A Belief-Evolution-Based Approach for Online Control of Fuzzy Discrete-Event Systems Under Partial Observation

Xiang Yin

***Abstract*—In this paper, we investigate the partially observed supervisor synthesis problem in the framework of fuzzy discrete-event systems (DESs). The goal is to synthesize a fuzzy supervisor such that the behavior of the closed-loop system is within a given fuzzy language. A new approach for solving this problem is proposed based on the idea of belief evolution. Specifically, we propose an algorithm that can be implemented in an online manner. We show that the proposed algorithm is both sound and complete, i.e., it effectively solves the supervisor synthesis problem. To the best of our knowledge, this is the first algorithm with such a property for fuzzy DESs, as previous works on this topic mostly focus on the supervisor existence condition rather than the supervisor synthesis problem.**

***Index Terms*—Belief evolution, fuzzy discrete-event systems (FDESs), partial observation, supervisor control.**

I. Introduction

Discrete-event systems (DESs) are a class of dynamic systems which have inherently event-driven behaviors and discrete state spaces. Modeling and analysis using DES have been successfully applied to many applications, e.g., flexible manufacturing systems, software systems, and communication protocols. However, standard (crisp) DES model cannot capture uncertainty, vagueness, or imprecision of the system. To address this issue, the theory of *fuzzy DES* (FDES) was developed a decade ago by Lin and Ying in [1] and Lin *et al.* in [2]. Since then, many results in the framework of FDES have been developed; see, e.g., [3]–[6].

The motivation of studying FDES is that, in many applications, it is impossible to obtain the precise model of the system. Instead, one may only have a vague knowledge about the system's behavior based on the experience. For example, such an issue arises in the medical treatment problem, since it is hard to tell whether a patient's condition is "good" or "bad" precisely. However, FDES provides a suitable framework for addressing the issues of uncertainty, vagueness, and imprecision. Due to these advantages, the theory of FDES has also been successfully applied to many applications, e.g., HIV/AIDS treatment [7], [8], robot motion planning [9]–[12], and fault diagnosis [13]–[16].

One of the most important problems in DES is the supervisory control problem initiated by Ramadge and Wonham [17]. In this problem, one is interested in synthesizing a supervisor to restrict the system's behavior such that some specifications are achieved. Due to measurement uncertainties and limited sensor capabilities, the issue of partial observation also arises in many applications and one has to handle this issue in the supervisory control problem [18], [19]. In the supervisory control theory, the most fundamental two problems are the *supervisor existence problem* and the *supervisor synthesis problem*. The former asks whether or not we can *exactly achieve* a given specification language, while the later asks to *synthesize* a supervisor such that behavior of the closed-loop system is a *sublanguage* of the given specification language.

The supervisory control problem has also drawn considerable attention in the framework of FDES; see, e.g., [6], [11], [20]–[31]. In particular, in [22], Cao and Ying investigated the supervisor existence problem under the partial observation setting, where the notions of fuzzy controllability and fuzzy observability were proposed. It was shown that these two notions together provide the necessary and sufficient conditions for the existence of a supervisor that achieves a given fuzzy language. In [25], Qiu and Liu proposed a more general framework for the supervisory control of partially observed FDES by considering fuzziness in both observation and control. Necessary and sufficient conditions for the supervisor existence problem were also derived.

Although the supervisory control of FDES has been investigated by many works, there are still several important problems remain. In particular, both [22] and [25] only focus on the supervisor existence problem. However, when the necessary and sufficient conditions for the supervisor existence problem do not hold, how to *synthesize* a supervisor has drawn little attention. Moreover, in both the framework of [22] and the framework of [25], the notion of observability is not preserved under union. Consequently, unlike the fully-observed case, there does not exist a supremal solution to the synthesis problem under the partial observation setting. To address this issue, in [22], the notion of fuzzy normality, which is strictly stronger than fuzzy observability, was proposed. Since fuzzy normality is preserved under union, we can compute the supremal fuzzy normal sublanguage in order to synthesize a supervisor. However, this approach has a limitation since normality may be too strong in many cases. Therefore, using the supremal normal approach to synthesize a supervisor is just sound but not complete, i.e., the supremal fuzzy normal language may be empty even if a nonempty supervisor exists. To the best of our knowledge, finding a sound and complete algorithm for the supervisor synthesis problem in partially observed FDES is still an open problem.

In this paper, we tackle the supervisor synthesis problem for partially observed FDES in the framework of Cao and Ying [22]. Specifically, we consider fuzzy languages recognized by max-min automata, where controllability is fuzzy and observability is crisp. We propose a new synthesis approach based on the concept of *belief evolution*. The idea of using belief sets is motivated by some related approaches in crisp DES; see, e.g., [32]–[35]. Roughly speaking, a belief is a sufficient statistic of all information available. In this paper, we generalize the idea of belief evolution from the crisp case to FDES by taking the degree of membership into account. We propose an effective online algorithm that synthesizes a supervisor recursively. Such an online mechanism is useful in many practical situations, where the system is time-varying or the system is too large to compute the supervisor offline all at once. Moreover, unlike the supremal fuzzy normal approach, which is sound but not complete, we show that the proposed algorithm is both sound and complete. Therefore, it effectively solves the supervisor synthesis

Manuscript received May 2, 2016; revised July 23, 2016; accepted September 19, 2016. Date of publication October 13, 2016; date of current version November 29, 2017.

The author is with the Department of Automation, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: xiangyin@umich.edu).

Digital Object Identifier 10.1109/TFUZZ.2016.2617361

problem for FDES. To the best of our knowledge, this is the first algorithm with such a property for FDES.

The rest of this paper is organized as follows. In Section II, we review the theory of FDES. In Section III, we first present the concept of belief evolution. Then, we propose an online algorithm that solves the supervisor synthesis problem. In Section IV, we show that the proposed algorithm is both sound and complete. An illustrative example is provided in Section V. Finally, we conclude the paper in Section VI.

II. Problem Formulation

A. FDES and Fuzzy Languages

Let Σ be a finite set of alphabets or events. We denote by Σ^* the set of finite strings over Σ. A language $L \subseteq \Sigma^*$ is a subset of Σ^*. For any language L, the prefix closure of L is defined as $\overline{L} = \{s \in \Sigma^* : \exists t \in \Sigma^* \text{ s.t. } st \in L\}$. For any string s, we write $\overline{s} = \overline{\{s\}}$ for simplicity. For any two languages L_1 and L_2, we denote by $L_1 L_2$ their concatenation, i.e., $L_1 L_2 = \{st \in \Sigma^* : s \in L_1, t \in L_2\}$.

A crisp DES is a deterministic finite-state automaton $G = (Q, \Sigma, \delta, q_0)$, where Q is a finite state of states, Σ is a finite set of events, $\delta : Q \times \Sigma \to Q$ is the partial transition function, and $q_0 \in Q$ is the initial state. The transition function is also extended to $\delta : Q \times \Sigma^* \to Q$ in the usual manner; see, e.g., [36]. The language generated by G is $\mathcal{L}_G = \{s \in \Sigma^* : \delta(q_0, s)!\}$, where "!" means "is defined."

