Preprints of the 22nd IFAC World Congress
Yokohama, Japan, July 9-14, 2023

Synthesis of Non-blocking Controllers for Linear Temporal Logic Tasks under Partial Observations ⋆

Shuaiyi Li * **Shaoyuan Li** * **Xiang Yin** *

* *Department of Automation, Shanghai Jiao Tong University, Shanghai 200240, China. (E-mail: lishuaiyi, yinxiang, syli@sjtu.edu.cn)*

Abstract: In this paper, we investigate the formal synthesis of discrete controllers for linear temporal logic tasks under partial information. Existing works on this topic mainly focus on find sure winning or almost sure winning control strategies under some assumptions regarding the system's atomic propositions. In this work, we consider non-blockingness as the metric as the achievement of the task. Specifically, we require that at each instant, the controller maintains the possibility to achieve the LTL task under some environment's behavior. We first present an offline algorithm for the computation of the winning region over the belief state space. Then we present an online control algorithm that effective solves the control synthesis problem. The proposed control algorithm is also illustrated by a case study of robot task planning.

Keywords: Formal Methods, Linear Temporal Logic, Task Planning, Partial Observation.

1. INTRODUCTION

Task planning and decision-making are the central problems in the field of autonomous robots. Recently, there has been a growing interest in task planning for robots for high-level specifications described by, e.g., linear temporal logic (LTL). In this setting, the robot needs to accomplish a complex task in a dynamic environment, such as go to some region before doing something or first do something and then visit some region infinitely. There have been considerable recent works on temporal logic based task planning; see, e.g., Kantaros et al. (2019); Yu et al. (2022).

Many existing works on temporal logic based task planning are based on the assumption that the controller can access the complete state information of the system. In practice, however, only partial information of the system might be available. Therefore, there has been growing interest in task planning problem under imperfect information. Such control synthesis problem under partial observations can be modeled by partially observable Markov decision processes, or by two player games with imperfect information see, e.g., Belta et al. (2017); Raskin et al. (2007); Chatterjee et al. (2016); Fu and Topcu (2016); Ramasubramanian et al. (2020); Sakakibara and Ushio (2020).

For games with partial observation, it was shown that the standard sure winning condition in Raskin et al. (2007) is difficult to achieve under the general setting. Therefore, researchers seek for synthesizing sure winning controllers and almost sure winning controllers under the assumption that all observational equivalent states have the same atomic proposition. In the context of supervisory control of discrete event system, there are also many results on synthesis of supervisors under partial observation Cai et al. (2014); Ru et al. (2014); Yin and Lafortune (2016b); Ma and Cai (2021); Ji et al. (2021). In particularly, Yin and Lafortune (2016a) solves the problem of synthesizing maximally permissive non-blocking supervisors, where the system always has the possibility to reach a desired state.

In this paper, we consider the problem of synthesizing controllers for (non-deterministic) labeled transition systems under partial observation in the general setting. Specifically, we consider a fragment of LTL formulae, which can be accepted by deterministic Büchi automata, as the control objectives. Then we solve a new non-blocking control problem for this setting. Specifically, our contributions are as follows. First, we present an approach for the *offline* computation of the winning region in the belief structure for such a problem. Then, based on the offline computed winning region, we design an *online* control algorithm solving the non-blocking control problem. Our result is different from the almost sure winning controller synthesis and Raskin et al. (2007) since no assumptions on the environment's strategy and observation equivalence are considered. Our online control scheme is also different from the non-blocking supervisory control algorithm Yin and Lafortune (2016a) where the supervisor is computed fully offline. Furthermore, we provide a sufficient condition under which the proposed non-blocking controller is also sure winning.

2. PRELIMINARY

2.1 System Model

We consider a system modeled as a *label transition system* (LTS), which is a tuple

$$\mathcal{T} = (X, Act, \Delta, X_0, \mathcal{AP}, L),$$

where X is a finite set of states representing, e.g., the physical locations or the status of the systems; Act is a

⋆ This work was supported by the National Natural Science Foundation of China (62061136004, 62173226, 61833012).

finite set of actions or control inputs; $\Delta : X \times Act \to 2^X$ is the non-deterministic transition function, where $x' \in \Delta(x, a)$ means that at state x, the system may move to state x' by taking action a; $X_0 \subseteq X$ is the set of possible initial states; $\mathcal{AP}$ is the set of atomic propositions representing basic properties of our interest; $L : X \to 2^{\mathcal{AP}}$ is the labeling function that assigns each state a set of atomic propositions.

Let A be a set. We denote by A^ω and A^* the set of all infinite sequences and finite sequences over A, respectively. For states $x, x' \in X$ and action $a \in Act$, we also write as $x \xrightarrow{a} x'$ if $x' \in \Delta(x, a)$ and as $x \to x'$ if $x \xrightarrow{a} x'$ for some a. Then a *path* generated by LTS $\mathcal{T}$ is an infinite sequence of states $\tau = x_0 x_1 x_2 \cdots \in X^\omega$ such that $x_0 \in X_0$ and $x_i \to x_{i+1}, \forall i \geq 0$. We denote by $\mathsf{Path}(\mathcal{T})$ and $\mathsf{Path}(\mathcal{T})^*$ the set of all paths and finite paths generated by $\mathcal{T}$. The trace of path $\tau \in \mathsf{Path}(\mathcal{T})$ is $\mathsf{Trace}(\tau) = L(x_0)L(x_1)L(x_2)\cdots \in (2^{\mathcal{AP}})^\omega$ and we denote by $\mathsf{Trace}^*(\mathcal{T})$ the set of all finite traces generated by $\mathcal{T}$. Let $\Omega \subseteq \Sigma^\omega$ be a set of infinite sequences. We denote the set of finite prefixes of Ω as $\overline{\Omega} = \{\rho \in \Sigma^* \mid \exists \omega \in \Sigma^\omega, \rho\omega \in \Omega\}$.

