Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 56-2 (2023) 3591–3597

You Don't Know When I Will Arrive: Unpredictable Controller Synthesis for Temporal Logic Tasks ⋆

Yu Chen * **Shuo Yang** ** **Rahul Mangharam** ** **Xiang Yin** *

* *Department of Automation, Shanghai Jiao Tong University, Shanghai 200240, China. e-mail: {yuchen26,yinxiang}@sjtu.edu.cn)*
** *Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA 19104, USA. (e-mail: {yangs1, rahulm}@seas.upenn.edu)*

Abstract: In this paper, we investigate the problem of synthesizing controllers for temporal logic specifications under security constraint. We assume that there exists a passive intruder (eavesdropper) that can partially observe the behavior of the system. For the purpose of security, we require that the system's behaviors are unpredictable in the sense that the intruder cannot determine for sure that the system will exactly accomplish the task in K steps ahead. This problem is particularly challenging since future information is involved in the synthesis process. We propose a novel information structure that predicts the effect of control in the future. A sound and complete algorithm is developed to synthesize a controller which ensures both task completion and security guarantee. The proposed approach is illustrated by a case study of robot task planning.

Keywords: Information-Flow Security, Robot Path Planning, Temporal Logic Tasks.

1. INTRODUCTION

With the development of information technologies, autonomous systems are required to accomplish complex tasks in dynamic environments. Temporal logics have drawn considerable attention in the control community in the recent years due to their high expressibility in describing complex tasks. Using automata-theoretic approaches, controllers can be effectively synthesized to achieve complex temporal logic tasks Kress-Gazit et al. (2018). When executing tasks, the system usually need to communicate with other components such as clouds or central stations to exchange information. However, the communication process may leak critical information to the outside world. In addition to the correctness requirement, security issue has also been becoming increasingly important consideration in controller synthesis Liu et al. (2022).

Our Contributions: In this paper, we formulate and solve a new security-aware controller synthesis problem for syntactically co-safe linear temporal logic (scLTL) specifications. We model the system as a non-deterministic transition system. The correctness requirement of the controller is to ensure the satisfaction of a given scLTL task. Furthermore, we assume that there is an eavesdropper/intruder monitoring the system's behaviors via an output function. For the purpose of security, we further require that the behaviors of system are *unpredicable* in the sense that the intruder can never determine for sure that the system will finish the task at some specific future instant, which is formally captured by *K-step unpredictability* proposed in this work. We propose a novel approach by "borrowing" information in the future and build an information structure in which all predictions are correct in the sense that they are consistent with what actually happens in the future. A backtracking-based algorithm is then proposed to synthesize a controller that achieves both the scLTL task and the unpredictable requirement.

Related Work: There has been an increasing interest in high-level controller synthesis for robotic specifications; see, e.g., Smith et al. (2011); Cai et al. (2020); Shi et al. (2022). However, these works only focus on the correctness of system without considering the security requirement. In the context of supervisory control of discrete event systems, algorithms have been developed for enforcing the notion of opacity Yin and Lafortune (2016); Tong et al. (2018); Xie et al. (2022); Liu et al. (2022). In the context of security-aware controller synthesis, our work is mostly related to Wang et al. (2020); Yang et al. (2020); Xie et al. (2021); Yu et al. (2022), where both LTL specifications and security constraints are enforced. However, the security constraints in these work require that the intruder can never infer that the robot started from a secret location either initially or at some specific instant. Such security requirements are only related to the past behaviors of the system, while we consider the unpredictability of the *future behavior* of the system.

2. MOTIVATING EXAMPLE

Before we formulate the problem, we first consider a motivating example. We consider a robot moving in a

⋆ This work was supported by the National Natural Science Foundation of China (62061136004, 62173226, 61833012). The first two authors contribute equally.

10.1016/j.ifacol.2023.10.1519

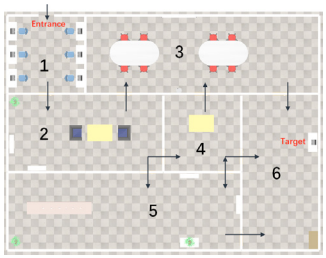


Fig. 1. Robot workspace.

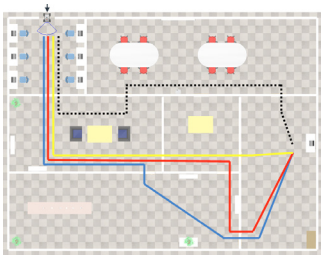


Fig. 2. Different paths.

workspace with six regions as shown in Fig. 1. The robot is initialized at Region 1. Black arrows in the figure represent the permitted moves of the robot from one region to another. Note that, from Region 2, the robot cannot move to Regions 4 or 5 for sure since two doors are too closed, i.e., these two transitions are non-deterministic. Similar situation happens when the robot tries to move from from Region 4 to Region 5 or 6.

The objective of the robot is to first reach Region 2 to collect a critical message and then move to Region 6 to broadcast it. Assume that there exists an intruder knowing the control policy and the current location of robot. The intruder intends to hack into the broadcasting channel to get the message. However, it only has one chance to hack the channel. Therefore, the intruder needs to predict the precise time instant when the robot will finish the task, i.e., reaching Region 6 for the first time.

Consider the control policy generating an unique path shown as the black dashed line in Fig. 2. Clearly, once robot has started from Region 1, the intruder knows for sure that the robot will be in Region 6 after 3 steps. Thus, this control policy is not "secure". However, the robot can choose to go to Region 4 or 5 from Region 2, and then go to Region 5 or 6 when in Region 4, and finally go to Region 6 when in Region 5. Under this control policy, robot will generate three possible paths which are shown as the colored lines in Fig. 2. When the robot has started from Region 1, after 3 steps, it can be in either Region 5 or 6. Therefore, the intruder cannot successfully predict that the robot will reach Region 6 in 3 steps for the first time.

