# Supervisory Control of Fuzzy Discrete Event Systems under Partial Observation

Feng Lin, *Fellow, IEEE*, and Hao Ying, *Fellow, IEEE*

***Abstract*—Supervisory control of fuzzy discrete event systems under partial observation is investigated in the paper. Without loss of generality, we consider fuzzy discrete event systems with constraints where a fuzzy discrete event system is modeled by a fuzzy automaton. Sequences of events that can be generated by the system are regarded as constraints and are modeled by a crisp automaton. A supervisor is designed to control the fuzzy discrete event system so that the supervised system is prevented from entering a pre-specified set of illegal/unsafe fuzzy states. A necessary and sufficient condition for the existence of a supervisor is obtained. When the condition is satisfied, an online supervisor can be designed. Fuzzy state estimation problem is first solved, as the supervisor is fuzzy-state-estimate-based. A method is developed to estimate fuzzy state iteratively after observation of each new event. We show that the supervisor so developed ensures the safety of the system and is least restrictive among all possible safe supervisors. Potential of the theoretical results for real-world application is illustrated through an example of HIV/AIDS treatment decision-making.**

***Index Terms*—Discrete event systems; fuzzy systems; fuzzy discrete event systems; supervisory control; state estimates; partial observation**

## I. Introduction

Discrete event systems (DES) are systems whose states are discrete and whose dynamics are event driven. Many real systems can be modeled as discrete event systems at some level of abstraction. The development of DES theory started in the 1980's [1], [2]. Since then, researchers have investigated supervisory control of DES [3], [4], diagnosability of DES [5], [6], opacity of DES [7], [8], and other topics in DES.

While discrete event systems are most suitable to describe engineering systems, where states are crisp, they are not suitable to describe biomedical systems , where states are vague. In [9], we propose to combine discrete event systems with fuzzy logic and introduce fuzzy discrete event systems (FDES). While in a DES, the system is in one and only one state at any time (e.g., a computer is either on or off), in an FDES, the system can be in several states with different memberships (e.g., a patient's health is excellent with membership 0.2 and good with membership 0.7 at the same time). We use fuzzy automata to model fuzzy discrete event systems and investigate observability of DES in [9]. Since the publication of [9], papers have been published by other researchers in FDES. Control of FDES is investigated in [10], [11], [12], [13], [14]. Observability of FDES is investigated in [15], [16], [17]. Diagnosability of FDES is investigated in [18], [19], [20]. Theoretical results in FDES have been applied to mobile robotics [21], where controllability and observability are investigated, information service system [22], and HIV/AIDS treatments [23], [24]. A recent survey of FDES can be found in [25].

In this paper, we investigate control of FDES under partial observation, which has never been studied in the literature before. Without loss of generality, we consider fuzzy discrete event systems with constraints (FDESwC) [26]. An FDESwC can be described by one fuzzy automaton and one crisp automaton. While the fuzzy automaton describes the fuzzy discrete event system, the crisp automaton constraints the sequences of events that can be generated by the system. An FDES without constraints is a special cases of FDESwC when the crisp automaton allows all possible sequences of events.

We use a controller called supervisor to control an FDESwC in a way similar to supervisory control of DES. We assume that some events are controllable and some events are observable. A supervisor can only observe observable events. Based on the sequence of observable events observed, the supervisor disables some controllable events. The goal of the supervisor is to ensure that the supervised system (the closed-loop system) never enters a pre-specified set of illegal/unsafe fuzzy states

Since the system is under partial observation, in order to design a supervisor, we first need to know how to estimate the current fuzzy state of the fuzzy automaton and the crisp state of the crisp automaton. This state estimation problem in FDES turns out to be much more difficult to solve than the corresponding state estimation problem in DES. The reason for this difficulty is that while the number of the elements in state space of a discrete event system is finite, the number of the elements in state space of a fuzzy discrete event system is infinite. As a result, in DES, the state estimate defined as the set of states the system may be in can be calculated off-line by constructing an observer (see, for example [4]); but in FDESwC, the state estimate cannot be calculated off-line, because, in the worst case, even if an observer exists, it may have infinitely many states.

Furthermore, to estimate states in DES, the history of past observations is irrelevant and can be forgotten. In other words, for DES, if the current state estimate is known and a new observable event is observed, then the new state estimate can be calculated based on the current state estimate and the new event without the need of knowing the past observations. However, this is not the case for FDESwC. In FDESwC, the new state estimate depends on not only the current state estimate and the new event, but also on the past observations.

This work is supported in part by the National Science Foundation of USA under grant 2146615.

Feng Lin and Hao Ying are with the Department of Electrical and Computer Engineering, Wayne State University, Detroit, MI 48202, USA. Corresponding author: Feng Lin, Tel: +1 313 5773428. E-mail: flin@wayne.edu.

Because of these differences in state estimates of DES and FDESwC, a new method is proposed in this paper to do state estimation for FDESwC. The new method is an online method that calculates state estimate iteratively. After observing a new (observable) event, the state estimate is updated by taking into consideration of the past observations. An algorithm is proposed to implement the new method. The correctness of the algorithm is formally proved.

Given a control specification, which is a set of illegal or unsafe fuzzy states, a supervisor can be designed based on state estimates. The supervisor is an online supervisor that updates its control action (a set of enabled events) after observing an observable event. A necessary and sufficient condition is obtained for a supervisor that achieves the control specification to exist.

If the necessary and sufficient condition is satisfied, an algorithm is proposed to calculate the control action online. Furthermore, we show that the control calculated by the algorithm is least restrictive (or most permissive) in the sense that it disables an event only when it is absolutely necessary. To show possible applications of the theoretical results, we use HIV/AIDS treatment decision making as an illustrative example.

There are significant differences between state estimation and supervisory control of crisp DES and those of fuzzy DES, including but not limited to the following. (1) In crisp DES, the number of illegal/unsafe states are finite, while in fuzzy DES, the number of illegal/unsafe states are infinite. (2) In crisp DES, the cardinality (number of elements) of state estimate is bounded as more events are observed, while in fuzzy DES the cardinality of state estimate is not bounded as more events are observed. (3) In crisp DES, a supervisor can be designed off-line, because the number of states in a supervisor is finite, while in fuzzy DES, a supervisor cannot be designed off-line, because the number of states in a supervisor may be infinite. (4) In crisp DES, state estimation does not require the history of events that occurred in the past, while in fuzzy DES, state estimation does require the history of events that occurred in the past. (5) In crisp DES, supervisory control does not require the history of events that occurred in the past, while in fuzzy DES, supervisory control does require the history of events that occurred in the past. Because of these differences, both state estimation and supervisory control are much more difficult in fuzzy DES than in crisp DES.

The paper is organized as follows. In Section II, fuzzy discrete event systems with constraints and other necessary definitions and notations are reviewed. In Section III, fuzzy state estimates are defined and the method to estimate fuzzy states is developed. In Section IV, supervisory control of fuzzy discrete event systems under partial observation is investigated. An existence condition for a supervisor is obtained. An algorithm is developed to calculate control if the existence condition is satisfied. In Section V, an illustrative example of HIV/AIDS treatment decision making is presented.

## II. Fuzzy discrete event systems with constraints

In [9], fuzzy discrete event systems are introduced and modeled by fuzzy automata of the form

$$\tilde{G} = (\tilde{Q}, \tilde{\Sigma}, \tilde{\delta}, \tilde{\theta}_o). \tag{1}$$

The elements of a fuzzy automaton $\tilde{G}$ are defined as follows. $\tilde{Q} = \{\tilde{q}_1, \tilde{q}_2, ..., \tilde{q}_n\}$ is the set of $n$ individual states. $\tilde{G}$ can be in a set of individual states with different memberships at the same time. The membership of $\tilde{G}$ in $\tilde{q}_i$ is denoted by $e_i$. We call the vector $\tilde{\theta} = [e_1 \quad e_2 \quad ... \quad e_n]$ the fuzzy state (vector). In other words, a fuzzy state is an $n$-dimensional (row) vector representing the memberships of $\tilde{G}$ in individual states. The initial fuzzy state is denoted by $\tilde{\theta}_o$. The system moves from one fuzzy state to another when an event occurs. An event $\tilde{\sigma}$ is represented by an $n \times n$ matrix with elements in the interval $[0, 1]$. The set of events are denoted by $\tilde{\Sigma}$. The movement of the system from one fuzzy state to another after the occurrence of an event is defined by the transition function $\tilde{\delta}$ as follows. If at the current fuzzy state $\tilde{\theta}$, event $\tilde{\sigma}$ occurs, then the next fuzzy state is $\tilde{\delta}(\tilde{\theta}, \tilde{\sigma})$. We define $\tilde{\delta}(\tilde{\theta}, \tilde{\sigma})$ as $\tilde{\delta}(\tilde{\theta}, \tilde{\sigma}) = \tilde{\theta} \circ \tilde{\sigma}$, where $\circ$ denotes a fuzzy reasoning operator. Examples of fuzzy reasoning operators include the max-product and max-min operators. The theoretical results in this paper, including all the theorems, are valid for any fuzzy reasoning operator. The transition function $\tilde{\delta}$ is extended to sequences of events by $\tilde{\delta}(\tilde{\theta}, \tilde{s}\tilde{\sigma}) = \tilde{\delta}(\tilde{\delta}(\tilde{\theta}, \tilde{s}), \tilde{\sigma})$.

Most fuzzy discrete event systems investigated in the literature are systems without constraints. Any sequences of events can occur in an unconstrained system. In practical applications, not all sequences of events can occur in a system. Hence, some constraints must be added to fuzzy discrete event systems. Therefore, we recently propose fuzzy discrete event systems with constraints in [26]. A crisp automaton [1], [2], [4] is used to specify constraints:

$$G = (Q, \Sigma, \delta, q_o, Q_m), \tag{2}$$

where $Q$ is the set of (crisp) states; $\Sigma$ is the set of events corresponding to $\tilde{\Sigma}$; $\delta : Q \times \Sigma \to Q$ is the transition function; $q_o \in Q$ is the initial state; and $Q_m$ is the set of marked states. The transition function $\delta$ is extended to sequences $\delta : Q \times \Sigma^* \to Q$ in the usual way [4], where $\Sigma^*$ is the set of all sequences of events in $\Sigma$, including the empty sequence $\varepsilon$.

The transition function $\delta(q, s)$ is a partial function (not defined for all $q$ and $s$). We use $\delta(q, s)!$ to denote the fact that $\delta(q, s)$ is defined. The language generated by $G$ is defined as

$$L(G) = \{s \in \Sigma^* : \delta(q_o, s)!\}.$$

The language marked by $G$ is defined as

$$L_m(G) = \{s \in L(G) : \delta(q_o, s) \in Q_m\}.$$

If $Q = Q_m$, then $L(G) = L_m(G)$.