2.2 Partial Observation Controller

The original behavior $\mathsf{Trace}(\mathcal{T})$ may contain some undesired traces. Therefore, a feedback controller is usually imposed on the system such that the closed-loop under control satisfies some requirement. In general, the controller may not be able to access the full state information of the system. Instead, it may have its partial observation described by a mapping $H : X \to O$, where O is a finite set of observation symbols. Mapping H also classifies the state space into equivalence classes $\{[x]_o \mid o \in O\}$, where $[x]_o = \{x \in X \mid H(x) = o\}$. For any finite path $\tau = x_0 x_1 \cdots \in \mathsf{Path}(\mathcal{T})^*$, we denote by $H(\tau) = H(x_0)H(x_1)\cdots \in O^*$ its finite observation sequence. Then a controller is a function $C : H(\mathsf{Path}(\mathcal{T})^*) \to Act$ that determines the control input based on its observation.

We denote by $\mathcal{T}_C$ the closed-loop system under controller C. Then a finite path $\tau = x_0 x_1 \cdots \in \mathsf{Path}(\mathcal{T})^*$ is generated by $\mathcal{T}_C$ if $\forall i \geq 0 : x_i \xrightarrow{a_i} x_{i+1}$, where $a_i = C(H(x_0 x_1 \cdots x_i))$. We also denote by $\mathsf{Path}(\mathcal{T}_C)^*$ and $\mathsf{Trace}^*(\mathcal{T}_C)$ the sets of all finite paths and traces generated by $\mathcal{T}_C$, respectively.

2.3 Linear Temporal Logic

The control objectives are described by Linear Temporal Logic (LTL) formulae with the following syntax

$$\phi ::= True \mid \Sigma \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \bigcirc\phi \mid \phi_1 \,\mathcal{U}\, \phi_2,$$

where $\Sigma \in \mathcal{AP}$ is an atomic proposition, $\neg$ and $\wedge$ are Boolean operators "negation" and "conjunction", respectively, and $\bigcirc$ and $\mathcal{U}$ are temporal operators "next" and "until", respectively. We can also define other temporal operators such as "eventually" by $\Diamond\phi := True\ \mathcal{U}\ \phi$ and "always" by $\Box\phi := \neg\Diamond\neg\phi$. LTL formulae are evaluated on infinite words (infinite sequences over $2^{\mathcal{AP}}$). For an infinite word $\sigma \in (2^{\mathcal{AP}})^\omega$, we denote by $\sigma \vDash \phi$ if it satisfies LTL formula ϕ. The reader is referred to Baier and Katoen (2008) for more details on LTL semantics. We denote by $\mathsf{Word}(\phi)$ the set of all infinite words satisfying ϕ.

Given an LTL formula ϕ, in general, $\mathsf{Word}(\phi)$ can be accepted by a *non-deterministic* Büchi automaton or a deterministic Rabin automaton. In this work, we assume that LTL formula ϕ can be accepted by a *deterministic* Büchi automaton (DBA). This assumption is restrictive since it does not hold for formulae such as $\Diamond\Box$. Formally, a DBA is a 5-tuple $\mathcal{A} = (S, \Sigma, \delta, s_0, S_F)$, where S is a finite set of states, $\Sigma = 2^{\mathcal{AP}}$ is a finite alphabet, $\delta : S \times \Sigma \to S$ is the transition function, s_0 is the initial state, and $S_F \subseteq S$ is the set of accepting states. For an infinite word $\rho = \sigma_1\sigma_2 \cdots \in \Sigma^\omega$, it visits a unique infinite sequence of infinite states $\rho = s_0 s_1 \cdots \in S^\omega$, called a run, in $\mathcal{A}$ such that $s_{i+1} = \delta(s_i, \sigma_{i+1})$. An infinite run $\pi \in S^\omega$ is said to be accepted by DBA $\mathcal{A}$ if $inf(\pi) \cap S_F \neq \emptyset$, where $inf(\pi)$ is the set of states that occurs infinitely in ρ. An infinite word ρ is said to be accepted if its run is accepted. We denote by $\mathcal{L}(\mathcal{A})$ the set of all accepted words. Then for any LTL formula ϕ, we denote by $\mathcal{A}_\phi$ the DBA such that $\mathcal{L}(\mathcal{A}_\phi) = \mathsf{Word}(\phi)$.

2.4 Problem Formulation

In the context of controller synthesis under partial observation, one problem is to synthesize a controller that achieve the LTL task for sure, i.e., $\mathsf{Trace}(\mathcal{T}_C) \subseteq \mathsf{Word}(\phi)$. However, as discussed in Raskin et al. (2007) that, sure winning controllers are generally very restricted and difficult to synthesize for partially observed systems. And almost sure winning controllers require assumptions on environment behavior. To the end, one may seek for synthesizing strategies that are less restrictive and do not require assumptions.

In this work, motivated by the work of supervisory control of discrete event systems, we adopt the notion of *nonblockingness* as the winning condition, which is further weaker than the almost sure winning condition. Specifically, we say a feedback controller is non-blocking w.r.t. a LTL task, if for any possible instant, the system always hold the possibility of generating infinite traces satisfying the LTL tasks. Note that the satisfaction of the task may need the cooperation of the environment. Therefore, a non-blocking system means that, as long as the environment is not fully adversarial or never making mistake, the system will eventually achieve the temporal logic task. Formally, a feedback controller C is said to be non-blocking for system $\mathcal{T}$ w.r.t. ϕ if $\mathsf{Trace}^*(\mathcal{T}_C) \subseteq \overline{\mathsf{Word}(\phi)}$. Our objective is to synthesize such a controller.

Problem 1. Given system $\mathcal{T}$, observation mapping H and LTL formula ϕ, synthesize a control $C : O^* \to Act$ such that $\mathsf{Trace}^*(\mathcal{T}_C) \subseteq \overline{\mathsf{Word}(\phi)}$.

3. CONSTRUCTION OF WINNING REGION

3.1 Belief Transition Systems

In order to incorporate the completion status of the LTL task ϕ into the system model, we first define the *product system*.