Let X be a universal set. A *fuzzy set* $\mathcal{A}$ w.r.t. X is a membership function $\mathcal{A} : X \to [0, 1]$ that assigns each element in X a value in $[0, 1]$. Namely, for any $x \in X$, $\mathcal{A}(x)$ denotes its *degree of membership* in $\mathcal{A}$. We denote by supp$[\mathcal{A}]$ the *support* of fuzzy set $\mathcal{A}$. Specifically, supp$[\mathcal{A}]$ is a crisp set defined by supp$[\mathcal{A}] = \{x \in X : \mathcal{A}(x) > 0\}$. Let supp$[\mathcal{A}] = \{x_1, \ldots, x_n\}$ be the support of $\mathcal{A}$. We also write $\mathcal{A}$ in Zadeh's notation [37] $\mathcal{A} = \frac{\mathcal{A}(x_1)}{x_1} + \frac{\mathcal{A}(x_2)}{x_2} + \cdots + \frac{\mathcal{A}(x_n)}{x_n}$. We use $\mathcal{F}(X)$ to denote the set of all fuzzy sets over X. Let $\mathcal{A}, \mathcal{B} \in \mathcal{F}(X)$ be two fuzzy sets. We denote by $\mathcal{A} \subseteq \mathcal{B}$ if $\forall x \in X : \mathcal{A}(x) \leq \mathcal{B}(x)$. We denote by $\mathcal{A} \subset \mathcal{B}$ if 1) $\mathcal{A} \subseteq \mathcal{B}$; and 2) $\exists x \in X : \mathcal{A}(x) < \mathcal{B}(x)$. The empty fuzzy set, denoted by $\mathcal{O}$, is a fuzzy set such that $\forall x \in X : \mathcal{O}(x) = 0$. We denote by $\vee$ and $\wedge$ the supremum and the infimum operators, respectively.

In this paper, we consider FDES modeled by *max–min automata*. Specifically, an FDES is a four-tuple $G = (Q, \Sigma, \delta, q_0)$, where Q is a finite set of states, Σ is a finite set of events, $\delta : Q \times \Sigma \times Q \to [0, 1]$ is the fuzzy transition function, and q_0 is the initial state. The fuzzy transition function is extended to $\delta : Q \times \Sigma^* \times Q \to [0, 1]$ inductively as follows: for any $q_1, q_2 \in Q$, $s \in \Sigma^*$, and $\sigma \in \Sigma$, we have

$$\delta(q_1, \epsilon, q_2) = \begin{cases} 1, & \text{if } q_1 = q_2, \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

$$\delta(q_1, s\sigma, q_2) = \vee_{q' \in Q} (\delta(q_1, s, q') \wedge \delta(q', \sigma, q_2)). \tag{2}$$

The *fuzzy language* generated by G, denoted by $\mathcal{L}_G$, is a fuzzy subset over Σ^*, i.e., $\mathcal{L}_G \in \mathcal{F}(\Sigma^*)$. Specifically, $\mathcal{L}_G$ is defined by $\forall s \in \Sigma^* : \mathcal{L}_G(s) = \vee_{q \in Q} \delta(q_0, s, q)$. Namely, $\mathcal{L}_G(s)$ denotes the degree of membership of string s in $\mathcal{L}_G$. By definition, we know that 1) $\mathcal{L}_G(\epsilon) = 1$; and 2) $\forall s, t \in \Sigma^* : \mathcal{L}_G(st) \leq \mathcal{L}_G(s)$. Moreover, if a fuzzy language $\mathcal{L} \in \mathcal{F}(\Sigma^*)$ satisfies these two conditions, then we can always construct an FDES G such that $\mathcal{L}_G = \mathcal{L}$; see, e.g., [22]. We will assume the above two conditions for any fuzzy language except $\mathcal{O}$.

B. Supervisory Control of FDES

We review the supervisory control theory of FDES in the framework of Cao and Ying [22]. In this framework, the event set is partitioned by $\Sigma = \Sigma_c \dot{\cup} \Sigma_{uc} = \Sigma_o \dot{\cup} \Sigma_{uo}$, where $\Sigma_c, \Sigma_{uc}, \Sigma_o$, and Σ_{uo} are the sets of controllable, uncontrollable, observable, and unobservable events, respectively. The natural projection $P : \Sigma^* \to \Sigma_o^*$ is defined in the usual manner; see, e.g., [36]. The natural projection is also extended to 2^{Σ^*}.

Then, a partial observation *fuzzy supervisor* is defined as a function $S : P(\text{supp}[\mathcal{L}_G]) \to \mathcal{F}(\Sigma)$ such that $\forall \alpha \in P(\text{supp}[\mathcal{L}_G]), \forall \sigma \in \Sigma_{uc} : S(\alpha)(\sigma) = 1$. For any observable string $\alpha \in P(\text{supp}[\mathcal{L}_G])$, $S(\alpha)(\sigma)$ denotes the *degree of enablement* of event σ. Clearly, we cannot disable an uncontrollable event in any degree.

Given a fuzzy supervisor S, S/G denotes the closed-loop system under control. The fuzzy language generated by S/G, denoted by $\mathcal{L}_{S/G}$, is defined recursively by [22]: 1) $\mathcal{L}_{S/G}(\epsilon) = 1$; and 2) $\mathcal{L}_{S/G}(s\sigma) = \mathcal{L}_G(s\sigma) \wedge \mathcal{L}_{S/G}(s) \wedge S(P(s))(\sigma)$.

Let $\mathcal{K} \in \mathcal{F}(\Sigma^*)$ be a fuzzy language. We say that $\mathcal{K}$ is *fuzzy controllable* (w.r.t. $\mathcal{L}_G$ and Σ_c) [21] if $(\forall s \in \Sigma^*)(\forall \sigma \in \Sigma_{uc})[\mathcal{K}(s\sigma) = \mathcal{K}(s) \wedge \mathcal{L}_G(s\sigma)]$. We say that $\mathcal{K}$ is *fuzzy observable* (w.r.t. $\mathcal{L}_G, \Sigma_o$ and Σ_c) [22] if $(\forall s, s' \in \text{supp}[\mathcal{K}] : P(s) = P(s'))(\forall \sigma \in \Sigma_c : s\sigma \in \text{supp}[\mathcal{K}])(\exists x \in [0, 1])$ s.t. $\mathcal{K}(s\sigma) = \mathcal{K}(s) \wedge \mathcal{L}_G(s\sigma) \wedge x$ and $\mathcal{K}(s'\sigma) = \mathcal{K}(s') \wedge \mathcal{L}_G(s'\sigma) \wedge x$.

It was shown in [22] that, given a fuzzy language $\mathcal{K}$, there exists a fuzzy supervisor S such that $\mathcal{L}_{S/G} = \mathcal{K}$, if and only if, $\mathcal{K}$ is fuzzy controllable and fuzzy observable. These two conditions are referred to as the *supervisor existence* conditions. However, in general, a fuzzy language may not be controllable and observable. Therefore, we need to *synthesize* a fuzzy supervisor S such that the closed-loop fuzzy language is *safe*, i.e., $\mathcal{L}_{S/G} \subseteq \mathcal{K}$. This problem is referred to as the *supervisor synthesis* problem, which is formulated as follows.

Problem 1: (Fuzzy Supervisor Synthesis Problem). Let G be an FDES with Σ_c and Σ_o. Let $\mathcal{K} \subseteq \mathcal{L}_G$ be a fuzzy specification language. Synthesize a fuzzy supervisor $S : P(\text{supp}[\mathcal{L}_G]) \to \mathcal{F}(\Sigma)$ such that $\mathcal{L}_{S/G} \subseteq \mathcal{K}$.

Remark II.1: It was shown in [21] that fuzzy controllability is closed under union. Therefore, the supremal fuzzy controllable sublanguage of $\mathcal{K}$ exists. Hereafter, we assume without the loss of generality that the specification language $\mathcal{K}$ is fuzzy controllable; otherwise, we can compute the supremal fuzzy controllable sublanguage of $\mathcal{K}$ to replace $\mathcal{K}$. However, fuzzy observability is not closed under union [22], i.e., there does not exist a supremal fuzzy observable sublanguage. This is the fundamental difficulty in the partially observed synthesis problem. In this paper, we will propose a constructive approach for synthesizing a supervisor in order to tackle this difficulty.