A sequence $s \in L(G)$ represents a trajectory of the system. A fuzzy discrete event system with constraints, introduced in [26], is then given by

$$FDESwC = (\tilde{G}, G). \tag{3}$$

$FDESwC$ is constrained because only sequences of events in $L(G)$ can occur in $FDESwC$. Let us illustrate $FDESwC$ by the following example.

*Example 1:* Let us consider the following fuzzy discrete event system with constraints $FDESwC = (\tilde{G}, G)$. For the fuzzy automaton $\tilde{G}$, the state space is $\tilde{Q} = \{\tilde{q}_1, \tilde{q}_2, \tilde{q}_3\}$ with $n = 3$; the events are $\tilde{\Sigma} = \{\tilde{\alpha}, \tilde{\beta}, \tilde{\gamma}, \tilde{\lambda}\}$ with

$$\tilde{\alpha} = \begin{bmatrix} 0.8 & 0.9 & 0.2 \\ 0.9 & 0.6 & 0.5 \\ 0.1 & 0.0 & 0.9 \end{bmatrix}$$
$$\tilde{\beta} = \begin{bmatrix} 0.9 & 0.2 & 0.1 \\ 0.1 & 0.4 & 0.4 \\ 0.9 & 0.8 & 0.9 \end{bmatrix}$$
$$\tilde{\gamma} = \begin{bmatrix} 0.7 & 0.0 & 0.6 \\ 0.9 & 0.8 & 0.7 \\ 0.6 & 0.9 & 0.7 \end{bmatrix}$$
$$\tilde{\lambda} = \begin{bmatrix} 0.4 & 0.0 & 0.2 \\ 0.3 & 0.4 & 0.1 \\ 0.1 & 0.3 & 0.1 \end{bmatrix}.$$

the initial state is $\tilde{\theta}_o = [0.3 \quad 0.7 \quad 0.5]$; and the transition function $\tilde{\delta}(\tilde{\theta}, \tilde{\sigma}) = \tilde{\theta} \circ \tilde{\sigma}$, where $\circ$ is the max-min fuzzy reasoning operator in fuzzy logic theory.

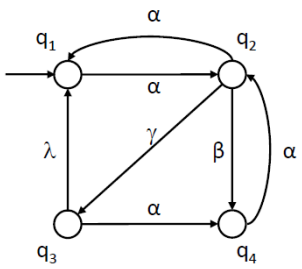


Fig. 1. Automaton $G$ of the system in Example 1. The initial state is $q_1$, denoted by $\rightarrow$.

The constraint automaton $G = (Q, \Sigma, \delta, q_o, Q)$ is shown in Figure 1. The states are $Q = \{q_1, q_2, q_3, q_4\}$. The events are $\Sigma = \{\alpha, \beta, \gamma, \lambda\}$. The initial state is $q_o = q_1$. The transition function is given in Figure 1 by showing the connections between the states.

If the sequence $\alpha\beta$ occurs in the system, the subsequent states in the constraint system $G$ is given by

$$\delta(q_1, \alpha) = q_2$$
$$\delta(q_1, \alpha\beta) = q_4.$$

The corresponding fuzzy states in $\tilde{G}$ can be calculated as follows:

$$\begin{aligned} \tilde{\delta}(\tilde{\theta}_o, \tilde{\alpha}) &= \tilde{\theta}_o \circ \tilde{\alpha} \\ &= [0.3 \quad 0.7 \quad 0.5] \circ \begin{bmatrix} 0.8 & 0.9 & 0.2 \\ 0.9 & 0.6 & 0.5 \\ 0.1 & 0.0 & 0.9 \end{bmatrix} \\ &= [0.7 \quad 0.6 \quad 0.5] \\ \tilde{\delta}(\tilde{\theta}_o, \tilde{\alpha}\tilde{\beta}) &= \tilde{\theta}_o \circ \tilde{\alpha} \circ \tilde{\beta} \\ &= [0.7 \quad 0.6 \quad 0.5] \circ \begin{bmatrix} 0.9 & 0.2 & 0.1 \\ 0.1 & 0.4 & 0.4 \\ 0.9 & 0.8 & 0.9 \end{bmatrix} \\ &= [0.7 \quad 0.5 \quad 0.5]. \end{aligned}$$

■

In the rest of the paper, we investigate fuzzy state estimation and supervisory control of $FDESwC$ under partial observation, which have never been investigated before.

## III. Fuzzy State Estimates under Partial Observation

If all events are observable, then the current fuzzy state is unique and can be calculated iteratively as follows. (1) The initial fuzzy state $\tilde{\theta}_o$ is given and known. (2) If the current fuzzy state is $\tilde{\theta}$, then after the occurrence of a new event $\tilde{\sigma}$, the new fuzzy state is $\tilde{\theta} \circ \tilde{\sigma}$.

If not all events are observable, then the current fuzzy state cannot be uniquely determined. The best we can do is to find the current (fuzzy) state estimate, which is defined as the set of all possible fuzzy states that the system may be in. We develop a method to calculate state estimates iteratively in this section.

Let $\Sigma_o \subseteq \Sigma$ be the set of all observable events and $\Sigma_{uo} = \Sigma - \Sigma_o$ be the set of all unobservable events. We make the following assumption:

(A1) There exists no loop of unobservable events in $G$, that is,

$$(\forall q \in Q)(\forall s \in \Sigma^*)\delta(q, s) = q \Rightarrow s \notin \Sigma_{uo}^* - \{\varepsilon\}. \qquad \text{(A1)}$$

Let $P : \Sigma^* \rightarrow \Sigma_o^*$ be the natural projection defined as

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

So, if a sequence of events $s \in L(G)$ occurs in the system, a supervisor will observe $w = P(s)$. The set of all possible observations is given by the language

$$P(L(G)) = \{w \in \Sigma^* : (\exists s \in L(G))w = P(s)\}.$$

If the current observation is $w \in P(L(G))$, then the set of all possible sequences that may have occurred in the system is given by

$$\rho(w) = P^{-1}(w) \cap L(G), \qquad (4)$$

where the inverse projection $P^{-1} : \Sigma_o^* \rightarrow 2^{\Sigma^*}$ is defined as

$$P^{-1}(w) = \{s \in \Sigma^* : P(s) = w\}.$$

The fuzzy state estimate after observing $w \in P(L(G))$ is formally defined as

$$\tilde{E}(w) = \{\tilde{\theta}_o \circ \tilde{s} : s \in \rho(w)\}. \qquad (5)$$

In order to calculate $\rho(w)$ and $\tilde{E}(w)$ iteratively, for each sequence $s \in \rho(w)$, let us remember its corresponding discrete state $q_s = \delta(q_o, s)$ and fuzzy state $\tilde{\theta}_s = \tilde{\delta}(\tilde{\theta}_o, \tilde{s}) = \tilde{\theta}_o \circ \tilde{s}$ as follows:

$$\pi(s) = (s, q_s, \tilde{\theta}_s) = (s, \delta(q_o, s), \tilde{\theta}_o \circ \tilde{s}).$$

Algorithm 1 calculates $\rho(w)$, $\tilde{E}(w)$, and $\pi(s)$, for all $s \in \rho(w)$ iteratively by first calculating $\rho(w)$, $\tilde{E}(w)$, and $\pi(s)$, for all $s \in \rho(w)$ for $w = \varepsilon$, that is, before any event is observed,

and then updating $\rho(w)$, $\tilde{E}(w)$, and $\pi(s)$, for all $s \in \rho(w)$ whenever a new event is observed.

*Algorithm 1:* State Estimation for FDESwC under Partial Observation

**Input:** $G = (Q, \Sigma, \delta, q_o)$, $\tilde{G} = (\tilde{Q}, \tilde{\Sigma}, \tilde{\delta}, \tilde{\theta}_o)$, and observation $w \in P(L(G))$

**Output:** State estimate $\tilde{E}(w)$;

1: *Initialization*:
2: $w = \varepsilon$;
3: $\rho(\varepsilon) = \{s \in \Sigma^* : s \in \Sigma_{uo}^* \wedge \delta(q_o, s)!\}$;
4: $\tilde{E}(\varepsilon) = \emptyset$;
5: **for** all $s \in \rho(\varepsilon)$ **do**
6: $q_s = \delta(q_o, s)$;
7: $\tilde{\theta}_s = \tilde{\theta}_o \circ \tilde{s}$;
8: $\pi(s) = (s, q_s, \tilde{\theta}_s)$;
9: $\tilde{E}(\varepsilon) = \tilde{E}(\varepsilon) \cup \{\tilde{\theta}_s\}$;
10: **end for**
11: wait for the next observable event $\sigma$ to be observed
12: *Iteration*:
13: $\rho(w\sigma) = \emptyset$;
14: **for** all $s \in \rho(w)$ **do**
15: $\mu(q_s, \sigma) = \{\sigma u \in \Sigma^* : u \in \Sigma_{uo}^* \wedge \delta(q_s, \sigma u)!\}$;
16: $\rho(w\sigma) = \rho(w\sigma) \cup s\mu(q_s, \sigma)$;
17: **end for**
18: $\tilde{E}((w\sigma) = \emptyset$;
19: **for** all $s\sigma u \in \rho(w\sigma)$ **do**
20: $q_{s\sigma u} = \delta(q_s, \sigma u)$;
21: $\tilde{\theta}_{s\sigma u} = \tilde{\theta}_s \circ \tilde{\sigma} \circ \tilde{u}$;
22: $\pi(s\sigma u) = (s\sigma u, q_{s\sigma u}, \tilde{\theta}_{s\sigma u})$;
23: $\tilde{E}(w\sigma) = \tilde{E}(w\sigma) \cup \{\tilde{\theta}_{s\sigma u}\}$;
24: **end for**
25: $w = w\sigma$;
26: Go to Line 11.

The complexity of Algorithm 1 is given in the following proposition, whose proof can be found in Appendix.

*Proposition 1:* The complexity of Algorithm 1 is linear with respect to the length of $w$.

The correctness of Algorithm 1 is given in the following theorem, whose proof can be found in Appendix.

*Theorem 1:* Under Assumption (A1), Algorithm 1 terminates finitely. Furthermore, $\tilde{E}(w)$ calculated in Algorithm 1 is the state estimate after observing $w$, that is, for all $w \in P(L(G))$,

$$\rho(w) = P^{-1}(w) \cap L(G)$$
$$\tilde{E}(w) = \{\tilde{\theta}_o \circ \tilde{s} : s \in \rho(w)\}.$$

Let us illustrate the results by the following example.