Definition 1. (Product Systems). Given LTS $\mathcal{T} = (X, Act, \Delta, X_0, AP, L)$ and DBA $\mathcal{A}_\phi = (S, \Sigma, \delta, s_0, S_F)$, the product system is a new LTS

$$\tilde{T} = \mathcal{T} \times \mathcal{A}_\phi = (\tilde{X}, Act, \tilde{\Delta}, \tilde{X}_0, \mathcal{AP}, L),$$

where $\tilde{X} = X \times S$ is the set of (product) states, Act is the same set of control inputs, $\tilde{\Delta} : \tilde{X} \times Act \to 2^{\tilde{X}}$ is the transition function defined by: for any $\tilde{x} = (x, s), \tilde{x}' = (x', s') \in \tilde{X}$ and $a \in Act$,

$$\tilde{x}' \in \tilde{\Delta}(\tilde{x}, a) \quad \text{iff} \quad x' \in \Delta(x, a) \wedge \delta(s, L(x')) = s'$$

$\tilde{X}_0 = X_0 \times \{s_0\}$ is the set of initial states, and $\tilde{L}$ is the labeling function defined by for any $\tilde{x} = (x, s) \in \tilde{X}$, we have $\tilde{L}(\tilde{x}) = L(x)$.

We denote by $\tilde{X}_F = \{(x, s) \in \tilde{X} \mid s \in S_F\}$ the set of product states whose second components are accepting. We also extend the observation function to $H : \tilde{X} \to O$ by for any $\tilde{x} = (x, s) \in \tilde{X}$, we have $H(\tilde{x}) = H(x)$.

Due to the partial observability of the system, the controller does not know the exact (product) state of the system. Instead, it only knows a set of its possible current states, which is referred to as a *belief state*. By issuing a control input and making a new observation, the controller can update its belief state. This belief evolution is captured by the following *belief transition systems*.

Definition 2. (Belief Transition Systems). Given product system $\tilde{T}$ and observation mapping H, the belief transition system is defined as a 5-tuple

$$\mathcal{B} = (Q, Act, \Delta_B, Q_0),$$

where

- $Q = \{q \in 2^{\tilde{X}} \mid \forall \tilde{x}, \tilde{x}' \in q, H(\tilde{x}) = H(\tilde{x}')\}$ is the set of belief states;
- Act is the same set of actions;
- $\Delta_B : Q \times Act \to 2^Q$ is the non-deterministic transition function defined by: for any $q, q' \in Q$ and $a \in Act$, we have $q' \in \Delta_B(q, a)$ if
$$q' = (\cup_{\tilde{x} \in q} \tilde{\Delta}(\tilde{x}, a)) \cap [\tilde{x}]_o \text{ for some } o \in O$$
- $Q_0 = \{q_0 \in 2^{\tilde{X}_0} : q_0 = \tilde{X}_0 \cap [\tilde{x}]_o \text{ for some } o \in O\}$ is the set of possible initial belief states.

Note that, given a belief state $q \in Q$, since all (product) states in it has the same observation, we can also write $H(q)$ as the same observation for all states in it, and we have $q \subseteq [\tilde{x}]_{H(q)}$. Clearly, we have $H(\mathsf{Path}(\mathcal{T})) = H(\mathsf{Path}(\mathcal{B}))$. Therefore, we can also design a controller for $\mathcal{B}$ and apply back to $\mathcal{T}$. It has been in Raskin et al. (2007) that, under the assumption of all observational-equivalent states having the same propositions, the sure winning synthesis problem for $\mathcal{T}$ can be directly solved as a Büchi game on $\mathcal{B}$ by considering those beliefs containing some accepting state (in fact, all states in it are accepting) as accepting beliefs. This also suggests that a belief-state-based strategy is sufficient for the synthesis problem. However, without the assumption on propositions, the synthesis problem becomes much challenging as shown by the following example.

Example 1. Given a LTS system $\mathcal{T}$ and a task described by LTL ϕ, assume that the product transition system $\tilde{T}$ is presented in Fig.1(a). The belief transition system $\mathcal{B}$ generated is in Fig.1(b). Clearly, there is no belief state in which *all states* are accepting. However, if we consider beliefs states, in which *some states* are accepting, as accepting beliefs, e.g., states $\{q_1, q_2, q_3\}$, then we will have the following issues. According the solution of Büchi games, the

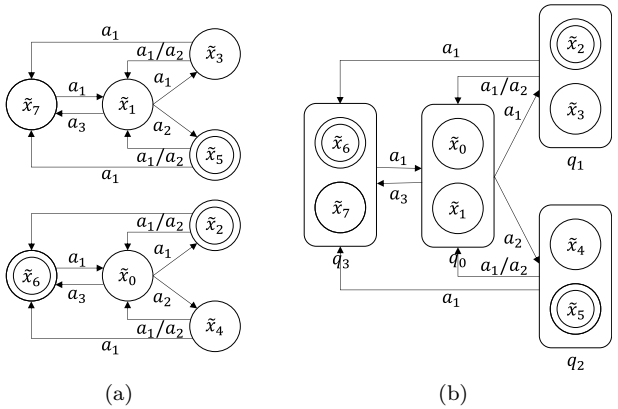

Fig. 1. An example of belief transition system (a) Product transition system $\tilde{T}$ (b) Belief transition system $\mathcal{B}$

controller can either take action a_1 at q_0, following action a_1 at q_1 and action a_1 at q_3 or action a_2 at q_1 returning to q_0; or take action a_2 at q_0 following action a_1 at q_2 and action a_1 at q_3 or action a_2 at q_2 returning to q_0. All these strategies do not consider history information and can infinitely visit accepting beliefs $\{q_1, q_2, q_3\}$. However, if we investigate into system's actual path induced by the aforementioned strategies, it can be discovered that some strategies cannot guarantee achieving the task. For example, if action a_1 is taken at belief q_0, then no matter what action the system takes at q_1, the future potential product path of $(\tilde{x}_3\tilde{x}_7)^\omega$ can be generated in practice, which visits none of the accepting states.