3. PRELIMINARY

3.1 System Model

Let A be a finite set of symbols. We denote by A^* and A^ω the sets of all finite and infinite sequences over A, respectively, and $\epsilon \in A^*$ is the empty sequence. For any finite sequence $s = a_1 \dots a_n \in A^*$, we denote by $|s| = n$ and $\mathsf{last}(s) = a_n$ its length and its last element, respectively.

The mobility of the robot in a workspace is modelled as a (non-deterministic) transition system

$$G = (X, x_0, U, \to, \mathcal{AP}, L),$$

where X is the set of *states* representing the locations of the robot; $x_0 \in X$ is the *initial state* of the robot; U is the set of *control inputs* or *actions* taken by the robot at each instant; $\to \subseteq X \times U \times X$ is the *transition relation*, where $(x, u, x') \in \to$ (also denoted by $x \xrightarrow{u} x'$) means that the robot can possibly move from state x to state x' under action u; $\mathcal{AP}$ is the set of *atomic propositions* representing some basic properties of our interest; and $L : X \to 2^{\mathcal{AP}}$ is the *labeling function* assigning each state x a set of atomic propositions that hold at state x.

Given an input sequence $u_1 \dots u_n \in U^*$, it may induce a finite state *run* $x_0 \xrightarrow{u_1} x_1 \xrightarrow{u_2} \cdots \xrightarrow{u_n} x_n$ such that $x_i \xrightarrow{u_{i+1}} x_{i+1}, \forall i = 0, \dots, n-1$; state sequence $x_0 \dots x_1 \in X^*$ is called a finite *path* of system G. Note that the run or path induced by an input sequence may not be unique when the system is non-deterministic. We denote by $\mathsf{Path}(G)$ the set of all finite paths generated by G starting from the initial state x_0. The *trace* of a path $\tau = x_0 \dots x_n \in \mathsf{Path}(G)$ is defined by $\mathsf{trace}(\tau) = L(x_0) \dots L(x_n)$, and we denote by $\mathsf{Trace}(G) = \{\mathsf{trace}(\tau) : \tau \in \mathsf{Path}(G)\}$ the set of all traces in G. System G is called *live* if for any $x \in X$, there exist $u \in U$ and $x' \in X$ such that $(x, u, x') \in \to$. We assume that system G is live. For each state $x \in X$, we define $U(x) = \{u \in U : \exists x' \in X \text{ s.t. } x \xrightarrow{u} x'\}$ as the set of active inputs at x. For a set of states $q \subseteq X$ and a control input $u \in U$, we define the successor states of q under u by

$$\mathbf{NX}_u(q) = \{x' \in X : \exists x \in q \text{ s.t. } x \xrightarrow{u} x'\}. \quad (1)$$

3.2 Linear Temporal Logic Specifications

The task of the robot is described by a syntactically co-safe Linear Temporal Logic (scLTL) formula φ over atomic propositions $\mathcal{AP}$. Formally, the syntax of scLTL is defined as

$$\varphi ::= true \mid a \mid \neg a \mid \varphi_1 \wedge \varphi_2 \mid \bigcirc \varphi \mid \varphi_1 U \varphi_2,$$

where $a \in AP$ is an atomic proposition; $\neg$ and $\wedge$ are Boolean operators "negation" and "conjunction", respectively; $\bigcirc$ and U are temporal operators "next" and "until", respectively.

In general, LTL formula are evaluated on infinite words over $2^{\mathcal{AP}}$. For any infinite word $\rho \in (2^{\mathcal{AP}})^\omega$, we denote by $\rho \models \varphi$ if word ρ satisfies LTL formula φ. The reader is referred to Baier and Katoen (2008) for details of the semantics of LTL. However, for an scLTL formula, its satisfaction can be determined in finite horizon. It is well-known that, for any infinite word $\rho = \rho_0 \rho_1 \dots \in (2^{\mathcal{AP}})^\omega$ such that $\rho \models \varphi$, it has a *finite good prefix* $\rho_{pref} = \rho_0 \rho_1 \dots \rho_n$ in the sense that $\rho_{pref} \rho' \models \varphi$ for any $\rho' \in (2^{\mathcal{AP}})^\omega$. We denote by $\mathsf{Word}_{pref}(\varphi)$ the set of all finite good prefixes for scLTL formula φ, and for a finite word $\rho \in (2^{\mathcal{AP}})^*$, we also write $\rho \models \varphi$ if $\rho \in \mathsf{Word}_{pref}(\varphi)$. For system G, we denote by $G \models \varphi$ if every infinite path in G has a finite good prefix. A good prefix is said to be minimal if it has no good prefix except itself, and we denote by $\mathsf{Word}^{min}_{pref}(\varphi)$ the set of minimal good prefixes. Intuitively, $\mathsf{Word}^{min}_{pref}(\varphi)$ represents all words that *exactly accomplishes the task* φ for the first time. We denote by $\sigma \models_1 \varphi$ if $\sigma \in \mathsf{Word}^{min}_{pref}(\varphi)$, i.e., σ exactly accomplishes the task for the first time. Therefore, $\sigma \not\models_1 \varphi$ means either the task is not yet satisfied or has been satisfied before.

The set of words satisfying an scLTL formula can be accepted by a *deterministic finite-state automata* (DFA). Formally, a DFA is a 5-tuple

$$A = (S, s_0, \xi, \Sigma, F),$$

where S is the set of states, $s_0 \in S$ is the initial state, Σ is the alphabet, $\xi : S \times \Sigma \to S$ is the deterministic transition function, and $F \subseteq S$ is the set of accepting

states. The transition function can be extended to $\xi : S \times \Sigma^* \to S$ recursively by: $\forall x \in S, \rho \in \Sigma^*, \sigma \in \Sigma, \xi(x, \rho\sigma) = \xi(\xi(x, \rho), \sigma)$. We denote by $\mathcal{L}(A)$ the set of all finite words *accepted* by A, i.e., $\mathcal{L}(A) = \{\rho \in \Sigma^* : \xi(s_0, \rho) \in F\}$.