Remark II.2: In this paper, we follow the framework of Cao and Ying [22], where controllability is fuzzy while observability is crisp. A general framework, where both controllability and observability are fuzzy, was proposed by Qiu and Liu [25]. In principle, the idea in this paper can be extended to the general framework. We choose the simple framework in order to simplify our technical development and to present the essence of our idea that solves the supervisor synthesis problem.

III. Synthesis Algorithm

In this section, we address the fuzzy supervisor synthesis problem. First, we discuss the concept of *belief evolution* in the partial-observed system. Then, we propose an *online approach* that effectively synthesizes a safe supervisor.

A. Belief Evolution

In the partially observed synthesis problem, one of the most important issues is what is our *belief* about the system. Roughly speaking, a belief is a sufficient statistic of all available information we have

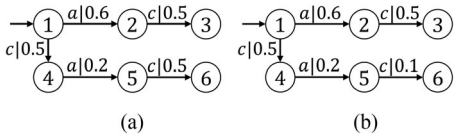

Fig. 1. FEDS G and $\mathcal{L}_H = \mathcal{K}$, where $\Sigma_c = \{c\}$ and $\Sigma_o = \{a\}$. (a) G. (b) H.

obtained so far. Hereafter, we will explore two language-based beliefs, $\hat{\mathtt{B}}$ and $\mathtt{B}$.

Let $S : P(\text{supp}[\mathcal{L}_G]) \to \mathcal{F}(\Sigma)$ be a fuzzy supervisor and $\alpha \in P(\text{supp}[\mathcal{L}_{S/G}])$ be an observable string. Specifically, $\hat{\mathtt{B}}(\alpha)$ is defined as the set of strings in $\text{supp}[\mathcal{L}_{S/G}]$ that are possible *immediately after* observing α, i.e.,

$$\hat{\mathtt{B}}(\alpha) = \{s \in \text{supp}[\mathcal{L}_{S/G}] : P(s) = \alpha \text{ and } s \in \Sigma^*\Sigma_o \cup \{\epsilon\}\}. \quad (3)$$

We define $\mathtt{B}(\alpha)$ as the set of strings in $\text{supp}[\mathcal{L}_{S/G}]$ that are consistent with observation α, i.e.,

$$\mathtt{B}(\alpha) = \{s \in \text{supp}[\mathcal{L}_{S/G}] : P(s) = \alpha\}. \quad (4)$$

Clearly, we know that $\hat{\mathtt{B}}(\alpha) \subseteq \mathtt{B}(\alpha)$, since the latter contains the unobservable tails while the former does not. Note that, by the definition of the supervisor, sets $\hat{\mathtt{B}}(\alpha)$ and $\mathtt{B}(\alpha)$ only depends on the control decisions that have been made so far and they do not depend on the control decisions in the future. Therefore, these two sets can also be computed recursively, as follows: for any $\alpha \in \Sigma_o^*, \sigma \in \Sigma_o$, we have

$$\hat{\mathtt{B}}(\epsilon) = \{\epsilon\} \quad (5)$$

$$\mathtt{B}(\alpha) = \{st \in \text{supp}[\mathcal{L}_G] : s \in \hat{\mathtt{B}}(\alpha) \wedge t \in (\Sigma_{uo} \cap \text{supp}[S(\alpha)])^*\} \quad (6)$$

$$\hat{\mathtt{B}}(\alpha\sigma) = \{s\sigma \in \text{supp}[\mathcal{L}_G] : s \in \mathtt{B}(\alpha) \wedge \sigma \in \text{supp}[S(\alpha)]\}. \quad (7)$$

Let $\sigma_1 \dots \sigma_n \in P(\text{supp}[\mathcal{L}_{S/G}])$ be a sequence of observable events. Then, the above recursive definition essentially yields a sequence alternating between $\hat{\mathtt{B}}$ and $\mathtt{B}$, i.e.,

$$\widehat{\mathtt{B}}(\epsilon) \xrightarrow{S(\epsilon)} \mathtt{B}(\epsilon) \xrightarrow{\sigma_1} \cdots \xrightarrow{\sigma_n} \widehat{\mathtt{B}}(\sigma_1 \dots \sigma_n) \xrightarrow{S(\sigma_1 \dots \sigma_n)} \mathtt{B}(\sigma_1 \dots \sigma_n).$$

Hereafter, we refer to $\mathtt{B}(\alpha)$ and $\hat{\mathtt{B}}(\alpha)$ as *belief sets* (or beliefs) and refer to the above sequence as *the belief evolution* along $\sigma_1 \dots \sigma_n$. Note that sets $\mathtt{B}(\alpha)$ and $\hat{\mathtt{B}}(\alpha)$ are both crisp and we will handle the issue of membership degree later.

Example III.1: Let us consider the system G shown in Fig. 1(a) and the specification language $\mathcal{K}$ generated by automaton H shown in Fig. 1(b). Suppose that $\Sigma_c = \{c\}$ and $\Sigma_o = \{a\}$. We consider a supervisor S defined by $S(\epsilon)(c) = 0.5, S(a)(c) = 0.1$ and $S(\epsilon)(a) = S(a)(a) = 1$. Initially, we have that $\hat{\mathtt{B}}(\epsilon) = \{\epsilon\}$. Once the first control decision $S(\epsilon)$ is made, we obtain $\mathtt{B}(\epsilon) = \{\epsilon, c\}$ according to (6). If $a \in \Sigma_o$ is observed, then we obtain $\hat{\mathtt{B}}(a) = \{a, ca\}$ according to (7). By making a new control decision $S(a)$, we move to $\mathtt{B}(a) = \{ac, cac\}$.

B. Online Synthesis Algorithm

We present a formal procedure to synthesize a fuzzy supervisor $S : P(\text{supp}[\mathcal{L}_G]) \to \mathcal{F}(\Sigma)$ such that $\mathcal{L}_{S/G} \subseteq K$. In our approach, instead of computing S directly all at once, we will *recursively* compute fuzzy set $S(\alpha) \in \mathcal{F}(\Sigma)$ for each observation $\alpha \in \Sigma_o^*$ encountered in an online manner. This is detailed as follows.

First, let $\alpha = \sigma_1 \dots \sigma_n \in \Sigma_o^*$ be the sequence of events been observed. Suppose that all control decisions up to $\sigma_1 \dots \sigma_{n-1}$, i.e., $S(\epsilon), S(\sigma_1), \dots, S(\sigma_1 \dots \sigma_{n-1})$, have been computed. By using the above information, belief set $\hat{B}(\alpha)$ can be effectively computed by the recursive procedure discussed earlier. Then our goal is to determine the next fuzzy control decision $S(\alpha) \in \mathcal{F}(\Sigma)$. To this end, we take two steps: 1) first we determine the support of $S(\alpha)$, denoted by $Dec \in 2^\Sigma$; 2) then we assign each element in Dec a degree of enablement in order to obtain fuzzy set $S(\alpha) \in \mathcal{F}(\Sigma)$.

Step-1: First, we determine the support of the control decision. Specifically, the crisp set $Dec = \text{supp}[S(\alpha)]$ is chosen such that the following conditions hold

- C1 $\Sigma_{uc} \subseteq Dec$.
- C2 $\mathtt{B}(\alpha)(\Sigma_o \cap Dec) \cap \text{supp}[\mathcal{L}_G] \subseteq \text{supp}[\mathcal{K}]$, where $\mathtt{B}(\alpha)$ is the belief set updated from $\hat{\mathtt{B}}(\alpha)$ and $\text{supp}[S(\alpha)] = Dec$ according to (5).
- C3 For any other $Dec' \in 2^\Sigma$ such that C1 and C2 hold, we have $Dec \not\subset Dec'$.

