathcal{A}(s)} V^\pi(s) \\
&= \sum_{s' \in S} P(s'|s,a) \{\mathcal{R}(s,a,s') + \gamma V^\pi(s')\} \\
&= \sum_{s' \in S} P(s'|s,a) \{\mathcal{R}(s,a,s') + \gamma \sum_{a' \in \mathcal{A}(s')} \pi(s',a') Q^\pi(s',a')\}. \qquad (2.2)
\end{aligned}$$

The above equations are called the *Bellman equation.*

Definition 2.5 (Optimal policy) 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.

We have the following *Bellman optimality functions* by the definition of optimal policies.

$$\begin{aligned}
V^*(s) :=& V^{\pi^*}(s) \\
=& \max_{\pi \in \Pi^{pos}} V^\pi(s) \\
=& \max_{\pi \in \Pi^{pos}} \sum_{a \in \mathcal{A}(s)} \pi(s,a) \sum_{s' \in S} P(s'|s,a) \{\mathcal{R}(s,a,s') + \gamma V^\pi(s')\} \\
=& \max_{a \in \mathcal{A}(s)} [\sum_{s' \in S} P(s'|s,a) \{\mathcal{R}(s,a,s') + \gamma V^{\pi^*}(s')\}], \qquad (2.3)
\end{aligned}$$

$$
\begin{aligned}
Q^*(s,a) :=& Q^{\pi^*}(s,a) \\
=& \max_{\pi \in \Pi^{pos}} Q^\pi(s,a) \\
=& \max_{\pi \in \Pi^{pos}} \sum_{s' \in S} P(s'|s,a)\{\mathcal{R}(s,a,s') + \gamma \sum_{a' \in \mathcal{A}(s')} \pi(s',a') Q^\pi(s',a')\} \\
=& \sum_{s' \in S} P(s'|s,a)\{\mathcal{R}(s,a,s') + \gamma \max_{\pi \in \Pi^{pos}} \sum_{a' \in \mathcal{A}(s')} \pi(s',a') Q^\pi(s',a')\} \\
=& \sum_{s' \in S} P(s'|s,a)\{\mathcal{R}(s,a,s') + \gamma \max_{a \in \mathcal{A}(s')} Q^\pi(s',a')\}. \qquad (2.4)
\end{aligned}
$$

We call V^* and Q^* the optimal state-value function and the optimal action-value function, respectively. $V^*(s)$ represents $Q^*(s,a)$ with an optimal action at the first step. Therefor, for any state $s \in S$, we have

$$V^*(s) = \max_{a \in \mathcal{A}(s)} Q^*(s,a)$$

In words, the set of optimal policies under V^* and the set of optimal polisies under Q^* are the same.

If we know the full and accurate information of the MDP such as the transition probability or the reward function, we can obtain an optimal policy by solving Eqs. 2.3 or 2.4 directly. we usually use *Dynamic Programming* by solving recursively equation such as Eqs 2.3 or 2.4. To find V^π or Q^π for a policy π to solve Eqs. 2.1 or 2.2 is referred to *Policy Evaluation.* For any state $s \in S$ or any state-action pair $(s,a) \in S \times A$, to update the policy π to increase the value of $V^\pi(s)$ or $Q^\pi(s,a)$ is referred to *Policy Improvement.* The method to find an optimal policy to update optimal value function repeatedly in accordance with Eqs. 2.3 ar 2.4 is referred to *Value Iteration.* The method to find an optimal policy to repeat policy evaluation and policy improvement alternately is referred to *Policy Iteration.*

2.2.2 Temporal Difference Learning

If the MDP M is unknown, we can not use dynamic programming such as value iteration or policy iteration to obtain an optimal policy. In the case that the MDP M is unknown, we often use reinforcement learning to find an optimal policy instead of dynamic programming.

Temporal difference learning (TD-learning) is the basic method of model-free reinforcement learning. The method does not require the prior knowledge about the environment and utilize a raw experience by one step. Unlike dynamic programing, we update a value function using the experience in an on-line and incremental manner as follows.

$$\hat{V}^{\pi_k}(s_k) \leftarrow \hat{V}^{\pi_k}(s_k) + \alpha_k\{r_{k+1} + \hat{V}^{\pi_k}(s_{k+1}) - \hat{V}^{\pi_k}(s_k)\}, \tag{2.5}$$

where π_k, s_k, and α_k arre the policy, the state, and the learning ratio at the time step k such that $\alpha \in (0,1]$, respectively, and $r_{k+1} = \mathcal{R}(s_k, a_k, s_{k+1})$. The quantity in the curly bracket in the right hand side is called *TD-error*

$$\Delta_k = r_{k+1} + \hat{V}^{\pi_k}(s_{k+1}) - \hat{V}^{\pi_k}(s_k). \tag{2.6}$$

TD-error represents the difference between the current estimated value function and the better estimated value function of current state based on the actual experience, namely $r_{k+1} + \hat{V}^{\pi_k}(s_{k+1})$. Intuitively, as the value function repeatedly updated, the errors is gradually reduced. Hence, the magnitude of the learning ratio α_k describes how much we influence the better estimated value at the current state based on the most recent experience on the current value at the current state. The overall procedure TD-learning for a state-value function is given by Algoritym .

TD-learning methods for an action-value function are classified as two main learning methods that are referred to *Q-learning* and *SARSA*.

2.3 Stochastic Discrete Event Systems

We represent a stochastic discrete event system (DES) as an MDP.

Definition 2.6 (Stochastic discrete event system) A DES is a tuple $D = (S, E, \mathcal{E}, P_T, P_E, s_{init}, AP, L)$, where S is a finite set of states; E is a finite set of events; $\mathcal{E} : S \to E$ is a mapping that maps each state to the set of feasible events at the state; $P_T : S \times S \times E \to [0,1]$ is a transition probability such that $\sum_{s' \in S} P_T(s'|s,e) = 1$ for any state $s \in S$ and any event $e \in \mathcal{E}(s)$ and $P_T(s'|s,e) = 0$ for any $e \notin \mathcal{E}(s)$; $P_E : E \times S \times 2^E \to [0,1]$ is the probability that an event occurs under a subset $\pi \in \mathcal{E}(s)$ of events allowed to occur at the state $s \in S$ such that $\Sigma_{e\in\pi} P_E(e|s,\pi) = 1$ and we call the subset the control pattern; for any $(s', s, \pi) \in S \times S \times 2^E$, we define the probability $P : S \times S \times 2^E \to [0,1]$ such that $P(s'|s,\pi) = \sum_{e\in\pi} P_E(e|s,\pi)P_T(s'|s,e)$ and $\Sigma_{s'\in S} P(s'|s,\pi) = 1$; $s_{init} \in S$ is the initial state; AP is a finite set of atomic propositions; and $L : S \times E \times S \to 2^{AP}$ is a labeling function that assigns a set of atomic propositions to each transition $(s, e, s') \in S \times E \times S$. We assume that E can be partitioned into the set of controllable events E_c and the set of uncontrollable events E_{uc} such that $E_c \cup E_{uc} = E$ and $E_c \cap E_{uc} = \emptyset$. Note that each event e occurs probabilistically depending on only the current state and the subset of feasible events at the state given by a controller.

In the DES D, an infinite path for the DES starting from a state $s_0 \in S$ is defined as a sequence $\rho^D = s_0\pi_0e_0s_1\ldots \in S(2^EES)^\omega$ such that $P_E(e_i|s_i,\pi_i) > 0$ and $P_T(s_{i+1}|s_i,e_i) > 0$ for any $i \in \mathbb{N}_0$. A finite path for the DES is a finite sequence in $S(2^EES)^*$. In addition, we sometimes represent ρ^D as ρ^D_{init} to emphasize that ρ^D starts from $s_0 = s_{init}$. For a path $\rho^D = s_0\pi_0e_0s_1\ldots$, we define the corresponding labeled path $L(\rho^D) = L(s_0,e_0,s_1)L(s_1,e_1,s_2)\ldots \in (2^{AP})^\omega$. For simplicity, we often represent infinite (resp., finite) path for the DES as infinite (resp., finite) path and omit superscript D of the paths. $InfPath^D$ (resp., $FinPath^D$) is defined as the set of infinite (resp., finite) paths starting from $s_0 = s_{init}$ in the DES D. For each finite path ρ, $last(\rho)$ denotes its last state.