*Example 2:* Let us consider the same fuzzy discrete event system with constraints $FDESwC = (\tilde{G}, G)$ as in Example 1. Let the observable events be $\Sigma_o = \{\alpha, \gamma, \lambda\}$. Hence, $\Sigma_{uo} = \{\beta\}$. Let us calculate some state estimates using Algorithm 1.

Initially, $q_o = q_1$, $\tilde{\theta}_o = [0.3 \;\; 0.7 \;\; 0.5]$, and $w = \varepsilon$. We have,

$$\begin{aligned}
\rho(\varepsilon) &= \{u \in \Sigma^* : u \in \Sigma_{uo}^* \wedge \delta(q_o, u)!\} = \{\varepsilon\} \\
\pi(\varepsilon) &= (\varepsilon, q_\varepsilon, \tilde{\theta}_\varepsilon) = (\varepsilon, q_o, \tilde{\theta}_o) \\
&= (\varepsilon, q_1, [0.3 \;\; 0.7 \;\; 0.5]) \\
\tilde{E}(\varepsilon) &= \{\tilde{\theta}_u : u \in \rho(\varepsilon)\} = \{\tilde{\theta}_\varepsilon\} \\
&= \{[0.3 \;\; 0.7 \;\; 0.5]\}.
\end{aligned}$$

If event $\alpha$ is first observed, then $w = \varepsilon$, $\sigma = \alpha$, and $\rho(w) = \rho(\varepsilon) = \{\varepsilon\}$. For $s \in \rho(\varepsilon)$, that is, $s = \varepsilon$, calculate $\mu(q_s, \alpha)$ as

$$\begin{aligned}
\mu(q_s, \alpha) &= \mu(q_\varepsilon, \alpha) \\
&= \{\alpha u \in \Sigma^* : u \in \Sigma_{uo}^* \wedge \delta(q_s, \alpha u)!\} \\
&= \{\alpha, \alpha\beta\}.
\end{aligned}$$

Hence,

$$\begin{aligned}
\rho(\alpha) &= \cup_{s \in \rho(w)} s\mu(q_s, \alpha) \\
&= \mu(q_\varepsilon, \alpha) \\
&= \{\alpha, \alpha\beta\}.
\end{aligned}$$

For $s \in \rho(\alpha)$, calculate $\pi(s)$ as

$$\begin{aligned}
\pi(\alpha) &= (\alpha, \delta(q_o, \alpha), \tilde{\theta}_o \circ \tilde{\alpha}) \\
&= (\alpha, \delta(q_1, \alpha), [0.3 \;\; 0.7 \;\; 0.5] \circ \tilde{\alpha}) \\
&= (\alpha, q_2, [0.7 \;\; 0.6 \;\; 0.5]) \\
\pi(\alpha\beta) &= (\alpha\beta, \delta(q_o, \alpha\beta), \tilde{\theta}_o \circ \tilde{\alpha} \circ \tilde{\beta}) \\
&= (\alpha\beta, \delta(q_1, \alpha\beta), [0.3 \;\; 0.7 \;\; 0.5] \circ \tilde{\alpha} \circ \tilde{\beta}) \\
&= (\alpha\beta, q_4, [0.7 \;\; 0.5 \;\; 0.5]).
\end{aligned}$$

Therefore,

$$\begin{aligned}
\tilde{E}(\alpha) &= \{\tilde{\theta}_s : s \in \rho(\alpha)\} \\
&= \{[0.7 \;\; 0.6 \;\; 0.5], [0.7 \;\; 0.5 \;\; 0.5]\}.
\end{aligned}$$

If another $\alpha$ is observed next, then $w = \alpha$, $\sigma = \alpha$, and $\rho(w) = \rho(\alpha) = \{\alpha, \alpha\beta\}$. Using Algorithm 1, we can find

$$\begin{aligned}
\tilde{E}(\alpha\alpha) &= \{\tilde{\theta}_s : s \in \rho(\alpha\alpha)\} \\
&= \{[0.7 \;\; 0.7 \;\; 0.5]\}.
\end{aligned}$$

■

## IV. Control of Fuzzy Discrete Event Systems under Partial Observation

In this section, we investigate control problem in fuzzy discrete event systems under partial observation. The supervisory control framework [1], [2], [4], [3] is used to ensure that the controlled FDESwC will never enter a set of illegal/unsafe fuzzy states.

The controller used in supervisory control is called supervisor and denoted by $\mathcal{S}$. A supervisor can observe events in the observable event set $\Sigma_o$. Based on its observation, the supervisor can disable some controllable events. Denote the set of all controllable events by $\Sigma_c$. The set of all uncontrollable events is denoted by $\Sigma_{uc} = \Sigma - \Sigma_c$.

When a sequence of events $s \in L(G)$ occurred in the system, the supervisor observes $w = P(s)$. Based on this observation, the supervisor $\mathcal{S}$ decides which events are enabled and which events are disabled. Hence, the supervisor issues a control commend $\mathcal{S}(w)$, which is a subset of events, that is, $\mathcal{S}(w) \subseteq \Sigma$. All events that are not in $\mathcal{S}(w)$ are disabled. Among the events in $\mathcal{S}(w)$, which event will actually occur is determined by the system $FDESwC$ (not by the supervisor $\mathcal{S}$). After a new event occurs, the supervisor may update its

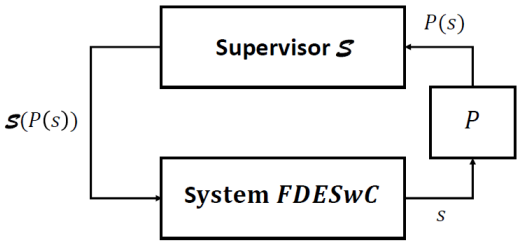


Fig. 2. Architecture of supervisory control. Projection $P$ describes partial observation and $\mathcal{S}(P(s))$ is the control after observing $P(s)$.

control (if the event is observable) or do nothing (if the event is unobservable). This supervisory control architecture is shown in Figure 2.

Therefore, a supervisor $\mathcal{S}$ can be formally defined as a mapping

$$\mathcal{S} : P(L(G)) \to 2^{\Sigma}, \tag{6}$$

where it is required that for all $w \in P(L(G))$, $\Sigma_{uc} \subseteq \mathcal{S}(w)$, that is, all uncontrollable events are always enabled.

The supervised system is denoted by $\mathcal{S}/G$. The language generated by the supervised system, denoted by $L(\mathcal{S}/G)$, is defined recursively as

$$\begin{aligned} &(1)\ \varepsilon \in L(\mathcal{S}/G), \\ &(2)\ (\forall s \in L(\mathcal{S}/G))(\forall \sigma \in \Sigma) s\sigma \in L(\mathcal{S}/G) \\ &\quad \Leftrightarrow (s\sigma \in L(G) \wedge \sigma \in \mathcal{S}(P(s))). \end{aligned} \tag{7}$$

Initially, no event has occurred and hence $\varepsilon \in L(\mathcal{S}/G)$. After the occurrence of sequence $s \in L(\mathcal{S}/G)$, the next event $\sigma$ can occur in the supervised system if and only if $\sigma$ is feasible in $G$ (that is, $s\sigma \in L(G)$), and $\sigma$ is allowed by $\mathcal{S}$ (that is, $\sigma \in \mathcal{S}(P(s))$).

The fuzzy discrete event systems under the supervisory control is denoted by

$$\mathcal{S}/FDESwC = (\tilde{G}, \mathcal{S}/G).$$

In other words, for $\mathcal{S}/FDESwC$, the new constraint is $\mathcal{S}/G$. The control objective is to ensure that the supervised system $\mathcal{S}/FDESwC$ never enters any illegal/unsafe fuzzy states. Denote the set of all illegal/unsafe fuzzy states by $\Theta_b \subseteq [0,1]^n$. Then the control objective can be formally specified as

$$(\forall s \in L(\mathcal{S}/G))\tilde{\delta}(\tilde{\theta}_o, \tilde{s}) \notin \Theta_b. \tag{8}$$

If a supervisor $\mathcal{S}$ satisfies Equation (8), then we say that $\mathcal{S}$ is valid. To design a valid supervisor, we make the following assumption.

(A2) There exists no loop of uncontrollable events in $G$, that is,

$$(\forall q \in Q)(\forall s \in \Sigma^*)\delta(q,s) = q \Rightarrow s \notin \Sigma_{uc}^* - \{\varepsilon\}. \tag{A2}$$

(A3) All controllable events are observable, that is,

$$(\forall \sigma \in \Sigma)\sigma \in \Sigma_c \Rightarrow \sigma \in \Sigma_o. \tag{A3}$$

Under Assumptions (A1), (A2), and (A3), a valid supervisor $\mathcal{S}^\circ$ can be designed as follows. After observing $w \in P(L(G))$, $\rho(w)$ and $\pi(s)$, for all $s \in \rho(w)$, can be calculated using Algorithm 1. We then use Algorithm 2 to calculates control $\mathcal{S}^\circ(w)$ iteratively by first calculating $\mathcal{S}^\circ(w)$ for $w = \varepsilon$, that is, before any event is observed, and then updating $\mathcal{S}^\circ(w)$ whenever a new event is observed.

*Algorithm 2:* Supervisory Control Design for FDESwC under Partial Observation

**Input:** $G = (Q, \Sigma, \delta, q_o)$, $\tilde{G} = (\tilde{Q}, \tilde{\Sigma}, \tilde{\delta}, \tilde{\theta}_o)$, $\Theta_b$, $\Sigma_{uc}$, $w \in P(L(G))$, $\rho(w)$ and $\pi(s) = (s, q_s, \tilde{\theta}_s)$, for all $s \in \rho(w)$

**Output:** $\mathcal{S}^\circ(w)$;

1: $\mathcal{S}^\circ(w) = \Sigma_{uc}$;
2: **for** all $\sigma \in \Sigma_c$ **do**
3:  **for** all $s \in \rho(w)$ **do**
4:  $\eta(q_s, \sigma) = \{\sigma u \in \Sigma^* : u \in \Sigma_{uc}^* \wedge \delta(q_s, \sigma u)!\}$ ;
5:  **end for**
6:  **if** $(\forall s \in \rho(w))(\forall \sigma u \in \eta(q_s, \sigma))\tilde{\delta}(\tilde{\theta}_s, \tilde{\sigma}\tilde{u}) \notin \Theta_b$ **then**
7:  $\mathcal{S}^\circ(w) = \mathcal{S}^\circ(w) \cup \{\sigma\}$;
8:  **end if**
9: **end for**
10: End.