For any scLTL formula φ, there always exists a DFA A_φ over $\Sigma = 2^{\mathcal{AP}}$ that only accepts all good prefixes, i.e., $\mathcal{L}(A_\varphi) = \mathsf{Word}_{pref}(\varphi)$. To distinguish the exact satisfaction of the task for the first time and after the first satisfaction, we further modify DFA A_φ by the following steps: (1) add a new accepting state s_F with self-loops for all labels in $\Sigma = 2^{\mathcal{AP}}$; and (2) remove all output transitions from the original accepting states F, and add new transitions from F to s_F with labels for all $\Sigma = 2^{\mathcal{AP}}$; and (3) add a new "bad" state s_B, and for each state s, if any $\sigma \in 2^{\mathcal{AP}}$ is undefined, then we add a transition from s to s_B labeled by σ. Clearly, the modified DFA generates all words $(2^{\mathcal{AP}})^*$ and accepts exactly the same words as the original DFA. Hereafter, we will use this modified version and still denote it by A_φ.

3.3 Controller

Since not all traces in the system satisfy the scLTL formula, a feedback *controller* (or policy) should be imposed. We consider a partially-observed setting, i.e., the controller cannot obtain the perfect state information of the system. To this end, we consider an observation mapping

$$H : X \to O,$$

where O is the set of observation symbols and for each $x \in X$, $H(x)$ represents the observation of the controller at state x. Then for each path $\tau = x_0 \dots x_n$, its corresponding observation is $H(\tau) = H(x_0) \dots H(x_n)$. We define $H(\mathsf{Path}(G)) = \{H(\tau) \in O^* : \tau \in \mathsf{Path}(G)\}$ the set of all observations. Then a controller for G is a function

$$C : H(\mathsf{Path}(G)) \to U,$$

which determines the control input based on the observation. The closed-loop system under controller C is denoted by G_C. Specifically, we say that a path $\tau = x_0 \xrightarrow{u_1} x_1 \xrightarrow{u_2} \cdots \xrightarrow{u_n} x_n$ is feasible under C if $\tau \in \mathsf{Path}(G)$ and $u_i = C(H(x_0 \dots x_{i-1})), \forall i \in \{1, \dots, n-1\}$. We denote by $\mathsf{Path}(G_C)$ and $\mathsf{trace}(G_C)$ the set of all feasible paths and traces in G under controller C, respectively. For technical purposes, we assume that for any two states having the same observation, they have the same set of active control inputs, i.e., $\forall x, x' \in X : H(x) = H(x') \Rightarrow U(x) = U(x')$.

4. INTENTION-AWARE SECURE SYNTHESIS PROBLEM

4.1 Intruder Model

During the execution of the system, the information available to the controller may also be released to the outside. In this work, we consider an attacker modelled as a passive observer having the following capabilities: 1) It knows the system model G, scLTL task φ, as well as the functionality of the controller C; and 2) It can also access the observation of the controller.

Therefore, for each path $\tau = x_0 \dots x_n$, the observation of the intruder is also $H(\tau) = H(x_0) \dots H(x_n)$.

Remark 1. The above intruder model has been widely considered in both computer science and control engineering literature; see, e.g., Lafortune et al. (2018); Liu et al. (2022). This model is motivated by the fact that offline designed policies are usually public information, while actual online trajectories are usually only partially released to the outsider. Therefore, the intruder needs to use this partial online information together with the offline model information to infer the behavior of the system.

4.2 Unpredictable Security Requirement

Depending on what information the system wants to hide, the security of the system can be defined differently. In this work, we consider the following security requirement:

- the system does not want the intruder to infer too early *when it will accomplish the task*.

In other words, the controlled behavior of the system needs to be *unpredictable* in the sense that, by observing external information-flow, the intruder is unable to infer confidentially that the robot will finish task exactly after a certain number of steps.

The above requirement was originally formulated by the notion of *K-step pre-opacity* in Yang and Yin (2022), where the completion of the task is modelled by reaching a target state. We then formulate our unpredictable controller synthesis problem by adopting this notion. First, we introduce the notion of K-step unpredictability w.r.t. an scLTL task.

Definition 1. (K-Step Unpredictability). Given system G, scLTL task φ and controller C with output function H, a finite path $\tau \in \mathsf{Path}(G_C)$ is said to be K-step unpredictable w.r.t. H and φ if

$$(\forall m \geq K)(\exists \tau_1\tau_2 \in \mathsf{Path}(G_C)) \text{ s.t. } [|\tau_2| = m] \wedge [H(\tau) = H(\tau_1)] \wedge [\mathsf{trace}(\tau_1\tau_2) \not\models_1 \varphi]. \quad (2)$$

We say that the controlled system G_C is K-step unpredictable if all paths in it are K-step unpredictable.

With this notion, we formulate the synthesis problem that we solve in this paper.

Problem 1. (Unpredictable Control Synthesis Problem). Given system G, scLTL task φ and output function H for both intruder and controller, determine a controller C such that (i) G_C is live; and (ii) $G_C \models \varphi$; and (iii) G_C is K-step unpredictable.

Remark 2. In the above problem formulation, we require that the controlled system G_C is live, i.e., it needs to work indefinitely. In practice, since we consider scLTL task here, the robot can stop once it knows for sure that the task has been completed. However, this assumption is mainly for the sake of simplification of analysis and is without loss of generality. For example, one can add a virtual "stop" state with self-loops to mimic liveness when the robot stops.

4.3 Product System and Problem Transformation

To incorporate the task information into the system model, we construct the *product* of system and the task DFA.