We define the supervisor as a controller for the DES that restricts the behaviors of the DES to satisfy a given specification.

Definition 2.7 (Supervisor) For the DES D, a supervisor $SV : FinPath^D \to 2^E$ is defined as a mapping that maps each finite path to a set of allowed events at the finite path and we call the set the control pattern. In the following, the supervisor we consider is *state-based*, namely for any $\rho \in FinPath^D$, $SV(\rho) = SV(last(\rho))$. Note that the relation $E_{uc} \subset SV(\rho) \subset E$ holds for any $\rho \in FinPath^D$. Let $InfPath^D_{SV}$ (resp., $FinPath^D_{SV}$) be the set of infinite (resp., finite) paths starting from $s_0 = s_{init}$ in the DES D under a supervisor SV. The behavior of an DES D under a supervisor SV is defined on a probability space $(InfPath^D_{SV}, \mathcal{F}_{InfPath^D_{SV}}, Pr^D_{SV})$.

We consider the objective function similar to the *Bellamn optimality function* defined by the definition 2.5.

Definition 2.8 (Optimal value function for DESs) From the view point of reinforcement learning, the DES can be interpreted as the environment controlled by the supervisor and the supervisor can be interpreted as the policy. We introduce the two following assumptions.

1. The relative frequency of occurrence of each event does not depend on the control pattern.
2. We define a reward function $\mathcal{R} : S \times 2^E \times E \times S \to \mathbb{R}$ and the reward $\mathcal{R}$ can be decomposed into $\mathcal{R}_1$ and $\mathcal{R}_2$. The first reward $\mathcal{R}_1 : S \times 2^E \to \mathbb{R}$ is determined by the control pattern selected by the supervisor, which depends on only the control pattern and the current state. The second reward $\mathcal{R}_2 : S \times E \times S \to \mathbb{R}$ is determined by the occurrence of an event and the corresponding state transition. For any $(s,\pi,e,s') \in S \times 2^E \times E \times S$, we then have

$$\mathcal{R}(s,\pi,e,s') = \mathcal{R}_1(s,\pi) + \mathcal{R}_2(s,e,s'). \tag{2.7}$$

Under the above assumptions, we have the following *Bellman optimality equation*.

$$
\begin{aligned}
Q^*(s,\pi) =& \sum_{s'\in S} P(s'|s,\pi) \left\{ \mathcal{R}(s,\pi,e,s') + \gamma \max_{\pi'\in 2^{\mathcal{E}(s')}} Q^*(s',\pi') \right\} \\
=& \sum_{s'\in S} \sum_{e\in\pi} P_E(e|s,\pi) P_T(s'|s,e) \left\{ \mathcal{R}_1(s,\pi) + \mathcal{R}_2(s,e,s') + \gamma \max_{\pi'\in 2^{\mathcal{E}(s')}} Q^*(s',\pi') \right\} \\
=& \mathcal{R}_1(s,\pi) + \sum_{e\in\pi} P_E(e|s,\pi) \sum_{s'\in S} P_T(s'|s,e) \left\{ \mathcal{R}_2(s'|s,e) + \gamma \max_{\pi'\in 2^{\mathcal{E}(s')}} Q^*(s',\pi') \right\},
\end{aligned}
\tag{2.8}
$$

where $\gamma \in [0,1)$.

We introduce the following function. $T^* : S \times E \to \mathbb{R}$ such that

$$
T^*(s,e) = \sum_{s'\in S} P_T(s'|s,e) \left\{ \mathcal{R}_2(s'|s,e) + \gamma \max_{\pi'\in 2^{\mathcal{E}(s')}} Q^*(s',\pi') \right\}. \tag{2.9}
$$

We then have

$$
Q^*(s,\pi) = \mathcal{R}_1(s,\pi) + \sum_{e\in\pi} P_E(e|s,\pi) T^*(s,e). \tag{2.10}
$$

Definition 2.9 (Optimal supervisor) We define an optimal supervisor SV^* as follows. For any state $s \in S$,

$$
SV^*(s) = \pi \in \arg\max_{\pi\in\mathcal{E}(s)} Q^*(s,\pi), \tag{2.11}
$$

2.4 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 recursively 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 $\bot := \neg\top$, $\varphi_1 \vee \varphi_2 := \neg(\neg\varphi_1 \wedge \neg\varphi)$, and $\varphi_1 \Rightarrow \varphi_2 := \neg\varphi_1 \vee \varphi_2$. The operators $\mathbf{X}$ and $\mathbf{U}$ are called "next" and "until", respectively. Using the operator $\mathbf{U}$, 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 $\rho = s_0a_0s_1\dots$ 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 $\rho[i:] = s_ia_is_{i+1}\dots$. We define the i-th state and i-th suffix of the infinite path for a DES in the same way.

Definition 2.10 (LTL semantics) For an LTL formula φ, an MDP M, and an infinite path $\rho = s_0a_0s_1\dots$ of M with $s_0 \in S$, the satisfaction relation $M, \rho \models \varphi$ is recursively defined as follows.

$$
\begin{aligned}
&M, \rho \models \top, \\
&M, \rho \models \alpha \in AP \Leftrightarrow \alpha \in L(s_0, a_0, s_1), \\
&M, \rho \models \varphi_1 \wedge \varphi_2 \Leftrightarrow M, \rho \models \varphi_1 \wedge M, \rho \models \varphi_2, \\
&M, \rho \models \neg\varphi \qquad \Leftrightarrow M, \rho \not\models \varphi, \\
&M, \rho \models \mathbf{X}\varphi \qquad \Leftrightarrow M, \rho[1:] \models \varphi, \\
&M, \rho \models \varphi_1\mathbf{U}\varphi_2 \;\Leftrightarrow \exists j \geq 0,\; M, \rho[j:] \models \varphi_2 \wedge \forall i, 0 \leq i < j,\; M, \rho[i:] \models \varphi_1.
\end{aligned}
$$

The next operator $\mathbf{X}$ requires that φ is satisfied by the next state suffix of ρ. The until operator $\mathbf{U}$ requires that φ_1 holds true until φ_2 becomes true over the path ρ. For the path in a DES, we define the LTL semantics in the same way. Using the operator $\mathbf{U}$ we can define two temporal operators: (1) *eventually*, $\mathbf{F}\varphi := \top\mathbf{U}\varphi$ and (2) *always*, $\mathbf{G}\varphi := \neg\mathbf{F}\neg\varphi$. In the following, we write $\rho \models \varphi$ for simplicity without referring to MDP M and DES D.

For any policy π and any supervisor SV, we denote the probability of all paths starting from s_{init} on the MDP M (resp., DES) that satisfy an LTL formula φ under the policy π (resp., the supervisor SV) as follows.

$$
\begin{aligned}
Pr^M_\pi(s_{init} \models \varphi) &:= Pr^M_\pi(\{\rho_{init} \in InfPath^M_\pi ; \rho_{init} \models \varphi\}), \\
Pr^D_{SV}(s_{init} \models \varphi) &:= Pr^D_{SV}(\{\rho_{init} \in InfPath^D_{SV} ; \rho_{init} \models \varphi\}).
\end{aligned}
$$

We say that an LTL formula φ is satisfied by a positional policy π (resp., a supervisor SV) if

$$
\begin{aligned}
Pr^M_\pi(s_{init} \models \varphi) > 0, \\
Pr^D_{SV}(s_{init} \models \varphi) > 0.
\end{aligned}
$$

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 Büchi automaton.

Definition 2.11 (Transition-based generalized Büchi automaton) 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, $\delta \subset X \times \Sigma \times X$

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.

Let Σ^ω be the set of all infinite words over Σ and let an infinite run be an infinite sequence $r = x_0\sigma_0x_1 \ldots \in X(\Sigma X)^\omega$ 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 2.12 (Sink state) A sink state in state set X of an augmented tLDBA $\bar{B}_\varphi = (\bar{X}, \bar{x}_{init}, \bar{\Sigma}, \bar{\delta}, \bar{\mathcal{F}})$ is defined as a state such that there exist no accepting transition of $\bar{B}_\varphi$ that is accessible from the state. We denote the set of sink states as $SinkSet$.

