Maximally Permissive Supervisory Control of Timed Discrete-Event Systems under Partial Observation ⋆

Ziteng Yang * **Xiang Yin** ** **Shaoyuan Li** **

* *Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China.*
** *Department of Automation, Shanghai Jiao Tong University, Shanghai 200240, China. (E-mail: yinxiang@sjtu.edu.cn)*

Abstract: In this paper, we investigate the supervisory control problem for timed discrete-event systems (TDES) under partial observation. In the timed setting, the system consists of both standard logical events and time event, where the former can be disabled directly by the supervisor if it is controllable while the latter can only be preempted by forcing the occurrences of forcible events. We consider a general control mechanism where the supervisor can choose which events to force dynamically online at each instant. The design objective is to synthesize a maximally-permissive supervisor to restrict the behavior of the system such that the closed-loop language is within a safe specification language. Effective procedure is presented to synthesize such a supervisor. To our knowledge, how to synthesize a maximally-permissive partial-observation supervisor has not been solved for timed DES. We provide a solution to this problem under a general control mechanism.

Keywords: Timed Discrete Event Systems, Supervisory Control, Partial Observation.

1. INTRODUCTION

The supervisory control theory (SCT) of Discrete Event Systems (DES) is a formal framework for the synthesis of correct-by-construction control logic for complex automated systems (Wonham and Cai, 2018). In the SCT, the system is modeled as a DES whose behavior is restricted by a supervisor that disables/enables events dynamically online based on its observation. Since the seminal work of Ramadge and Wonham (1987), the SCT has been developed very extensively in the past 30 years and has been applied to many real-world systems; some recent applications of the SCT include, e.g., MRI scanners (Theunissen et al., 2013), gene regulatory networks (Baldissera et al., 2015), warehouse robots (Tatsumoto et al., 2018) and waterway-lock systems (Goorden et al., 2019).

In the supervisory control theory, one of the central problems is how to synthesize a *maximally-permissive safe* supervisor. Specifically, the safety requirement is modeled as a specification sub-language and the closed-loop language is said to be safe if its behavior is within the specification. Furthermore, we want that the synthesized supervisor works in a *least-restrictive manner*, i.e., it disables events only when it is necessary to do so. The supervisor synthesis problem is particularly challenging in the partial-observation setting as the property of observability is not preserved under union. Many different approaches have been proposed in the literature to handle this problem; see, e.g., Heymann and Lin (1994); Ben Hadj-Alouane et al. (1996); Takai and Ushio (2003); Komenda et al. (2011); Cai et al. (2015). Particularly, in our recent works (Yin and Lafortune, 2016a,b, 2017), a uniform framework was proposed for synthesizing maximally-permissive partial-observation supervisors that are both safe and non-blocking.

In many real-world systems, *timing* information is crucial as the occurrence of an event may be within a designated time bound. Therefore, in Brandin and Wonham (1994), the framework of *timed* discrete events systems (TDES) was proposed to capture this temporal specification. The SCT has also been developed extensively in the timed setting; see, e.g., Lin and Wonham (1995); Takai (2000); Schafaschek et al. (2016); Lin et al. (2018). In the control of TDES, a new event *tick* representing a time unit is introduced in addition to standard logic events; such an event cannot be disabled directly and it can only be *preempted* via some forcible events. More recently, Zhao et al. (2015); Alves et al. (2017); Rashidinejad et al. (2018); Pruekprasert and Ushio (2019) considered how to control a TDES in a networked environment with communication delays and losses. State-based control for TDES was also investigated by Rahnamoon and Wonham (2018). The authors in Zhang et al. (2013); Zhang and Cai (2019) studied the supervisor localization problem for TDES. Also, decentralized control of TDES was investigated by Nomura and Takai (2011); Miura and Takai (2018).

Based on the original framework of Brandin and Wonham, in Takai and Ushio (2006), a generalized framework for the control of TDES was proposed. Specifically, this framework allows the supervisor to choose which events to force dynamically at each instant, where in the original

⋆ This work was supported by the National Natural Science Foundation of China (61803259, 61833012).

framework events that are forced are "static". However, in both the original framework of Brandin and Wonham and the generalized framework of Takai and Ushio, the maximally-permissive supervisor synthesis problem has not yet been solved even for safety specifications only. For example, in Takai and Ushio (2006), the authors provide a normality based solution. A solution based on relative observability in the timed setting was provided in Cai et al. (2016). However, none of the existing solutions is maximally-permissive.

In this paper, we tackle the safe supervisor synthesis problem for TDES under the partial-observation setting. More specifically, we adopt the general framework of Takai and Ushio so that the supervisor can choose forcible events dynamically online. We propose a new information structure that captures all safe control decisions, and based on such a structure, a maximally-permissive supervisor can be synthesized. The proposed approach is motivated by our recent work on supervisory control of untimed DES (Yin and Lafortune, 2016a,b). However, the control mechanisms of the timed setting and the untimed setting are quite different as the former also needs to handle forcible events appropriately to "control" the time event. Therefore, new plant and control operators, as well as their properties, are proposed to handle this issue. To our knowledge, how to synthesize a maximally-permissive safe supervisor in the timed setting was not solved before.

The rest of this paper is organized as follows. Section 2 presents some basic preliminaries and formulates the problem. Section 3 defines the concept of unobservable reach and observable reach. Section 4 proposes the inclusive controller for the timed setting, which is further generalized to the all inclusive controller in Section 5 by taking the safety requirement into account. The supervisor synthesis procedure is also discussed in Section 5. Finally, we conclude the paper in Section 6.

2. PRELIMINARY AND PROBLEM FORMULATION

2.1 System Model

A Timed Discrete Event System (TDES) in the framework of Brandin and Wonham is modeled as a deterministic finite state automaton

$$G = (X, \Sigma, \delta, x_0),$$

where X is the finite set of states, Σ is the finite set of events, $\delta : X \times \Sigma \to X$ is partial transition function, $x_0 \in X$ is the initial state. The event set is further partitioned as $\Sigma := \Sigma_{act} \cup \{tick\}$, where Σ_{act} is the set of usual events in untimed systems and *tick* is a special event representing a "time unit". Let Σ^* be the set of all finite strings over Σ including the empty string ϵ. Then the transition function is also extended to $\delta : X \times \Sigma^* \to X$ in the usual manner; see, e.g., Cassandras and Lafortune (2008). The language generated by G is define by $\mathcal{L}(G) = \{s \in \Sigma^* : \delta(x_0, s)!\}$, where ! means "is defined". We also define $E_{\mathcal{L}(G)}(s) := \{\sigma \in \Sigma : s\sigma \in \mathcal{L}(G)\}$ as the set of events defined upon the occurrence of string s in G. For any language $L \subseteq \Sigma^*$, we define $\overline{L} = \{s \in \Sigma^* : \exists w \in \Sigma^* \text{ s.t. } sw \in L\}$ as the prefix-closure of L; we call L prefix-closed if $\overline{L} = L$. Similarly, for any state $x \in X$, we define $E_G(x) := \{\sigma \in \Sigma : \delta(x, \sigma)!\}$ as the set of events defined at state $x \in X$ in G.

