Fault-Tolerant Control of Discrete-Event Systems with Controllability Failures

Arun Raman and R. S. Sreenivas

***Abstract*— A *supervisory policy* controls a *Discrete-Event System* (DES) by appropriately disabling a subset of events, known as *controllable events*, based on the observed event string generated by the supervised DES thus far. We consider supervisory control of DES in the presence of an extraneous *fault* that renders an arbitrary subset of controllable events to be temporarily uncontrollable. The fault is detected at the first occurrence of a controllable event that was disabled by the supervisor. It is rectified after finitely-many such unintended occurrences of controllable events following which the supervisor regains control of all controllable events and can prevent them from occurring when deemed necessary.**

We present a necessary and sufficient condition for the existence of a supervisor that enforces a desired language specification in the paradigm of Ramadge and Wonham [1], under the fault semantics described above. We also prove that such a supervisor, if it exists, can always be synthesized if the language of the plant and the specification is *regular*.

I. Introduction

A *Discrete Event System* (DES) is a discrete-state system, where the state changes at discrete-time instants due to the occurrence of events. Queuing, manufacturing and communication systems etc. are some examples of DES. If a DES does not follow a desired behaviour, then it is of interest to design supervisory policies that constrain the DES to the desired behaviour, if possible. A *supervisory policy* controls a DES by disabling a subset of events known as *controllable events*. The supervisory policy can only disable a controllable event, it cannot force an event to occur.

Oftentimes the behaviour of the system is degraded due to the occurrence of faults. Faults can occur in different parts of the system and can be of different types. The most common way of modeling a fault is as an uncontrollable event which results in a different plant behaviour than expected. Faults have also been looked at in the context of uncertain system models [2] and changes in observability [3]. Fault-tolerance in DES modeled by Petri Nets (PNs) [4] has largely been explored in the context of unreliable resources (tokens) ([5], [6], [7], [8], [9]). Faults in supervisor-structure modeled by PNs– in places and transitions, is considered in [10].

In this paper, we consider DES modeled by an automaton and faults in which a subset of controllable events becomes temporarily uncontrollable at an arbitrary discrete-time instant. In other words, faults which lead to *change in controllability* of the DES. This could be due to a device- or line-fault, where communication between supervisor and plant is temporarily unavailable; or due to the activity of a malicious-user. The fault is an extraneous event and is not modeled by the DES. An arbitrary subset of controllable events becoming uncontrollable is the consequence of the fault.

The objective of a fault-tolerant controller is to ensure that a certain specification is fulfilled despite the faulty behaviour by the system. Reference [11] presents a survey of fault-tolerant control methods for DES. There are two approaches for designing fault tolerant supervisors – *passive* and *active*. In the passive approach, a single supervisor is used to enforce the specification before and after the occurrence of faults. An active fault tolerant supervisor changes the supervision-parameters or supervision-structure based on the occurrence of faults. It consists, generally, of a diagnoser that detects, and identifies the fault and other necessary information based on which the supervisor adapts to the changed plant while still enforcing the required specification.

In this paper, we consider active fault tolerant controllers. The fault-event is not directly observed but inferred from the first occurrence of an event that was originally disabled by the supervisor. A fault can occur any time, and can be rectified only after the detection of k_r-many unintended occurrences of DES-events affected by the fault. The controller adapts depending on the number of detected occurrences of controllable events that became uncontrollable due to the occurrence of the fault-event. We present a necessary and sufficient condition for the existence of a supervisor tolerant to such faults for a given value of k_r. We also prove that such a supervisor can always be synthesized, if it exists, if the language of the plant and the specification is *regular*.

The rest of the paper is organized as follows. Section II presents the notations and definitions that we use in the paper. It also formally introduces the fault semantics and its model. The main results are discussed in Section III. We illustrate the results with an example in Section IV. We conclude with suggestions for future research in Section V.