Definition 2.13 (Transition-based limit-deterministic generalized Büchi automaton) A tGBA $B = (X, x_{init}, \Sigma, \delta, \mathcal{F})$ is limit-deterministic (tLDBA) if the following conditions hold.

- $\exists X_{initial},\ X_{final} \subset X$ s.t. $X = X_{initial} \cup X_{final} \wedge X_{initial} \cap X_{final} = \emptyset$,
- $F_j \subset X_{final} \times \Sigma \times X_{final}$, $\forall j \in \{1, ..., n\}$,
- $|\{(x, \sigma, x') \in \delta; x' \in X_{initial}\}| \leq 1$, $\forall x \in X_{initial}, \forall \sigma \in \Sigma$,
- $|\{(x, \sigma, x') \in \delta; x' \in X_{final}\}| \leq 1$, $\forall x \in X_{final}, \forall \sigma \in \Sigma$,
- $|\{(x, \sigma, x') \in \delta; x' \in X_{initial}\}| = 0$, $\forall x \in X_{final}, \forall \sigma \in \Sigma$.

A tLDBA is a tGBA whose state set can be partitioned into the initial part $X_{initial}$ and the final part X_{final}, and they are connected by a single "guess". The final part has all accepting sets. The transitions in each part are deterministic. It is known that, for any LTL formula φ, there exists a tLDBA that accepts all words satisfying φ [13]. In particular, we represent a tLGBA recognizing an LTL formula φ as B_φ, whose input alphabet is given by $\Sigma = 2^{AP}$.

Chapter 3

Reinforcement-Learning-Based Synthesis of Control Policy

We introduce an automaton augmented with binary 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 $V = \{(v_1, \dots, v_n)^T \ ; \ v_i \in \{0,1\}, \ i \in \{1, \dots, n\}\}$ 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 tLDBA 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 $e \in \delta$, $visitf(e) = (v_1, \dots, v_n)^T$, where

$$v_i = \begin{cases} 1 & \text{if } e \in F_i, \\ 0 & \text{otherwise.} \end{cases}$$

For any $v \in V$,

$$reset(v) = \begin{cases} \mathbf{0} & \text{if } v = \mathbf{1}, \\ v & \text{otherwise.} \end{cases}$$

For any $v, u \in V$, $Max(v,u) = (l_1, \dots, l_n)^T$, where $l_i = max\{v_i, u_i\}$ for any $i \in \{1, \dots, n\}$.

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 3.1 For a tLDBA $B_\varphi = (X, x_{init}, \Sigma, \delta, \mathcal{F})$, its augmented automaton is a tLDBA $\bar{B}_\varphi = (\bar{X}, \bar{x}_{init}, \bar{\Sigma}, \bar{\delta}, \bar{\mathcal{F}})$, 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 $\bar{\mathcal{F}} = \{\bar{F}_1, \dots, \bar{F}_n\}$ is defined as

$\bar{F}_i = \{((x,v),\bar{\sigma},(x',v')) \in \bar{\delta}\ ;\ (x,\sigma,x') \in F_i,\ v_i = 0,\ visitf((x,\bar{\sigma},x'))_i = 1\}$ for each $i \in \{1,...,n\}$, where $visitf((x,\bar{\sigma},x'))_i$ is the i-th element of $visitf((x,\bar{\sigma},x'))$.

Definition 3.2 Given an augmented tLDBA $\bar{B}_\varphi$ and an MDP M, a tuple $M \otimes \bar{B}_\varphi = M^\otimes = (S^\otimes, A^\otimes, \mathcal{A}^\otimes, s^\otimes_{init}, P^\otimes, \delta^\otimes, \mathcal{F}^\otimes)$ is a product MDP, where $S^\otimes = S \times \bar{X}$ is the finite set of states, $A^\otimes = A$ is the finite set of actions, $\mathcal{A}^\otimes : S^\otimes \to 2^{A^\otimes}$ is the mapping defined as $\mathcal{A}^\otimes((s,\bar{x})) = \mathcal{A}(s)$, $s^\otimes_{init} = (s_{init}, \bar{x}_{init})$ is the initial states, $P^\otimes : S^\otimes \times S^\otimes \times A^\otimes \to [0,1]$ is the transition probability defined as

$$P^\otimes(s^{\otimes\prime}|s^\otimes,a) = \begin{cases} P(s'|s,a) & \text{if } (\bar{x}, L((s,a,s')), \bar{x}') \in \bar{\delta}, \\ 0 & \text{otherwise,} \end{cases}$$

$\delta^\otimes = \{(s^\otimes, a, s^{\otimes\prime}) \in S^\otimes \times A^\otimes \times S^\otimes ; P^\otimes(s^{\otimes\prime}|s^\otimes,a) > 0\}$ is the set of transitions, and $\mathcal{F}^\otimes = \{\bar{F}^\otimes_1, \ldots, \bar{F}^\otimes_n\}$ is the acceptance condition, where $\bar{F}^\otimes_i = \{((s,\bar{x}),a,(s',\bar{x}')) \in \delta^\otimes\ ;\ (\bar{x}, L(s,a,s'), \bar{x}') \in \bar{F}_i\}$ for each $i \in \{1,\ldots,n\}$.

Definition 3.3 The reward function $\mathcal{R} : S^\otimes \times A^\otimes \times S^\otimes \to \mathbb{R}_{\geq 0}$ is defined as

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

where r_p is a positive value.

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, maximizing the expected discounted reward produces a policy satisfying φ.

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

Lemma 3.1 For any policy π and any recurrent class $R^{\otimes i}_\pi$ in the Markov chain $MC^\otimes_\pi$, $MC^\otimes_\pi$ satisfies one of the following conditions.

1. $\delta^\otimes_{\pi,i} \cap \bar{F}^\otimes_j \neq \emptyset$, $\forall j \in \{1,\ldots,n\}$,
2. $\delta^\otimes_{\pi,i} \cap \bar{F}^\otimes_j = \emptyset$, $\forall j \in \{1,\ldots,n\}$.

Lemma 3.1 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 3.1 Let $M^{\otimes}$ be the product MDP corresponding to an MDP M and an LTL formula φ. If there exists a positional policy satisfying φ, then there exists a discount factor γ^* such that any algorithm that maximizes the expected reward with $\gamma > \gamma^*$ will find a positional policy satisfying φ.

We show the overall procedure of our proposed method in Algorithm 1. We employ Q-learning in Algorithm 1, but any algorithms maximizing the expected discounted reward can be applied to our proposed method.

Algorithm 1 RL-based synthesis of control policy on the MDP with the augmented tLDBA.

```
Input: LTL formula $\varphi$ and MDP $M$
Output: Optimal policy $\pi^*$ on the product MDP $M^{\otimes}$
1: Translate $\varphi$ into tLDBA $B_{\varphi}$.
2: Augment $B_{\varphi}$ to $\bar{B}_{\varphi}$.
3: Construct the product MDP $M^{\otimes}$ of $M$ and $\bar{B}_{\varphi}$.
4: Initialize $Q : S^{\otimes} \times A^{\otimes} \to \mathbb{R}_{\geq 0}$.
5: Initialize episode length $T$.
6: while $Q$ is not converged do
7:    $s^{\otimes} \leftarrow (s_{init}, (x_{init}, \mathbf{0}))$.
8:    for $t = 1$ to $T$ do
9:       Choose the action $a$ by a policy $\pi$.
10:      Observe the next state $s^{\otimes\prime}$.
11:      $Q(s^{\otimes}, a) \leftarrow Q(s^{\otimes}, a) + \alpha\{\mathcal{R}(s^{\otimes}, a, s^{\otimes\prime}) + \gamma \max_{a'} Q(s^{\otimes\prime}, a') - Q(s^{\otimes}, a)\}$
12:      $s^{\otimes} \leftarrow s^{\otimes\prime}$
13:   end for
14: end while
```

3.1 Example

In this section, we evaluate our proposed method and compare it with an existing work. We consider a path planning problem of a robot in an environment consisting of eight rooms and one corridor as shown in Fig. 3.1. 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

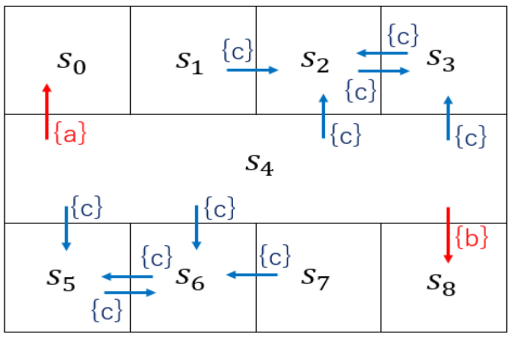


Fig. 3.1 The environment consisting of eight rooms and one corridor. Red arcs are the transitions that we want to occur infinitely often, while blue arcs are the transitions that we never want to occur. s_7 is the initial state.