The complexity of Algorithm 2 is given in the following proposition, whose proof can be found in Appendix.

*Proposition 2:* The complexity of Algorithm 2 is linear with respect to the length of $w$.

The correctness of Algorithm 2 is given in the following theorem, whose proof can be found in Appendix.

*Theorem 2:* Under Assumptions (A1), (A2), and (A3), Algorithm 2 terminates finitely. Furthermore, a valid supervisor exists if and only if

$$(\forall u \in \Sigma_{uc}^* \cap L(G))\tilde{\delta}(\tilde{\theta}_o, \tilde{u}) \notin \Theta_b.$$

If the above condition is satisfied, then supervisor $\mathcal{S}^\circ$ obtained in Algorithm 2 is a valid supervisor, that is,

$$(\forall s \in L(\mathcal{S}^\circ/G))\tilde{\delta}(\tilde{\theta}_o, \tilde{s}) \notin \Theta_b.$$

Let us now investigate how to design a valid supervisor that is least restrictive (or most permissive). Formally, we say that supervisor $\mathcal{S}$ is less restrictive than supervisor $\mathcal{S}'$ [4], [3], denoted by $\mathcal{S}' \leq \mathcal{S}$, if

$$(\forall w \in P(L(G)))\mathcal{S}'(w) \subseteq \mathcal{S}(w).$$

The proof of the following proposition can be found in Appendix.

*Proposition 3:* If supervisor $\mathcal{S}$ is less restrictive than supervisor $\mathcal{S}'$, that is, $\mathcal{S}' \leq \mathcal{S}$, then the language generated by $\mathcal{S}'$ is equal to or contained in the language generated by $\mathcal{S}$, that is,

$$L(\mathcal{S}'/G) \subseteq L(\mathcal{S}/G).$$

Proposition 3 implies that if a supervisor $\mathcal{S}$ is valid, then a more restrictive supervisor $\mathcal{S}' \leq \mathcal{S}$ is also valid. We want to find the least restrictive valid supervisor if possible. To this end, denote the set of all valid supervisors with respect to $\Theta_b$ as

$$VS(\Theta_b) = \{\mathcal{S} : (\forall s \in L(\mathcal{S}/G))\tilde{\delta}(\tilde{\theta}_o, \tilde{s}) \notin \Theta_b\}.$$

We say that a valid supervisor $\mathcal{S}^\diamond \in VS(\Theta_b)$ is least restrictive if

$$(\forall \mathcal{S} \in VS(\Theta_b))L(\mathcal{S}/G) \subseteq L(\mathcal{S}^\diamond/G).$$

We can now have the following theorem, whose proof can be found in Appendix.

*Theorem 3:* Under Assumptions (A1), (A2), and (A3), if $(\forall u \in \Sigma_{uc}^* \cap L(G))\tilde{\delta}(\tilde{\theta}_o, \tilde{u}) \notin \Theta_b$, then supervisor $\mathcal{S}^\circ$ obtained in Algorithm 2 is the least restrictive valid supervisor.

Note that Assumptions (A2) and (A3) together imply Assumption (A1).

Let us illustrate the results of this section using the following example.

*Example 3:* Let us consider again the same $FDESwC = (\tilde{G}, G)$ as in Examples 1 and 2. Let the controllable events be $\Sigma_c = \{\alpha, \gamma\}$. Hence, $\Sigma_{uc} = \{\beta, \lambda\}$. Define the unsafe/illegal (fuzzy) states as

$$\Theta_b = \{\tilde{\theta} = [e_1 \quad e_2 \quad e_3] : e_i < 0.45, i = 1, 2, 3\}.$$

We design supervisor online using Algorithm 2 as follows.

Initially, $w = \varepsilon$ and $\rho(w) = \{\varepsilon\}$. Let us determine if $\sigma = \alpha \in \Sigma_c$ shall be enabled or disabled. For $s = \varepsilon \in \rho(w)$ (the only string in $\rho(w)$), $q_s = q_1$ and $\tilde{\theta}_s = [0.3 \quad 0.7 \quad 0.5]$. Using Algorithm 2,

$$\begin{aligned}
\eta(q_s, \alpha) &= \{\alpha u \in \Sigma^* : u \in \Sigma_{uc}^* \wedge \delta(q_s, \alpha u)!\} \\
&= \{\alpha, \alpha\beta\} \\
\tilde{\delta}(\tilde{\theta}_s, \tilde{\alpha}) &= [0.7 \quad 0.6 \quad 0.5] \notin \Theta_b \\
\tilde{\delta}(\tilde{\theta}_s, \tilde{\alpha}\tilde{\beta}) &= [0.7 \quad 0.7 \quad 0.5] \notin \Theta_b.
\end{aligned}$$

Since $(\forall s \in \rho(w))(\forall \alpha u \in \eta(q_s, \alpha))\tilde{\delta}(\tilde{\theta}_s, \tilde{\alpha}\tilde{u}) \notin \Theta_b$ is true, $\alpha$ is enabled.

Similarly, we can determine $\gamma$ shall be enabled because $\eta(q_s, \gamma) = \emptyset$ and hence $(\forall s \in \rho(w))(\forall \gamma u \in \eta(q_s, \gamma))$ $\tilde{\delta}(\tilde{\theta}_s, \tilde{\gamma}\tilde{u}) \notin \Theta_b$ is true. Note that since $\gamma$ is not defined in $q_1$, its enablement or disablement will not make any difference.

Since $\alpha$ is the only event that can occur in $\mathcal{S}^\circ/G$ initially, it will occur and be observed. Then $w = \alpha$ and $\rho(w) = \{\alpha, \alpha\beta\}$.

For $s = \alpha \in \rho(w)$, $q_s = q_2$ and $\tilde{\theta}_s = [0.7 \quad 0.6 \quad 0.5]$. Using Algorithm 2,

$$\begin{aligned}
\eta(q_s, \alpha) &= \{\alpha\} \\
\tilde{\delta}(\tilde{\theta}_s, \tilde{\alpha}) &= [0.7 \quad 0.7 \quad 0.5] \notin \Theta_b \\
\eta(q_s, \gamma) &= \{\gamma, \gamma\lambda\} \\
\tilde{\delta}(\tilde{\theta}_s, \tilde{\gamma}) &= [0.7 \quad 0.6 \quad 0.6] \notin \Theta_b \\
\tilde{\delta}(\tilde{\theta}_s, \tilde{\gamma}\tilde{\lambda}) &= [0.4 \quad 0.4 \quad 0.2] \in \Theta_b.
\end{aligned}$$

For $s = \alpha\beta \in \rho(w)$, $q_s = q_4$ and $\tilde{\theta}_s = [0.7 \quad 0.5 \quad 0.5]$. Using Algorithm 2,

$$\begin{aligned}
\eta(q_s, \alpha) &= \{\alpha\} \\
\tilde{\delta}(\tilde{\theta}_s, \tilde{\alpha}) &= [0.7 \quad 0.7 \quad 0.5] \notin \Theta_b \\
\eta(q_s, \gamma) &= \emptyset.
\end{aligned}$$

Since $(\forall s \in \rho(w))(\forall \alpha u \in \eta(q_s, \alpha))\tilde{\delta}(\tilde{\theta}_s, \tilde{\alpha}\tilde{u}) \notin \Theta_b$ is true, $\alpha$ is enabled.

Since $(\forall s \in \rho(w))(\forall \gamma u \in \eta(q_s, \gamma))\tilde{\delta}(\tilde{\theta}_s, \tilde{\gamma}\tilde{u}) \notin \Theta_b$ is false, $\gamma$ is disabled.

■

## V. Illustrative Example of HIV/AIDS Treatment Decision Making

In this section, we present an illustrative example from HIV/AIDS treatment decision making [27], [28], [29] to demonstrate the theoretical results obtained in the previous sections. The background on using fuzzy discrete event systems for HIV/AIDS treatment decision making can be found in [23], [24], [30]. While most medicine-related aspects of this example reflect a real-world situation (e.g., patient states and drugs used), it is still only an illustrative example because the event matrices used in this section are not obtained from patient data. Indeed, finding event matrices from patient data is a big job, involving collecting patient data and deriving the event matrices, probably using some self-learning methods such as those discussed in [31], [32], [33], [34], [35]. This is beyond the scope of this paper.

For HIV/AIDS treatment decision making, the fuzzy automaton $\tilde{G} = (\tilde{Q}, \tilde{\Sigma}, \tilde{\delta}, \tilde{\theta}_o)$ is given as follows. The individual states are

$$\tilde{Q} = \{\tilde{q}_1, \tilde{q}_2, \tilde{q}_3\},$$

where

$\tilde{q}_1$ denotes "CD4 cell counts is low"
$\tilde{q}_2$ denotes "HIV RNA level is high"
$\tilde{q}_3$ denotes "side effects are low"

and "low" and "high" are characterized by fuzzy sets.

Initially, an HIV/AIDS patient's CD4 cell counts is somewhat low (say, less than 200), HIV RNA level is somewhat high (say, more than 100,000), and there are no side effects of drug (drug is yet to be taken). Hence, the initial fuzzy state is

$$\tilde{\theta}_o = [1 \quad 1 \quad 1].$$

The goal of HIV/AIDS treatment is to increase CD4 cell counts (to raise body's defense against the virus) and decrease HIV RNA level (to kill the virus) with as little side effects as possible. Thus, the target fuzzy state is

$$\tilde{\theta}_{target} = [0 \quad 0 \quad 1].$$

The treatment by a doctor consists of selecting one of the following four oral medication regimens [30]:

Regimen 1: efavirenz + zidovudine/lamivudine
Regimen 2: nelfinavir + zidovudine/lamivudine
Regimen 3: nevirapine + zidovudine/lamivudine
Regimen 4: abacavir/zidovudine/lamivudine

A patient receiving the drugs prescribed by the doctor may fully adhere to the treatment (i.e., taking the drugs as instructed by the doctor). For various reasons (e.g., homeless, side effects of the drugs), some patients may fail to take the drugs sometimes or all the times, which is clinically referred as the adherence problem. In HIV/AIDS treatment, adherence is very important. If the patient adheres to the regimen, then his/her CD4 cell counts will increase and HIV RNA level will decrease. If the patient does not adhere to the regimen, then not only his/her CD4 cell counts and HIV RNA level will not reach the desired levels, but also the virus will develop resistance to the regimen. This will lead to the regimen to become ineffective. When that happens, the doctor may prescribe a different regimen. Since the number of regimens are limited

(4 in this example), if the patient does not adhere to the prescribed regimen, he/she will eventually run out of treatment options and suffers declined health.