2.2 Supervisory Control of TDES

We assume that event set Σ_{act} is partitioned as

$$\Sigma_{act} = \Sigma_c \dot{\cup} \Sigma_{uc},$$

where Σ_c is the set of controllable events and Σ_{uc} is the set of uncontrollable events. That is, we can disable the occurrences of events in Σ_c. In the timed setting, however, event *tick* cannot be disabled directly. Instead, it can be *preempted* by *forcible* events. We denote by $\Sigma_{for} \subseteq \Sigma_{act}$ the set of all forcible events. The reader is referred to Brandin and Wonham (1994) for the physical meaning of forcible events as well as how a TDES is modeled.

Furthermore, we assume that the system is *partially-observed*. To this end, event set is also partitioned as

$$\Sigma = \Sigma_o \dot{\cup} \Sigma_{uo},$$

where Σ_o is the set of observable events and Σ_{uo} is the set of unobservable events. The natural projection $P : \Sigma^* \to \Sigma_o^*$ is defined recursively by:

$$P(\epsilon) = \epsilon \text{ and } P(s\sigma) = \begin{cases} P(s)\sigma & \text{if } \sigma \in \Sigma_o \\ P(s) & \text{if } \sigma \in \Sigma_{uo} \end{cases}.$$

Natural projection is also extended to $P : 2^{\Sigma^*} \to 2^{\Sigma_o^*}$ by for any $L \subseteq \Sigma^*$, we have $P(L) := \{P(s) : s \in L\}$.

In this paper, we consider a general class of supervisors for TDES proposed by Takai and Ushio (2006). In this setting, a supervisor needs to make the following two decisions at each instant

- what events need to be enabled; and
- what events need to be forced (to preempt *tick*).

Therefore, a supervisor S is defined as a function

$$S : P(\mathcal{L}(G)) \to 2^{\Sigma_{act}} \times 2^{\Sigma_{for}} \tag{1}$$

such that for any $\alpha \in P(\mathcal{L}(G))$, $S(\alpha) = (S_a(\alpha), S_f(\alpha))$ satisfies the following two conditions

- $\Sigma_{uc} \subseteq S_a(\alpha)$; and
- $S_f(\alpha) \subseteq S_a(\alpha) \cap \Sigma_{for}$.

That is, a supervisor should always enable uncontrollable events and can only force forcible events that are enabled. Event set $S(\alpha)$ satisfying the above two conditions is also referred to an *admissible control decision* and we denote by Γ the set of all admissible control decisions.

Then the closed-loop language of TDES G under supervisor S, denoted by $\mathcal{L}(S/G)$, is defined recursively as follows:

- $\epsilon \in \mathcal{L}(S/G)$;
- For any $s \in \mathcal{L}(S/G)$ and $\sigma \in \Sigma$
  - if $\sigma \in \Sigma_{act}$, then
    $$s\sigma \in \mathcal{L}(S/G) \Leftrightarrow \sigma \in E_{\mathcal{L}(G)}(s) \cap S_a(P(s))$$
  - if $\sigma = tick$, then
    $$s\sigma \in \mathcal{L}(S/G) \Leftrightarrow [\sigma \in E_{\mathcal{L}(G)}(s)] \wedge [E_{\mathcal{L}(G)}(s) \cap S_f(P(s)) = \emptyset].$$

The intuition of the above definition is as follows. For any standard event in Σ_{act}, it can happen if it is feasible and is enabled by S. However, for event *tick*, it can happen only when it is feasible and no feasible events are forced, i.e., event *tick* is not preempted.

2.3 Supervisor Synthesis Problem

The goal of the supervisor is to restrict the behavior of the system such that the closed-loop system satisfies some desired property. In this paper, we consider *safety* as the control objective. Formally, we consider a prefix-closed sub-language $K = \overline{K} \subseteq \mathcal{L}(G)$ as the safety specification. Then the maximally-permissive supervisor synthesis problem that we solve in this paper is formulated as follows.

Problem 1. Given a TDES G and a safety specification $K \subseteq \mathcal{L}(G)$, find a partial-observation supervisor $S : P(\mathcal{L}(G)) \to \Gamma$ such that

- S is safe, i.e., $\mathcal{L}(S/G) \subseteq K$; and
- S is maximally-permissive, i.e., for any S' that is safe, we have $\mathcal{L}(S/G) \not\subset \mathcal{L}(S'/G)$.

For the sake of simplicity, we assume that the specification language K is recognized by a strict sub-automaton $H = (X_H, \Sigma, \delta_H, x_0)$ of G, i.e., $\mathcal{L}(H) = K$, such that the following conditions hold:

- $\forall s \in \mathcal{L}(H) : \delta_H(x_0, s) = \delta(x_0, s)$; and
- $\forall s \in \mathcal{L}(G) \setminus \mathcal{L}(H) : \delta(x_0, s) \notin X_H$.

Note that this assumption is without loss of generality as we can always refine the state space of H such that the above conditions hold. Therefore, $X_H \subseteq X$ is essentially the set of *legal states* and a supervisor is safe if and only if $\forall s \in \mathcal{L}(S/G) : \delta(x_0, s) \in X_H$.

Remark 2. In some applications, one wants to synthesize a supervisor that exactly achieves K. This problem is referred to as the *supervisor existence problem* and it has been shown by Takai and Ushio (2006) that controllability together with weak observability provide necessary and sufficient conditions for the supervisor existence problem. However, the supervisor synthesis problem considered here is more challenging. Partial solutions have been provided in Takai and Ushio (2006) and Cai et al. (2016), but none of the solutions is maximally permissive. To our knowledge, how to synthesize a maximally-permissive safe supervisor has not yet been solved in the timed setting.

3. UNOBSERVABLE REACH AND OBSERVABLE REACH

In this section, we first focus on investigating how the information about the system, i.e., state estimate, evolves during the control process in the timed setting.

Specifically, when a supervisor controls a TDES, there are two instants the information about the plant should be updated:

- when a new observable event occurs; and
- when a new control decision is issued.

The first scenario is captured by the *observable reach* defined as follows.

Definition 3. (Observable Reach) Let $\imath \subseteq X$ be a set of states, $\gamma = (\gamma_a, \gamma_f)$ be a control decision that is applied currently and $\sigma \in \Sigma_o$ be a new observable event. Then the observable reach of $\imath$ upon the occurrence of σ under control decision γ, where $\sigma \in \gamma_a \cup \{tick\}$, denoted by $OR_\sigma(\imath \mid \gamma)$, is the set of states that can be reached immediately. Formally,

- If $\sigma \in \Sigma_{act}$, then
$$OR_\sigma(\imath \mid \gamma) := \{\delta(x, \sigma) \in X : x \in \imath\}.$$
- If $\sigma = tick$, then
$OR_\sigma(\imath \mid \gamma) := \{\delta(x, \sigma) \in X : x \in \imath \wedge E_G(x) \cap \gamma_f = \emptyset\}$.