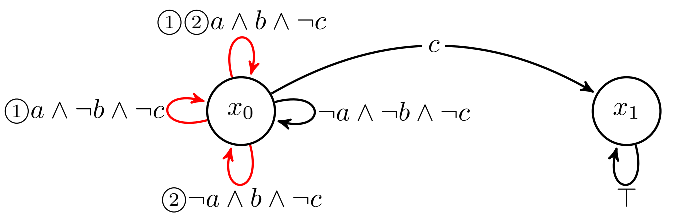

①②$a \wedge b \wedge \neg c$
c
①$a \wedge \neg b \wedge \neg c$
x_0
$\neg a \wedge \neg b \wedge \neg c$
x_1
②$\neg a \wedge b \wedge \neg c$
$\top$

Fig. 3.2 The tLDBA 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, e.g., ①$a \wedge \neg b \wedge \neg c$ means the transition labeled by it belongs to the accepting set F_1.

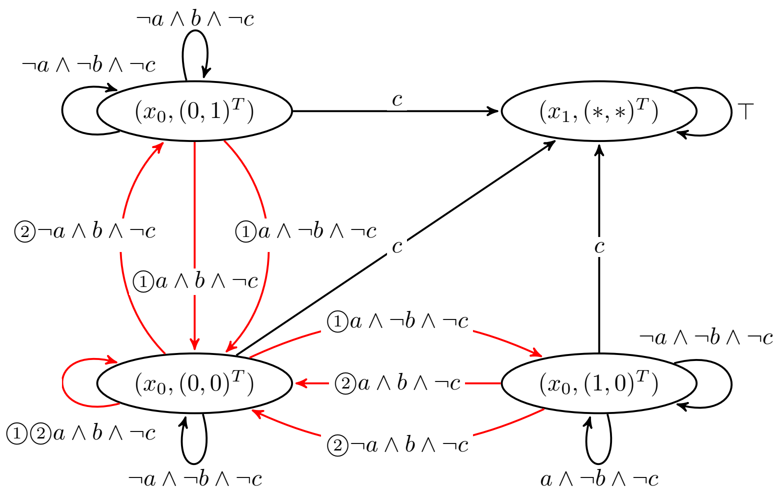

$\neg a \wedge b \wedge \neg c$
$\neg a \wedge \neg b \wedge \neg c$
$(x_0, (0,1)^T)$
c
$(x_1, (*,*)^T)$
$\top$
②$\neg a \wedge b \wedge \neg c$
①$a \wedge \neg b \wedge \neg c$
c
c
①$a \wedge b \wedge \neg c$
①$a \wedge \neg b \wedge \neg c$
$\neg a \wedge \neg b \wedge \neg c$
$(x_0, (0,0)^T)$
②$a \wedge b \wedge \neg c$
$(x_0, (1,0)^T)$
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. 3.3 The augmented automaton for the tLDBA in Fig. 3.2 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 $(x_1, (*,*)^T)$.

state. The labeling function is as follows.

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

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 following LTL formula.

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

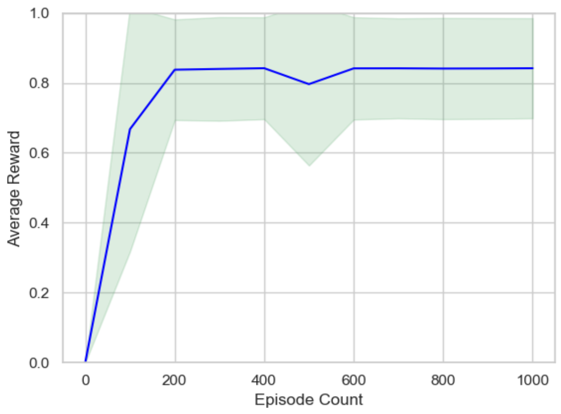
The above LTL formula requires the robot to keep on entering the two rooms s_0 and s_8 from the corridor s_4 regardless of the order of entries, while avoiding entering the four rooms s_2, s_3, s_5, and s_6.

We use Owl [22] to obtain the tLDBA corresponding to the LTL formula. The tLDBA $B_\varphi = (X, x_{init}, \Sigma, \delta, \mathcal{F})$ and its augmented automaton $\bar{B}_\varphi = (\bar{X}, \bar{x}_{init}, \bar{\Sigma}, \bar{\delta}, \bar{\mathcal{F}})$ are shown in Figs. 3.2 and 3.3, respectively. Specifically, the acceptance condition $\mathcal{F}$ of the tLDBA is given by $\mathcal{F} = \{F_1, F_2\}$, where $F_1 = \{(x_0, \{a\}, x_0),\ (x_0, \{a,b\}, x_0)\}$ and $F_2 = \{(x_0, \{b\}, x_0),\ (x_0, \{a,b\}, x_0)\}$.

We use Q-learning 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 visits to state $s^\otimes$ within t time steps [21], and the discount factor $\gamma = 0.9$. The learning rate α varies in accordance with *the Robbins-Monro condition*.

We also evaluate the method by Hasanbeig *et al.*[14] with the same example. They use state-based LDBAs for LTL formulas and construct the product MDP of an MDP and a state-based LDBA to synthesize a policy satisfying the LTL formula. They proposed the accepting frontier function $Acc : X \times 2^X \to 2^X$ where X is the set of states of the state-based LDBA. Under initializing a set of states $\mathbb{F}$ with the union of the all accepting sets of the state-based LDBA, the function receives the state x after each transition and the set $\mathbb{F}$. If x is in $\mathbb{F}$, then Acc removes the accepting sets containing x from $\mathbb{F}$. The reward function is based on the varying set $\mathbb{F}$. We conduct the same example with their method using the tLDBA instead.

Figs. 3.4 and 3.5 show the average reward and the optimal policy, respectively, as a result of the learning when using our proposed method and the method in [14] after 10000 iterations and 1000 episodes. The arithmetic mean of average reward in each episode for 20 learning sessions is displayed per 100 episodes in Fig. 3.4.



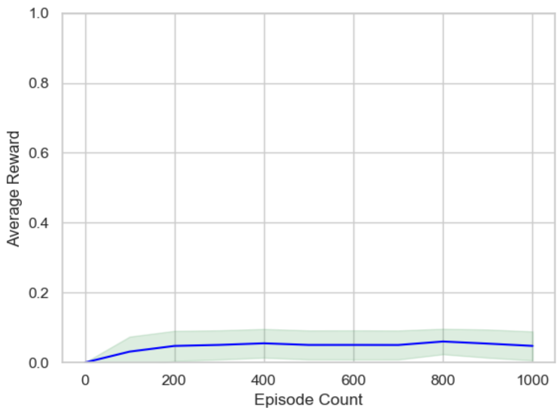
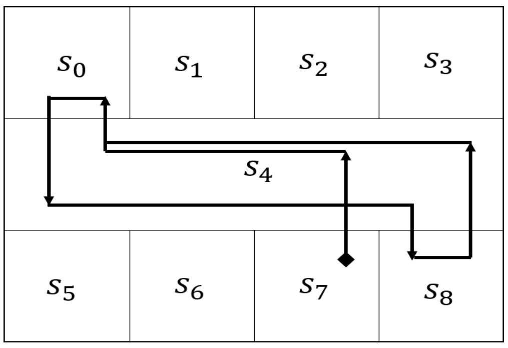


Fig. 3.4 The arithmetic mean of average reward in each episode for 20 learning sessions obtained from our proposed method (left) and the method by Hasanbeig *et al.*[14] (right). They are plotted per 100 episodes and the green areas represent the range of standard deviations.