II. Notations and Definitions

A. DES Model

We model the DES by an automaton $G = (Q, \Sigma, \delta, q_0, Q_m)$ [1], [12]. Here Q and Σ denote the set of *states* and *events* associated with DES G respectively. The *initial state* of the DES is $q_0 \in Q$, and $Q_m \subseteq Q$ is called the set of *marked states*. The function $\delta : \Sigma \times Q \to Q$ is the *state transition function*, where for $q \in Q, \sigma \in \Sigma, \delta(\sigma, q) = \widehat{q}$,

This work was supported in part by the Arthur Davis Faculty Scholar Endowment at the University of Illinois at Urbana-Champaign.

The authors are with the Department of Industrial and Enterprise Systems Engineering and Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, Illinois, USA. raman12{rsree}@illinois.edu

implies there is an event labeled σ from state q to state $\widehat{q}$. In general, the state transition function δ is a partial function. The *active event function* $\Gamma : Q \to 2^{\Sigma}$, identifies the set of all events $\sigma \in \Sigma$ for which $\delta(\sigma, q)$ is defined.

Σ^* denotes the set of all finite strings of elements of Σ including the empty string ϵ. We write $\delta^*(\omega, q) = \widehat{q}$ if there is a string of events $\omega \in \Sigma^*$ from state q to state $\widehat{q}$. For a given DES G, the set of all admissible strings of events is denoted by $L(G)$ and is referred to as the language generated by the DES G over the alphabet Σ. Formally

$$L(G) = \{\omega \in \Sigma^* : \delta^*(\omega, q_0) \text{ is defined}\} \tag{1}$$

The marked language, $L_m(G) \subseteq L(G)$, generated by a DES G is

$$L_m(G) = \{\omega \in \Sigma^* : \delta^*(\omega, q_0) \in Q_m\}$$

A string u is a *prefix* of a string $v \in \Sigma^*$ if for some $w \in \Sigma^*$ $v = uw$. If v is an admissible string in G, then so are all its prefixes. We define the *prefix closure* of $L \subseteq \Sigma^*$ to be the language $\overline{L}$ defined as:

$$\overline{L} = \{u : uv \in L \text{ for some } v \in \Sigma^*\}$$

If $L = \overline{L}$, then we say that L is *prefix closed*. From equation 1, it follows that $L(G)$ is prefix-closed. Additionally, $\overline{L_m}(G) \subseteq L(G)$ in general; if $\overline{L_m}(G) = L(G)$, we say G is *non-blocking*. If G is non-blocking, all strings in $L(G)$ can be extended to a string in $L_m(G)$. We only consider non-blocking DES in this paper.

An automaton is represented by a directed graph whose nodes correspond to the states of the automaton, with an edge from q to $\widehat{q}$ labeled σ for each triple $(\sigma, q, \widehat{q})$ such that $\widehat{q} = \delta(\sigma, q)$. There is a path from every node to a marked node in the graphical representation (state transition diagram) of a non-blocking DES.

Given two automata $G_1 = (Q_1, \Sigma_1, \delta_1, q_{01})$ and $G_2 = (Q_2, \Sigma_2, \delta_2, q_{02})$, we say that G_1 is a *subautomaton* of G_2, denoted by $G_1 \sqsubseteq G_2$, if:

$$h \cdot \delta_1(s, q_{01}) = \delta_2(s, q_{02}) \quad \forall s \in L(G_1)$$

where $h : Q_1 \to Q_2$ is a unique function mapping the states of Q_1 to Q_2. Note that this condition implies that $Q_1 \subseteq Q_2, q_{01} = q_{02}$, and $L(G_1) \subseteq L(G_2)$. This definition also implies that the state transition diagram of G_1 is a subgraph of that of G_2 [12].

B. Supervisory Control

The set of events (Σ) is partitioned into the set of *controllable* events (Σ_c) and *uncontrollable* events (Σ_u). More specifically, Σ_c (resp. Σ_u) denotes the set of events that can (resp. cannot) be disabled by the supervisor. The supervisor controls the behavior of G by appropriately disabling controllable events, based on the observed admissible behavior of the supervised system.