Now, suppose that the state estimate of the system is $\imath \subseteq X$. When the supervisor makes a new control decision γ, the system may reach a set of new states via some unobservable strings; such a set of states is called the *unobservable reach*. However, unlike the untimed case, where the enablement status of an event is fixed within its unobservable reach, the enablement status of event *tick* is dynamic within its unobservable reach in the timed setting as whether or not it is preempted depends on the existence of a feasible forcible event at that point. Therefore, we propose the following new definition of unobservable reach.

Definition 4. (Unobservable Reach) Let $\imath \subseteq X$ be a set of states and $\gamma = (\gamma_a, \gamma_f) \in 2^{\Sigma_{act}} \times 2^{\Sigma_{for}}$ be control decision. Then the unobservable reach of $\imath$ under γ, denoted by $UR_\gamma(\imath)$, is defined recursively as follows:

- $\imath \subseteq UR_\gamma(\imath)$;
- For any $x \in UR_\gamma(\imath)$, $\sigma \in \gamma_a \cap \Sigma_{uo}$ such that $\delta(x, \sigma) = x'$, we have $x' \in UR_\gamma(\imath)$;
- For any $x \in UR_\gamma(\imath)$ such that $E_G(x) \cap \gamma_f = \emptyset$, $\delta(x, tick) = x'$ and $tick \in \Sigma_{uo}$, we have $x' \in UR_\gamma(\imath)$.

Remark 1. In general, event *tick* can be either observable or unobservable depending on whether or not the system has a clock. According to Definitions 3 and 4, if $tick \in \Sigma_o$, then the unobservable reach is the same as the untimed setting, while the observable reach is not. On the other hand, if $tick \in \Sigma_{uo}$, then the observable reach is the same as the untimed setting, while the unobservable reach is not. Our definitions aim to capture both cases in a general manner.

The following results establish some of the properties of the proposed operators. The first lemma shows that the unobservable reach defined indeed yields a reachability closure.

Lemma 5. For any set of states $\imath \subseteq X$ and any control decision $\gamma \in \Gamma$, we have $UR_\gamma(\imath) = UR_\gamma(UR_\gamma(\imath))$.

Proof. By Definition 4,we know $UR_\gamma(\imath) \subseteq UR_\gamma(UR_\gamma(\imath))$. Hence, it suffices to show that $UR_\gamma(UR_\gamma(\imath)) \subseteq UR_\gamma(\imath)$. To this end, we assume that $UR_\gamma(UR_\gamma(\imath)) \not\subseteq UR_\gamma(\imath)$. Note that $UR_\gamma(UR_\gamma(\imath))$ is expended recursively from $UR_\gamma(\imath)$. Therefore, we know that there exist states $x \in X$ and event σ, such that

- $x \in UR_\gamma(\imath) \cap UR_\gamma(UR_\gamma(\imath))$; and
- $\delta(x, \sigma) \notin UR_\gamma(\imath)$; and
- either (i) $\sigma \in \gamma_a \cap \Sigma_{uo}$; or (ii) $[\sigma \in \{tick\} \cap \Sigma_{uo}] \wedge [E_G(x) \cap \gamma_f = \emptyset]$.

That is, $\delta(x, \sigma)$ is the first state that is not in $UR_\gamma(\imath)$. However, by Definition 4, the first and the third condition also imply that $\delta(x, \sigma) \in UR_\gamma(\imath)$, which is a contradiction. Therefore, we must have $UR_\gamma(UR_\gamma(\imath)) \subseteq UR_\gamma(\imath)$. □

Lemma 6. For any state $x \in UR_\gamma(\imath)$, there exists a state $x' \in \imath$ and a sequence of unobservable events

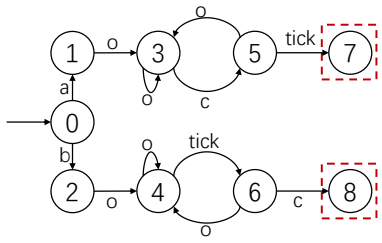

Fig. 1. A TDES G with $\Sigma_o = \{o\}, \Sigma_c = \{c\}$ and $\Sigma_{for} = \{o\}$. States 7 and 8 are illegal states.

$u_1u_2\dots u_m \in \Sigma_{uo}^*$ such that $x = \delta(x', u_1u_2\dots u_m)$ and $\delta(x', u_1u_2\dots u_i) \in UR_\gamma(\imath)$ for any $i \le m$.

Proof. By the definition of $UR_\gamma(\imath)$, we know that

- If $x \in \imath \subseteq UR_\gamma(S)$, then assign $x' = x$ and $u_1u_2, \dots, u_m = \epsilon$, we have $\delta(x', \epsilon) = x \in UR_\gamma(\imath)$
- For any $x \in UR_\gamma(\imath)$ satisfying this property, i.e. $x = \delta(y, u_1u_2\dots u_m)$ where $y \in UR_\gamma(\imath)$, we consider every $\sigma \in E_G(x)$:
  - If $\sigma \in E_G(x) \cap \Sigma_{uo}$, then $x' = \delta(x, \sigma) \in UR_\gamma(\imath)$;
  - If $\sigma \in \{tick\} \cap \Sigma_{uo}$ and $E_G(x) \cap \gamma_a = \emptyset$, then $x' = \delta(x, \sigma) \in UR_\gamma(\imath)$

  Rewrite σ as u_{m+1} for a certain σ, we have $\forall 1 \le i \le m+1$, $\delta(x', u_1u_2\dots u_i) \in UR_\gamma(\imath)$

Since above is the only possible way that $UR_\gamma(\imath)$ was constructed inductively, the lemma was proved. □

The following example illustrates how the observable reach and the unobservable reach are computed.

Example 7. Let us consider TDES G shown in Figure 1. Let $\imath_0 = \{0, 1, 2\}$, i.e., the system is possibly at states 0, 1 or 2, and the control decision is $\gamma_0 = (\Sigma_{uc}, \emptyset)$, i.e., all controllable events are disabled and no event is forced. When new observable event $o \in \Sigma_o$ occurs, we have $\imath_1 = OR_o(\imath_0 \mid \gamma_0) = \{3, 4\}$. From state $\imath_1$, if we pick control decision $\gamma_1 = (\Sigma_{uc} \cup \{c\}, \{o\})$, then we have $UR_{\gamma_1}(\imath_1) = \{3, 4, 5\}$ since event o is feasible and forced at state 4 which preempts event *tick*. On the other hand, if we pick control decision $\gamma_2 = (\Sigma_{uc} \cup \{c\}, \emptyset)$, then we have $UR_{\gamma_2}(\imath_1) = \{3, 4, 5, 6, 7, 8\}$.