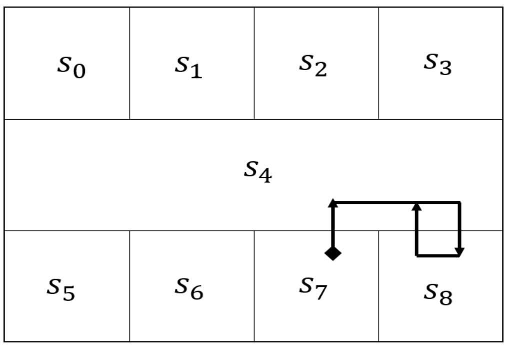


Fig. 3.5 The optimal policy obtained from our proposed method (left) and the method by Hasanbeig *et al.*[14] (right).

The results suggest that our proposed method can synthesize a policy satisfying φ on the MDP, while the method in [14] cannot. This is because it is impossible that the transitions labeled by $\{a\}$ and $\{b\}$ occur from s_4 infinitely often by any positional policy with the tLDBA. In detail, the state of the tLDBA is always x_0 while the agent does not move to states s_2, s_3, s_5, and s_6. Thus, the state of the product MDP is always (s_4, x_0) while the agent stays in s_4. Therefore, the method in [14] may not synthesize policies satisfying LTL specifications depending on the setting of MDPs or LTL specifications.

Chapter 4

4th Chapter

Definition 4.1 The two reward functions $\mathcal{R}_1 : S^{\otimes} \times 2^{E^{\otimes}} \to \mathbb{R}$ and $\mathcal{R}_2 : S^{\otimes} \times E^{\otimes} \times S^{\otimes} \to \mathbb{R}$ are defined as follows.

$$\mathcal{R}_1(s^{\otimes}, \pi) = r_{n1}(|E| - |\pi|), \tag{4.1}$$

where $|E|$ means number of elements in the set E and r_{n1} is a negative value.

$$\mathcal{R}_2(s^{\otimes}, e, s^{\otimes\prime}) = \begin{cases} r_p \text{ if } \exists i \in \{1, \ldots, n\},\ (s^{\otimes}, e, s^{\otimes\prime}) \in \bar{F}_i^{\otimes}, \\ r_{n2} \text{ if } [\![s^{\otimes\prime}]\!]_q \in \mathit{SinkSet}, \\ 0 \ \text{ otherwise}, \end{cases} \tag{4.2}$$

4.1 Reinforcement-learning-based Synthesis of Supervisor

We make the supervisor learn how to give the control patterns to satisfy an LTL specification while keeping costs associated with disabled events low. We use Q-learning to estimate the function T^*. We then use Bayesian inference to robustly estimate the probability P_E. For the inference, we model P_E as Categorical distribution as $p^k_{s,\pi,e}$, where $p^k_{s,\pi,e}$ represents the estimated probability of $P_E(e|s, \pi)$ at the time step k and the prior distribution $\phi^k_{s,\pi}$ for the parameter of $p^k_{s,\pi,e}$ is defined as Dirichlet.

To reflect the events prohibition by the supervisor on the estimated probability of the occurrence of allowed events, we introduce the function $\mathit{RestProb} : (0, 1)^{|E|} \times 2^E \to [0, 1]^{|E|}$ defined as

$$\mathit{RestProb}(\phi_{s,\pi}, \pi)_i = \begin{cases} \dfrac{\phi^i_{s,\pi}}{\sum_{e_j \in \pi} \phi^j_{s,\pi}} & \text{if } e_i \in \pi, \\ 0 & \text{otherwise}, \end{cases} \tag{4.3}$$

where $\phi^i_{s,\pi}$ is the i-th element of $\phi_{s,\pi}$ and $\mathit{RestProb}(\phi_{s,\pi},\pi)_i$ is the i-th element of $\mathit{RestProb}(\phi_{s,\pi},\pi)$.

Let $p^k_{s,\pi}$ denote the probability vector at the time step k as $p^k_{s,\pi} = (p^k_{s,\pi,e_1},\dots,p^k_{s,\pi,e_{|E|}})$. Let $n^k_{s,\pi,e}$ be the number of the occurrence of the event $e \in E$ up to the time step k under the state $s \in S$ and the control pattern $\pi \in \mathcal{E}(s)$, let $n^k_{s,\pi}$ denote $(n^k_{s,\pi,e_1},\dots,n^k_{s,\pi,e_{|E|}})$, and let $\bar{p}^k_{s,\pi}$ denote the expected value of $p^k_{s,\pi}$. The procedure of the inference is shown in Algorithm 1.

Algorithm 2 P_E inference.

```
Input: the event occurrence count $n^k_{s,\pi}$, a threshold $\xi^k_{s,\pi}$ for $p^k_{s,\pi}$
Output: the posterior distribution $p^k_{s,\pi}$
1: repeat
2:     $\phi^k_{s,\pi} \sim Dir(\cdot|n^k_{s,\pi})$
3:     $p^k_{s,\pi} = \mathit{RestProb}(\phi^k_{s,\pi},\pi)$
4: until $||p^k_{s,\pi} - \bar{p}^k_{s,\pi}||_1 < \xi^k_{s,\pi}$
```

Under the estimation of P_E, we use TD-learning to estimate Q^* with the TD-error defined as $\mathcal{R}_1(s^{\otimes},\pi) + \sum_{e\in\pi} p_{s^{\otimes},\pi,e} T(s^{\otimes},e) - Q(s^{\otimes},\pi)$.

We show the all procedure of learning algorithm in Algorithm 3.

4.2 Example

We evaluate the two algorithms by the maze of the cat and the mouse shown in Fig. 4.1. At the beginning, we define the settings for the example. The corresponding DES is as follows. The state set is $S = \{(s^{cat}, s^{mouse}); s^{cat}, s^{mouse} \in \{s_0,s_1,s_2,s_3\}\}$. The set of events (to open the corresponding door) is $E = \{m_0,m_1,m_2,m_3,c_0,c_1,c_2,c_3\}$, where $E_c = \{m_0,m_1,m_2,m_3,c_0,c_1,c_2\}$ and $E_{uc} = \{c_3\}$ and $\mathcal{E}(s) = E$ for any $s \in S$. The initial state is $s_{init} = (s_0,s_2)$. If the door of the room with the cat (resp., mouse) opens, the cat (resp., mouse) moves, with probability 0.95, to the room next to the room with it where the door is open or stays in the same room with probability 0.05. Otherwise, the cat (resp., mouse) stays in the same room with probability 1. The labeling function is

$$L((s,a,s')) = \begin{cases} \{a\} & \text{if } s'_c = s_1, \\ \{b\} & \text{if } s'_m = s_1, \\ \{c\} & \text{if } s'_c = s'_m, \\ \emptyset & \text{otherwise,} \end{cases}$$

where s'_c and s'_m is the next room where the cat and the mouse is, respectively, i.e., $s' = (s'_c, s'_m)$.