The above example reveals that, for the general case, we cannot simply characterizing belief states as accepting and non-accepting, and reduce the partial observation synthesis problem to the case of full observation. To tackle the synthesis problem, we need to further investigate the actual interconnection between states in belief states, and belief-based-strategy may not be sufficient.

3.2 Computation of Preliminary Winning Regions

First of all, we define

$$Q_F = \{q \in Q \mid q \cap \tilde{X}_F \neq \emptyset\}$$

as the set of accepting belief states. As we discussed in Example 1, visiting accepting belief states is only a *necessary condition* for achieving the task. We denote by $\mathtt{Win}(Q_F) \subseteq Q$ the set of belief states from which the controller can ensure visiting Q_F infinitely often, i.e., the winning region of Büchi game in $\mathcal{B}$ w.r.t. Q_F. Note that, since staying in $\mathtt{Win}(Q_F)$ is only necessary to achieve the task, we refer $\mathtt{Win}(Q_F)$ to as the *preliminary winning region*, which can be computed by the standard Büchi game techniques.

Formally, let $\imath \subseteq Q$ be a set of belief states in $\mathcal{B}$ one wants to reach. We define the *controlled predecessor* of $\imath$ by

$$\mathtt{CPre}(\imath) = \{q \in Q \mid \exists a \in Act \text{ s.t. } \Delta_B(q, a) \subseteq \imath\}$$

The *k-step attractor* $\mathtt{Attr}^{(i)}(\imath)$ of $\imath$ is defined by

$$\mathtt{Attr}^{(0)}(\imath) = \imath$$
$$\mathtt{Attr}^{(k+1)}(\imath) = \mathtt{Attr}^{(k)}(\imath) \cup \mathtt{CPre}(\mathtt{Attr}^{(k)}(\imath))$$

Then the *attractor* of $\imath$ is defined by

$$\texttt{Attr}(\imath) = \bigcup_{k\geq 0} \texttt{Attr}^{(k)}(\imath).$$

Similarly, let $\imath \subseteq Q$ be a set of belief states in $\mathcal{B}$ one wants to avoid. We define the *adversarial predecessor* of $\imath$ by

$$\texttt{APre}(\imath) = \{q \in Q \mid \forall a \in Act \text{ s.t. } \Delta_B(q,a) \cap \imath \neq \emptyset\}$$

The *k-step avoid-region* $\texttt{Avoid}^{(k)}(\imath)$ of $\imath$ is defined inductively by

$$\texttt{Avoid}^{(0)}(\imath) = \imath$$
$$\texttt{Avoid}^{(k+1)}(\imath) = \texttt{Avoid}^{(k)}(\imath) \cup \texttt{APre}(\texttt{Avoid}^{(k)}(\imath))$$

Then the *avoid-region* of $\imath$ is defined by

$$\texttt{Avoid}(\imath) = \bigcup_{k\geq 0} \texttt{Avoid}^{(k)}(\imath).$$

Note that not all accepting belief states in Q_F can be visited infinitely often. For example, in order to visit Q_F again, we should not start from a state in $\texttt{Avoid}(Q \setminus \texttt{Attr}(Q_F))$. Therefore, we shrink the set of accepting belief states $Q_F^{(0)} \supseteq Q_F^{(1)} \supseteq \cdots$ inductively by

$$Q_F^{(0)} = Q_F$$
$$Q_F^{(k+1)} = Q_F^{(k)} \setminus \texttt{Avoid}(Q \setminus \texttt{Attr}(Q_F^{(k)}))$$

We define $\texttt{Rec}(Q_F) = \lim_{k\to\infty} Q_F^{(k)}$ as the set of *recurrent* accepting belief states of Q_F. Then, we have

$$\texttt{Win}(Q_F) = \texttt{Attr}(\texttt{Rec}(Q_F)).$$

3.3 Computation of Actual Winning Regions

The preliminary winning region $\texttt{Win}(Q_F)$ is very rough and needs to be further pruned iteratively to get the actual winning region denoted by $\texttt{Win}^+(Q_F)$. Because the belief transition system can only describe the relation between groups of states and cannot represent the details of how system's states are actually connected with each other.

The adjustment process of preliminary winning region is conducted iteratively. In each iteration, a zoomed transition system is constructed to zoom into the belief transition system to identify and delete unqualified belief states. Then a new winning region is recomputed. The iteration stops when an convergence of the final winning region is reached. We will illustrate the details of winning region's adjustment as follows.

The adjustment of preliminary winning region is conducted based on a product transition system called the interconnected transition system.

Definition 3. (Interconnected Transition Systems). Given a belief transition system $\mathcal{B}$ and a set of belief states $R \subseteq Q$, which will be referred to as the *reference region*, the *interconnected transition system* induced by reference region R is a tuple

$$\hat{\mathcal{B}}(R) = (\hat{Q}, Act, \hat{\Delta}, \hat{Q}_0),$$

where

- $\hat{Q} = \{(\tilde{x}, q) \mid q \in R, \tilde{x} \in q\}$ is the set of all pairs of belief states in R and augmented states in each belief states;
- Act is the same set of actions;
- $\hat{\Delta} : \hat{Q} \times Act \to 2^{\hat{Q}}$ is the non-deterministic transition function defined by: for any $\hat{q} = (\tilde{x}, q), \hat{q}' = (\tilde{x}', q')$ and $a \in Act$, we have $\hat{q}' \in \hat{\Delta}(\hat{q}, a)$ if
  - $\tilde{x}' \in \tilde{\Delta}(\tilde{x}, a)$; and
  - $q' \in \Delta_B(q, a) \subseteq R$.
- $\hat{Q}_0 = \{(\tilde{x}, q) \mid q \in R \cap Q_0, \tilde{x} \in q\}$ is the set of initial states.