To model the treatment, we define the following events:

$$\tilde{\Sigma} = \{\tilde{\alpha}_i, \tilde{\beta}_i, \tilde{\gamma}_i : i = 1, 2, 3, 4\},$$

where

$\tilde{\alpha}_i$ denotes "Regimen $i$ is prescribed by the doctor"

$\tilde{\beta}_i$ denotes "the patient adheres to Regimen $i$, leading to higher regiment potency"

$\tilde{\gamma}_i$ denotes "the patient does not adhere to Regimen $i$, leading to lower regiment potency"

Let us now define fuzzy event matrices as follows. Since $\tilde{\alpha}_i$ denotes "Regimen $i$ is prescribed by the doctor" and the patient has not taken the regimen yet, the event matrices of $\tilde{\alpha}_i$ are all $3 \times 3$ identity matrix, that is,

$$\tilde{\alpha}_1 = \tilde{\alpha}_2 = \tilde{\alpha}_3 = \tilde{\alpha}_4 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Among the four regimens, Regimen 1 is most potent and Regimen 4 is least potent. On the other hand, Regimen 4 has the least side effects and Regimen 2 has the most side effects [30]. Accordingly, the event matrices of $\tilde{\beta}_i$ are given as follows:

$$\tilde{\beta}_1 = \begin{bmatrix} 0.1 & 0.1 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.8 \end{bmatrix} \quad \tilde{\beta}_2 = \begin{bmatrix} 0.15 & 0.1 & 0 \\ 0.1 & 0.15 & 0 \\ 0 & 0 & 0.7 \end{bmatrix}$$
$$\tilde{\beta}_3 = \begin{bmatrix} 0.15 & 0 & 0 \\ 0.1 & 0.15 & 0 \\ 0 & 0 & 0.8 \end{bmatrix} \quad \tilde{\beta}_4 = \begin{bmatrix} 0.2 & 0.1 & 0 \\ 0.1 & 0.2 & 0 \\ 0 & 0 & 0.9 \end{bmatrix}.$$

If the patient does not adhere to the regimen, then the potency of the regimen will be significantly reduced. Accordingly, the event matrices of $\tilde{\gamma}_i$ are given as follows:

$$\tilde{\gamma}_1 = \begin{bmatrix} 0.5 & 0.1 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.8 \end{bmatrix} \quad \tilde{\gamma}_2 = \begin{bmatrix} 0.6 & 0.1 & 0 \\ 0.1 & 0.6 & 0 \\ 0 & 0 & 0.7 \end{bmatrix}$$
$$\tilde{\gamma}_3 = \begin{bmatrix} 0.6 & 0 & 0 \\ 0.1 & 0.6 & 0 \\ 0 & 0 & 0.8 \end{bmatrix} \quad \tilde{\gamma}_4 = \begin{bmatrix} 0.7 & 0.1 & 0 \\ 0.1 & 0.7 & 0 \\ 0 & 0 & 0.9 \end{bmatrix}.$$

Let us now consider the crisp automaton $G = (Q, \Sigma, \delta, q_o, Q)$ describing the constrains. Clearly, the patient can take a regimen only after it is prescribed by the doctor. Hence, constraint automaton $G_i$ for Regimen $i$ is as shown in Fig. 3.

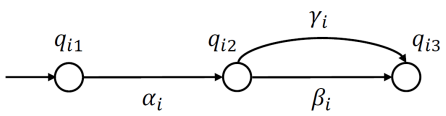


Fig. 3. Automaton $G_i$ for Regimen $i$.

Since a treatment sequence consists of using four different regimens one by one, there are $4! = 24$ possible treatment sequences. For example, the treatment sequence of using Regimen 1, then Regimen 2, Regimen 3, and finally Regimen 4 is shown in Fig. 4, which is a part of the constraint automaton. The entire constraint automaton $G$ contains all the 24 treatment sequences and is too big to be shown here.

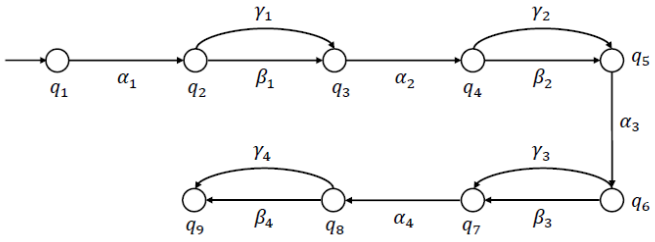


Fig. 4. Part of automaton $G$ for the treatment sequence of using Regimen 1, Regimen 2, Regimen 3, and Regimen 4 .

In $G = (Q, \Sigma, \delta, q_o, Q)$, the corresponding event set is denoted by $\Sigma = \{\alpha_i, \beta_i, \gamma_i : i = 1, 2, 3, 4\}$. Clearly, events $\beta_i$ and $\gamma_i$ are uncontrollable and unobservable (from doctor's point of view), that is, $\Sigma_o = \Sigma_c = \{\alpha_i : i = 1, 2, 3, 4\}$. It can be checked that Assumptions (A1), (A2), and (A3) are all satisfied.

Let us now design a supervisor $\mathcal{S}$ such that the supervised system $\mathcal{S}/FDESwC$ will never enter illegal/unsafe fuzzy states $\Theta_b$. Depending on patient's condition, $\Theta_b$ may be different. Let us consider two cases.

*Case 1*: If a patient condition is very bad, then it is important to increase patient's CD4 cell counts and decrease HIV RNA level. In this case, side effects are of a lesser concern. Hence, the illegal/unsafe fuzzy states are given by

$$\Theta_b = \{\tilde{\theta} = [\tilde{\theta}_1 \;\; \tilde{\theta}_2 \;\; \tilde{\theta}_3] : \tilde{\theta} \neq \tilde{\theta}_o \wedge \tilde{\theta}_1 > 0.65 \wedge \tilde{\theta}_2 > 0.65\}.$$

Before the start of treatment, the supervisor observes nothing: $w = \varepsilon$. Thus,

$$\rho(w) = P^{-1}(w) \cap L(G) = \{\varepsilon\}$$
$$\pi(\varepsilon) = (\varepsilon, q_\varepsilon, \tilde{\theta}_\varepsilon) = (\varepsilon, q_o, \tilde{\theta}_o).$$

Let us calculate $\mathcal{S}^\circ(\varepsilon)$ using Algorithm 2 as follows. Initially,

$$\mathcal{S}^\circ(\varepsilon) = \Sigma_{uc} = \{\beta_i, \gamma_i, : i = 1, 2, 3, 4\}.$$

For $\sigma = \alpha_1 \in \Sigma_c$,

$$\eta(q_\varepsilon, \alpha_1) = \{\alpha_1, \alpha_1\beta_1, \alpha_1\gamma_1\}$$
$$\tilde{\delta}(\tilde{\theta}_\varepsilon, \alpha_1) = [1 \;\; 1 \;\; 1]$$
$$\tilde{\delta}(\tilde{\theta}_\varepsilon, \alpha_1\beta_1) = [0.1 \;\; 0.1 \;\; 0.8]$$
$$\tilde{\delta}(\tilde{\theta}_\varepsilon, \alpha_1\gamma_1) = [0.5 \;\; 0.5 \;\; 0.8].$$

Since $(\forall s \in \rho(\varepsilon))(\forall \sigma u \in \eta(q_s, \sigma))\tilde{\delta}(\tilde{\theta}_s, \tilde{\sigma}\tilde{u}) \notin \Theta_b$ is true,

$$\mathcal{S}^\circ(\varepsilon) = \mathcal{S}^\circ(\varepsilon) \cup \{\alpha_1\}.$$

In other words, prescribing Regimen 1 is allowed (enabled) by the supervisor.

Similarly, for $\sigma = \alpha_2$ and $\sigma = \alpha_3$, we have $\alpha_2 \in \mathcal{S}^\circ(\varepsilon)$ and $\alpha_3 \in \mathcal{S}^\circ(\varepsilon)$. Hence, prescribing Regimens 2 and 3 are also allowed by the supervisor.

On the other hand, for $\alpha_4$, we have

$$\begin{aligned}\eta(q_\varepsilon, \alpha_4) &= \{\alpha_4, \alpha_4\beta_4, \alpha_4\gamma_4\}\\ \tilde{\delta}(\tilde{\theta}_\varepsilon, \alpha_4) &= [1 \quad 1 \quad 1]\\ \tilde{\delta}(\tilde{\theta}_\varepsilon, \alpha_4\beta_4) &= [0.2 \quad 0.2 \quad 0.7]\\ \tilde{\delta}(\tilde{\theta}_\varepsilon, \alpha_4\gamma_4) &= [0.7 \quad 0.7 \quad 0.9].\end{aligned}$$

Since $(\forall s \in \rho(\varepsilon))(\forall \sigma u \in \eta(q_s, \sigma))\tilde{\delta}(\tilde{\theta}_s, \tilde{\sigma}\tilde{u}) \notin \Theta_b$ is not true, $\alpha_4 \notin \mathcal{S}^\circ(\varepsilon)$. In other words, for patients whose conditions are very bad, Regimen 4 shall not be used. Hence,

$$\mathcal{S}^\circ(\varepsilon) = \{\alpha_1, \alpha_2, \alpha_3\}.$$

Now, suppose that Regimen 1 is prescribed by the doctor, that is, $w = \alpha_1$. Then,

$$\begin{aligned}\rho(\alpha_1) &= P^{-1}(\alpha_1) \cap L(G) = \{\alpha_1, \alpha_1\beta_1, \alpha_1\gamma_1\}\\ \pi(\alpha_1) &= (\alpha_1, q_{\alpha_1}, [1 \quad 1 \quad 1])\\ \pi(\alpha_1\beta_1) &= (\alpha_1\beta_1, q_{\alpha_1\beta_1}, [0.1 \quad 0.1 \quad 0.8])\\ \pi(\alpha_1\gamma_1) &= (\alpha_1\gamma_1, q_{\alpha_1\gamma_1}, [0.5 \quad 0.5 \quad 0.8]).\end{aligned}$$

Let us calculate $\mathcal{S}^\circ(\alpha_1)$ using Algorithm 2 as follows. Initially,

$$\mathcal{S}^\circ(\alpha_1) = \Sigma_{uc} = \{\beta_i, \gamma_i, : i = 1, 2, 3, 4\}.$$