Remark III.1: Let us explain the intuition behind the above three conditions. Condition C1 simply requires that the supervisor cannot disable any uncontrollable event. Condition C2 requires that the supervisor only allows strings in $\text{supp}[\mathcal{K}]$ before the next control decision can be made. This condition guarantees that the closed-loop language is safe if we assign the degree of membership properly. Finally, the last condition C3 says that Dec is a maximal crisp set satisfying C1 and C2 in order to enable as many events as possible. In the next section, we will prove that such a control decision satisfying C1–C3 always exists for any $\alpha \in \Sigma_o^*$ encountered.

Step-2: Once the support of $S(\alpha)$, i.e., Dec, is determined, we need to assign a degree of membership to each event in Dec in order to obtain fuzzy control decision $S(\alpha)$. This is defined as follows: for each event $\sigma \in Dec$, we have

$$S(\alpha)(\sigma) = \begin{cases} \bigwedge\{\Xi(s,\sigma) : s \in \mathtt{B}(\alpha), s\sigma \in \text{supp}[\mathcal{L}_G]\} & \text{if } \sigma \in \Sigma_c \\ 1 & \text{if } \sigma \in \Sigma_{uc} \end{cases} \quad (8)$$

where

$$\Xi(s,\sigma) = \begin{cases} \mathcal{K}(s\sigma), & \text{if } \mathcal{K}(s\sigma) < \mathcal{K}(s) \\ 1, & \text{otherwise.} \end{cases} \quad (9)$$

Remark III.2: Let us explain the intuition behind (8) and (9). For any string s and any event σ, $\Xi(s,\sigma)$ is essentially the largest degree of membership we can enable for event σ following string s. Specifically, for each controllable event $\sigma \in \Sigma_c$, if $\mathcal{K}(s) = \mathcal{K}(s\sigma)$, then there is no need to disable σ for any degree since we are considering max–min system and the degree of $s\sigma$ is bounded by s. On the other hand, if $\mathcal{K}(s\sigma) < \mathcal{K}(s)$, then we do need to disable σ with degree $\mathcal{K}(s\sigma)$ to make sure that the closed-loop fuzzy language is still in $\mathcal{K}$. The enablement degree for σ is taken as the infimum of $\Xi(s,\sigma)$ for any string in belief $\mathtt{B}(\alpha)$. The reason why we need to take the infimum rather than the supremum is still that we need to be conservative in order to guarantee safety.

In order to make the supervisor synthesis approach more clear, Algorithm ONLINE-SYNTHESIS is proposed which essentially summarizes the recursive procedure we discussed above. Specifically, whenever a new event is observed, we compute the fuzzy control decision $S(\alpha)$ and update our beliefs $\mathtt{B}$ and $\hat{\mathtt{B}}$, where α is the entire observation up to this point. Then, we just wait for the next observable event and repeat the above procedure. Therefore, the supervisor is essentially implementable recursively in an *online* manner.

The following example illustrates the synthesis procedure.

Example III.2: Let us still consider system G and specification H shown in Fig. 1(a) and (b), respectively, where $\Sigma_c = \{c\}$ and $\Sigma_o = \{a\}$. Initially, Algorithm ONLINE-SYNTHESIS starts with $\hat{\mathtt{B}}(\epsilon) = \{\epsilon\}$ and we go to line 5 to choose the support of $S(\epsilon)$. Since the only

Algorithm 1: ONLINE-SYNTHESIS $(\mathcal{L}_G, \mathcal{K}, \Sigma_c, \Sigma_o)$

1 $\alpha \leftarrow \epsilon$, $\hat{\mathtt{B}}(\alpha) \leftarrow \{\epsilon\}$;
2 Go to line 5;
while *new event* $\sigma \in \Sigma_o$ *is observed* **do**
3 Obtain $\hat{\mathtt{B}}(\alpha\sigma)$ by $\mathtt{B}(\alpha)$ based on Eq. (7);
4 $\alpha \leftarrow \alpha\sigma$;
5 Obtain supp$[S(\alpha)]$ by $\hat{\mathtt{B}}(\alpha)$ based on C1-C3;
6 Obtain $\mathtt{B}(\alpha)$ by $\hat{\mathtt{B}}(\alpha)$ and supp$[S(\alpha)]$ based on Eq. (6);
7 Compute fuzzy control decision $S(\alpha)$ using $\mathtt{B}(\alpha)$ and supp$[S(\alpha)]$ based on Eqs. (8) and (9);

controllable event is c, we have two choices for supp$[S(\epsilon)]$: $\{a\}$ or $\{a, c\}$. Note that, both decisions $\{a\}$ and $\{a, c\}$ satisfy conditions C1 and C2. For example, if we choose $\{a\}$, then the next belief is $\mathtt{B}(\epsilon) = \{\epsilon\}$ and we have $\{\epsilon\}\{a\} \cap \text{supp}[\mathcal{L}_G] \subseteq \text{supp}[\mathcal{K}]$; if we choose $\{a, c\}$, then the next belief is $\mathtt{B}(\epsilon) = \{\epsilon, c\} \subseteq \text{supp}[\mathcal{K}]$ and we still have $\{\epsilon, c\}\{a\} \cap \text{supp}[\mathcal{L}_G] \subseteq \text{supp}[\mathcal{K}]$. Therefore, according to condition C3, we need to choose $\{a, c\}$ for supp$[S(\epsilon)]$, since it strictly contains $\{a\}$. Then, in line 6, we update our belief by using this choice and get $\mathtt{B}(\epsilon) = \{\epsilon, c\}$. Next, we move to line 7 and we need to determine the degree of membership for each element in supp$[S(\epsilon)]$. For event $a \in \Sigma_{uc}$, which is uncontrollable, we have that $S(\epsilon)(a) = 1$. For event $c \in \Sigma_c$, which is controllable, we have that $\Xi(\epsilon, c) = 0.5$. Also, we note that for, $c \in \mathtt{B}(\epsilon)$, we have $cc \notin \text{supp}[\mathcal{L}_G]$. Therefore, we obtain $S(\epsilon)(c) = \Xi(\epsilon, c) = 0.5$.

Once a new event $a \in \Sigma_o$ is observed, according to line 3, first we need to update our belief to $\hat{\mathtt{B}}(a) = \{a, ca\}$. Then, we need to choose the support of $S(a)$ in line 5; we can still choose supp$[S(a)] = \{a, c\}$. Next, in line 7, for $a \in \Sigma_{uc}$, we still have that $S(a)(a) = 1$. For event $c \in \Sigma_c$, this time we have $\Xi(a, c) = 0.5$ and $\Xi(ca, c) = 0.1$. Therefore, according to (8), we have $S(a)(c) = \Xi(a, c) \wedge \Xi(ca, c) = 0.5 \wedge 0.1 = 0.1$. Since the language in this example is finite, there will be no further observation. In general, the while loop will not terminate since the system may not terminate.

Remark III.3: In [22], a supervisor construction approach was provided in order to *exactly achieve* $\mathcal{K}$ when $\mathcal{K}$ is fuzzy observable. Specifically, the supervisor is defined by: for any $\alpha \in P(\text{supp}[\mathcal{L}_G])$, we have

$$S(\alpha)(\sigma) = \begin{cases} \vee\{\mathcal{K}(s\sigma) : P(s) = \alpha\}, & \text{if } \sigma \in \Sigma_c, \\ 1, & \text{if } \sigma \in \Sigma_{uc}. \end{cases} \tag{10}$$

Note that, this supervisor is valid only when $\mathcal{K}$ is fuzzy observable; otherwise, the closed-loop system may not be safe. For example, if we use this supervisor for the system in Fig. 1, then we have $S(\epsilon)(c) = S(a)(c) = 0.5$. This gives us closed-loop fuzzy language $\mathcal{L}_{S/G} = \frac{1}{\epsilon} + \frac{0.6}{a} + \frac{0.5}{ac} + \frac{0.5}{c} + \frac{0.2}{ac} + \frac{0.2}{cac}$, which is not safe. Compared with this supervisor construction in [22], our supervisor synthesis procedure has the following important features. First, unlike (10), where string s is chosen from supp$[\mathcal{K}]$, in (8), string s is chosen from $\mathtt{B}(\alpha)$. This is because that supervisor defined by (10) aims to match $\mathcal{K}$ exactly and, therefore, it knows *a priori* that the support of the closed-loop language is supp$[\mathcal{K}]$. However, such an information is missing in the synthesis problem and we have to use our control decisions in history to formulate our belief. Second, unlike (10) where the membership degree is taken as the supremum, our membership degree is taken as the infimum; we have to be conservative in order to guarantee the safety requirement.