We define the set of accepting pairs in $\hat{\mathcal{B}}(R)$ by

$$\hat{Q}_{F,R} = \{(\tilde{x}, q) \in \tilde{X} \times Q \mid \tilde{x} \in \tilde{X}_F \wedge \tilde{x} \in q\} \subseteq \hat{Q}$$

Then let $R \subseteq Q$ be a reference region. We denote by $\widehat{\texttt{Win}}_R(\cdot)$ the Büchi winning region operators in interconnected transition system $\hat{\mathcal{B}}(R)$, respectively.

Now, let us assume that initially the reference region R is $\texttt{Win}(Q_F)$. Then constructing its interconnected transition system $\hat{\mathcal{B}}(R)$, one may find that some state in it may not actually be able to reach an accepting state, e.g.,

$$(\tilde{x}, q) \notin \widehat{\texttt{Win}}_R(\hat{Q}_{F,R}).$$

In this case, we should remove state $(\tilde{x}, q)$ from $\hat{\mathcal{B}}(R)$. In fact, since the controller cannot distinguish states inside of belief state q, all state of $(\cdot, q)$ should be remove from $\hat{\mathcal{B}}(R)$. This suffices to remove belief state q from the reference region R.

Therefore, we iteratively shrink the reference regions $R_0 \supseteq R_1 \supseteq \cdots$ as follows:

$$R_0 = \texttt{Win}(Q_F) \tag{1}$$
$$R_{i,unq} = \{q \in R_i \mid \exists (\tilde{x}, q) \in \hat{Q} \setminus \widehat{\texttt{Win}}_{R_i}(\hat{Q}_{F,R_i})\} \tag{2}$$
$$R_{i,rem} = \texttt{Avoid}(R_{i,unq} \cup (Q \setminus R_i)) \cap R_i \tag{3}$$
$$R_{i+1} = \texttt{Win}(Q_F \cap (R_i \setminus R_{i,rem})) \tag{4}$$

We define $Q_{win} = \lim_{k\to\infty} R^{(k)} \subseteq Q$ be the limit of the above reference region sequence, which is also referred to as the *actual winning region*. Q_{win} induces a interconnected transition system $\hat{B}(Q_{win})$. We denote the set of all pairs and accepting pairs appear in $\hat{B}(Q_{win})$ as $\hat{Q}_{win}$ and $\hat{Q}_{acc}$, and the set of all states and accepting states appear in Q_{win} as $\tilde{X}_{win}$ and $\tilde{X}_{acc}$.

The following result formally shows that, the above computed winning region is indeed necessary for the purpose of non-blocking controller synthesis. In fact, the sufficiency will be clear once our synthesis algorithm is presented.

Proposition 1. Given an arbitrary controller C, the induced closed-loop system $\mathcal{T}_C$, its constructed belief transition system $\mathcal{B}_C$ and LTL formula ϕ, if $\exists \tau \in \mathsf{Path}(\mathcal{B}_C)$ satisfies $\tau \notin (Q_{win})^\omega$, then $\mathsf{Trace}(\mathcal{T}_C) \nsubseteq \mathsf{Word}(\phi)$.

4. ONLINE CONTROLLER ALGORITHM

4.1 Overview of the Algorithm

The main idea of our online control strategy is as follows. Initially, we have an initial belief and we do not know the previous state of the system. Therefore, we need to alternatively assign each possible initial state a chance to reach accepting state. Specifically, our strategy is to alternately transfer initial belief state's inner states towards accepting states until all of them have reached accepting

states, which means that only one initial inner state will be focused at each time until all of its potential paths have reached accepting states. Extra memories are needed to assemble history information and record the implicit relations between current inner states and their origination from initial states. So we introduce *augmented state* and *action generation state* to record the history information and make current step control decision. The non-blocking control strategy C for controller is constructed by online information update and offline decision generation, shown in Algorithm.1 and Algorithm.2. Firstly, the existence of non-blocking control strategy will be determined. If the initial state is outside the winning region computed above, then there do not exist non-blocking control strategy. Given an initial observation and corresponding belief state, its augmented state, and current action generation state will be firstly initialized. Then a decision will be made by offline decision generation function `Action` based on current action generation state, and the system will take this action. After that, the system receives a new observation representing another belief state, its corresponding augmented state will be constructed. This alternating process ends when the system reaches a special augmented state marking that all the possible product states originated from initial belief state are either impossible to happen according to the information along the path or have already reached an accepting state. Then a new iteration will start initialized by the current belief state.

Algorithm 1: Online Control Algorithm

receive the initial observation o
$q \leftarrow \tilde{X}_0 \cap [\tilde{x}]_o$, $q_{\mathtt{aug}} \leftarrow \{(x, x, 0) \mid x \in q\}$
$x_{\mathtt{act}} \leftarrow$ randomly pick a state from q
make control decision $a \leftarrow \mathtt{Action}(x_{\mathtt{act}}, q, \hat{\mathcal{B}})$
while receive new observation $o \in O$ **do**
  $q \leftarrow \Delta_B^{obs}(q, a, o)$, $q_{\mathtt{aug}} \leftarrow \Delta_B^{aug}(q_{\mathtt{aug}}, a, o)$
  if $\forall (x_{int}, x_{cur}, b) \in q_{aug} : b = 1$ **then**
    $q_{\mathtt{aug}} \leftarrow \{(x, x, 0) \mid x \in q\}$
    $x_{\mathtt{act}} \leftarrow$ randomly pick a state from q
  if $(\Delta(x_{act}, a) \cap [x]_o) \backslash \tilde{X}_{acc} \neq \emptyset$ **then**
    randomly pick a state $x'_{\mathtt{cur}} \in (\Delta(x_{\mathtt{act}}, a) \cap [x]_o) \backslash \tilde{X}_{acc}$ and set $x_{\mathtt{act}} \leftarrow x'_{\mathtt{cur}}$
  else
    if $\exists (x_{int}, x'_{cur}, 0) \in q_{aug}$ **then**
      randomly pick an augmented state $(x_{\mathtt{int}}, x'_{\mathtt{cur}}, 0) \in q_{\mathtt{cur}}$ and set $x_{\mathtt{act}} \leftarrow x'_{\mathtt{cur}}$
    else
      randomly pick an augmented state $(x'_{\mathtt{int}}, x'_{\mathtt{cur}}, 0) \in q_{\mathtt{cur}}$ and set $x_{\mathtt{act}} \leftarrow x'_{\mathtt{cur}}$
  make control decision $a \leftarrow \mathtt{Action}(x_{\mathtt{act}}, q, \hat{\mathcal{B}})$