Because of the constraint automaton $G$, $\alpha_1$ is no longer possible ($\eta(q_s, \alpha_1) = \emptyset$). For $\sigma = \alpha_2 \in \Sigma_c$ and $s = \alpha_1\beta_1$, we have

$$\begin{aligned}\eta(q_{\alpha_1\beta_1}, \alpha_2) &= \{\alpha_2, \alpha_2\beta_2, \alpha_2\gamma_2\}\\ \tilde{\delta}(\tilde{\theta}_{\alpha_1\beta_1}, \alpha_2) &= [0.1 \quad 0.1 \quad 0.8]\\ \tilde{\delta}(\tilde{\theta}_{\alpha_1\beta_1}, \alpha_2\beta_2) &= [0.1 \quad 0.1 \quad 0.7]\\ \tilde{\delta}(\tilde{\theta}_{\alpha_1\beta_1}, \alpha_2\gamma_2) &= [0.1 \quad 0.1 \quad 0.7].\end{aligned}$$

Since $(\forall s \in \rho(\varepsilon))(\forall \sigma u \in \eta(q_s, \sigma))\tilde{\delta}(\tilde{\theta}_s, \tilde{\sigma}\tilde{u}) \notin \Theta_b$ is true,

$$\mathcal{S}^\circ(\varepsilon) = \mathcal{S}^\circ(\varepsilon) \cup \{\alpha_2\}.$$

In other words, prescribing Regimen 2 is allowed by the supervisor. Similarly, prescribing Regimens 3 and 4 are also allowed by the supervisor. Therefore,

$$\mathcal{S}^\circ(\alpha_1) = \{\alpha_2, \alpha_3, \alpha_4\}.$$

This process will continue until all the four regimens are used.

*Case 2*: If a patient can hardly tolerate side effects (say, because of another illness), then it is important to keep the side effects of a regimen to be prescribed low. In this case, the illegal/unsafe fuzzy states are given by

$$\Theta_b = \{\tilde{\theta} = [\tilde{\theta}_1 \quad \tilde{\theta}_2 \quad \tilde{\theta}_3] : \tilde{\theta} \neq \tilde{\theta}_o \wedge \tilde{\theta}_3 > 0.75\}.$$

Using Algorithm 2, the control is calculated as follows. Initially

$$\mathcal{S}^\circ(\varepsilon) = \{\alpha_1, \alpha_3, \alpha_4\}.$$

In other words, Regimen 2 shall not be prescribed. If Regimen 3 is prescribed by the doctor, then the next control is calculated as

$$\mathcal{S}^\circ(\alpha_3) = \{\alpha_1, \alpha_2, \alpha_4\}.$$

## VI. Conclusions

We investigate supervisory control of fuzzy discrete event systems with constraints under partial observation in this paper, which is a much more difficult problem than the problem of supervisory control under full observation. The main contributions of the paper are summarized as follows. (1) We propose supervisory control of fuzzy discrete event systems by using a supervisor to control the constraint automaton so that the supervised system will never enter illegal or unsafe fuzzy states. (2) An algorithm is developed to estimate fuzzy states iteratively online after each and every observation of observable event. (3) A necessary and sufficient condition is derived for the existence of a supervisor that achieves the control objective. (4) Another algorithm is developed to calculate control online if the necessary and sufficient is satisfied. (5) We show that the control calculated by the algorithm is least restrictive. (6) The application of the theoretical results to HIV/AIDS treatment decision making is illustrated.

Nowadays, more and more systems are networked systems in the sense that systems and controllers are connected via shared communication networks. In such networked systems, communication delays and losses are unavoidable. In the future, we plan to investigate supervisory control of networked fuzzy discrete event systems, where there are delays and losses in both observation channels and control channels.

## VII. Appendix

*Proof of Theorem 1:*

Because of Assumption (A1), for all $w \in P(L(G))$, $\rho(w)$ is a finite set. Hence, Algorithm 1 terminates finitely. We prove that

$$\begin{aligned}\rho(w) &= P^{-1}(w) \cap L(G)\\ \tilde{E}(w) &= \{\tilde{\theta}_o \circ \tilde{s} : s \in \rho(w)\}\end{aligned}$$

by induction on the length $|w|$ of $w$.

*Base*: For $|w| = 0$, that is, $w = \varepsilon$, we have, by Line 3 of Algorithm 1,

$$\begin{aligned}\rho(\varepsilon) &= \{s \in \Sigma^* : s \in \Sigma_{uo}^* \wedge \delta(q_o, s)!\}\\ &= P^{-1}(\varepsilon) \cap L(G).\end{aligned}$$

By Lines 4-10 of Algorithm 1, for all $s \in \rho(\varepsilon)$,

$$\begin{aligned}\pi(s) &= (s, q_s, \tilde{\theta}_s)\\ &= (s, \delta(q_o, s), \tilde{\theta}_o \circ \tilde{s}).\end{aligned}$$

Also,

$$\begin{aligned}\tilde{E}(\varepsilon) &= \cup_{s \in \rho(\varepsilon)}\{\tilde{\theta}_s\}\\ &= \{\tilde{\theta}_s : s \in \rho(\varepsilon)\}\\ &= \{\tilde{\theta}_o \circ \tilde{s} : s \in \rho(\varepsilon)\}.\end{aligned}$$

*Induction Hypothesis*: Assume that for all $w \in P(L(G))$ such that $|w| \leq n$,

$$\rho(w) = P^{-1}(w) \cap L(G)$$
$$\tilde{E}(w) = \{\tilde{\theta}_o \circ \tilde{s} : s \in \rho(w)\}.$$

*Induction Step*: We prove that for all $w\sigma \in P(L(G))$ such that $\sigma \in \Sigma_o$ and $|w\sigma| = n+1$,

$$\rho(w\sigma) = P^{-1}(w\sigma) \cap L(G)$$
$$\tilde{E}(w\sigma) = \{\tilde{\theta}_o \circ \tilde{s} : s \in \rho(w\sigma)\}.$$

By Lines 13-17 of Algorithm 1, we have

$$\begin{aligned}
\rho(w\sigma) =& \cup_{s\in\rho(w)}\, s\mu(q_s,\sigma)\\
=&\{s\sigma u \in \Sigma^* : s \in \rho(w) \wedge \sigma u \in \mu(q_s,\sigma)\}\\
=&\{s\sigma u \in \Sigma^* : s \in \rho(w) \wedge u \in \Sigma_{uo}^*\\
&\wedge \delta(q_s,\sigma u)!\}\\
=&\{s\sigma u \in \Sigma^* : s \in \rho(w) \wedge P(\sigma u) = \sigma\\
&\wedge \delta(q_s,\sigma u)!\}\\
=&\{s\sigma u \in \Sigma^* : s \in P^{-1}(w) \cap L(G)\\
&\wedge P(\sigma u) = \sigma \wedge \delta(q_s,\sigma u)!\}\\
&\text{(by Induction Hypothesis)}\\
=&\{s\sigma u \in \Sigma^* : P(s) = w \wedge s \in L(G)\\
&\wedge P(\sigma u) = \sigma \wedge \delta(q_s,\sigma u)!\}\\
=&\{s\sigma u \in \Sigma^* : P(s) = w \wedge P(\sigma u) = \sigma\\
&\wedge s\sigma u \in L(G)\}\\
=&\{s\sigma u \in \Sigma^* : P(s\sigma u) = w\sigma \wedge s\sigma u \in L(G)\}\\
=&\{s\sigma u \in \Sigma^* : P(s\sigma u) = w\sigma\} \cap L(G)\\
=&P^{-1}(w\sigma) \cap L(G).
\end{aligned}$$

By Lines 18-24 of Algorithm 1, we have, for all $s\sigma u \in \rho(w\sigma)$,

$$\begin{aligned}
\pi(s\sigma u) &= (s\sigma u, q_{s\sigma u}, \tilde{\theta}_{s\sigma u})\\
&= (s\sigma u, \delta(q_s,\sigma u), \tilde{\theta}_s \circ \tilde{\sigma} \circ \tilde{u})\\
&= (s\sigma u, \delta(q_o, s\sigma u), \tilde{\theta}_o \circ \tilde{s} \circ \tilde{\sigma} \circ \tilde{u}).
\end{aligned}$$

Also,

$$\begin{aligned}
\tilde{E}(w\sigma) =& \cup_{s\sigma u\in\rho(w\sigma)}\, \{\tilde{\theta}_{s\sigma u}\}\\
=&\{\tilde{\theta}_{s\sigma u} : s\sigma u \in \rho(w\sigma)\}\\
=&\{\tilde{\theta}_o \circ \tilde{s} \circ \tilde{\sigma} \circ \tilde{u} : s\sigma u \in \rho(w\sigma)\}\\
=&\{\tilde{\theta}_o \circ \tilde{s'} : s' \in \rho(w\sigma)\}\\
&\text{(let } s' = s\sigma u).
\end{aligned}$$

■

*Proof of Theorem 2:*

Because of Assumptions (A1) and (A2), for all $w \in P(L(G))$ and $s \in \rho(w)$, $\rho(w)$ and $\eta(q_s,\sigma)$ are finite sets. Hence, Algorithm 2 terminates finitely. We now prove the rest of the theorem.

("IF" Part) We prove that if $(\forall u \in \Sigma_{uc}^* \cap L(G))\tilde{\delta}(\tilde{\theta}_o, \tilde{u}) \notin \Theta_b$ is true, then the supervisor $\mathcal{S}^\circ$ obtained in Algorithm 2 is a valid supervisor by contradiction as follows.

Suppose that $\mathcal{S}^\circ$ is not a valid supervisor, then

$$\begin{aligned}
&\neg(\forall s \in L(\mathcal{S}^\circ/G))\tilde{\delta}(\tilde{\theta}_o, \tilde{s}) \notin \Theta_b\\
\Leftrightarrow&(\exists s \in L(\mathcal{S}^\circ/G))\tilde{\delta}(\tilde{\theta}_o, \tilde{s}) \in \Theta_b.
\end{aligned}$$

Let $s \in L(\mathcal{S}^\circ/G)$ be the shortest sequence such that $\tilde{\delta}(\tilde{\theta}_o, \tilde{s}) \in \Theta_b$. In other words, $(\forall s' \leq s)s' \neq s \Rightarrow \tilde{\delta}(\tilde{\theta}_o, \tilde{s'}) \notin \Theta_b$, where $s' \leq s$ denotes that $s'$ is a prefix of $s$. We consider two possible cases for $s$.