Definition 2. (Product System). Given system G and scLTL task φ, let $A_\varphi = (S, s_0, \xi, \Sigma, F \cup \{s_F\})$ be the

(modified) DFA such that $\mathcal{L}(A_\varphi) = \mathsf{Word}_{pref}(\varphi)$. The *product* of G and A_φ is a new transition system

$$\tilde{G} = (\tilde{X}, \tilde{x}_0, U, \rightarrow_\otimes, \mathcal{AP}, \tilde{L}),$$

where $\tilde{X} \subseteq X \times X_F$ is the set of states; $\tilde{x}_0 = (x_0, \xi(s_0, L(x_0))) \in \tilde{X}$ is the initial state; U is the set of control inputs; $\rightarrow_\otimes \subseteq \tilde{X} \times U \times \tilde{X}$ is transition relation defined by: for any $\tilde{x}_1 = (x_1, s_1) \in \tilde{X}, \tilde{x}_2 = (x_2, s_2) \in \tilde{X}$ and $u \in U$, we have $\tilde{x}_1 \xrightarrow{u}_\otimes \tilde{x}_2$ if (i) $x_1 \xrightarrow{u} x_2$ and (ii) $s_2 = \xi(s_1, L(x_2))$; $\mathcal{AP}$ is the same set of atomic propositions; and $\tilde{L} : \tilde{X} \to 2^{\mathcal{AP}}$ is labeling function such that, for any $\tilde{x} = (x, s)$, we have $\tilde{L}(\tilde{x}) = L(x)$.

It is easy to know that any controller C designed for $\tilde{G}$ can be applied to G, and vice versa. Hereafter, we will focus on the product system $\tilde{G}$.

Note that, the information of the satisfaction of scLTL task φ has also been encoded in the second component of $\tilde{G}$. To see this more clearly, we define

$$\tilde{X}_F = \{(q, s) \in \tilde{X} : s \in F\},$$
$$\tilde{X}_{F \cup \{s_F\}} = \{(q, s) \in \tilde{X} : s \in F \cup \{s_F\}\}.$$

Then for any path $\tau \in \mathsf{Path}(\tilde{G})$, we have the followings:

$$\mathsf{trace}(\tau) \models_1 \varphi \Leftrightarrow \mathsf{last}(\tau) \in \tilde{X}_F, \tag{3}$$
$$\mathsf{trace}(\tau) \models \varphi \Leftrightarrow \mathsf{last}(\tau) \in \tilde{X}_{F \cup \{s_F\}}. \tag{4}$$

We also call $\tilde{X}_F$ *secret states* since they represent those states whose visits do not want to be predicted by the intruder. In addition to replace $\mathsf{trace}(\tau_1\tau_2) \not\models_1 \varphi$ in Definition 1 by $\mathsf{last}(\tau_2) \notin \tilde{X}_F$, here, we provide a stronger result showing that the requirement of "$\forall m \geq K$" can also be simplified by only considering $m = K$.

Proposition 1. Given system $\tilde{G}$ and controller C, $\tilde{G}_C$ is K-step unpredictable w.r.t. H and φ if and only if for any finite path $\tau \in \mathsf{Path}(\tilde{G}_C)$ there exists a path $\tau_1\tau_2 \in \mathsf{Path}(\tilde{G}_C)$ s.t.

$$[|\tau_2| = K] \wedge [\tilde{H}(\tau) = \tilde{H}(\tau_1)] \wedge [\mathsf{last}(\tau_2) \notin \tilde{X}_F].$$

Based on the product system $\tilde{G}$ and Proposition 1, we can transform Problem 1 regarding system G and scLTL task φ to a problem depending only on $\tilde{G}$. That is, our objective beomces to synthesize a live controller C for $\tilde{G}$ such that all traces in $\tilde{G}_C$ end up with $\tilde{X}_F$ are K-steps unpredictable as characterized by Proposition 1. To simplify the notation, in the rest of this paper, we denote system $\tilde{G} = (\tilde{X}, \tilde{x}_0, U, \rightarrow_\otimes, \mathcal{AP}, \tilde{L})$ by $G = (X, x_0, U, \rightarrow, \mathcal{AP}, L)$ by understanding how the product is constructed. We also denote states set $\tilde{X}_F$ and $\tilde{X}_{F \cup \{s_F\}}$ by X_F and $X_{F \cup \{s_F\}}$, respectively.

5. PREDICTION SETS AND BIPARTITE TRANSITION SYSTEMS

5.1 Prediction Sets

Compared with the existing security-aware controller synthesis problems for current-state opacity, our K-step unpredictable controller synthesis problem has the following new challenge. In the synthesis for current-state opacity, whether or not the secret of the system is revealed to the intruder can be completely determined by the historical information up to the current instant. However, in our problem, whether or not the system is secure depends on its behavior *in the future*. This is not a verification problem for open-loop system since all future behaviors are given and fixed. However, in the synthesis problem, the future behaviors of the system are unknown and depend on the control decisions in the future, which have not yet been determined. This *future dependency issue* makes our synthesis problem particularly challenging.

To resolve the future dependency issue, we propose a novel approach by "borrowing" information in the future. The general idea is as follows. At each instant, the system makes a *prediction* regarding in how many number of steps, the scLTL task will be accomplished, or equivalently, secret states X_F will be reached. In order to make the prediction valid, the system should accomplish the task as it predicted; otherwise, such a prediction will be considered invalid and be truncated.

To formalize the above idea, we define a *prediction* as a $(K+1)$-dimensional binary vector $h = (h[0], h[1], \cdots, h[K]) \in \{0, 1\}^{K+1}$, where for each $i = 0, 1, \ldots, K$,

- $h[i] = 0$ means that the system *may not reach* secret states X_F in exactly i steps; and
- $h[i] = 1$ means that the system *will reach* secret states X_F *for sure* in exactly i steps.

We denote by $\mathcal{H} = \{0, 1\}^{K+1}$ the prediction set. We augment the state space of G with prediction set and denote by $\hat{X} = X \times \mathcal{H}$ the augmented state space. For each augmented state $\hat{x} = (x, h) \in \hat{X}$, we define $\mathsf{state}(\hat{x}) = x$ and $\mathsf{pred}(\hat{x}) = h$ as its state and prediction components, respectively. Then, for a set of augmented states $\imath = \{(x_1, h_1), ..., (x_n, h_n)\} \subseteq \hat{X}$, we define $\mathsf{state}(\imath) = \{x_1, ..., x_n\}$ and $\mathsf{pred}(\imath) = \{h_1, ..., h_n\}$. With some abuse of notation, for $(x, h) = \hat{x} \in \hat{X}$, we also write $H(\hat{x}) = H(x)$.