Formally, a supervisor $S : L(G) \to 2^{\Sigma}$ is a function from the language generated by G to the power set of Σ. $S(\omega)$ is the set of events enabled by the supervisor after a string $\omega \in L(G)$ is observed by the supervisor. An event $\sigma \in \Gamma(\delta(\omega, q_0))$ can occur in the supervised system only if $\sigma \in S(\omega)$. The supervisor cannot disable any event in Σ_u, and consequently $\Sigma_u \cap \Gamma(\delta(\omega, q_0)) \subseteq S(\omega)$ for any $\omega \in L(G)$. That is, the uncontrollable events are always enabled by the supervisor. We use S/G to denote the controlled DES (CDES) in which supervisor S is controlling the DES G. The language generated by the CDES can be recursively defined as follows:

1) $\epsilon \in L(S/G)$
2) $((\omega \in L(S/G)) \wedge (\omega\sigma \in L(G)) \wedge (\sigma \in S(\omega))) \Leftrightarrow (\omega\sigma \in L(S/G))$

The marked language generated by the CDES is $L_m(S/G) = L(S/G) \cap L_m(G)$. Next, we discuss behaviors that can be achieved by supervision for a non-blocking DES G.

Let $K \subseteq L_m(G) (K \neq \emptyset)$ be a non-empty supervisor specification. There is a supervisor S such that $L(S/G) = \overline{K}$ and $L_m(S/G) = K$ if and only if (i) $\overline{K}\Sigma_u \cap L(G) \subseteq \overline{K}$, and (ii) $\overline{K} \cap L_m(S/G) = K$ (cf. Proposition 4.1, [1]). We only consider non-empty specifications in this paper (i.e. $K \neq \emptyset$).

Condition (i) listed above is the *controllability condition*. If this condition is not satisfied, it is of interest to find the *supremal controllable sublanguage* of K,

$$K^{\uparrow} = \bigcup_i \{K_i : (K_i \subseteq K) \wedge (\overline{K}_i\Sigma_u \cap L(G) \subseteq \overline{K}_i)\}. \tag{2}$$

Condition (ii) is the *non-blocking condition*.

Let $\mathcal{L}$ be the set of all languages over Σ. Following reference [13], we define an operator $\Omega : \mathcal{L} \times \Sigma \to \mathcal{L}$ for a fixpoint characterization of $K^{\uparrow}$

$$\Omega(K^i, \Sigma_u) = \{\omega : \omega \in K \text{ and } \overline{\omega}\Sigma_u \cap L(G) \subseteq \overline{K}^i\} \tag{3}$$

By Proposition 2.1 in [13] we have $K^{\uparrow} = \Omega(K^{\uparrow}, \Sigma_u)$ and $K^{\uparrow} \supseteq \widehat{K}$ for every $\widehat{K}$ such that $\widehat{K} = \Omega(\widehat{K}, \Sigma_u)$. We can define a sequence to compute $K^{\uparrow}$ by iterating over Ω

$$\begin{aligned} K^0 &= K \\ K^{j+1} &= \Omega(K^j, \Sigma_u) \end{aligned} \tag{4}$$

We recall the definition of *Nerode Equivalence* before moving to Theorem 1. Let $L \subseteq \Sigma^*$. Two strings $s, t \in \Sigma^*$ are equivalent (mod L), written as $s \equiv_L t$, if the set of suffixes of s and t to form strings in L are the same. That is, $s \equiv_L t$ if

$$\{s' : s' \in \Sigma^* \text{ and } ss' \in L\} = \{t' : t' \in \Sigma^* \text{ and } tt' \in L\}$$

The language is regular if the number of equivalence classes $\equiv_L$ in Σ^* is finite. Proposition 2.2 and Theorem 3.1 in [13] state that:

Theorem 1: The set-theoretic limit

$$M = \lim_{j \to \infty} K^j$$

exists and $K^{\uparrow} \subseteq M$. If K and $L(G)$ are regular, then the sequence of languages K^j defined by Equation 4 converges after a finite number of terms to $K^{\uparrow}$. Furthermore, $K^{\uparrow}$ is also a regular language.