4. INCLUSIVE CONTROLLER

In this section, we introduce the notion of *Inclusive Controller* (IC) that combines the observable reach and the unobservable reach in order to describe the entire control process.

4.1 Definition of the Inclusion Controller

Definition 8. (Inclusive Controller) An inclusive controller T w.r.t. G is a 5-tuple

$$T = (Q_T, \Sigma_o, \Gamma, h_T, Q_{0,T}),$$

where

- $Q_T \subseteq 2^X \times \Gamma$ is the set of states in T, where each Q-state q is in the form of $(\imath, \gamma) = (\imath, (\gamma_a, \gamma_f))$;
- Σ_o is the set of observable events in G;
- $\Gamma \subseteq 2^{\Sigma_{act}} \times 2^{\Sigma_{for}}$ is the set of admissible control decisions of G;
- $h_T : Q_T \times \Sigma_o \to 2^{Q_T}$ is the partial *non-deterministic* transition function from a state to a set of states satisfying the following constraints: for any $q_1 = (\imath_1, \gamma_1) = (\imath_1, (\gamma_{1,a}, \gamma_{1,f})), q_2 = (\imath_2, \gamma_2) = (\imath_2, (\gamma_{2,a}, \gamma_{2,f})) \in Q_T$ and $\sigma \in \Sigma_o$ such that $q_2 \in h_T(q_1, \sigma)$, we have
  - $\sigma \in \gamma_{1,a} \cup \{tick\}$
  - $\imath_2 = UR_{\gamma_2}(OR_\sigma(\imath_1 \mid \gamma_1))$
- $Q_{0,T} \subseteq Q_T$ is the set of initial states in T, which satisfies the following constraint: for any $q = (\imath, \gamma) \in Q_{0,T}$, we have $\imath = UR_\gamma(\{x_0\})$.

A state $(\imath, \gamma)$ in T is also referred to as an *information state*, which captures the following two information. The first component of $\imath$ captures the current-state estimate of the system, i.e., all possible states the system can be in currently, and the second component γ captures the current control decision applied. Therefore, a transition $q_2 \in h(q_1, \sigma)$ denotes that a new event σ is observed at state q_1 and a new control decision γ_2, which is the second component of q_2, is issued, which yields a new state estimate $\imath_2 = UR_{\gamma_2}(OR_\sigma(\imath_1 \mid \gamma_1))$ as the first component of q_2. Note that the transition function is non-deterministic in general as the choice of control decision upon the occurrence of an observable event is not unique and we would like to investigate the effects of all possible control decisions in a single structure. For convenience, hereafter, for each state $q \in Q_T$, we denote by $I(q)$ its state estimate component and denote by $C(q)$ its control decision component with $C(q) = (C_a(q), C_f(q))$.

In the definition of the IC, an event σ can occur from state q only when $\sigma \in C_a(q)$ or $\sigma = tick$. Moreover, by the definition the observable reach, such an event should also be feasible from some plant state in $I(q)$. For the purpose of control, a supervisor should be able to react to all such feasible events. This leads to the concept of completeness.

Definition 9. (Completeness) An inclusive controller $T = (Q_T, \Sigma_o, \Gamma, h_T, Q_{0,T})$ is said to be *complete* if, for any $q \in Q_T$ and $\sigma \in \Sigma_o$, we have

- If $\sigma \in \Sigma_{act}$, then
$$h_T(q, \sigma)! \Leftrightarrow \exists x \in I(q) : [\delta(x, \sigma)!] \wedge [\sigma \in C_a(q)].$$
- If $\sigma = tick$, then
$$h_T(q, \sigma)! \Leftrightarrow \exists x \in I(q) : [\delta(x, \sigma)!] \wedge [E_G(x) \cap C_f(q) = \emptyset].$$

As examples, consider G in Figure 1, two complete ICs T_1 and T_2 w.r.t. G are shown in Figure 2.

Definition 10. (Subsystem) Given two inclusive controllers $T_i = (Q_{T_i}, h_{T_i}, \Sigma_o, \Gamma, Q_{0,T_i}), i = 1, 2$, we say T_1 is a subsystem of T_2, denoted by $T_1 \sqsubseteq T_2$, if

- $Q_1 \subseteq Q_2$ and $Q_{0,1} \subseteq Q_{0,2}$; and
- $\forall q \in Q_1, \gamma \in \Gamma, h_{T_1}(q, \gamma) \subseteq h_{T_2}(q, \gamma)$.

Recall that the transition function h_T is non-deterministic in general, i.e., when $h_T(q, \sigma)$ is defined, its successor states may not be unique. Note that, in Definition 8, we just require that any transition in h should satisfy the transition constraints; but not all transitions satisfying the constraints have to be defined. We call an inclusive controller T *total* if it contains all transitions satisfying the constraints. We denote by $\mathcal{T}(G) = (Q, \Sigma_o, \Gamma, h, Q_0)$ the total inclusive controller for G, where we drop the subscript for the sake of simplicity. Clearly, the total

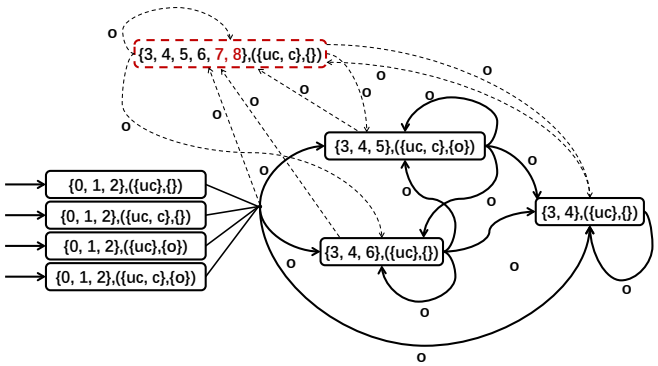

Fig. 2. Inclusive Controllers T_1 (depicted with both solid lines and dashed lines) and T_2 (depict with only solid lines) w.r.t. G_1. For the sake of simplicity, we use uc to denote all uncontrollable events here. Also, the dot on the right hand side of initial states means that all states on the two sides of the dot are pairwise connected.

inclusive controller $\mathcal{T}(G)$ is complete and for any inclusive controller T, we have $T \sqsubseteq \mathcal{T}(G)$. For example, T_1 in Figure 2, it in fact is the total inclusive controller $\mathcal{T}(G)$ for system G in Figure 1.

4.2 Properties of the Inclusive Controller

In this subsection, we show that the inclusive controller and the transition rules defined indeed give the state estimate of the closed-loop control system.