4.2 Non-blocking Control Algorithm

Given an arbitrary belief state $q \in Q_{win}$, we want to ensure all the inside product states which are not accepting can reach accepting states in future. When the system transfers to a belief state q', we want to track which inner states of q' can be originated from which non-accepting initial inner state of q. So augmented state $q_{\mathtt{aug}} \in 2^{\tilde{X} \times \tilde{X} \times \{0,1\}}$ is designed, and each element of it is constructed by three parts: $(x_{\mathtt{int}}, x_{\mathtt{cur}}, b) \in \tilde{X} \times \tilde{X} \times \{0, 1\}$. Current inner state $x_{\mathtt{cur}} \in q'$ is an inner state of current belief state q'. Initial inner state $x_{\mathtt{int}} \in q \backslash \tilde{X}_{acc}$ is the potential origination of current inner state. Boolean variable b records whether this path originate from $x_{\mathtt{int}}$ have already reached accepting states. Apart from the relationship between initial inner states and current inner states, the algorithm also needs to determine which current inner state will be used to generate action decision. At each step only action generation state $x_{\mathtt{act}} \in q'$ is considered and transferred towards accepting states.

When the system is initialized at belief state q according to initial observation, augmented state $q_{\mathtt{aug}}$ is concomitantly initialized in line 1-2. Current action generation state $x_{\mathtt{act}}$ is initialized by randomly picking from q. Control decision is made and conducted by offline function `Action` in line 3-4. After receiving a new observation, the system will update its belief state and augmented state as follows. Let $q_{\mathtt{aug}} \in 2^{\tilde{X} \times \tilde{X} \times \{0,1\}}$. For $a \in Act, o \in O$, we define the update functions as follows.

$$\Delta_B^{aug}(q_{\mathtt{aug}}, a, o) = \left\{ (x_{\mathtt{int}}, x'_{\mathtt{cur}}, b') \mid \begin{array}{c} \exists (x_{\mathtt{int}}, x_{\mathtt{cur}}, b) \in q_{\mathtt{aug}} \text{ s.t.} \\ [x'_{\mathtt{cur}} \in \tilde{\Delta}(x_{\mathtt{cur}}, a) \cap [x]_o] \text{ and} \\ [b' = 1 \text{ iff } b = 1 \vee x'_{\mathtt{cur}} \in \tilde{X}_F] \end{array} \right\} \quad (5)$$

$$\Delta_B^{obs}(q, a, o) = \{q' \in \Delta_B(q, a) \mid H(q') = o\} \quad (6)$$

Function Δ_B^{aug} updates the augmented state. This function means that if product state $x'_{\mathtt{cur}}$ originated from $x_{\mathtt{cur}}$ and $x_{\mathtt{int}}$ is proven to be impossible according to received observation after taking action a, then its successors will be excluded from future augmented states. Also, if such product state $x'_{\mathtt{cur}}$ reaches an accepting state, it will be marked by setting Boolean variable $b' = 1$. Function Δ_B^{obs} updates the belief state according to the observation.

After the update of belief state and augmented state, action generation state $x_{\mathtt{act}}$ is picked in line 7-15, considering both current $q_{\mathtt{aug}}$ and last step's $x_{\mathtt{act}}$. If all states have reached accepting states, then $q_{\mathtt{aug}}$ and $x_{\mathtt{act}}$ are initialized for a new iteration in line 7-9. If there exist successors of last step's $x_{\mathtt{act}}$ under current observation that have not reached accepting states, then randomly pick one successor to update $x_{\mathtt{act}}$ in line 10-11. Otherwise, the update principle of action generation state in line 13-16 means that if there no longer exist current inner state $x_{\mathtt{act}}$'s direct successors, then in the next step, current inner product states originated from other initial inner states $x'_{\mathtt{int}}$ will still not be resolved when there still exists unfinished current inner states originated from $x_{\mathtt{int}}$.

The control decision is generated by an offline function `Action`, based on the interconnected transition system of the winning region computed in Subsection 3. Given an action generation state $x_{\mathtt{act}}$, arbitrary belief state q and interconnected transition system $\hat{\mathcal{B}}$, if $(x_{\mathtt{act}}, q)$ is in system's (i+1)-step attractor, we enforce its visit to i-step attractor to approach accepting states. Note that this computation is conducted in interconnected transition system, reflecting the system's dynamic in reality. We assume the system's actual state to be $x_{\mathtt{act}}$ and ignore all other potential inner states of current belief state. Such an assumption is rea-

sonable because all the other unconsidered inner states will not be neglected permanently but considered alternately in the future.

Algorithm 2: `Action`

given $x_{\mathtt{act}}, q, \hat{\mathcal{B}}$
compute $\hat{Q}_{win}, \hat{Q}_{acc}$ out of $\hat{\mathcal{B}}$
find i such that
$(x_{\mathtt{act}}, q) \in \mathtt{Attr}^{(i+1)}(\hat{Q}_{acc}) \backslash \mathtt{Attr}^{(i)}(\hat{Q}_{acc})$
randomly pick an action a such that
$\hat{\Delta}((x_{\mathtt{act}}, q), a) \subseteq \mathtt{Attr}^{(i)}(\hat{Q}_{acc})$

4.3 Properties of Problem Solution

In the proposed algorithm, for an arbitrary belief state inside winning region, whether a product state inside it have visited accepting states is described by Boolean variable b in augmented state. And arriving at an augmented state with all the Boolean variable $b = 1$ means that the system must have visited accepting states during its past transitions regardless of its uncertainty of the environment, which is illustrated in Proposition 2.