IV. PROPERTIES OF THE ALGORITHM

In this section, we prove some properties of the proposed algorithm. Given a problem and an algorithm, we say that the algorithm is *sound* if its output is a solution to the problem, i.e., it never returns a wrong answer. Also, we say that an algorithm is *complete* if it never returns "no solution" when one exists. Next, we show that, unlike the supremal fuzzy normal approach, which is just sound but not complete, our synthesis algorithm is both sound and complete. Therefore, it effectively solves the fuzzy supervisor synthesis problem under partial observation.

First of all, we show that the proposed algorithm is sound.

Lemma IV.1: Let G be the system, $\mathcal{K}$ be the specification, and S be the fuzzy supervisor synthesized by Algorithm ONLINE-SYNTHESIS. Then, we have $\mathcal{L}_{S/G} \subseteq \mathcal{K}$.

Proof: It suffices to show that for any $s \in \text{supp}[\mathcal{L}_{S/G}]$: 1) $s \in \text{supp}[\mathcal{K}]$; and 2) $\mathcal{L}_{S/G}(s) \leq \mathcal{K}(s)$. Hereafter, we show that 1) and 2) hold by induction on the length of $P(s)$.

Induction Basis: For $|P(s)| = 0$, we know that $s \in \mathtt{B}(\epsilon)$. By condition C2, control decision $S(\epsilon)$ is chosen such $\mathtt{B}(\epsilon) \subseteq \text{supp}[\mathcal{K}]$. Therefore, 1) holds. Next, we show that 2) holds by contradiction. Assume that 2) does not hold for $|P(s)| = 0$, i.e., there exists a string s such that $|P(s)| = 0$ and $\mathcal{K}(s) < \mathcal{L}_{S/G}(s)$. Note that $s \neq \epsilon$, since $\mathcal{K}(s) = \mathcal{L}_{S/G}(s) = 1$ by definition. Therefore, we write $s = t\sigma \in \Sigma_{uo}^*$, where $\sigma \in \Sigma_{uo}$ is the last event in s. We also assume without the loss of generality that $\forall w \in \overline{t} : \mathcal{L}_{S/G}(w) \leq \mathcal{K}(w)$; otherwise, it suffices to choose the shortest prefix of s such that this assumption holds. If $\sigma \in \Sigma_{uc}$, by the assumption that $\mathcal{K}$ is fuzzy controllable, $\mathcal{L}_{S/G}(t\sigma) \leq \mathcal{K}(t\sigma)$, which contradicts the assumption that 2) does not hold. If $\sigma \in \Sigma_c$, then we know that $\mathcal{K}(t\sigma) < \mathcal{L}_{S/G}(t\sigma) \leq \mathcal{L}_{S/G}(t) \leq \mathcal{K}(t)$. Therefore, by (9), $\Xi(t, \sigma) = \mathcal{K}(t\sigma) < \mathcal{K}(t)$. Since $t \in \mathtt{B}(\epsilon)$, by (8), we know that $S(\epsilon)(\sigma) \leq \Xi(t, \sigma) = \mathcal{K}(t\sigma)$. Overall, we have that $\mathcal{L}_{S/G}(t\sigma) = \mathcal{L}_G(t\sigma) \wedge \mathcal{L}_{S/G}(t) \wedge S(\epsilon)(\sigma) \leq \mathcal{L}_G(t\sigma) \wedge \mathcal{L}_{S/G}(t) \wedge \mathcal{K}(t\sigma) = \mathcal{K}(t\sigma)$. However, this contradicts to $\mathcal{K}(t\sigma) < \mathcal{L}_{S/G}(t\sigma)$.

Induction Hypothesis: We assume that 1) and 2) hold for string s such that $|P(s)| = k$.

Induction Step: To proceed, we consider two cases for string s such that $|P(s)| = k + 1$.

Case 1: String s ends up with an observable event.

We write $s = t\sigma$, where $\sigma \in \Sigma_o$ and $|P(t)| = k$. By the induction hypothesis, we know that $t \in \text{supp}[\mathcal{K}]$ and $t \in \mathtt{B}(P(t))$. By condition C2, $S(P(t))$ is chosen such that $\mathtt{B}(P(t))(\Sigma_o \cap Dec) \cap \text{supp}[\mathcal{L}_G] \subseteq \text{supp}[\mathcal{K}]$, which implies that $t\sigma \in \text{supp}[\mathcal{K}]$, i.e., 1) holds. For 2), if $\sigma \in \Sigma_{uc}$, by the assumption that $\mathcal{K}$ is fuzzy controllable, we know that $\mathcal{L}_{S/G}(t\sigma) \leq \mathcal{K}(t\sigma)$. If $\sigma \in \Sigma_c$, then we still assume that $\mathcal{K}(t\sigma) < \mathcal{L}_{S/G}(t\sigma)$ for the sake of contradiction. Since $t \in \mathtt{B}(P(t))$, $\sigma \in \text{supp}[S(P(t))]$ and the assumption that $\mathcal{K}(t\sigma) < \mathcal{L}_{S/G}(t\sigma)$, by (8), $\Xi(t, \sigma) = \mathcal{K}(t\sigma) < \mathcal{K}(t)$, which means that $S(P(t))(\sigma) \leq \Xi(t, \sigma) = \mathcal{K}(t\sigma)$. Similar to the induction basis, we have that $\mathcal{L}_{S/G}(t\sigma) = \mathcal{L}_G(t\sigma) \wedge \mathcal{L}_{S/G}(t) \wedge S(P(t))(\sigma) \leq \mathcal{L}_G(t\sigma) \wedge \mathcal{L}_{S/G}(t) \wedge \mathcal{K}(t\sigma) = \mathcal{K}(t\sigma)$. This contradicts to $\mathcal{K}(t\sigma) < \mathcal{L}_{S/G}(t\sigma)$.

Case 2: String s ends up with an unobservable event.

We write $s = t\sigma_1 w \sigma_2$, where $\sigma_1 \in \Sigma_o, w \in \Sigma_{uo}^*, \sigma_2 \in \Sigma_{uo}$. By the induction hypothesis, $t \in \text{supp}[\mathcal{K}]$. Moreover, $t\sigma_1 \in \hat{\mathtt{B}}(P(t)\sigma_1)$ and $t\sigma_1 w\sigma_2 \in \mathtt{B}(P(t)\sigma_1)$. By condition C2, control decision $S(P(t)\sigma_1)$ is chosen such that $\mathtt{B}(P(t)\sigma_1) \subseteq \text{supp}[\mathcal{K}]$, which implies that $t\sigma_1 w\sigma_2 \in \text{supp}[\mathcal{K}]$, i.e., 1) holds. Next, we still show that 2) holds by contradiction. We assume that $\mathcal{K}(t\sigma) < \mathcal{L}_{S/G}(t\sigma)$. By the induction hypothesis, $\mathcal{L}_{S/G}(t) \leq \mathcal{K}(t)$. Moreover, we assume w.l.o.g. that $\forall v \in \overline{t\sigma_1 w} : \mathcal{L}_{S/G}(v) \leq \mathcal{K}(v)$; otherwise, it either suffices to choose