As we discussed earlier, transition function $h : Q \times \Sigma_o \to 2^Q$ is non-deterministic as we may choose different control decisions after the same observation. Therefore, by incorporating the information of the control decision, the transition function becomes deterministic. To this end, we define a *control-couple* as a pair of an observable event and a control decision in the form of $(\sigma, \gamma) \in (\Sigma_o \times \Gamma)$. Then, for an inclusive control T, we define another transition function

$$H_T : Q \times (\Sigma_o \times \Gamma) \to Q$$

by: for any $q, q' \in Q, \sigma \in \Sigma_o$ and $\gamma \in \Gamma$, we have $H_T(q, (\sigma, \gamma)) = q'$ if $q' \in h_T(q, \sigma)$ and $\gamma = C(q')$. Function H_T can also be extended to $H_T : Q \times (\Sigma_o \times \Gamma)^* \to Q$ recursively.

Now, let $S : P(\mathcal{L}(G)) \to \Gamma$ be a supervisor for TDES G and $\alpha = \sigma_1 \dots \sigma_n \in P(\mathcal{L}(S/G))$ be a string of observable events where $\sigma_i \in \Sigma_o$. Then α induces the following sequence of control-couples

$$\xi_{\alpha,S} := (\sigma_1, S(\sigma_1))(\sigma_2, S(\sigma_1\sigma_2)) \dots (\sigma_n, S(\alpha)) \in (\Sigma_o \times \Gamma)^* \quad (2)$$

Note that the above sequence of control-couples does not contain the initial control decision $S(\epsilon)$. To encode this information, we define the initial state under S by

$$q_{0,S} = (UR_{S(\epsilon)}(\{x_0\}), S(\epsilon)). \quad (3)$$

Clearly, $q_{0,S}$ is also an initial state in $\mathcal{T}(G)$. Therefore, starting from $q_{0,S}$ and by executing $\xi_{\alpha,S}$, we reach state $q_{\alpha,S} := H_T(q_{0,S}, \xi_{\alpha,S})$ in the inclusive controller.

Example 11. Recall the total inclusive controller T_1 in Figure 2. Let us consider a supervisor S that works as follows. Initially, S neither enable nor force any events, and then enable event c and force event o after o is observed, i.e., $S(\epsilon) = (\Sigma_{uc}, \emptyset)$ and $S(o) = (\Sigma_{uc} \cup \{c\}, \{o\})$. Let us consider $\alpha = o$. Then we have $q_{0,S} = (\{0, 1, 2\}, (\Sigma_{uc}, \emptyset))$ and $q_{\alpha,S} = H_T(q_{0,S}, (o, S(o))) = (\{3, 4, 5\}, (\Sigma_{uc} \cup \{c\}, \{o\}))$.

The following result reveals that such a state reached is indeed the state estimate of the closed-loop system.

Theorem 12. Let $S : P(\mathcal{L}(G)) \to \Gamma$ be a supervisor for TDES G and $\alpha \in P(\mathcal{L}(S/G))$ be an observable string. Then we have

$$I(q_{\alpha,S}) = \{\delta(x_0, s) \in X : s \in \mathcal{L}(S/G) \wedge P(s) = \alpha\}.$$

Proof. (RHS⊆LHS) It suffices to show that, for any $s \in \mathcal{L}(S/G)$ such that $P(s) = \alpha$, we have $\delta(x_0, s) \in I(q_{\alpha,S})$. We prove by induction on the length of s. For the induction basis, we have $x_0 \in I((UR_{S(\epsilon)}(\{x_0\}), S(\epsilon)))$ immediately. Now, we assume that $\delta(x_0, s) \in I(q_{\alpha,S})$ for $|s| = n$ and we consider string $s\sigma \in \mathcal{L}(S/G)$, where $\sigma \in \Sigma$. We consider the following cases.

Since $s\sigma \in \mathcal{L}(S/G)$, we have

Case 1: $\sigma \in \Sigma_{act}$. For this case, we have $\sigma \in E_{\mathcal{L}(G)}(s) \cap S_a(\alpha)$. This implies that $\sigma \in E_G(\delta(x_0, s)) \cap C_a(q_{\alpha,S})$. If $\sigma \in \Sigma_o$, by Definition 3, we have

$$\delta(x_0, s\sigma) \in OR_\sigma(I(q_{\alpha,S}) \mid C_a(q_{\alpha,S})).$$

Moreover, by Definition 4, we have

$$OR_\sigma(I(q_{\alpha,S}) \mid C_a(q_{\alpha,S})) \subseteq UR_{S(\alpha\sigma)}(OR_\sigma(I(q_{\alpha,S}) \mid C_a(q_{\alpha,S}))).$$

Also, by transition function H_T, we have

$$I(q_{\alpha\sigma,S}) = UR_{S(\alpha\sigma)}(OR_\sigma(I(q_{\alpha,S}) \mid C_a(q_{\alpha,S}))).$$

Therefore, we have $\delta(x_0, s\sigma) \in I(q_{P(s\sigma),S}) = I(q_{\alpha\sigma,S})$. On the other hand, if $\sigma \in \Sigma_{uo}$, then we have $\delta(x_0, s\sigma) = \delta(\delta(x_0, s), \sigma) \in I(q_{P(s\sigma),S}) = I(q_{\alpha,S})$ immediately.

Case 2: $\sigma = tick$. For this case, we have $\sigma \in E_{\mathcal{L}(G)}(s)$ and $E_{\mathcal{L}(G)}(s) \cap S_f(\alpha) = \emptyset$. Which implies that $\sigma \in E_G(\delta(x_0, s))$ and $E_G(\delta(x_0, s)) \cap C_f(q_{\alpha,S}) = \emptyset$. Similar to Case 1, we also have $\delta(x_0, s\sigma) \in I(q_{\alpha,S})$ if $\sigma \in \Sigma_{uo}$ and $\delta(x_0, s\sigma) \in I(q_{\alpha\sigma,S})$ if $\sigma \in \Sigma_o$.

For both cases, we have $\delta(x_0, s\sigma) \in I(q_{P(s\sigma),S})$, which proves this direction.

(LHS⊆RHS) Let us consider an arbitrary state $x \in I(q_{\alpha,S})$. We prove by induction on the length of α that there exists a string $s \in \mathcal{L}(S/G)$ such that $P(s) = \alpha$ and $\delta(x_0, s) = x$.

When $|\alpha| = 0$, we have $I(q_{\epsilon,S}) = UR_{S(\epsilon)}(\{x_0\})$. By Lemma 6, we know that a sequence of unobservable events $u_1u_2 \dots u_m \in \Sigma_{uo}^*$ such that $x = \delta(x_0, u_1u_2 \dots u_m)$ and $x_i := \delta(x_0, u_1u_2 \dots u_i) \in UR_{S(\epsilon)}(\{x_0\})$ for any $i \leq m$. By Definition 4, for each $0 \leq i < m$, we have

- $u_{i+1} \in E_G(x_i) = E_{\mathcal{L}(G)}(u_1 \dots u_i)$; and
- $u_{i+1} \in S_a(\epsilon)$ when $u_{i+1} \in \Sigma_{act}$; and
- $S_f(\epsilon) \cap E_{\mathcal{L}(G)}(u_1 \dots u_i) = \emptyset$ when $u_{i+1} = tick$.