We focus our attention on the realization of the supremal controllable sublanguage. Let

$$Q_{ac} = \{q : q \in Q \text{ and } \delta(\omega, q_0) = q \text{ for some } \omega \in \Sigma^*\}$$
$$Q_{co} = \{q : q \in Q \text{ and } \delta(\omega, q) \in Q_m \text{ for some } \omega \in \Sigma^*\}$$

An automaton is *accessible* if $Q_{ac} = Q$, *coaccessible* if $Q_{co} = Q$, and *trim* if $Q_{ac} = Q_{co} = Q$. Let $Q_{tr} = Q_{ac} \cap Q_{co}$, and $\delta_{tr}(\sigma, q) = \delta(\sigma, q)$ if $q \in Q_{tr}$, and is not defined otherwise. Then the trim component of the automaton G, if it is not empty, is given by $Tr(G) = (Q_{tr}, \Sigma, \delta_{tr}, q_0, Q_m \cap Q_{tr})$. $L(Tr(G)) = L(G)$ and $Tr(G)$ is effectively constructible [13]. All *Minimal Automata* (cf. section 4.4.3, [14]) are *trim*, but the converse is not necessarily true.

We say that an automaton G_1 refines G_2 if it is a sub-automaton of G_2. Consider an automaton $G = (Q, \Sigma, \delta, q_0, Q)$ with an associated active event function $\Gamma : Q \to 2^{\Sigma}$, and $G_j = (Q_j, \Sigma, \delta_j, q_{0j}, Q_{mj})$ such that $L(G_j) = K^j$ and G_j refines G. Lemma 6.1 of [13] proved that G_j can be *refined* to obtain an automaton $G_{j+1}(Q_{j+1}, \Sigma, \delta_{j+1}, q_{0(j+1)}, Q_{m(j+1)})$ such that $L(G_{j+1}) = \Omega(K^j, \Sigma_0)$ $(= K^{j+1})$ by removing those states of G_j that do not satisfy the *active event constraint*:

$$\Gamma(h(q_j)) \cap \Sigma_0 \subseteq \Gamma(q_j) \tag{5}$$

where $q_j \in Q_j$. Here, $h : Q_j \to Q$ is a unique function satisfying $h \cdot \delta_j(s, q_{0j}) = \delta(s, q_0)$. We refer the reader to [13] for an elaborate discussion of these results.

C. Fault Model

We use the term *fault-event*, denoted by ϕ, to refer to an extraneous discrete-event where an arbitrary subset of controllable events, $\Sigma_f \subseteq \Sigma_c$, becomes temporarily uncontrollable. We assume that the fault-event ϕ is followed by an extraneous *rectification-event*, denoted by ρ, but not necessarily immediately. After the rectification event, all events in $\Sigma_f \subseteq \Sigma_c$ become controllable again. Between the events ϕ and ρ, the set of uncontrollable (resp. controllable) events is effectively $\Sigma_f \cup \Sigma_u$ (resp. $\Sigma_c - \Sigma_f$). Before the event ϕ, and after the event ρ, the set of uncontrollable (resp. controllable) events is Σ_u (resp. Σ_c). The fault-event ϕ is not directly observed, its occurrence has to be inferred by the unintended occurrence of an event disabled by the supervisor. The fault- and rectification-event are assumed to be extraneous and are not modeled by the DES.

Due to cost considerations or system limitations, a fault may not be rectified immediately after detection. We quantify the tolerance of G to a fault-event by associating a positive integer k_r with it; where k_r is the mandatory number of unintended occurrences of disabled events that are to be detected before the fault is rectified. Under this semantics, the fault-free scenario is represented as the case when $k_r = 0$.

Suppose k_d-many ($k_d < k_r$) unintended occurrences of events affected by the fault were detected after $\omega \in \Sigma^*$ is generated by the CDES. The $(k_d + 1)$-th unintended occurrence of an event affected by the fault will be detected at $\delta(\omega, q_0)$ only if some event $\sigma \in \Gamma(\delta(\omega, q_0)) - S(\omega)$ occurs after ω. That is, an event σ occurred despite the supervisor disabling it. The rectification-event ρ occurs *immediately after* k_r-many unintended occurrences of disabled events are detected.