Algorithm 3 RL-based synthesis of a supervisor satisfying a given LTL specification.

```
Input: LTL formula $\varphi$, DES $M$
Output: optimal supervisor $SV^*$ on the product DES $M^\otimes$
 1: Convert $\varphi$ into tLDBA $B_\varphi$.
 2: Augment $B_\varphi$ to $\bar{B}_\varphi$.
 3: Construct the product DES $M^\otimes$ of $M$ and $\bar{B}_\varphi$.
 4: Initialize $T : S^\otimes \times E^\otimes \to \mathbb{R}$.
 5: Initialize $Q : S^\otimes \times 2^{E^\otimes} \to \mathbb{R}$.
 6: Initialize $n : S^\otimes \times 2^{E^\otimes} \times E^\otimes \to \mathbb{R}$.
 7: Initialize episode length $L$.
 8: while $Q$ is not converged do
 9:     $s^\otimes \leftarrow (s_{init}, (x_{init}, \mathbf{0}))$.
10:     $t \leftarrow 0$
11:     while $t < L$ and $[\![s^\otimes]\!]_q \notin SinkSet$ do
12:         Choose the control pattern $\pi \in \mathcal{E}(s^\otimes)$ by the supervisor $SV$.
13:         Observe the occurrence of the event $e \in E$.
14:         Observe the next state $s^{\otimes\prime}$.
15:         $T(s^\otimes, e) \leftarrow (1-\alpha)T(s^\otimes, e) + \alpha\{\mathcal{R}_2(s^\otimes, e, s^{\otimes\prime}) + \gamma \max_{\pi' \in 2^{\mathcal{E}(s^{\otimes\prime})}} Q(s^{\otimes\prime}, \pi')\}$
16:         $n(s^\otimes, \pi, e) \leftarrow n(s^\otimes, \pi, e) + 1$
17:         Obtain $p_{s^\otimes,\pi}$ by the $P_E$ inference.
18:         $Q(s^\otimes, \pi) = (1-\beta)Q(s^\otimes, \pi) + \beta\{\mathcal{R}_1(s^\otimes, \pi) + \sum_{e\in\pi} p_{s^\otimes,\pi,e} T(s^\otimes, e)\}$
19:         $s^\otimes \leftarrow s^{\otimes\prime}$
20:         $t \leftarrow t + 1$
21:     end while
22: end while
```

In the example, we want the supervisor to learn to give control patterns satisfying that the cat and the mouse take the food in the room 1 (s_1) avoiding they come across. This is formally specified by the following LTL formula.

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

The tLDBA $B_\varphi = (X, x_{init}, \Sigma, \delta, \mathcal{F})$ corresponding to φ is shown in Fig. 4.2. B_φ has the acceptance condition of two accepting sets.

We use ε-greedy policy and gradually reduce ε to 0 to learn an optimal supervisor asymptotically. We set the rewards $r_p = 10$, $r_{n1} = -0.1.-0.5, and -1$, and $r_{n2} = -100$; the epsilon greedy parameter $\varepsilon = \frac{1}{\sqrt{episode}}$, where *episode* is the number of the current episode; and the discount factor $\gamma = 0.99$. $\xi^k_{s^\otimes,\pi}$ is initially set to 1 and changes to 0.6 during 1/3 to 2/3 of all episodes and to 0.3 after 2/3 of all episodes for any $(s^\otimes, \pi) \in S^\otimes \times 2^{E^\otimes}$. The learning rate α and β vary in accordance with *the Robbins-Monro condition.*

Figs. 4.5, **??**, and **??** show the estimated optimal state value function at the initial state $V(s^{\otimes}_{init})$ with $r_{n1} = -0.1. -0.5$, and -1, respectively, for each episode when learning 5000 iterations and 15000 episodes by the algorithm 3. Fig. 4.8, **??**, and **??** shows the average reward from $\mathcal{R}_2$ and the average cost from $\mathcal{R}_1$ with $r_{n1} = -0.1. -0.5$, and -1, respectively, of 5000 iteration and 1000 episodes by the supervisor obtained from the learning.

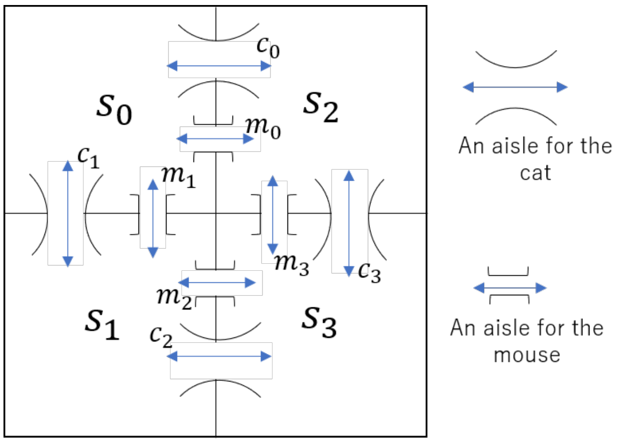
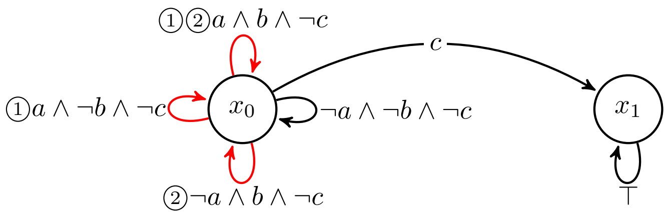


Fig. 4.1 The maze of the cat and the mouse. the initial state of the cat and the mouse is s_0 and s_2, respectively. the food for them is in the room 1 (s_1).



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. 4.2 The tLDBA 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, e.g., ①$a \wedge \neg b \wedge \neg c$ means the transition labeled by it belongs to the accepting set F_1.

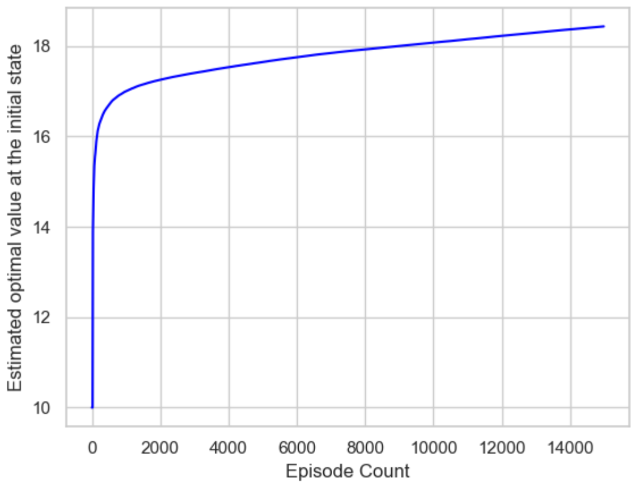


Fig. 4.3 the estimated optimal state value function at the initial state $V(s^{\otimes}_{init})$ with $r_{n1} = -0.1$ when using the algorithm 3.

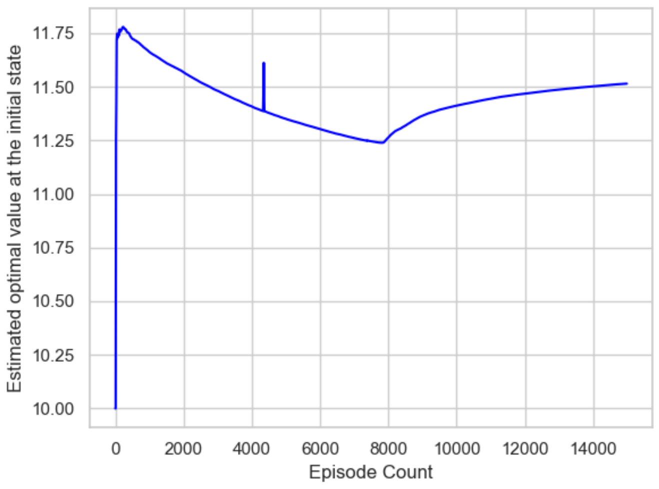


Fig. 4.4 the estimated optimal state value function at the initial state $V(s^{\otimes}_{init})$ with $r_{n1} = -0.5$ when using the algorithm 3.

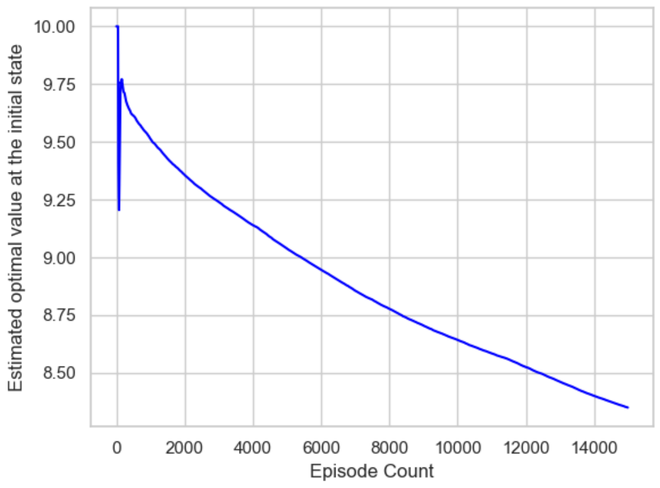


Fig. 4.5 the estimated optimal state value function at the initial state $V(s^{\otimes}_{init})$ with $r_{n1} = -1$ when using the algorithm 3.