Now, we discuss what properties a "good" prediction should have. First, in an augmented state $\hat{x} = (x, h) \in \hat{X}$, its prediction for the current instant, i.e., $h[0]$, should be consistent with the fact of the current state x. This is captured by the following definition.

Definition 3. (Current Consistency). An augmented state $\hat{x} \in \hat{X}$ is called *currently consistent* if

$$\mathsf{pred}(\hat{x})[0] = 1 \Leftrightarrow \mathsf{state}(\hat{x}) \in X_F. \tag{5}$$

We denote by $\hat{X}_{cons} \subseteq \hat{X}$ the set of all currently consistent augmented states.

Second, let $\hat{x} \in \hat{X}$ be an augmented state, and $\imath \in 2^{\hat{X}}$ be a set of augmented states representing all successor states of $\hat{x}$ in *one step*. Then the predictions of $\imath$ at the next instant should be consistent with the predictions of $\hat{x}$ at the current instant. Formally, we have the following definition.

Definition 4. (One-Step Consistency). Let $h \in \mathcal{H}$ be a prediction and $H \in 2^{\mathcal{H}}$ be a set of predictions. We say h and H are *one-step consistent* if for each $i = 1, \ldots, K$,

$$\begin{aligned} h[i]=1 \;&\Rightarrow\; \forall h' \in H : h'[i-1]=1, \\ h[i]=0 \;&\Rightarrow\; \exists h' \in H : h'[i-1]=0. \end{aligned} \tag{6}$$

We denote by $(h, H) \in \Delta$ if they are one-step consistent.

Due to the partial observability, the system may not know the precise state and the prediction. Instead, it may hold a *belief* regarding the augmented state.

Definition 5. (Belief States). A *belief state* is a set of currently consistent augmented states $\imath \in 2^{\hat{X}_{cons}}$ such that

$$\forall (x,h),(x',h') \in \imath : x = x' \Rightarrow h = h'.$$

We denote by $\mathcal{B} \subseteq 2^{\hat{X}_{cons}}$ the set of all belief states.

If all states in a belief state has the same prediction, then even if the belief state is not a singleton, we can still make a conclusion regarding when the secret state will be reached.

Definition 6. (Secure Belief States). A belief state $\imath \in \mathcal{B}$ is said to be *insecure* if

$$\forall \hat{x} \in \imath : \mathsf{pred}(\hat{x})[K] = 1.$$

Otherwise, $\imath$ is called *secure*. We denote by $\mathcal{B}_{ins}$ and $\mathcal{B}_{sec}$ the set of insecure and secure belief states, respectively.

5.2 Bipartite Transition Systems

In order to synthesize an unpredictable controller, our approach is to enumerate all possible control actions over the belief state space. Then, we solve a safety game to avoid those insecure belief states at which the intention of accomplishing the task in K steps is revealed.

Definition 7. (Bipartite Transition Systems) A bipartite transition system (BTS) of G is a 7-tuple

$$T = (Q_Y, Q_Z, \delta_{YZ}, \delta_{ZY}, U, O, Y_0),$$

where

- $Q_Y \subseteq \mathcal{B}_{sec} \times O$ is the set of Y-states;
- $Q_Z \subseteq \mathcal{B}_{sec} \times U$ is the set of Z-states;
- $\delta_{YZ} : Q_Y \times U \to 2^{Q_Z}$ is a non-deterministic transition function from Y-states to Z-states defined by: for any $y = (\imath, o) \in Q_Y, u \in U$, and $z = (\imath', u') \in Q_Z$, we have $z \in \delta_{YZ}(y,u)$ if (i) $u = u'$; and (ii) $\mathsf{state}(\imath') = \mathbf{NX}_u(\mathsf{state}(\imath))$; and (iii) for $\hat{x} \in \imath$, we have $(\mathsf{pred}(\hat{x}), \mathsf{pred}(\imath'(\hat{x},u))) \in \Delta$, where
$$\imath'(\hat{x},u) = \{\hat{x}' \in \imath' : \mathsf{state}(\hat{x}') \in \mathbf{NX}_u(\mathsf{state}(\hat{x}))\}.$$
- $\delta_{ZY} : Q_Z \times O \to Q_Y$ is the deterministic transition function from Z-states to Y-states defined by: for any $z = (\imath, u) \in Q_Z, o \in O$, and $y = (\imath', o') \in Q_Y$, we have $\delta_{ZY}(z,o) = y$ if (i) $o' = o$; and (ii) $\imath' = \{\hat{x} \in \imath : H(\hat{x}) = o\}$.
- U is the set of input;
- O is the output set;
- $Y_0 = \{(\{(x_0,h)\}, H(x_0)) : (x_0,h) \in \mathcal{B}_{sec}\}$ is the set of possible initial Y-states.

Note that for $(\imath, o) \in Q_Y$ and $(\imath', u) \in Q_Z$, o and u are redundant information since they are uniquely determined by their input transitions, and we use them mainly to distinguish between Y and Z-states. With some abuse of notation, for $q = (\imath, b) \in Q_Y \cup Q_Z$, we sometimes omit b and also write $\mathsf{state}(q) = \mathsf{state}(\imath)$ and $\mathsf{pred}(q) = \mathsf{pred}(\imath)$.

For the purpose of control, we need to further require that, in each Y-state, there exists a control input under which it has successor and, in each Z-state all feasible observations are defined. Formally, given a BTS T, we say

- a Y-state $y = (\imath, o) \in Q_Y$ is *complete* if there exists a control input defined at y in T, i.e., $\exists u \in U : \delta_{YZ}(y,c) \neq \emptyset$;
- a Z-state $z = (\imath, u) \in Q_Z$ is *complete* if all feasible observations are defined at z in T, i.e., $\forall \hat{x} \in \imath : \delta_{ZY}(z, H(\hat{x}))!$.

Then BTS T is said to be complete if all Y-states and Z-states in it are complete.