*Case 1:* All events in $s$ are uncontrollable, that is, $s \in \Sigma_{uc}^*$. In this case, $s \in \Sigma_{uc}^* \cap L(G) \wedge \tilde{\delta}(\tilde{\theta}_o, \tilde{s}) \in \Theta_b$, which contradicts the assumption

$$(\forall u \in \Sigma_{uc}^* \cap L(G))\tilde{\delta}(\tilde{\theta}_o, \tilde{u}) \notin \Theta_b.$$

*Case 2:* There exists at least one controllable event in $s$, that is, $s \notin \Sigma_{uc}^*$. Let $\sigma \in \Sigma_c$ be the last controllable event in $s$. By Assumption (A3), $\sigma \in \Sigma_o$. Hence, Algorithm 2 can make the decision on enabling or disabling $\sigma$ just before its occurrence.

We can write $s$ as $s = s'\sigma u$, where $u \in \Sigma_{uc}^*$. Let $w = P(s')$. Then $s' \in \rho(w) = P^{-1}(w) \cap L(G)$. Thus, we have

$$\begin{aligned}
&(\exists s' \in \rho(w))(\exists \sigma u \in \eta(q_{s'},\sigma))\tilde{\delta}(\tilde{\theta}_{s'}, \tilde{\sigma}\tilde{u}) \in \Theta_b\\
&\text{(because } \tilde{\delta}(\tilde{\theta}_o, \tilde{s}) \in \Theta_b)\\
\Leftrightarrow&\neg(\forall s' \in \rho(w))(\forall \sigma u \in \eta(q_{s'},\sigma))\tilde{\delta}(\tilde{\theta}_{s'}, \tilde{\sigma}\tilde{u}) \notin \Theta_b.
\end{aligned}$$

Hence, the condition in the "if" statement of Line 6 in Algorithm 2 is not satisfied. Therefore, by the definition of $L(\mathcal{S}^\circ/G)$,

$$\sigma \notin \mathcal{S}^\circ(w) \Rightarrow s \notin L(\mathcal{S}^\circ/G),$$

which is a contradiction.

("ONLY IF" Part) We need to prove that if $(\forall u \in \Sigma_{uc}^* \cap L(G))\tilde{\delta}(\tilde{\theta}_o, \tilde{u}) \notin \Theta_b$ is false, then no valid supervisor exists. Indeed,

$$\begin{aligned}
&\neg(\forall u \in \Sigma_{uc}^* \cap L(G))\tilde{\delta}(\tilde{\theta}_o, \tilde{u}) \notin \Theta_b\\
\Leftrightarrow&(\exists u \in \Sigma_{uc}^* \cap L(G))\tilde{\delta}(\tilde{\theta}_o, \tilde{u}) \in \Theta_b.
\end{aligned}$$

Because any supervisor $\mathcal{S}$ must enable all uncontrollable events, $u \in \Sigma_{uc}^*$ implies $u \in L(\mathcal{S}/G)$. Hence, for any supervisor $\mathcal{S}$,

$$(\exists u \in L(\mathcal{S}/G))\tilde{\delta}(\tilde{\theta}_o, \tilde{u}) \in \Theta_b,$$

that is, $\mathcal{S}$ is not valid. Therefore, no valid supervisor exists.

■

*Proof of Proposition 1:*

Clearly, the complexity of Algorithm 1 is determined (that is, bounded) by the cardinality of $\rho(w) = P^{-1}(w) \cap L(G)$, denoted by $|\rho(w)|$. To find $|\rho(w)|$, let us denote the automaton marking $w$ by $H(w)$. $H(w)$ can be constructed as follows. Denote $w = \sigma_1\sigma_2...\sigma_{|w|}$, where $|w|$ is the length of $w$. Then

$$H(w) = (X, \Sigma_o, \xi, x_o, X_m).$$

The state set of $H(w)$ is $X = \{0, 1, 2, ..., |w|\}$. The transition function $\xi$ is defined as, for $x \in X$ and $\sigma \in \Sigma_o$,

$$\xi(x, \sigma) = \begin{cases} x + 1 & \text{if } \sigma = \sigma_{x+1} \\ \text{undefined} & \text{otherwise.} \end{cases}$$

The initial state is $x_o = 0$. The marked state set is $X_m = \{|w|\}$. Clearly $L_m(H(w)) = \{w\}$. Using $H(w)$, we have

$$\begin{aligned} \rho(w) &= P^{-1}(w) \cap L(G) \\ &= P^{-1}(L_m(H(w))) \cap L(G) \\ &= L_m(SL(H_m(w)) \times G), \end{aligned}$$

where $SL(.)$ denotes the operator of adding self-loops of unobservable events at all states, and $\times$ denotes the product of automata. It is well-known [4] that the number of states in $SL(H_m(w)) \times G$ is bounded by $|X| \times |Q|$.

By Assumption (A1), $\rho(w)$ is finite. Thus, there are no loops in the automaton $SL(H_m(w)) \times G$. Hence, the cardinality of $\rho(w)$ is bounded by the number of states in $SL(H_m(w)) \times G$. Therefore,

$$|\rho(w)| \leq |X| \times |Q| = (|w| + 1) \times |Q|.$$

In other words, the cardinality of $\rho(w)$ increases linearly as the length of $w$ increases, which implies that the complexity of Algorithm 1 is linear with respect to the length of $w$.

■

*Proof of Proposition 2:*

The proof of Proposition 2 is similar to that of Proof of Proposition 1.

■

*Proof of Proposition 3:* Assume $\mathcal{S}' \leq \mathcal{S}$. We prove that, for all $s \in L(G)$,

$$s \in L(\mathcal{S}'/G) \Rightarrow s \in L(\mathcal{S}/G)$$

by induction on the length $|s|$ of $s$.

*Base:* By definition, $\varepsilon \in L(\mathcal{S}'/G)$ and $\varepsilon \in L(\mathcal{S}/G)$. Therefore, for $|s| = 0$, that is, $s = \varepsilon$, we have

$$s \in L(\mathcal{S}'/G) \Rightarrow s \in L(\mathcal{S}/G).$$

*Induction Hypothesis:* Assume that for all $s \in L(G)$, $|s| \leq m$,

$$s \in L(\mathcal{S}'/G) \Rightarrow s \in L(\mathcal{S}/G).$$

*Induction Step:* We show that for all $s\sigma \in L(G)$, $\sigma \in \Sigma$, $|s\sigma| = m + 1$,

$$s\sigma \in L(\mathcal{S}'/G) \Rightarrow s\sigma \in L(\mathcal{S}/G).$$

Indeed, by the definition of large language, Equation (7), we have,

$$\begin{aligned} & s\sigma \in L(\mathcal{S}'/G) \\ \Rightarrow & s \in L(\mathcal{S}'/G) \wedge s\sigma \in L(G) \wedge (\sigma \in \Sigma_{uc} \vee \sigma \in \mathcal{S}(P(s'))) \\ \Rightarrow & s \in L(\mathcal{S}/G) \wedge s\sigma \in L(G) \wedge (\sigma \in \Sigma_{uc} \vee \sigma \in \mathcal{S}(P(s'))) \\ & \text{(by Induction Hypothesis )} \\ \Rightarrow & s \in L(\mathcal{S}/G) \wedge s\sigma \in L(G) \wedge (\sigma \in \Sigma_{uc} \vee \sigma \in \mathcal{S}(P(s))) \\ & \text{(because } \mathcal{S}' \leq \mathcal{S}) \\ \Rightarrow & s\sigma \in L(\mathcal{S}/G). \end{aligned}$$

■

*Proof of Theorem 3:*

Let us prove the result by contradiction. Suppose that the supervisor $\mathcal{S}^\circ$ obtained in Algorithm 2 is not the least restrictive valid supervisor. Then

$$\begin{aligned} & (\exists \mathcal{S} \in VS(\Theta_b)) L(\mathcal{S}/G) \not\subseteq L(\mathcal{S}^\circ/G) \\ \Rightarrow & (\exists \mathcal{S} \in VS(\Theta_b))(\exists s'' \in L(G)) \\ & s'' \in L(\mathcal{S}/G) \wedge s'' \notin L(\mathcal{S}^\circ/G) \\ \Rightarrow & (\exists \mathcal{S} \in VS(\Theta_b))(\exists s\sigma \in L(G))\sigma \in \Sigma \\ & \wedge s\sigma \in L(\mathcal{S}/G) \wedge s \in L(\mathcal{S}^\circ/G) \wedge s\sigma \notin L(\mathcal{S}^\circ/G) \\ & (\text{let } s\sigma \text{ be the shortest } s''). \end{aligned}$$

For such $s\sigma$, we have

$$\begin{aligned} & s \in L(\mathcal{S}^\circ/G) \wedge s\sigma \notin L(\mathcal{S}^\circ/G) \\ \Rightarrow & \neg(\sigma \in \Sigma_{uc} \vee \sigma \in \mathcal{S}^\circ(P(s))) \\ \Rightarrow & \sigma \in \Sigma_c \wedge \sigma \notin \mathcal{S}^\circ(P(s)) \\ \Rightarrow & \sigma \in \Sigma_c \wedge \neg(\forall s' \in \rho(P(s)))(\forall \sigma u \in \eta(q_{s'}, \sigma)) \\ & \tilde{\delta}(\tilde{\theta}_{s'}, \tilde{\sigma}\tilde{u}) \notin \Theta_b \\ & \text{(by Line 6 of Algorithm 2)} \\ \Rightarrow & \sigma \in \Sigma_c \wedge (\exists s' \in \rho(P(s)))(\exists \sigma u \in \eta(q_{s'}, \sigma)) \\ & \tilde{\delta}(\tilde{\theta}_{s'}, \tilde{\sigma}\tilde{u}) \in \Theta_b. \end{aligned}$$

Since $s\sigma \in L(\mathcal{S}/G)$, all events in $s\sigma$ are enabled by $\mathcal{S}$. In particular, all observable events in $s\sigma$, that is, events in $w = P(s\sigma)$, are enabled by $\mathcal{S}$.

(1) Since $s' \in \rho(P(s)) = P^{-1}(P(s)) \cap L(G)$, we have $P(s'\sigma) = P(s\sigma) = w$, that is, $s'\sigma$ and $s\sigma$ have the same sequence $w$ of observable events. Hence, all observable events in $s'\sigma$ are enabled by $\mathcal{S}$.

(2) By Assumption (A3), all controllable events are observable. Hence, all unobservable events are uncontrollable and must be enables. Therefore, all unobservable events in $s'\sigma$ are enabled by $\mathcal{S}$.