According to the definition of $\mathcal{L}(S/G)$, we know that $u_1u_2 \dots u_m \in \mathcal{L}(S/G)$.

Now, let us assume that for any $x \in I(q_{\alpha,S})$, there exists a string $s \in \mathcal{L}(S/G)$ such that $P(s) = \alpha$ and $\delta(x_0, s) = x$ when $|\alpha| = n$. We consider a state $x \in I(q_{\alpha\sigma,S})$ where

$|\alpha| = n$ and $\sigma \in \Sigma_o$. By Definition 3, Definition 4 and Lemma 6, we know that there exists a state $\hat{x} \in I(q_{\alpha,S})$ and a sequence of unobservable events $u_1 u_2 \dots u_m \in \Sigma_{uo}^*$ such that

- $x = \delta(\hat{x}, \sigma u_1 u_2 \dots u_m)!$;
- $\sigma \in S_a(\alpha)$ when $\sigma \in \Sigma_{act}$; and
- $S_f(\alpha) \cap E_G(\hat{x}) = \emptyset$ when $\sigma = tick$; and
- $x_i := \delta(\hat{x}, \sigma u_1 u_2 \dots u_i) \in UR_{S(\alpha\sigma)}(OR(I(q_{\alpha,S}) \mid S(\alpha)))$ for any $i \leq m$.

By the induction hypothesis, we know that there exists a string $\hat{s}$ such that $\hat{s} \in \mathcal{L}(S/G)$ such that $P(\hat{s}) = \alpha$ and $\delta(x_0, \hat{s}) = \hat{x}$. Therefore, $E_G(\hat{x}) = E_{\mathcal{L}(G)}(\hat{s})$. By Definition 4, similar to the argument in the induction basis for sequence $u_1 \dots u_m$, we have $\hat{s}\sigma u_1 u_2 \dots u_m \in \mathcal{L}(S/G)$. Since $P(\hat{s}\sigma u_1 u_2 \dots u_m) = \alpha\sigma$, we prove the induction step. This completes the proof. □

5. SUPERVISOR SYNTHESIS PROCEDURE

5.1 All Inclusive Controller for Safety

By Theorem 12, we know that $I(q_{\alpha,S})$ essentially captures all possible states reachable in the closed-loop system. Then the following theorem says that, to guarantee safety for a supervisor, it suffices to make sure that it will not reach a state whose first component contains an illegal state.

Theorem 13. Supervisor S is safe if and only if

$$\forall \alpha \in P(\mathcal{L}(S/G)) : I(q_{\alpha,S}) \subseteq X_H.$$

Proof. By Theorem 12, we have $I(q_{\alpha,S}) = \{\delta(x_0, s) \in X : s \in \mathcal{L}(S/G) \wedge P(s) = \alpha\}$. Therefore, if S is safe, i.e. $\mathcal{L}(S/G) \subseteq K$, we have $\forall s \in \mathcal{L}(S/G), s \in K$. Therefore $\forall s \in \mathcal{L}(S/G), x = \delta(x_0, s) \in X_H$ as K is recognized by H, which suggests $I(q_{\alpha,S}) \subseteq X_H$;

On the other hand, if $I(q_{\alpha,S}) \subseteq X_H$, i.e. $\forall s \in \mathcal{L}(S/G) : \delta(x_0, s) \in X_H$. Since H recognizes K, we have $s \in K$, suggesting that $\mathcal{L}(S/G) \subseteq K$. □

The above theorem suggests an approach for synthesizing a safe controller. In order to maintain safety, it suffices to make sure that any information state reached in the inclusive controller is safe. Formally, we say that an inclusive controller $T = (Q_T, h_T, \Sigma_o, \Gamma, Q_{0,T})$ is *safe* if for any $q \in Q_T$, we have $I(q) \subseteq X_H$. Then we define the All Inclusive Controller for Safety (AIC-Safe) as the "largest" safe inclusive controller as follows.

Definition 14. (All Inclusive Controller for Safety) The all inclusive controller for safety is a complete and safe inclusive controller

$$\mathcal{A}(G) = (Q_A, h_A, \Sigma_o, \Gamma, Q_{0,A})$$

such that, for any complete inclusive controller T that is safe, we have $T \sqsubseteq \mathcal{A}(G)$.

The AIC-Safe can be constructed as follows. First, starting from all possible initial-states, we expend the entire state-space in which all states are subsets of X_H. Then, we iteratively remove states that violates the completeness requirement, i.e., a state from which some feasible events are not defined (in order to guarantee safety). Note that removing incomplete state may introduce new incomplete states; hence this step needs to be performed iteratively until the resulting subsystem is complete. Such a construction procedure is the same as the untimed case and the reader is referred to Yin and Lafortune (2016b) for more details. Here, we use an example to illustrate the AIC and how it is constructed.

Example 15. Still, we consider TDES G shown in Figure 1 with $X \setminus X_H = \{7, 8\}$. Then the inclusive controller T_2 in Figure 2 it is in fact the AIC-Safe $\mathcal{A}(G)$ for G. Compared with the total inclusive controller $T_1 = \mathcal{T}(G)$ in the same figure, the dashed-line states and transitions are removed since $7, 8 \notin X_H$. By removing this state, the remaining structure is already complete, which is the AIC-Safe.

5.2 Property of the AIC-Safe

Let T be an inclusive controller, $q \in Q_T$ be a state and $\sigma \in \Sigma_o$ be an observable event. We define

$$C_T(q, \sigma) := \{C(q') \in \Gamma : q' \in h_T(q, \sigma)\}$$

as the set of control decisions that may be issued upon the occurrence of σ from state q. Then we can relate a supervisor and an inclusive controller with the help of the following definition.

Definition 16. Given a complete inclusive controller T, a supervisor S is said to be *included* in T if

- $q_{0,S} \in Q_{0,T}$; and
- for any $\alpha\sigma \in P(\mathcal{L}(S/G))$, where $\alpha \in \Sigma_o^*$ and $\sigma \in \Sigma_o$, we have $S(\alpha\sigma) \in C_T(H_T(q_{0,S}, \xi_{\alpha,S}), \sigma)$.

The set of all supervisors included in T is denoted by $\mathcal{S}(T)$.

The following result shows that the AIC-Safe includes all safe supervisors.

Theorem 17. A supervisor S is safe iff $S \in \mathcal{S}(\mathcal{A}(G))$.