Note that a complete BTS T may contain multiple controlled behaviors. In order to "decode" a controller from T, we need to resolve the above nondeterminism.

Definition 8. (Deterministic BTS). A complete BTS $T = (Q_Y, Q_Z, \delta_{YZ}, \delta_{ZY}, U, O, Y_0)$ is said to be *deterministic* if

(i) The initial Y-state is unique, i.e., $Y_0 = \{y_0\}$; and
(ii) For any Y-state, there exists a unique control input defined, i.e., $\forall y \in Q_Y : |\{u : \delta_{YZ}(y,u)!\}| = 1$; and
(iii) For each Y-state and control input, the transition is deterministic, i.e., $\forall y \in Q_Y, u \in U : |\delta_{YZ}(y,u)| \leq 1$.

For a deterministic BTS T, we denote by $U_T(y)$ the unique control input defined at Y-state $y \in Q_Y$ in T. Let $T = (Q_Y, Q_Z, \delta_{YZ}, \delta_{ZY}, U, O, y_0)$ be a deterministic BTS and $\pi = o_0 u_0 o_1 u_1 \dots o_{n-1} u_{n-1} o_n \in O(UO)^*$ be an alternating sequence of observations and control inputs. Then π visits a unique sequence of states $y_0 z_0 y_1 z_1 \dots y_{n-1} z_{n-1} y_n$, where y_0 is the initial Y-state, and for each $i \in \{0,1,\dots,n-1\}$, we have $z_i = h_{ZY}(y_i, u_i)$ and $y_{i+1} = h_{ZY}(z_i, o_{i+1})$. We denote $Y_T(\pi) = y_n$ as the last Y-state reached by π. Note that, since T is deterministic, the above state sequence is uniquely determined by its observation part $\mathbf{o} = o_0 o_1 \dots o_n$. Therefore, we can also write $Y_T(\mathbf{o})$ as the unique Y-state reached by a sequence whose observation part is $\mathbf{o}$. Then for a deterministic BTS T, we can "decode" a controller, denoted by C_T, as follows: for any observation $\mathbf{o} \in H(\mathsf{Path}(G))$, we have

$$C_T(\mathbf{o}) = U_T(Y_T(\mathbf{o})), \tag{7}$$

which is the unique control input defined at the unique Y-state reached by $\mathbf{o}$.

5.3 Properties of the BTS

Now we show that for a deterministic BTS, the prediction of each state indeed corresponds to the information of when it will reach a secret state X_F, i.e., complete the task, under the induced controller. To this end, for any path $\tau \in \mathsf{Path}(G_C)$, we define

$$\mathsf{Reach}^i_{G_C}(\tau) = \{\mathsf{last}(\tau\tau') \in X : \tau\tau' \in \mathsf{Path}(G_C), |\tau'| = i\}$$

as the set of states the controlled system G_C can reach in i steps following τ. Then, we have the following result.

Proposition 2. Let T be a deterministic BTS and C_T be its induced controller. For path $\tau \in \mathsf{Path}(G_C)$, let $(\imath, o) = Y_T(H(\tau))$ be the Y-state reached in T along the observation of τ. Then we have

(i) there exists a unique augmented state $\hat{x} \in \imath$ such that $\mathsf{state}(\hat{x}) = \mathsf{last}(\tau)$; and
(ii) for the unique augmented state $\hat{x} \in \imath$, we have
$$\forall i = \{0,\dots,K\} : \mathsf{pred}(\hat{x})[i] = 1 \Leftrightarrow \mathsf{Reach}^i_{G_{C_T}}(\tau) \subseteq X_F.$$

Proposition 2 leads to the following result.

Proposition 3. Let T be a deterministic BTS and C_T be its induced controller. Then the controlled system G_{C_T} is K-step unpredictable.

6. CONTROLLER SYNTHESIS PROCEDURE

In the previous section, we have shown how to "decode" an unpredictable controller from a deterministic BTS. In this section, we discuss how to build such a deterministic BTS.

First, we introduce the notion of all enforcement structure.

Definition 9. (All Enforcement Structure). Given system G, its *all enforcement structure* $AES(G) = (Q_Y^{AES}, Q_Z^{AES}, \delta_{YZ}^{AES}, \delta_{ZY}^{AES}, U, O, Y_0^{AES})$ is defined as the largest complete BTS. By "largest", we mean that for any complete BTS T, we have that $T \sqsubseteq AES(G)$, where $\sqsubseteq$ denotes the standard sub-system (graph) inclusion.

Next, we need to extract a deterministic BTS T from the AES such that scLTL task is enforced. Since our purpose is to reach states in $X_{F\cup\{s_F\}}$, we define

$$Q_Y^{F(0)} = \{y \in Q_Y^{AES} \mid \mathsf{state}(y) \subseteq X_{F\cup\{s_F\}}\},$$
$$Q_Z^{F(0)} = \{z \in Q_Z^{AES} \mid \mathsf{state}(z) \subseteq X_{F\cup\{s_F\}}\},$$

as the sets of Y and Z states such that their state estimate components are subsets of $X_{F\cup\{s_F\}}$, respectively. Therefore, our goal is to choose an initial Y-state from the AES, and choose a control input and a prediction for each Y-state, such that $Q_Y^{F(0)} \cup Q_Z^{F(0)}$ is reached in a finite number of steps. This is essentially a reachability game on the AES by considering all Y-states as control nodes and all Z-states as adversary nodes. To this end, we define $Q_Y^{F(k)}$ and $Q_Z^{F(k)}$ as follows:

$$Q_Y^{F(k+1)} = \{y \in Q_Y^{AES} : \exists u \in U \text{ s.t. } \delta_{YZ}^{AES}(y,u) \cap Q_Z^{F(k)} \neq \emptyset\} \cup Q_Y^{F(k)}, \tag{8}$$

$$Q_Z^{F(k+1)} = \{z \in Q_Z^{AES} : \forall o \in O \text{ s.t. } \delta_{ZY}^{AES}(z,o) \subseteq Q_Y^{F(k)}\} \cup Q_Z^{F(k)}. \tag{9}$$