By (1) and (2), all events in $s'\sigma$ are enabled by $\mathcal{S}$. Thus, $s'\sigma \in L(\mathcal{S}/G)$. Since $\sigma u \in \eta(q_{s'}, \sigma) = \{\sigma u \in \Sigma^* : u \in \Sigma_{uc}^* \wedge \delta(q_{s'}, \sigma u)!\}$ implies $u$ are uncontrollable, $v = s'\sigma u \in L(\mathcal{S}/G)$. Therefore,

$$\begin{aligned} & v = s'\sigma u \in L(\mathcal{S}/G) \wedge \tilde{\delta}(\tilde{\theta}_o, \tilde{v}) = \tilde{\delta}(\tilde{\theta}_{s'}, \tilde{\sigma}\tilde{u}) \in \Theta_b \\ \Rightarrow & (\exists v \in L(\mathcal{S}/G))\tilde{\delta}(\tilde{\theta}_o, \tilde{v}) \in \Theta_b \\ \Rightarrow & \neg(\forall v \in L(\mathcal{S}/G))\tilde{\delta}(\tilde{\theta}_o, \tilde{v}) \notin \Theta_b, \end{aligned}$$

which contradicts $\mathcal{S} \in VS(\Theta_b)$ is a valid supervisor.

■

## REFERENCES


[1] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM journal on control and optimization*, vol. 25, no. 1, pp. 206–230, 1987.

[2] F. Lin and W. M. Wonham, "On observability of discrete-event systems," *Information sciences*, vol. 44, no. 3, pp. 173–198, 1988.

[3] W. M. Wonham, K. Cai, *et al.*, "Supervisory control of discrete-event systems," 2019.

[4] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. Springer Nature, 3 ed., 2021.

[5] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis, "Diagnosability of discrete-event systems," *IEEE Transactions on automatic control*, vol. 40, no. 9, pp. 1555–1575, 1995.
[6] F. Lin, "Diagnosability of discrete event systems and its applications," *Discrete Event Dynamic Systems*, vol. 4, no. 2, pp. 197–212, 1994.
[7] A. Saboori and C. N. Hadjicostis, "Notions of security and opacity in discrete event systems," in *2007 46th IEEE Conference on Decision and Control*, pp. 5056–5061, IEEE, 2007.
[8] R. Jacob, J.-J. Lesage, and J.-M. Faure, "Overview of discrete event systems opacity: Models, validation, and quantification," *Annual reviews in control*, vol. 41, pp. 135–146, 2016.
[9] F. Lin and H. Ying, "Fuzzy discrete event systems and their observability," in *Proceedings Joint 9th IFSA World Congress and 20th NAFIPS International Conference (Cat. No. 01TH8569)*, vol. 3, pp. 1271–1276, IEEE, 2001.
[10] D. Qiu, "Supervisory control of fuzzy discrete event systems: a formal approach," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 35, no. 1, pp. 72–88, 2005.
[11] Y. Cao and M. Ying, "Supervisory control of fuzzy discrete event systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 35, no. 2, pp. 366–371, 2005.
[12] W. Deng and D. Qiu, "Supervisory control of fuzzy discrete-event systems for simulation equivalence," *IEEE Transactions on Fuzzy Systems*, vol. 23, no. 1, pp. 178–192, 2014.
[13] F. Lin and H. Ying, "State-feedback control of fuzzy discrete-event systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 40, no. 3, pp. 951–956, 2009.
[14] F. Lin and H. Ying, "Modeling and control of probabilistic fuzzy discrete event systems," *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2021.
[15] Y. Cao and M. Ying, "Observability and decentralized control of fuzzy discrete-event systems," *IEEE Transactions on Fuzzy Systems*, vol. 14, no. 2, pp. 202–216, 2006.
[16] D. Qiu and F. Liu, "Fuzzy discrete-event systems under fuzzy observability and a test algorithm," *IEEE Transactions on Fuzzy Systems*, vol. 17, no. 3, pp. 578–589, 2008.
[17] E. Kılıç and K. Leblebicioğlu, "From classic observability to a simple fuzzy observability for fuzzy discrete-event systems," *Information Sciences*, vol. 187, pp. 224–232, 2012.
[18] E. Kilic, "Diagnosability of fuzzy discrete event systems," *Information Sciences*, vol. 178, no. 3, pp. 858–870, 2008.
[19] F. Liu and D. Qiu, "Diagnosability of fuzzy discrete-event systems: A fuzzy approach," *IEEE Transactions on Fuzzy Systems*, vol. 17, no. 2, pp. 372–384, 2009.
[20] F. Liu and L. Wu, "Decentralized safe diagnosis of fuzzy discrete-event systems," in *2018 37th Chinese Control Conference (CCC)*, pp. 1970–1975, IEEE, 2018.
[21] A. Jayasiri, G. K. Mann, and R. G. Gosine, "Behavior coordination of mobile robotics using supervisory control of fuzzy discrete event systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 41, no. 5, pp. 1224–1238, 2011.
[22] Y. Zhang and S. Shao, "The application of fuzzy discrete event systems in information service system," *JOURNAL-SHANGHAI JIAOTONG UNIVERSITY-CHINESE EDITION-*, vol. 40, no. 11, p. 1901, 2006.
[23] H. Ying, F. Lin, R. D. MacArthur, J. A. Cohn, D. C. Barth-Jones, H. Ye, and L. R. Crane, "A fuzzy discrete event system approach to determining optimal hiv/aids treatment regimens," *IEEE Transactions on Information Technology in Biomedicine*, vol. 10, no. 4, pp. 663–676, 2006.
[24] F. Lin, H. Ying, R. D. MacArthur, J. A. Cohn, D. Barth-Jones, and L. R. Crane, "Decision making in fuzzy discrete event systems," *Information Sciences*, vol. 177, no. 18, pp. 3749–3763, 2007.
[25] N. J. Khan, G. Ahamad, M. Naseem, and Q. R. Khan, "Fuzzy discrete event system (fdes): A survey," in *Renewable Power for Sustainable Growth*, pp. 531–544, Springer, 2021.
[26] A. O. Mekki, F. Lin, and H. Ying, "On detectabilities of fuzzy discrete event systems," *IEEE Transactions on Fuzzy Systems*, vol. 30, no. 2, pp. 426–436, 2022.
[27] G. Awungafac, E. T. Amin, A. Fualefac, N. F. Takah, L. A. Agyingi, J. Nwobegahay, P. Ondoa, and P. A. Njukeng, "Viral load testing and the use of test results for clinical decision making for hiv treatment in cameroon: An insight into the clinic-laboratory interface," *PLoS One*, vol. 13, no. 6, p. e0198686, 2018.
[28] M. Sayan, N. Sultanoglu, B. Uzun, F. S. Yildirim, T. Sanlidag, and D. U. Ozsahin, "Determination of post-exposure prophylaxis regimen in the prevention of potential pediatric hiv-1 infection by the multi-criteria decision making theory," in *2019 Advances in Science and Engineering Technology International Conferences (ASET)*, pp. 1–5, IEEE, 2019.
[29] J. L. Marcus, K. A. Katz, D. S. Krakower, and S. K. Calabrese, "Risk compensation and clinical decision making—the case of hiv preexposure prophylaxis," *The New England journal of medicine*, vol. 380, no. 6, p. 510, 2019.
[30] H. Ying, F. Lin, R. D. MacArthur, J. A. Cohn, D. C. Barth-Jones, H. Ye, and L. R. Crane, "A self-learning fuzzy discrete event system for hiv/aids treatment regimen selection," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 37, no. 4, pp. 966–979, 2007.
[31] H. Ying, F. Lin, and R. Sherwin, "Fuzzy discrete event systems with gradient-based online learning," in *2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pp. 1–6, IEEE, 2019.
[32] H. Ying and F. Lin, "Online self-learning fuzzy discrete event systems," *IEEE Transactions on Fuzzy Systems*, 2019.
[33] H. Ying and F. Lin, "Online learning of event transition matrix of the fuzzy discrete event systems when post-event states are unknown," *IEEE Transactions on Cybernetics*, 2020.
[34] F. Lin and H. Ying, "Supervised learning of multievent transition matrices in fuzzy discrete-event systems," *IEEE Transactions on Cybernetics*, 2022.
[35] F. Lin, "Supervised learning in neural networks: Feedback-network-free implementation and biological plausibility," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

**Feng Lin** (S85-M88-SM07-F09) received his B.Eng. degree in electrical engineering from Shanghai Jiao Tong University, Shanghai, China, in 1982, and the M.A.Sc. and Ph.D. degrees in electrical engineering from the University of Toronto, Toronto, ON, Canada, in 1984 and 1988, respectively. He was a Post-Doctoral Fellow with Harvard University, Cambridge, MA, USA, from 1987 to 1988. Since 1988, he has been with the Department of Electrical and Computer Engineering, Wayne State University, Detroit, MI, USA, where he is currently a Professor. His current research interests include discrete event systems, hybrid systems, neural networks, robust control, and their applications in alternative energy, biomedical systems, machine learning, and automotive control. He authored a book entitled "Robust Control Design: An Optimal Control Approach" and coauthored a paper that received a George Axelby outstanding paper award from the IEEE Control Systems Society. He was an associate editor of IEEE Transactions on Automatic Control.

**Hao Ying** (S88-M90-SM97-F12) received the B.S. and M.S. degrees in electrical engineering from Donghua University, Shanghai, China, in February 1982 and July 1984, respectively, and the Ph.D. degree in biomedical engineering from the University of Alabama at Birmingham, USA, in 1990. He is a Professor with the Department of Electrical and Computer Engineering, Wayne State University, USA. He has authored one book and coauthored another on fuzzy control, and also published 129 journal papers and over 160 conference papers. His Google h-index is 51 with the total number of citations for all his publications being more than 11,000. He is ranked among top 25 percent of the 100,000 most-cited authors across all 22 scientific fields, which are selected from nearly 7 million scientists worldwide. He is a recipient of the 2023 IEEE Computational Intelligence Society Fuzzy Systems Pioneer Award with the citation "For fundamental contributions to model-free fuzzy control theory and its biomedical applications." Professor Ying is an Associate Editor or a Member of Editorial Board for 13 international journals, including the IEEE Transactions on Fuzzy Systems and the IEEE Transactions on Systems, Man, and Cybernetics: Systems. He is a Member of the 2023 Fellows Committee of the IEEE Computational Intelligence Society (was also on the committee in 2020 and 2021), and was on the same committee of the IEEE Systems, Man, Cybernetics Society in 2016, 2017, and 2020. He is a Member of the Fuzzy Systems Technical Committee of the IEEE Computational Intelligence Society for many years. He was a Program Chair/Co-Chair for four international conferences and was a Program/Technical Committee Member for more than 140 international conferences.