A supervisor in the presence of faults, $S_\phi : L(G) \times \{k_r, k_r - 1, \ldots, 0\} \to 2^{\Sigma}$, determines the set of events that are permitted to occur as a function of the observed event strings generated by the (supervised) plant, and the number of unintended occurrences of events that are to be observed before the rectification event occurs (i.e. the number of detections *to-go*). The uncontrollable events are always permitted by the supervisor for all values of the arguments.

The language generated by the CDES under the influence of faults after k_d-many unintended occurrences of disabled events have been detected (that is when there are $(k_r - k_d)$-many unintended occurrences of disabled events *to-go* before rectification), $L(S_\phi/G/k_r - k_d)$, can be recursively defined as follows:

1) $\epsilon \in L(S_\phi/G/k_r)$, that is ϵ is a member of the set of supervised behaviors when no faults have been detected,
2) $(\omega \in L(S_\phi/G/k+1)) \Rightarrow (\omega \in L(S_\phi/G/k))$ for $0 \leq k \leq k_r - 1$,
3) $((\omega \in L(S_\phi/G/k)) \wedge (\omega\sigma \in L(G)) \wedge (\sigma \in S_\phi(\omega, k))) \Leftrightarrow (\omega\sigma \in L(S_\phi/G/k))$ for $0 \leq k \leq k_r$, and
4) $((\omega \in L(S_\phi/G/k)) \wedge (\omega\sigma \in L(G)) \wedge (\sigma \notin S_\phi(\omega, k)) \wedge (\sigma \in \Sigma_f)) \Rightarrow (\omega\sigma \in L(S_\phi/G/k-1))$ for $1 \leq k \leq k_r$.

In Item 4 above, the number of unintended occurrences of disabled events is incremented (and detections to-go decremented from k to $k-1$) as a controllable event σ that was disabled by the supervisor occurred. The fault-free behaviour is analogous to the case when $k_r = 0$. The following observations are a direct consequence of the definition.

Observation 1: For every $\omega_1 \in L(S_\phi/G/k)$, $\exists\ \omega_2 \in L(S_\phi/G/k+1)$ such that $\omega_2 = pr(\omega_1)$, where $pr()$ denotes the prefix of the string argument.

Observation 2: $L(S_\phi/G/k) \subseteq L(S_\phi/G/k-1)\ \forall\ 0 < k_d \leq k_r$

The marked language generated by the CDES under the fault semantics is $L_m(S_\phi/G/k) = L(S_\phi/G/k) \cap L_m(G)$.

III. Main Results

Let $K \subseteq L_m(G) (K \neq \emptyset)$ be a non-empty, desired set of event strings to be generated under supervision under the fault semantics of the previous section. In subsequent discussion we assume K to be controllable with respect to $L(G)$. That is, $\overline{K}\Sigma_u \cap L(G) \subseteq \overline{K}$; this would be the case if we let $K \leftarrow K^{\uparrow}$ (cf. Equation 2). In this section, we are interested in behaviours that can be achieved by supervision for a DES G in the presence of faults.

Definition 1: A CDES G is said to be *compliant* to a specification $K (= K^{\uparrow}) \subseteq L_m(G)$, $K \neq \emptyset$, in the presence of controllability faults if:

1) $L(S_\phi/G/k) \subseteq \overline{K}$, $\forall k \in \{0, 1, \ldots, k_r\}$,
2) $L(S_\phi/G/0) = \overline{K}$.

That is, before the rectification event the CDES generates a subset of $\overline{K}$; and the CDES should be able to generate any string $\overline{K}$ after the rectification event.

In Theorem 2, we define a sequence of sets $K_0(=K) \supseteq K_1 \supseteq \ldots \supseteq K_{k_r}$ and show that $L(S_\phi/G/k) = \overline{K}_k$, $\forall k \in \{0, \ldots k_r\}$, is necessary and sufficient for the existence of a fault-tolerant supervisor compliant with the specification $K \subseteq L_m(G)(K \neq \emptyset)$.