Proof. If S is safe, then for $\alpha \in P(\mathcal{L}(S/G))$, we have $I(q_{\alpha,S}) \subseteq X_H$ by Theorem 13, which means S is included in a complete and safe inclusive controller. Thus $S \in \mathcal{S}(\mathcal{A}(G))$. If $S \in \mathcal{S}(\mathcal{A}(G))$, then S is included in $\mathcal{A}(G)$. Thus for any $\alpha\sigma \in P(\mathcal{L}(S/G))$ where $\alpha\sigma \in \Sigma_o^*$, we have $S(\alpha\sigma) \in C_T(H(q_{0,S}, \xi_{\alpha,S}))$. Since $\mathcal{A}(G)$ is complete and $I(H(q_{0,S}, \xi_{\alpha,S})) \subseteq X_H$, we know that no illegal state can be reached under S, i.e., S is safe according to Theorem 13.

□

5.3 Extract a Supervior from AIC-Safe

By Theorem 17, to synthesize a safe supervisor, it suffices to "extract" a subsystem from $\mathcal{A}(G)$. Specifically, we want to extract a subsystem that enables as many events as possible at each instant. To this end, let $q = (\imath, (\gamma_a, \gamma_f)) \in 2^X \times \Gamma$ be a state in the inclusive controller. We define

$$\textsc{Feas}(q) = \left\{\sigma \in \Sigma : x \in \imath \wedge \delta(x, \sigma)! \wedge \begin{bmatrix} (\sigma \in \Sigma_{act} \wedge \sigma \in \gamma_a) \text{ or} \\ (\sigma = tick \wedge E_G(x) \cap \gamma_f = \emptyset) \end{bmatrix}\right\} \tag{4}$$

as the set of events that are feasible under control decision (γ_a, γ_f) at state $\imath$. Therefore, when comparing the permissiveness of two control decisions, we should not just compare these sets directly since some enabled events may not be feasible. Instead, we should compare the corresponding set of feasible events under each control decision.

Based on the above discuss, we propose the following approach for exacting a subsystem of $\mathcal{A}(G)$.

- Initially, we choose an initial state $q_0 \in Q_{0,A}$ that contains the maximum number of feasible events among all initial states, i.e.,
$$\forall q_0' \in Q_{0,A} : |\textsc{Feas}(q_0')| \leq |\textsc{Feas}(q_0)|.$$
- At each state q reached, upon the occurrence of observable event $\sigma \in \Sigma_o$, we choose a successor state $q' \in h_A(q, \sigma)$ that contains the maximum number of feasible events among all successor states,
$$\forall q'' \in h_A(q, \sigma) : |\textsc{Feas}(q'')| \leq |\textsc{Feas}(q')|.$$
- We repeat the above procedure until all reachable states are visited (either by a depth-first search or a breath-first search) and denote by $T^* \sqsubseteq \mathcal{A}(G)$ the resulting inclusive controller.

Clearly, T^* includes a unique supervisor as the successor state upon the occurrence of each event is unique, and we denote by S^* such a supervisor. Then the following theorem shows that S^* indeed solves Problem 1.

Theorem 18. S^* solves Problem 1, i.e., S^* is a maximally-permissive safe supervisor.

Proof. By Theorem 17, S^* is safe. Next, we show that S^* is also maximally-permissive. We prove by contradiction: suppose S^* is not maximally-permissive, i.e., there exists a safe supervisor S such that $\mathcal{L}(S^*/G) \subset \mathcal{L}(S/G)$. This implies that there exists a string $s \in \mathcal{L}(S/G)$ such that

- $\textsc{Feas}(q_{P(s),S^*}) \subset \textsc{Feas}(q_{P(s),S})$; and
- For any prefix α of $P(s)$ such that $\alpha \neq P(s)$, we have $\textsc{Feas}(q_{\alpha,S}) = \textsc{Feas}(q_{\alpha,S^*})$.

If $P(s) = \epsilon$, then the first condition implies that $|\textsc{Feas}(q_{0,S^*})| < |\textsc{Feas}(q_{0,S})|$, which contradicts to the fact that we choose an initial state with the maximum number of feasible events. Similarly, if $P(s) \neq \epsilon$, then the second condition implies that $q_{\alpha',S} = q_{\alpha',S^*}$, where α' is the longest prefix of $P(s)$ such that $\alpha' \neq P(s)$. Let $P(s) = \alpha'\sigma$. Then we know that $\{q_{P(s),S}, q_{P(s),S^*}\} \subseteq h_A(q_{\alpha',S}, \sigma)$. However, again, the first condition implies a contradiction since $q_{P(s),S^*}$ is chosen such that it has the maximum number of feasible events in $h_A(q_{\alpha',S^*}, \sigma)$, but $|\textsc{Feas}(q_{P(s),S^*})| < |\textsc{Feas}(q_{P(s),S})|$. Therefore, S^* has to be maximally permissive. □

We illustrate how to synthesize a safe and maximally-permissive supervisor by the following example.

Example 19. Again, we consider our running example. To extract a supervisor from $\mathcal{A}(G)$ in Figure 2, we apply the method mentioned above and thus obtain $T_3 \sqsubseteq \mathcal{A}(G)$ shown in Figure 3. Initially, for any $q \in Q_0$, we have $|\textsc{Feas}(q)| = 1$; thus we can choose $(\{0, 1, 2\}, (\Sigma_{uc} \cup \{c\}, \{o\}))$ as the initial state. Then after observing o, we choose $(\{3, 4, 5\}, (\Sigma_{uc} \cup \{c\}, \{o\}))$ as the successor state. Again, if o is observed, we can choose successor state $(\{3, 4, 6\}, (\Sigma_{uc}, \emptyset))$. This results in the inclusive controller T^* shown in Figure 3. Let S^* be the unique supervisor included in T^*. The closed-loop language under control $\mathcal{L}(S^*/G)$ is shown in Figure 4.

Remark 2. Note that solution to Problem 1 is not unique since for each set of successor states $h_A(q, \sigma)$, there may have two different maximal control decisions such that

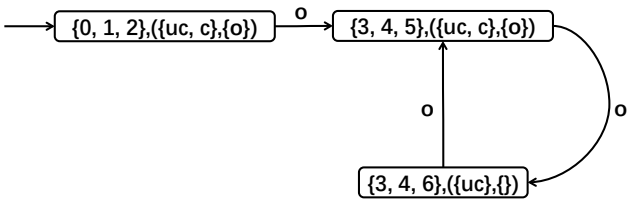

Fig. 3. The IC T_3 where the supervisor extracted by our methods from $\mathcal{A}(G)$ was uniquely included in.

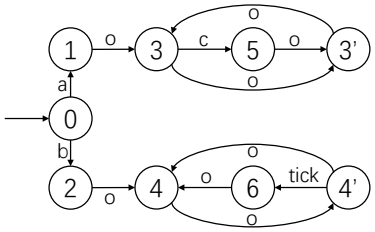

Fig. 4. Automaton that generates the closed-loop language $\mathcal{L}(S^*/G)$.

$|\textsc{Feas}(q)| = |\textsc{Feas}(q')|$ but $\textsc{Feas}(q) \neq \textsc{Feas}(q')$. Our algorithm just randomly choose a successor state with the maximum number of feasible events without any additional criterion.