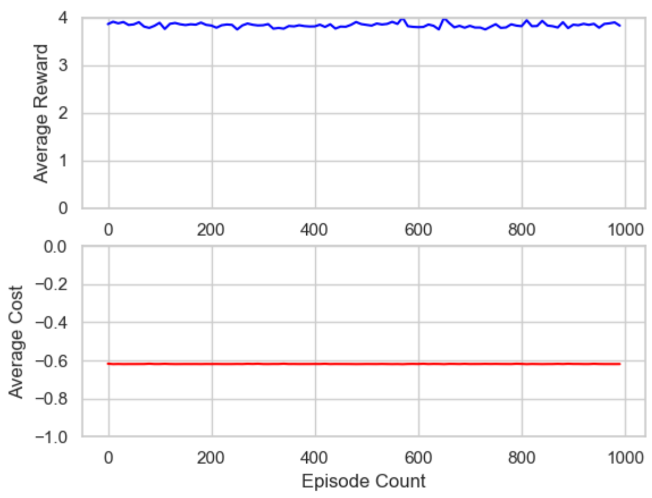


Fig. 4.6 The average reward and average cost by the supervisor obtained from the learning with $r_{n1} = -0.1$.

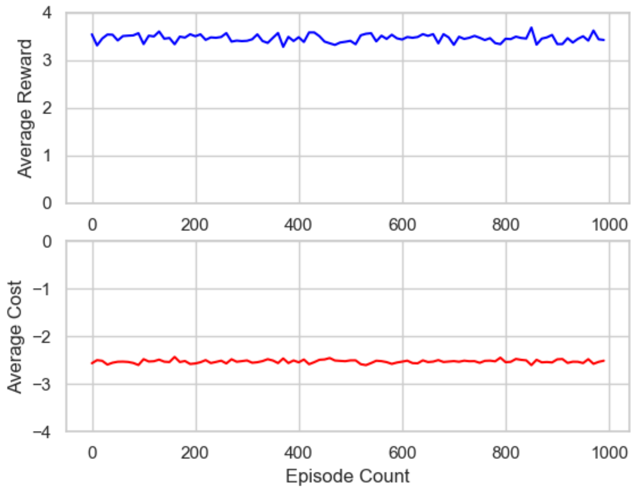


Fig. 4.7 The average reward and average cost by the supervisor obtained from the learning with $r_{n1} = -0.5$.

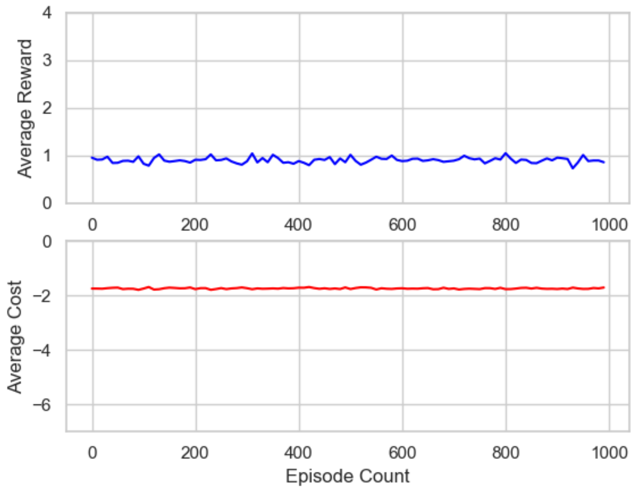


Fig. 4.8 The average reward and average cost by the supervisor obtained from the learning with $r_{n1} = -1$.

Chapter 5

Conclusions

A

Proofs

Proof （Proof of lemma 3.1） Suppose that $MC_{\pi}^{\otimes}$ satisfies neither conditions 1 nor 2. Then, there exists a policy π, $i \in \{1,\dots,h\}$, and j_1, $j_2 \in \{1,\dots,n\}$ such that $\delta_{\pi,i}^{\otimes} \cap \bar{F}_{j_1}^{\otimes} = \emptyset$ and $\delta_{\pi,i}^{\otimes} \cap \bar{F}_{j_2}^{\otimes} \neq \emptyset$. In other words, there exists a nonempty and proper subset $J \in 2^{\{1,\dots,n\}} \setminus \{\{1,\dots,n\},\emptyset\}$ such that $\delta_{\pi,i}^{\otimes} \cap \bar{F}_{j}^{\otimes} \neq \emptyset$ for any $j \in J$. For any transition $(s^{\otimes}, a, s^{\otimes\prime}) \in \delta_{\pi,i}^{\otimes} \cap \bar{F}_{j}^{\otimes}$, the following equation holds by the properties of the recurrent states in $MC_{\pi}^{\otimes}$[18].

$$\sum_{k=0}^{\infty} p^k((s^{\otimes}, a, s^{\otimes\prime}), (s^{\otimes}, a, s^{\otimes\prime})) = \infty, \tag{A.1}$$

where $p^k((s^{\otimes}, a, s^{\otimes\prime}), (s^{\otimes}, a, s^{\otimes\prime}))$ is the probability that the transition $(s^{\otimes}, a, s^{\otimes\prime})$ occurs again after the occurrence of itself in k time steps. Eq. (A.1) means that the agent obtains a reward infinitely often. This contradicts the definition of the acceptance condition of the product MDP $M^{\otimes}$. □

Proof （Proof of theorem 3.1） 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 $M^{\otimes}$, $\delta_{\pi^*,i}^{\otimes} \cap \bar{F}_{j}^{\otimes} = \emptyset$ holds by Lemma **??**. 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 $s_{init}^{\otimes}$ in the set of transient states, it holds that

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

where the action a is selected by π^*. By the property of the transient states, for any state $s^{\otimes}$ in $T^{\otimes}_{\pi^*}$, there exists a bounded positive value m such that $\sum_{k=0}^{\infty} \gamma^k p^k(s^{\otimes}_{init}, s^{\otimes}) \leq \sum_{k=0}^{\infty} p^k(s^{\otimes}_{init}, s^{\otimes}) < m$ [18]. Therefore, there exists a bounded positive value $\bar{m}$ such that $V^{\pi^*}(s^{\otimes}_{init}) < \bar{m}$. Let $\bar{\pi}$ be a positional policy satisfying φ. We consider the following two cases.

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

$$V^{\bar{\pi}}(s^{\otimes}_{init}) = \sum_{k=0}^{\infty} \sum_{s^{\otimes} \in R^{\otimes i}_{\bar{\pi}}} \gamma^k p^k(s^{\otimes}_{init}, s^{\otimes}) \sum_{s^{\otimes\prime} \in R^{\otimes i}_{\bar{\pi}}} P^{\otimes}_{\bar{\pi}}(s^{\otimes\prime} \mid s^{\otimes}) \mathcal{R}(s^{\otimes}, a, s^{\otimes\prime}),$$

where the action a is selected by $\bar{\pi}$. Since $s^{\otimes}_{init}$ is in $R^{\otimes i}_{\bar{\pi}}$, there exists a positive number $\bar{k} = \min\{k \ ; \ k \geq n, p^k(s^{\otimes}_{init}, s^{\otimes}_{init}) > 0\}$ [18]. We consider the worst scenario in this case. It holds that

$$\begin{aligned} V^{\bar{\pi}}(s^{\otimes}_{init}) \geq & \sum_{k=n}^{\infty} p^k(s^{\otimes}_{init}, s^{\otimes}_{init})(\gamma^k + \gamma^{k-1} + ... + \gamma^{k-n+1}) r_p \\ \geq & \sum_{k=1}^{\infty} p^{k\bar{k}}(s^{\otimes}_{init}, s^{\otimes}_{init})(\gamma^{k\bar{k}} + ... + \gamma^{k\bar{k}-n+1}) r_p \\ > & r_p \sum_{k=1}^{\infty} \gamma^{k\bar{k}} p^{k\bar{k}}(s^{\otimes}_{init}, s^{\otimes}_{init}), \end{aligned}$$