the shortest prefix of $t\sigma_1 w$ such that this assumption holds or reduces to Case 1. If $\sigma_2 \in \Sigma_{uc}$, by the assumption that $\mathcal{K}$ is fuzzy controllable, we know that $\mathcal{L}_{S/G}(t\sigma_1 w\sigma_2) \leq \mathcal{K}(t\sigma_1 w\sigma_2)$, which yields a contradiction immediately. If $\sigma_2 \in \Sigma_c$, then we still follow the same strategy we used in the induction basis and Case 1. First, $t\sigma_1 w \in \mathtt{B}(P(t)\sigma_1)$ and $\sigma_2 \in \text{supp}[S(P(t)\sigma_1)]$. Since $\mathcal{K}(t\sigma_1 w\sigma_2) < \mathcal{L}_{S/G}(t\sigma_1 w\sigma_2)$, by (8), $\Xi(t\sigma_1 w, \sigma_2) = \mathcal{K}(t\sigma_1 w\sigma_2) < \mathcal{K}(t\sigma_1 w)$, which means that $S(P(t)\sigma_1)(\sigma_2) \leq \Xi(t\sigma_1 w, \sigma_2) = \mathcal{K}(t\sigma_1 w\sigma_2)$. Therefore, $\mathcal{L}_{S/G}(t\sigma_1 w\sigma_2) = \mathcal{L}_G(t\sigma_1 w\sigma_2) \wedge \mathcal{L}_{S/G}(t\sigma_1 w) \wedge S(P(t)\sigma_1)(\sigma_2) \leq \mathcal{L}_G(t\sigma_1 w\sigma_2) \wedge \mathcal{L}_{S/G}(t\sigma_1 w) \wedge \mathcal{K}(t\sigma_1 w\sigma_2) = \mathcal{K}(t\sigma_1 w\sigma_2)$. This contradicts to $\mathcal{K}(t\sigma_1 w\sigma_2) < \mathcal{L}_{S/G}(t\sigma_1 w\sigma_2)$. ■

Next, we show Algorithm ONLINE-SYNTHESIS is sound.

Lemma IV.2: Algorithm ONLINE-SYNTHESIS always effectively synthesizes a supervisor under the assumption that $\mathcal{K}$ is fuzzy controllable.

Proof: To prove this, it suffices to show that, for each belief $\hat{\mathtt{B}}(\alpha)$ encountered in line 5 in the algorithm, there always exists a crisp set $Dec = \text{supp}[S(\alpha)]$ satisfying conditions C1–C3. We show this by induction on the length of α.

Induction Basis: Initially, we have $\hat{\mathtt{B}}(\epsilon) = \{\epsilon\}$. Let us consider $Dec = \Sigma_{uc}$. Clearly, it satisfies C1. Since $\mathcal{K}$ is fuzzy controllable, we know that $\{\epsilon\}\Sigma_{uc}^* \cap \text{supp}[\mathcal{L}_G] \subseteq \text{supp}[\mathcal{K}]$, i.e., $Dec = \Sigma_{uc}$ also satisfies C2. Since 2^Σ is finite, there must exist a control decision $\text{supp}[S(\epsilon)] \in 2^\Sigma$ satisfying C1–C3.

Induction Hypothesis: We assume that $\forall \alpha \in \Sigma_o^* : |\alpha| = k$, the control decision Dec is well defined for $\hat{\mathtt{B}}(\alpha)$.

Induction Step: Let us consider string $\alpha\sigma \in \Sigma_o^*$, where $\sigma \in \Sigma_o$ and $|\alpha| = k$. By the induction hypothesis, all control decisions up to $S(\alpha)$ are well defined. Therefore, we can effectively compute $\mathtt{B}(\alpha)$ and $\hat{\mathtt{B}}(\alpha\sigma)$. Moreover, by the proof of Lemma IV.1, we know that $\hat{\mathtt{B}}(\alpha\sigma) \subseteq \text{supp}[\mathcal{K}]$. Let us still consider $Dec = \Sigma_{uc}$ as the support of $S(\alpha\sigma)$, which satisfies C1. Since we have assumed that $\mathcal{K}$ is fuzzy controllable, we know that $\hat{\mathtt{B}}(\alpha\sigma)\Sigma_{uc}^* \cap \text{supp}[\mathcal{L}_G] \subseteq \text{supp}[\mathcal{K}]$. Therefore, $Dec = \Sigma_{uc}$ also satisfies C2. Still, there are only finite control decisions in 2^Σ that are strictly larger than Σ_{uc}. Therefore, there must exist a control decision $\text{supp}[S(\alpha\sigma)] \in 2^\Sigma$ satisfying C1–C3 for $\hat{\mathtt{B}}(\alpha\sigma)$. ■

Combing Lemma IV.1 and IV.2, we obtain the following result.

Theorem IV.1: Algorithm ONLINE-SYNTHESIS is both sound and complete, i.e., it effectively solves the fuzzy supervisor synthesis problem.

So far, we have shown that Algorithm ONLINE-SYNTHESIS is sound and complete. Another relevant question in the supervisory control theory is what is the permissiveness of the supervisor. Note that, unlike the fully observed case, in general, there does not exist a supremal supervisor for the partially observed case. Next, we prove an important feature of the synthesized supervisor. We show that the closed-loop fuzzy language is *locally maximal* in terms of its support.

Lemma IV.3: Let G be the system, $\mathcal{K}$ be the specification, and S be the fuzzy supervisor synthesized by Algorithm ONLINE-SYNTHESIS. Then, there does not exist another safe fuzzy supervisor $S' : P(\text{supp}[\mathcal{L}_G]) \to \mathcal{F}(\Sigma)$ such that $\mathcal{L}_{S'/G} \subseteq \mathcal{K}$ and $\text{supp}[\mathcal{L}_{S/G}] \subset \text{supp}[\mathcal{L}_{S'/G}]$.

Proof: By contradiction. We assume that there exists a safe fuzzy supervisor S' such that $\text{supp}[\mathcal{L}_{S/G}] \subset \text{supp}[\mathcal{L}_{S'/G}]$. This implies that there exists a string $s \in \text{supp}[\mathcal{L}_G]$ such that $\forall \alpha \in \overline{P(s)} \setminus \{P(s)\} : \text{supp}[S(\alpha)] = \text{supp}[S'(\alpha)]$; and $\text{supp}[S(P(s))] \subset \text{supp}[S'(P(s))]$. Since all decisions made by S and S' up to $P(s)$ are the same, the belief $\hat{\mathtt{B}}(P(s))$ are the same under both S and S'. Note that $\text{supp}[S(P(s))]$ is chosen such that C1–C3 hold for $\hat{\mathtt{B}}(P(s))$. Since $\text{supp}[S(P(s))] \subset \text{supp}[S'(P(s))]$, $\text{supp}[S'(P(s))]$ also satisfies C1. Since $\mathcal{L}_{S'/G} \subseteq \mathcal{K}$ and $\mathtt{B}(P(s))(\Sigma_o \cap \text{supp}[S'(P(s))]) \cap \text{supp}[\mathcal{L}_G] \subseteq \mathcal{L}_{S'/G}$, taking $S'(P(s))$ at $\hat{\mathtt{B}}(P(s))$ also satisfies C2. Recall that $\text{supp}[S'(P(s))]$ strictly contains $\text{supp}[S(P(s))]$. This contracts to C3. ■

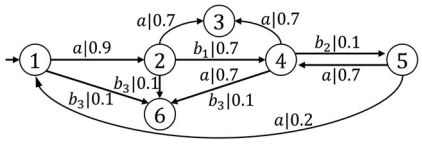

Fig. 2. FEDS G for the treatment process.