Remark 3. The complexity of the synthesis procedure is exponential in the size of the system. Specifically, the resulting IC T^*, which is essentially the supervisor realization, contains at most $2^{|X|+|\Sigma_{act}|+|\Sigma_{for}|}$ states and $|\Sigma_o|2^{|X|+|\Sigma_{act}|+|\Sigma_{for}|}$ transitions. However, it is well-known that such an exponential complexity is unavoidable for partially-observed synthesis problem Tsitsiklis (1989).

6. CONCLUSION

In this paper, we solved the problem of synthesizing maximally-permissive safe supervisors for TDES under partial observation. We considered a general setting where the supervisor can choose the set of events to force. We investigate how information evolves in the closed-loop system in the timed setting. A new automaton containing possible safe control decisions called the AIC-safe was defined. We showed how to synthesize a maximally-permissive safe supervisor from the AIC-safe. Our results generalize previous synthesis techniques from the untimed setting to the timed setting. In the future, we plan to investigate the non-blocking control problem for TDES.

REFERENCES

Alves, M., Carvalho, L., and Basilio, J. (2017). Supervisory control of timed networked discrete event systems. In *56th IEEE Conference on Decision and Control*, 4859–4865.

Baldissera, F., Cury, J., and Raisch, J. (2015). A supervisory control theory approach to control gene regulatory networks. *IEEE Transactions on Automatic Control*, 61(1), 18–33.

Ben Hadj-Alouane, N., Lafortune, S., and Lin, F. (1996). Centralized and distributed algorithms for on-line synthesis of maximal control policies under partial observation. *Discrete Event Dyn. Syst.: Theory Appl.*, 6(4), 379–427.

Brandin, B. and Wonham, W. (1994). Supervisory control of timed discrete-event systems. *IEEE Transactions on Automatic Control*, 39(2), 329–342.

Cai, K., Zhang, R., and Wonham, W. (2015). Relative observability of discrete-event systems and its supremal sublanguages. *IEEE Transactions on Automatic Control*, 60(3), 659–670.

Cai, K., Zhang, R., and Wonham, W. (2016). Relative observability and coobservability of timed discrete-event systems. *IEEE Transactions on Automatic Control*, 61(11), 3382–3395.

Cassandras, C. and Lafortune, S. (2008). *Introduction to Discrete Event Systems*. Springer, 2nd edition.

Goorden, M., van de Mortel-Fronczak, J., Reniers, M., Fokkink, W., and Rooda, J. (2019). Structuring multi-level discrete-event systems with dependency structure matrices. *IEEE Transactions on Automatic Control*.

Heymann, M. and Lin, F. (1994). On-line control of partially observed discrete event systems. *Discrete Event Dynamic Systems: Theory & Applications*, 4(3), 221–236.

Komenda, J., Masopust, T., and Van Schuppen, J. (2011). Synthesis of controllable and normal sublanguages for discrete-event systems using a coordinator. *Systems & Control Letters*, 60(7), 492–502.

Lin, F. and Wonham, W. (1995). Supervisory control of timed discrete-event systems under partial observation. *IEEE Transactions on Automatic Control*, 40(3), 558–562.

Lin, L., Wonham, W., and Su, R. (2018). Timed discrete-event systems are synchronous product structures. *arXiv:1807.06725*.

Miura, S. and Takai, S. (2018). Decentralized control of timed discrete event systems with conditional decisions for enforcement of forcible events. In *IEEE Conference on Decision and Control*, 3956–3961.

Nomura, M. and Takai, S. (2011). Decentralized supervisory control of timed discrete event systems. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 94(12), 2802–2809.

Pruekprasert, S. and Ushio, T. (2019). Supervisory control of communicating timed discrete event systems for state avoidance problem. *IEEE Control Systems Letters*, 4(1), 259–264.

Rahnamoon, S. and Wonham, W. (2018). State-based control of timed discrete-event systems. In *IEEE Conference on Decision and Control*, 4833–4838.

Ramadge, P. and Wonham, W. (1987). Supervisory control of a class of discrete event processes. *SIAM Iournal on Control and Optimization*, 25(1), 206–230.

Rashidinejad, A., Reniers, M., and Feng, L. (2018). Supervisory control of timed discrete-event systems subject to communication delays and non-fifo observations. In *14th International Workshop on Discrete Event Systems*, 456–463.

Schafaschek, G., de Queiroz, M.H., and Cury, J.E. (2016). Local modular supervisory control of timed discrete-event systems. *IEEE Transactions on Automatic Control*, 62(2), 934–940.

Takai, S. (2000). Robust supervisory control of a class of timed discrete event systems under partial observation. *Systems & Control Letters*, 39(4), 267–273.

Takai, S. and Ushio, T. (2003). Effective computation of an $L_m(G)$-closed, controllable, and observable sublanguage arising in supervisory control. *Systems & Control Letters*, 49(3), 191–200.

Takai, S. and Ushio, T. (2006). A new class of supervisors for timed discrete event systems under partial observation. *Discrete Event Dynamic Systems*, 16(2), 257–278.

Tatsumoto, Y., Shiraishi, M., Cai, K., and Lin, Z. (2018). Application of online supervisory control of discrete-event systems to multi-robot warehouse automation. *Control Engineering Practice*, 81, 97–104.

Theunissen, R., Petreczky, M., Schiffelers, R., van Beek, D., and Rooda, J. (2013). Application of supervisory control synthesis to a patient support table of a magnetic resonance imaging scanner. *IEEE Transactions on Automation Science and Engineering*, 11(1), 20–32.

Tsitsiklis, J. (1989). On the control of discrete-event dynamical systems. *Mathematics of Control, Signals and Systems*, 2(2), 95–107.

Wonham, W. and Cai, K. (2018). *Supervisory Control of Discrete-Event Systems*. Springer.

Yin, X. and Lafortune, S. (2016a). Synthesis of maximally permissive supervisors for partially observed discrete event systems. *IEEE Transactions on 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 Transactions on Automatic Control*, 61(8), 2140–2154.

Yin, X. and Lafortune, S. (2017). Synthesis of maximally-permissive supervisors for the range control problem. *IEEE Transactions on Automatic Control*, 62(8), 3914–3929.

Zhang, R. and Cai, K. (2019). Supervisor localization of timed discrete-event systems under partial observation. *IEEE Transactions on Automatic Control*.

Zhang, R., Cai, K., Gan, Y., Wang, Z., and Wonham, W. (2013). Supervision localization of timed discrete-event systems. *Automatica*, 49(9), 2786–2794.

Zhao, B., Lin, F., Wang, C., Zhang, X., Polis, M.P., and Wang, L.Y. (2015). Supervisory control of networked timed discrete event systems and its applications to power distribution networks. *IEEE Transactions on Control of Network Systems*, 4(2), 146–158.