Define $Q_Y^F = \bigcup_{k\geq 0} Q_Y^{F(k)}$ and $Q_Z^F = \bigcup_{k\geq 0} Q_Z^{F(k)}$. Intuitively, for each Y-state y, $y \in Q_Y^{F(k)} \setminus Q_Y^{F(k-1)}$ means that the controller can ensure to reach $X_{F\cup\{s_F\}}$ in k steps. Then we define a distance function $\mathsf{dist} : Q_Y^{AES} \cup Q_Z^{AES} \to \mathbb{N} \cup \{\infty\}$ by: for each Y-state $y \in Q_Y^{AES}$,

$$\mathsf{dist}(y) = \begin{cases} k & \text{, if } y \in Q_Y^{F(k)} \setminus Q_Y^{F(k-1)} \\ \infty & \text{, if } y \notin Q_Y^F \end{cases}, \tag{10}$$

and the same for each Z-state $z \in Q_Z^{AES}$.

The complete synthesis algorithm is shown in Algorithm 1. After constructing the AES and the distance function dist, we extract a deterministic BTS from the AES by procedure $\texttt{Extract}(\cdot)$. The complexity of the algorithm is exponential in the size of the system. However, it is known that such an exponential complexity is unavoidable for partially-observed synthesis problem. The following theorem shows the correctness of the proposed algorithm.

Theorem 1. Algorithm 1 is both sound and complete, i.e., its output is a solution to Problem 1, and if it returns "no solution", then Problem 1 has no solution.

7. ILLUSTRATIVE CASE STUDY

We revisit the motivating example in Section 2. The mobility of the robot can be modeled as the non-deterministic transition system shown in Fig. 3(a). For instance, both $(2, c_1, 4)$ and $(2, c_1, 5)$ are legal transitions, which means that the robot can reach Region 4 or 5 from Region 2 under the control action c_1, but it cannot decide for sure which region it will reach. We choose $\mathcal{AP} = \{P_1, P_2\}$ and the labeling function is defined by $L(2) = \{P_1\}$, $L(6) = \{P_2\}$, and $L(s) = \emptyset$ otherwise.

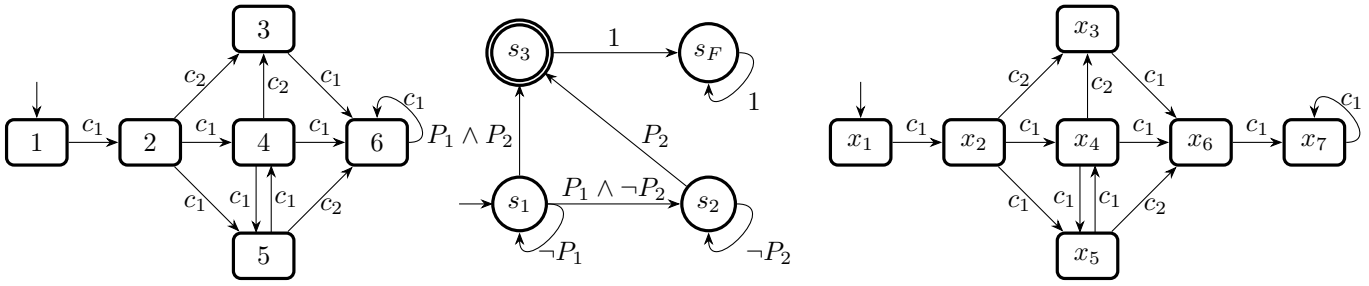
The task of the robot is to reach Region 2 first and then reach Region 6 eventually, which can be expressed by the sc-LTL formula $\varphi = \Diamond(P_1 \wedge \Diamond P_2)$. The (modified) DFA satisfying $\mathcal{L}(A_\varphi) = \mathsf{Word}_{pref}(\varphi)$ is shown in Fig. 3(b). The product system is presented in Fig. 3(c), where $X_F = \{x_6\}$ and $X_{F\cup\{s_F\}} = \{x_6, x_7\}$. Since both controller and intruder have full state information of the robot, we have $\tilde{H}((x,s)) = H(x) = x$, i.e., output function is an identity map.

Let us consider parameter $K = 3$, which means that the intruder should not predict the exact satisfication time of the robot 3 steps ahead. The $AES(G)$ is shown in Fig. 4, where circle states represent Y-states and rectangular states represent Z-states. We use q_i to denote the augmented state such that $\mathsf{state}(q_i) = x_i$. Also, for different j, q_i^j represent different predictions for x_i. For instance, $\mathsf{pred}(q_4^1) = (0,0,0,0)$ and $\mathsf{pred}(q_4^2) = (0,0,1,0)$. The deterministic BTS T in line 8 of Algorithm 1 is presented by the part within the dashed line of Fig. 4. The controller C decoded from it works as follows: $C(x_1) = c_1$, $C(x_1x_2) = c_1$, $C(x_1x_2x_4) = c_1$, $C(x_1x_2x_5) = c_2$, $C(x_1x_2x_4x_5) = c_2$, $C(\tau) = c_1$ for any other $\tau \in \mathsf{Path}(G_C)$. Regarding predictions, for example, we have $h = \mathsf{pred}(q_4^1) = (0,0,0,0)$, $H = \mathsf{pred}(\{q_5^2, q_6\}) = \{(0,1,0,0),(1,0,0,0)\}$, and $(h,H) \in \Delta$. The reasons are as followings: (i) $h[0] = 0$ since $x_4 \notin X_F$; and (ii) $h[1] = 0$ since x_4 has the successor x_5; and (iii) for $i \in \{2,3\}$, $h[i] = 0$ since $x_4x_6(x_7)^*$ are paths starting from x_4 whose last elements are not in X_F. Thus, this