We have shown that the synthesis algorithm is not only sound and complete, but the solution is also locally maximal in terms of its support. One may also ask whether or not it is also locally maximal in terms of the fuzzy language, i.e., there does not exist another supervisor S' such that $\mathcal{L}_{S'/G} \subseteq \mathcal{K}$ and $\mathcal{L}_{S/G} \subset \mathcal{L}_{S'/G}$. Unfortunately, the synthesized supervisor needs not be locally maximal in terms of the fuzzy language in general. This is illustrated by the following example.

Example IV.1: Let us consider system language $\mathcal{L}_G = \frac{1}{\epsilon} + \frac{1}{\sigma_1} + \frac{1}{\sigma_1\sigma_2} + \frac{1}{\sigma_1\sigma_2\sigma_1}$ and specification language $\mathcal{K} = \frac{1}{\epsilon} + \frac{0.3}{\sigma_1} + \frac{0.2}{\sigma_1\sigma_2} + \frac{0.1}{\sigma_1\sigma_2\sigma_1}$, where we have $\Sigma_c = \Sigma_{uo} = \{\sigma_1, \sigma_2\}$, i.e., all events are controllable but unobservable. Then, the supervisor just needs to make one control decision $S(\epsilon)$ at the very beginning. According to conditions C1–C3, we have $\text{supp}[S(\epsilon)] = \{\sigma_1, \sigma_3\}$. Since $\Xi(\epsilon, \sigma_1) = 0.3, \Xi(\sigma_1\sigma_2, \sigma_1) = 0.1$ and $\Xi(\sigma_1, \sigma_2) = 0.2$, we have that $S(\epsilon)(\sigma_1) = 0.3 \wedge 0.1 = 0.1$ and $S(\epsilon)(\sigma_2) = 0.2$ and the closed-loop fuzzy language is $\mathcal{L}_{S/G} = \frac{1}{\epsilon} + \frac{0.1}{\sigma_1} + \frac{0.1}{\sigma_1\sigma_2} + \frac{0.1}{\sigma_1\sigma_2\sigma_1}$. However, we can design another fuzzy supervisor S' defined by $S'(\epsilon)(\sigma_1) = 0.3$ and $S(\epsilon)(\sigma_2) = 0.1$. In this case, we obtain a closed-loop language $\mathcal{L}_{S'/G} = \frac{1}{\epsilon} + \frac{0.3}{\sigma_1} + \frac{0.1}{\sigma_1\sigma_2} + \frac{0.1}{\sigma_1\sigma_2\sigma_1}$, which is strictly more permissive than the synthesized one.

Remark IV.1: We explain why the synthesized supervisor may not be maximal in terms of the fuzzy language. In (8), we choose the degree of membership for each event based on $\mathcal{K}$. Although this guarantees the soundness and the completeness of the algorithm, it may be conservative, i.e., the optimality of the algorithm may be affected. Particularly, in the above example, the membership degree of $\sigma_1\sigma_2\sigma_1$ is changed in the closed-loop language if we decide to enable σ_2 with degree 0.1. In this case, if we enable σ_2 with degree 0.1, then there is no need to restrict σ_1 with degree 0.1 since the safety for the second σ_1 has already been taken care by its predecessor event σ_2. In other words, the best choice of membership degree for an event may depend on the choice for other event. How to resolve this issue and improve the permissiveness of the closed-loop fuzzy language is an interesting future direction.

V. ILLUSTRATIVE EXAMPLE

In this section, we illustrate the effectiveness of proposed algorithm by an illustrative example. Specifically, we adopt the medical treatment example from [21], [22]. In principle, our result can also be applied to other applications, e.g., robot motion planning [9]–[12].

Suppose that a patient is infected by a disease and a physician wants to conduct a treatment. Based on the physician's experience, it is known that the patient's condition during the treatment can be represented by fuzzy automaton G shown in Fig. 2. Specifically, event a denotes a stage-by-stage therapy and events b_1, b_2, and b_3 denote three possible anaphylaxis, respectively. Also, state 1 denotes the patient's initial condition, state 2 denotes a fair condition, state 3 denotes a good condition, states 4 and 5 denote two bad conditions with different levels of anaphylaxis, and state 6 denotes a seriously bad condition.

Suppose that anaphylaxis represented by event b_3 can be avoided by using other medicine, but anaphylaxis represented by events b_1 and b_2 cannot be controlled. On the other hand, only anaphylaxis b_2 can be directly monitored by the physician. Therefore, $\Sigma_c = \{a, b_2\}$ and $\Sigma_o = \{a, b_1\}$.

Let us consider the following requirements for the treatment. First, we want that the patient should never be in a seriously bad condition with any degree, i.e., anaphylaxis b_3 should never occur. Second, once anaphylaxis b_1 occurs, the degree of the therapy should be decreased. Finally, the treatment should be stopped once anaphylaxis b_2 occurs. These three requirements can be formally described by the following specification fuzzy language $\mathcal{K} = \frac{1}{\epsilon} + \frac{0.9}{a} + \frac{0.7}{aa} + \frac{0.7}{ab_1} + \frac{0.3}{ab_1a} + \frac{0.1}{ab_1b_2}$. With parameters G, $\mathcal{K}$, Σ_c, and Σ_o, the decision process of the treatment is formulated as a fuzzy supervisor synthesis problem and our goal is to synthesize a safe fuzzy supervisor.

Note that this problem cannot be handled by any existing approach in the literature due to the following reasons. First, the specification language $\mathcal{K}$ is not fuzzy observable, since for $a, ab_1 \in \text{supp}[\mathcal{K}]$ and $a \in \Sigma$, we have $P(a) = P(ab_1)$ but we cannot find a number $x \in [0, 1]$ such that $\mathcal{K}(aa) = \mathcal{K}(a) \wedge \mathcal{L}_G(aa) \wedge x$ and $\mathcal{K}(ab_1a) = \mathcal{K}(ab_1) \wedge \mathcal{L}_G(ab_1a) \wedge x$, i.e., $0.7 \wedge x = 0.3 \wedge x = 0.7$ cannot be satisfied. Second, the supremal fuzzy controllable and normal sublanguage of $\mathcal{K}$ is the empty fuzzy set $\mathcal{O}$. This can be seen from the fact that supremal fuzzy controllable and normal supervisor does not allow the disablement of controllable but unobservable event. However, in our example, event $b_3 \in \Sigma_c \cap \Sigma_{uo}$ has to be disabled for safety purpose. Therefore, the supremal fuzzy normal approach [22] also fails to handle this synthesis problem.

However, by applying the synthesis algorithm proposed in Section III, we obtain fuzzy supervisor $S : P(\text{supp}[\mathcal{L}_G]) \to \mathcal{F}(\Sigma)$ defined as follows: $S(\epsilon) = \frac{0.9}{a} + \frac{1}{b_1} + \frac{1}{b_2}$, $S(a) = \frac{0.3}{a} + \frac{1}{b_1} + \frac{1}{b_2}$, $S(aa) = S(ab_2) = \frac{1}{b_1} + \frac{1}{b_2}$. This supervisor yields the following closed-loop fuzzy language $\mathcal{L}_{S/G} = \frac{1}{\epsilon} + \frac{0.9}{a} + \frac{0.3}{aa} + \frac{0.7}{ab_1} + \frac{0.3}{ab_1a} + \frac{0.1}{ab_1b_2}$, which is a sublanguage of $\mathcal{K}$. Therefore, the supervisor synthesis problem is solved.

VI. Conclusion

In this paper, we proposed a new approach for synthesizing a safe fuzzy supervisor for partially observed FDES. The proposed algorithm can be implemented in an online manner by recursively computing the belief of the system. We showed that the proposed algorithm is both sound and complete in the sense that it always synthesizes a supervisor when one exists. Therefore, it effectively solves the supervisor synthesis problem for partially observed FDES.

Acknowledgment

The author would like to thank the anonymous reviewers for their useful comments on improving the paper.