Proposition 2. Given a belief transition system $\mathcal{B}$, an arbitrary initial belief state $q \in Q_{win}$ and a non-blocking controller C, if the system visits a state q' with $\forall(x_{\mathtt{int}}, x_{\mathtt{cur}}, b) \in q_{\mathtt{aug}}, b = 1$ then all finite paths from q to q' have reached accepting states.

The algorithm maintains the belief states transition inside winning region to always avoiding the system from possible permanent failure, which is proved by Proposition 1. Also, Proposition 2 shows that the algorithm infinitely guides all the system's potential states towards accepting. Based on the above properties, the proposed algorithm generates a non-blocking controller providing the future satisfaction of given task, which is presented in Proposition 1.

Proposition 3. Given a belief transition system $\mathcal{B}$ with an arbitrary belief state $q_{init} \in Q_{win}$ and a LTL specification ϕ, under C the closed-loop system $\mathcal{T}_C$ satisfies $\mathsf{Trace}^*(\mathcal{T}_C) \subseteq \overline{\mathsf{Word}(\phi)}$.

We now discuss the conditions when our proposed non-blocking controller can deterministically satisfy LTL task by surely winning the game. Given system $\mathcal{T}$, observation mapping H and LTL formula ϕ, controller $C : O^* \to Act$ is a sure winning controller if it induces a closed-loop system $\mathcal{T}_C$ such that $\mathsf{Trace}(\mathcal{T}_C) \subseteq \mathsf{Word}(\phi)$.

It can be obviously concluded that the non-blocking control algorithm becomes a sure winning control algorithm when reaching an convergence, which provides a sufficient condition for sure winning problem solution, shown in Proposition 4.

Proposition 4. Given a belief transition system $\mathcal{B}$ under a non-blocking controller C, if $\forall q \in Q_{win}$, the system will always visits states satisfying $\forall(x_{\mathtt{int}}, x_{\mathtt{cur}}, b) \in q_{\mathtt{aug}}, b = 1$ in finite steps then C is a sure winning controller.

We also provide two sufficient conditions of non-blocking control strategy's convergence, which are common and reasonable in practice.

Proposition 5. We define $Actions(x_{\mathtt{cur}}, q, \hat{\mathcal{B}}) \subseteq Act$ as the set of all actions possibly generated by function $\mathtt{Action}(x_{\mathtt{cur}}, q, \hat{\mathcal{B}})$. Given a belief transition system $\mathcal{B}$ under a non-blocking controller C, if $\forall q \in Q_{win}, \forall x_{\mathtt{cur}}, x'_{\mathtt{cur}} \in q$ satisfy $Actions(x_{\mathtt{cur}}, q, \hat{\mathcal{B}}) = Actions(x'_{\mathtt{cur}}, q, \hat{\mathcal{B}})$ then C is a sure winning controller.

Proposition 6. Given a belief transition system $\mathcal{B}$ under a non-blocking controller C, C is a sure winning controller if the following condition is satisfied: $\forall q_0 \in Q_{win}$, and $\forall \tau_B \in \mathsf{Path}(\mathcal{B}_C)^*$ with initial state of q, if $\exists \tau \in \mathsf{Path}(\mathcal{B}_C)^*$ with $H(\tau) = H(\tau_B)$, such that τ has not visited accepting states, then there does not exist a belief state that appears more than once in τ.

5. CASE STUDY

In this section, our strategy design method is applied to control a robot in a pick-delivery scenario. The robot's mission is to pick up cargo and deliver them to receiving spot while avoiding moving and stationary obstacles. After finishing a delivery mission, a new pair of pick-up and delivery spots will be randomly allocated. The workspace of robot is simplified to a gridworld of 5×5 shown in Fig.2(a). Red regions of (3,1) and (5,1) are pick-up spots of cargo, green regions of (1,3) and (1,5) are delivery destination, black regions of (5,4), (5,5), (4,5) and (3,4) are dangerous regions that the robot must avoid. Blue circle at (1,1) represents robot's charging station. There exists a moving obstacle, whose movement cannot be controlled by robot. The moving obstacle's movement pattern follows a determined trajectory simplified by black dash arrow while randomly waiting at waiting point at (4,4). There exist slopes around pick-up spots, which may affect robot's movement, denoted by grey dash arrows in regions around red regions. Robot's movement opposite to the direction of slope will not be changed, otherwise the environment may force the moving result towards arrow direction. The robot has a limited knowledge of the environment. The robot will not know where the cargo is coming out until it reaches a pick-up spot, so does the case of delivery at entrance doors. Also, the robot has a limited sensing ability so that it will only be alerted of moving obstacle's existence within ambient 8 grids without the precise location. The robot's available actions contain moving one step in four directions and waiting, denoted by $Act = \{\uparrow, \downarrow, \rightarrow, \leftarrow, -\}$. The transition function describes the dynamic of system, and as the robot moves, the environment changes as well. The system state is initialized to be $x_0 = (x_{A0}, (x_{p0}, x_{d0}, x_{w0})) \in X_0$, where $x_{A0} = (1,1), x_{p0} \in \{(3,1),(5,1)\}, x_{d0} \in \{(1,3),(1,5)\}, x_{w0} = (4,4)$. The robot's mission is described by LTL specification $\phi = \square\neg danger \wedge \square\diamondsuit(pick \wedge \diamondsuit deliver) \wedge \square\diamondsuit charge$ and can be described by a DBA $\mathcal{B}$.