Algorithm 1: Unpredictable Controller Synthesis

Input: system G, scLTL formula φ
Output: controller C_T

1 construct DFA A_φ for φ
2 construct product system of G and A_φ
3 construct the AES $AES(G)$
4 compute distance function dist by Equation (10)
5 **if** $Y_0^{AES} \cap Q_Y^F = \emptyset$ **then**
6 **return** no solution exists
7 **else**
8 $T \leftarrow \texttt{Extract}(AES(G), \mathsf{dist})$
9 **return** decoded controller C_T from T
10 **procedure** $\texttt{Extract}(AES(G), \mathsf{dist})$
11 pick an initial $y_0 \in Y_0^{AES} \cap Q_Y^F$ and add y_0 into T
12 **while** T *is not complete* **do**
13 **for** *incomplete* Y*-state* y *in* T **do**
14 find $y \xrightarrow{u} z$ in the AES s.t. $\mathsf{dist}(z) < \mathsf{dist}(y)$
15 add state z to Q_Z^T and transition $y \xrightarrow{u} z$ to δ_{YZ}^T
16 **for** *incomplete* Z*-state* z *in* T **do**
17 add all successor Y-states and the associated transitions in the AES to T



(a) Transition system G (b) DFA A_φ (c) Product system $\tilde{G}$

Fig. 3. For $\tilde{G}$, we have: $x_1 = (1, s_1)$, $x_2 = (2, s_2)$, $x_3 = (3, s_2)$, $x_4 = (4, s_2)$, $x_5 = (5, s_2)$, $x_6 = (1, s_3)$, $x_7 = (7, s_F)$.

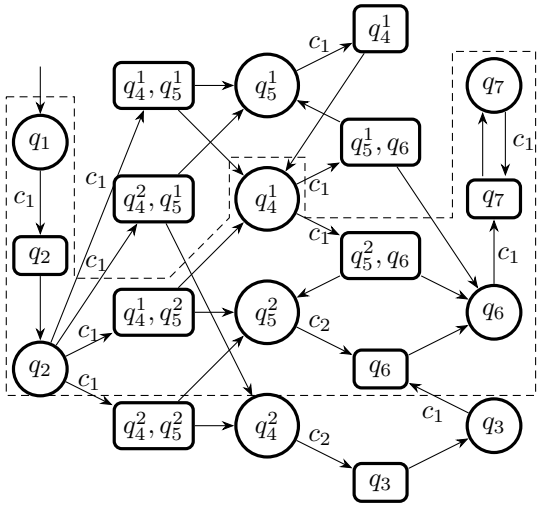


Fig. 4. $AES(G)$. Circle states are Y-states, rectangular states are Z-states. Predictions for each state are $h_1 = (0,0,0,0)$, $h_2 = (0,0,0,0)$, $h_3 = (0,0,0,0)$, $h_4^1 = (0,0,0,0)$, $h_4^2 = (0,0,1,0)$, $h_5^1 = (0,0,0,0)$, $h_5^2 = (0,1,0,0)$, $h_6 = (1,0,0,0)$, $h_7 = (0,0,0,0)$, where h_i denotes the prediction for q_i and h_i^j for q_i^j.

controller may generate three possible paths shown as the colored lines in Fig. 2. As we have discussed previously, all these three paths are 3-step unpredictable.

8. CONCLUSION

In this paper, we formulated and solved a security-aware controller synthesis problem. The synthesized controller can ensure both a given scLTL task and the unpredictability of the satisfaction time of the task. A novel information structure incorporating the effect of control decisions in the future was provided. We show that our synthesis algorithm is both sound and complete to the problem. In this work, we assume that the controller and the intruder have the same observation. In the future, we plan to relax this assumption by considering controllers and intruders with incomparable observation.

REFERENCES

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

Cai, M., Peng, H., Li, Z., and Kan, Z. (2020). Learning-based probabilistic LTL motion planning with environment and motion uncertainties. *IEEE Trans. Automatic Control*, 66(5), 2386–2392.

Kress-Gazit, H., Lahijanian, M., and Raman, V. (2018). Synthesis for robots: Guarantees and feedback for robot behavior. *Annual Review of Control, Robotics, and Autonomous Systems*, 1, 211–236.

Lafortune, S., Lin, F., and Hadjicostis, C.N. (2018). On the history of diagnosability and opacity in discrete event systems. *Annual Reviews in Control*, 45, 257–266.

Liu, S., Trivedi, A., Yin, X., and Zamani, M. (2022). Secure-by-construction synthesis of cyber-physical systems. *Annual Reviews in Control*.

Shi, W., He, Z., Tang, W., Liu, W., and Ma, Z. (2022). Path planning of multi-robot systems with boolean specifications based on simulated annealing. *IEEE Robotics and Automation Letters*, 7(3), 6091–6098.

Smith, S.L., Tůmová, J., Belta, C., and Rus, D. (2011). Optimal path planning for surveillance with temporal-logic constraints. *The Int. J. Robotics Research*, 30(14), 1695–1708.

Tong, Y., Li, Z., Seatzu, C., and Giua, A. (2018). Current-state opacity enforcement in discrete event systems under incomparable observations. *Discrete Event Dynamic Systems*, 28(2), 161–182.

Wang, Y., Nalluri, S., and Pajic, M. (2020). Hyperproperties for robotics: Planning via hyperltl. In *IEEE Int. Conf. Robotics and Automation*, 8462–8468.

Xie, Y., Yin, X., and Li, S. (2022). Opacity enforcing supervisory control using nondeterministic supervisors. *IEEE Transactions on Automatic Control*, 67(12), 6567–6582.

Xie, Y., Yin, X., Li, S., and Zamani, M. (2021). Secure-by-construction controller synthesis for stochastic systems under linear temporal logic specifications. In *IEEE Conf. Decision and Control*, 7015–7021. IEEE.

Yang, S. and Yin, X. (2022). Secure your intention: On notions of pre-opacity in discrete-event systems. *IEEE Trans. Automatic Control*.

Yang, S., Yin, X., Li, S., and Zamani, M. (2020). Secure-by-construction optimal path planning for linear temporal logic tasks. In *IEEE Conf. Decision and Control*, 4460–4466.

Yin, X. and Lafortune, S. (2016). 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 Eng. Practice*, 123, 105130.