References

[1] F. Lin and H. Ying, "Modeling and control of fuzzy discrete event systems," *IEEE Trans. Syst., Man, Cybern., Cybern.*, vol. 32, no. 4, pp. 408–415, Aug. 2002.

[2] F. Lin *et al.*, "Decision making in fuzzy discrete event systems," *Inf. Sci.*, vol. 177, no. 18, pp. 3749–3763, 2007.

[3] X. Du, H. Ying, and F. Lin, "Theory of extended fuzzy discrete-event systems for handling ranges of knowledge uncertainties and subjectivity," *IEEE Trans. Fuzzy Syst.*, vol. 17, no. 2, pp. 316–328, Apr. 2009.

[4] Y. Cao, G. Chen, and E. Kerre, "Bisimulations for fuzzy-transition systems," *IEEE Trans. Fuzzy Syst.*, vol. 19, no. 3, pp. 540–552, Jun. 2011.

[5] M. Nie and W. Tan, "Theory of generalized fuzzy discrete-event systems," *IEEE Trans. Fuzzy Syst.*, vol. 23, no. 1, pp. 98–110, Feb. 2015.

[6] W. Deng and D. Qiu, "Bifuzzy discrete event systems and their supervisory control theory," *IEEE Trans. Fuzzy Syst.*, vol. 23, no. 6, pp. 2107–2121, Dec. 2015.

[7] H. Ying *et al.*, "A fuzzy discrete event system approach to determining optimal HIV/AIDS treatment regimens," *IEEE Trans. Inf. Technol. Biomed.*, vol. 4, no. 10, pp. 663–676, Oct. 2006.

[8] H. Ying, "A self-learning fuzzy discrete event system for HIV/AIDS treatment regimen selection," *IEEE Trans. Syst., Man, Cybern., Cybern.*, vol. 37, no. 4, pp. 966–979, Aug. 2007.

[9] R. Huq, G. Mann, and R. Gosine, "Distributed fuzzy discrete event system for robotic sensory information processing," *Exp. Syst.*, vol. 23, no. 5, pp. 273–289, 2006.

[10] R. Huq, G. Mann, and R. Gosine, "Behavior-modulation technique in mobile robotics using fuzzy discrete event system," *IEEE Trans. Robot.*, vol. 22, no. 5, pp. 903–916, Oct. 2006.

[11] K. Schmidt and Y. Boutalis, "Fuzzy discrete event systems for multiobjective control: Framework and application to mobile robot navigation," *IEEE Trans. Fuzzy Syst.*, vol. 20, no. 5, pp. 910–922, Oct. 2012.

[12] R. Liu, Y.-X. Wang, and L. Zhang, "An FDES-based shared control method for asynchronous brain-actuated robot," *IEEE Trans. Cybern.*, vol. 46, no. 6, pp. 1452–1462, Jun. 2016.

[13] E. Kilic, "Diagnosability of fuzzy discrete event systems," *Inf. Sci.*, vol. 178, no. 3, pp. 858–870, 2008.

[14] F. Liu and D. Qiu, "Diagnosability of fuzzy discrete-event systems: A fuzzy approach," *IEEE Trans. Fuzzy Syst.*, vol. 17, no. 2, pp. 372–384, Apr. 2009.

[15] M. Luo, Y. Li, F. Sun, and H. Liu, "A new algorithm for testing diagnosability of fuzzy discrete event systems," *Inf. Sci.*, vol. 185, no. 1, pp. 100–113, 2012.

[16] F. Liu, "Safe diagnosability of fuzzy discrete-event systems and a polynomial-time verification," *IEEE Trans. Fuzzy Syst.*, vol. 23, no. 5, pp. 1534–1544, Oct. 2015.

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

[18] F. Lin and W. Wonham, "On observability of discrete-event systems," *Inf. Sci.*, vol. 44, no. 3, pp. 173–198, 1988.

[19] R. Cieslak, C. Desclaux, A. Fawaz, and P. Varaiya, "Supervisory control of discrete-event processes with partial observations," *IEEE Trans. Autom. Control*, vol. AC-33, no. 3, pp. 249–260, Mar. 1988.

[20] D. Qiu, "Supervisory control of fuzzy discrete event systems: A formal approach," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 35, no. 1, pp. 72–88, Feb. 2005.

[21] Y. Cao and M. Ying, "Supervisory control of fuzzy discrete event systems," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 35, no. 2, pp. 366–371, Feb. 2005.

[22] Y. Cao and M. Ying, "Observability and decentralized control of fuzzy discrete-event systems," *IEEE Trans. Fuzzy Syst.*, vol. 14, no. 2, pp. 202–216, Apr. 2006.

[23] Y. Cao, M. Ying, and G. Chen, "State-based control of fuzzy discrete-event systems," *IEEE Trans. Syst., Man, Cybern., Cybern.*, vol. 37, no. 2, pp. 410–424, Apr. 2007.

[24] F. Liu and D. Qiu, "Decentralized supervisory control of fuzzy discrete event systems," *Eur. J. Control*, vol. 14, no. 3, pp. 234–243, 2008.

[25] D. Qiu and F. Liu, "Fuzzy discrete-event systems under fuzzy observability and a test algorithm," *IEEE Trans. Fuzzy Syst.*, vol. 17, no. 3, pp. 578–589, Jun. 2009.

[26] F. Lin and H. Ying, "State-feedback control of fuzzy discrete-event systems," *IEEE Trans. Syst., Man, Cybern., Cybern.*, vol. 40, no. 3, pp. 951–956, Jun. 2010.

[27] F. Wang, Z. Feng, and P. Jiang, "Reliable decentralized supervisory control of fuzzy discrete event systems," *Fuzzy Sets Syst.*, vol. 161, no. 12, pp. 1657–1668, 2010.

[28] A. Jayasiri, G. Mann, and R. Gosine, "Generalizing the decentralized control of fuzzy discrete event systems," *IEEE Trans. Fuzzy Syst.*, vol. 20, no. 4, pp. 699–714, Aug. 2012.

[29] A. Jayasiri, G. Mann, and R. Gosine, "Modular supervisory control and hierarchical supervisory control of fuzzy discrete-event systems," *IEEE Trans. Autom. Sci. Eng.*, vol. 9, no. 2, pp. 353–364, Apr. 2012.

[30] H. Xing, Q. Zhang, and K. Huang, "Analysis and control of fuzzy discrete event systems using bisimulation equivalence," *Theor. Comput. Sci.*, vol. 456, pp. 100–111, 2012.

[31] W. Deng and D. Qiu, "Supervisory control of fuzzy discrete-event systems for simulation equivalence," *IEEE Trans. Fuzzy Syst.*, vol. 23, no. 1, pp. 178–192, Feb. 2015.

[32] N. Ben Hadj-Alouane, S. Lafortune, and F. Lin, "Centralized and distributed algorithms for on-line synthesis of maximal control policies under partial observation," *Discrete Event Dynamic Syst.: Theory Appl.*, vol. 6, no. 4, pp. 379–427, 1996.

[33] X. Yin and S. Lafortune, "Synthesis of maximally permissive supervisors for partially observed discrete event systems," *IEEE Trans. Autom. Control*, vol. 61, no. 5, pp. 1239–1254, May 2016.

[34] X. Yin and S. Lafortune, "A uniform approach for synthesizing property-enforcing supervisors for partially-observed discrete-event systems," *IEEE Trans. Autom. Control*, vol. 61, no. 8, pp. 2140–2154, Aug. 2016.

[35] X. Yin, "Supervisor synthesis for mealy automata with output functions: A model transformation approach," *IEEE Trans. Autom. Control*, 2016, doi: 10.1109/TAC.2016.2601118.

[36] C. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*, 2nd ed. New York, NY, USA: Springer, 2008.

[37] L. Zadeh, "Fuzzy sets," *Inf. Control*, vol. 8, no. 3, pp. 338–353, 1965.