whereas all states in $R(MC^{\otimes}_{\bar{\pi}})$ are positive recurrent because $|S^{\otimes}| < \infty$ [19]. Obviously, $p^{k\bar{k}}(s^{\otimes}_{init}, s^{\otimes}_{init}) \geq (p^{\bar{k}}(s^{\otimes}_{init}, s^{\otimes}_{init}))^k > 0$ holds for any $k \in (0, \infty)$ by the Chapman-Kolmogorov equation [18]. Furthermore, we have $\lim_{k \to \infty} p^{k\bar{k}}(s^{\otimes}_{init}, s^{\otimes}_{init}) > 0$ by the property of irreducibility and positive recurrence [20]. Hence, there exists $\bar{p}$ such that $0 < \bar{p} < p^{k\bar{k}}(s^{\otimes}_{init}, s^{\otimes}_{init})$ for any $k \in (0, \infty]$ and we have

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

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

2. Assume that the initial state $s^{\otimes}_{init}$ is in the set of transient states $T^{\otimes}_{\bar{\pi}}$. $P^{M^{\otimes}}_{\bar{\pi}}(s^{\otimes}_{init} \models \varphi) > 0$ holds by the definition of $\bar{\pi}$. For a recurrent class $R^{\otimes i}_{\bar{\pi}}$ such that $\delta^{\otimes}_{\bar{\pi},i} \cap \bar{F}^{\otimes}_j \neq \emptyset$ for each accepting set $\bar{F}^{\otimes}_j$, there exist a number $\bar{l} > 0$, a state $\hat{s}^{\otimes}$ in $Post(T^{\otimes}_{\bar{\pi}}) \cap R^{\otimes i}_{\bar{\pi}}$, and a subset of transient states $\{s^{\otimes}_1, \dots, s^{\otimes}_{\bar{l}-1}\} \subset T^{\otimes}_{\bar{\pi}}$ such that $p(s^{\otimes}_{init}, s^{\otimes}_1) > 0$,

$p(s_i^{\otimes}, s_{i+1}^{\otimes}) > 0$ for $i \in \{1, ..., \bar{l}-2\}$, and $p(s_{\bar{l}-1}^{\otimes}, \hat{s}^{\otimes}) > 0$ by the property of transient states. Hence, it holds that $p^{\bar{l}}(s_{init}^{\otimes}, \hat{s}^{\otimes}) > 0$ for the state $\hat{s}^{\otimes}$. Thus, by ignoring rewards in $T_{\bar{\pi}}^{\otimes}$, we have

$$V^{\bar{\pi}}(s_{init}^{\otimes}) \geq P_{\bar{\pi}}^{M^{\otimes}}(s_{init}^{\otimes} \models \varphi)\gamma^{\bar{l}} p^{\bar{l}}(s_{init}^{\otimes}, \hat{s}^{\otimes}) \sum_{k=0}^{\infty} \sum_{s^{\otimes\prime} \in R_{\bar{\pi}}^{\otimes i}} \gamma^k p^k(\hat{s}^{\otimes}, s^{\otimes\prime})$$

$$\sum_{s^{\otimes\prime\prime} \in R_{\bar{\pi}}^{\otimes i}} P_{\bar{\pi}}^{\otimes}(s^{\otimes\prime\prime}|s^{\otimes\prime})\mathcal{R}(s^{\otimes\prime}, a, s^{\otimes\prime\prime})$$

$$> P_{\bar{\pi}}^{M^{\otimes}}(s_{init}^{\otimes} \models \varphi)\gamma^{\bar{l}} p^{\bar{l}}(s_{init}^{\otimes}, \hat{s}^{\otimes}) r_p \bar{p} \gamma^{\bar{k}'} p^{\bar{k}'}(\hat{s}^{\otimes}, \hat{s}^{\otimes}) \frac{1}{1-\gamma^{\bar{k}'}},$$

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

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

Proof (Proof of theorem ??) Let π^* be optimal but $Pr_{\pi^*}^{M}(\rho_{init} \models \varphi) < 1$. By the definition of π^*, there exist a recurrent class $R_{\pi^*}^{\otimes i}$ that is accessible from the initial state s_{init} such that $\delta_{\pi^*,i}^{\otimes} \cap \bar{F}_j^{\otimes} = \emptyset$ for any accepting set $\bar{F}_j^{\otimes}$. Let $s^{\otimes}$ be a state in $R_{\pi^*}^{\otimes i}$. By lemma 3.1, $V^{\pi^*}(s^{\otimes}) = 0$. Let $\bar{\pi}$ be a policy such that $Pr_{\bar{\pi}}^{M}(\rho_{init} \models \varphi) = 1$. We consider the following two cases.

1. Assume that the state $s^{\otimes}$ is in a recurrent class $R_{\bar{\pi}}^{\otimes i}$

□

Acknowledgment

I would like to express my deep sense of gratitude to my adviser Professor Toshimitsu Ushio, Graduate School of Engineering Science, Osaka University, for his invaluable, constructive advice and constant encouragement during this work. Professor Ushio's deep knowledge and his eye for detail have inspired me much.

References

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

[2] E. M. Clarke, Jr., O. Grumberg, D. Kroening, D. Peled, and H. Veith, *Model Checking*, 2nd Edition. MIT Press, 2018.

[3] M. Kloetzer, C. Belta, "A fully automated framework for control of linear systems from temporal logic specifications," *IEEE Trans. Autom. Contr.*, vol. 53, no. 1, pp. 287–297, 2008.

[4] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Temporal-logic-based reactive mission and motion planning," *IEEE Trans. Robotics*, vol. 25, no. 6, pp. 1370–1381, 2009.

[5] T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Receding horizon temporal logic planning," *IEEE Trans. Autom. Contr.*, vol. 57, no. 11, pp. 2817–2830, 2012.

[6] A. Sakakibara and T. Ushio, "Decentralized supervision and coordination of concurrent discrete event systems under LTL constraints," in *Proc. 14th International Workshop on Discrete Event Systems*, 2018, pp. 18-23.

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

[8] M. L. Puterman, *Markov Decison Processes, Discrete Stochastic Dynamic Programming.* John Wiley & Sons, Inc., 1994.

[9] E. M. Wolff, U. Topcu, and R. M. Murray, "Robust control of uncertain Markov decision processes with temporal logic specifications," in *Proc. 51st IEEE Conference on Decision and Control*, 2012, pp. 3372–3379.

[10] 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.

[11] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd Edition. MIT Press, 2018.

[12] M. Hiromoto and T. Ushio, "Learning an optimal control policy for a Markov decision process under linear temporal logic specifications," in *Proc. 2015 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, 2015, pp. 548-555.

[13] S. Sickert, J. Esparaza, S. Jaax, and J. Křetìnský, "Limit-deterministic Büchi au-

tomata for linear temporal logic," in *International Conference on Computer Aided Verification*, 2016, pp. 312-332.

[14] M. Hasanbeig, A. Abate, and D. Kroening, "Logically-constrained reinforcement learning," *arXiv:1801.08099v8*, Feb. 2019.

[15] 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.

[16] 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.

[17] 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.

[18] R. Durrett, *Essentials of Stochastic Processes*, 2nd Edition. ser. Springer texts in statistics. New York; London; Springer, 2012.

[19] L. Breuer, "Introduction to Stochastic Processes", [Online]. Available: https://www.kent.ac.uk/smsas/personal/lb209/files/sp07.pdf

[20] S.M. Ross, *Stochastic Processes*, 2nd Edition. University of California, Wiley, 1995.

[21] 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.

[22] J. Kretínský, T. Meggendorfer, S. Sickert, "Owl: A library for ω-words, automata, and LTL," in *Proc. 16th International Symposium on Automated Technology for Verification and Analysis*, 2018, pp. 543–550.