Robot's potential decisions at each position under our strategy are shown in Fig.2(b), according to Proposition 6, the algorithm reaches an convergence and becomes sure winning. The robot will firstly move towards the nearest pick-up spot at (3,1). After reaching (3,1), the robot will update its belief of environment and if the cargo is picked up here, it will then directly move towards the nearest potential delivery destination at (1,3). If the cargo is not picked, the robot will make decision based on its obser-

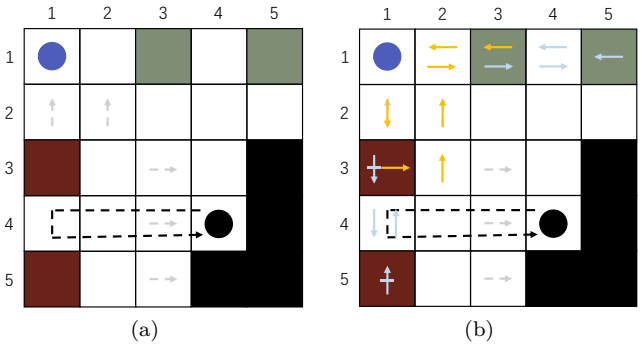

Fig. 2. (a) Workspace simplification for robot in a delivery scenario. (b) Robot's potential action under non-blocking control strategy.

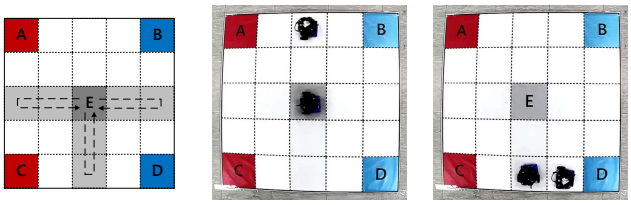

Fig. 3. (a) Gridworld of the experiment (b) Snapshot at initial time point (c) Snapshot during the experiment

vation of the moving obstacle's location. If the moving obstacle is detected nearby, the robot will wait for it to leave, otherwise, the robot will move towards pick-up spot at (5,1). In this case, if the moving obstacle is sensed after arriving at (5,1), the robot will also wait before moving towards the nearest delivery destination. After reaching the delivery destination at (1,3), if the cargo is take away, the robot will return to charging station, otherwise, it will move towards the other destination at (1,5) before its return. Under our proposed strategy, the robot will always complete its mission under any combination of target region while avoiding moving obstacles.

We conduct a simplified real-world experiment of the delivery scenario that only involves random task assignment and moving obstacle avoidance. The workspace is simplified to a gridworld of 5×5 shown in Fig.3(a). The red and blue regions represent pick-up and deliver spots. The moving obstacle is represented by a robot, and its movement is restricted to gray regions represented by dash arrows. The maximum continuous waiting time for robot and moving obstacle are both restricted for better presentation. The initial starting spot is shown in Fig.3(b). The pick-up and delivery spots are initially set to be A and D and changed to C and B. The complete experiment video is available in `https://github.com/LiShuaiyi/ifac2023`.

6. CONCLUSION

In this paper, we formulated and solved a non-blocking control synthesis problem for a fragment LTL tasks under partial observation without assumptions on observations and atomic propositions. Compared with existing works, which focus on (almost) sure winning strategies, we adopted the notion of non-blockingness as the metric to evaluate the winning condition of the controller. We first presented an offline algorithm for computing the winning region of the synthesis problem. Then we presented an online control algorithm for generating control actions on-the-fly. In the future, we aim to identify necessary and sufficient condition under which the non-blocking controller is also sure winning.

REFERENCES

Baier, C. and Katoen, J.P. (2008). *Principles of model checking*. MIT press.

Belta, C., Yordanov, B., and Aydin Gol, E. (2017). Finite temporal logic control. In *Formal Methods for Discrete-Time Dynamical Systems*, 81–108. Springer.

Cai, K., Zhang, R., and Wonham, W.M. (2014). Relative observability of discrete-event systems and its supremal sublanguages. *IEEE Trans. Automatic Control*, 60(3), 659–670.

Chatterjee, K., Chmelik, M., Gupta, R., and Kanodia, A. (2016). Optimal cost almost-sure reachability in POMDPs. *Artificial Intelligence*, 234, 26–48.

Fu, J. and Topcu, U. (2016). Synthesis of joint control and active sensing strategies under temporal logic constraints. *IEEE Trans. Automatic Control*, 61(11), 3464–3476.

Ji, Y., Yin, X., and Lafortune, S. (2021). Optimal supervisory control with mean payoff objectives and under partial observation. *Automatica*, 123, 109359.

Kantaros, Y., Guo, M., and Zavlanos, M.M. (2019). Temporal logic task planning and intermittent connectivity control of mobile robot networks. *IEEE Trans. Automatic Control*, 64(10), 4105–4120.

Ma, Z. and Cai, K. (2021). Optimal secret protections in discrete-event systems. *IEEE Trans. Automatic Control*, 67(6), 2816–2828.

Ramasubramanian, B., Niu, L., Clark, A., Bushnell, L., and Poovendran, R. (2020). Secure control in partially observable environments to satisfy ltl specifications. *IEEE Trans. Automatic Control*, 66(12), 5665–5679.

Raskin, J.F., Henzinger, T.A., Doyen, L., and Chatterjee, K. (2007). Algorithms for omega-regular games with imperfect information. *Logical Methods in Computer Science*, 3.

Ru, Y., Cabasino, M.P., Giua, A., and Hadjicostis, C.N. (2014). Supervisor synthesis for discrete event systems under partial observation and arbitrary forbidden state specifications. *Discrete Event Dynamic Systems*, 24(3), 275–307.

Sakakibara, A. and Ushio, T. (2020). On-line permissive supervisory control of discrete event systems for scltl specifications. *IEEE Control Systems Letters*, 4(3), 530–535.

Yin, X. and Lafortune, S. (2016a). Synthesis of maximally permissive supervisors for partially-observed discrete-event systems. *IEEE Trans. Automatic Control*, 61(5), 1239–1254.

Yin, X. and Lafortune, S. (2016b). A uniform approach for synthesizing property-enforcing supervisors for partially-observed discrete-event systems. *IEEE Trans. Automatic Control*, 61(8), 2140–2154.

Yu, X., Yin, X., Li, S., and Li, Z. (2022). Security-preserving multi-agent coordination for complex temporal logic tasks. *Control Engineering Practice*, 123, 105130.