Theorem 2: For a DES $G = (Q, \Sigma, \delta, \Gamma, q_0)$, where $\Sigma_u \subseteq \Sigma$ ($\Sigma_c = \Sigma - \Sigma_u$) denotes the set of uncontrollable (resp. controllable) events, there exists a supervisor S_ϕ, under the influence of fault-event ϕ, with tolerance k_r such that the CDES is compliant to a non-empty specification $K(= K^\uparrow) \subseteq L_m(G)$, if and only if there exists a sequence of languages $K_0 \supseteq K_1 \supseteq \ldots \supseteq K_{k_r}$, such that:

1) $K_0 = K$,
2) Each K_i is controllable with respect to $L(G)$. That is, $\overline{K}_i\Sigma_u \cap L(G) \subseteq \overline{K}_i$ $\forall i \in \{0, \ldots k_r\}$, and
3) $\overline{K}_j\Sigma_c \cap L(G) \subseteq \overline{K}_{j-1}$ $\forall j \in \{1, \ldots, k_r\}$.

Proof: (If) Let $S_\phi : L(G) \times \{k_r, k_r - 1, \ldots, 0\} \to 2^\Sigma$ be a supervisory policy defined as:

$$(\sigma \in S_\phi(\omega, k)) \Leftrightarrow (\omega\sigma \in \overline{K}_k) \quad \forall\ 0 \leq k \leq k_r$$

Using an induction argument on the number of unintended event occurrences *to-go*, we show that $L(S_\phi/G/k) = \overline{K}_k \cap L(G), \forall k_r \geq k \geq 0$. The base-case of induction is established by letting $k = k_r$, since K_r is controllable with respect to $L(G)$, we have $L(S_\phi/G/k_r) = \overline{K}_{k_r} (\subseteq \overline{K})$. As the induction hypothesis, let $L(S_\phi/G/k) = \overline{K}_k (\subseteq \overline{K})$, for some k where $k_r \geq k > 0$.

For the induction step, consider a string $\omega \in \overline{K}_k$. Let us suppose the next event that occurs under supervision is $\sigma \in \Sigma$. The number of unintended event occurrences *to-go* is updated from k to $k-1$ only if $\sigma \notin S_\phi(\omega, k)$. From item 3 above, and the fact that $\omega\sigma \in L(G)$, it follows that if $\sigma \notin S_\phi(\omega, k)$ then $\omega\sigma \in \overline{K}_{k-1} \Rightarrow L(S_\phi/G/k-1) = \overline{K}_{k-1} (\subseteq \overline{K})$. The implication is a result of the definition of $L(S_\phi/G/k)$ in Section II.

When $k = 0$, we have $L(S_\phi/G/0) = \overline{K}_0 (= \overline{K})$. By Observation 1, every string in $L(S_\phi/G/k)$ is a prefix of some string in $L(S_\phi/G/k-1)$ for $0 < k \leq k_r$. Therefore, every string in $L(S_\phi/G/k)$ for $0 < k \leq k_r$ can be extended to a string in $L(S_\phi/G/0) = \overline{K}_0$.

(Only If) We have

1) $L(S_\phi/G/k) \subseteq \overline{K})$, $\forall k \in \{0, 1, \ldots, k_r\}$,
2) $L(S_\phi/G/0) = \overline{K}$.

If no event in Σ_c is disabled by the supervisor S_ϕ, then let $K_0 = K_1 = \ldots = K_{k_r} = K$ and items 1, 2 and 3 in the theorem are trivially satisfied.

Suppose there exists an event that is disabled by the supervisor and consider the case when $(k_r - 1)$-many detection of events affected by fault have occurred (that is, one unintended occurrence to-go). Then by Observation 2, we have $L(S_\phi/G/1) \subseteq \overline{K}$. If $L(S_\phi/G/1) = K$ then the firing of a controllable event that is disabled by the supervisor will result in a string not in K. More specifically, $L(S_\phi/G/1)$ cannot have any string ω such that $\omega\Sigma_c \notin K$ and $L(S_\phi/G/1)$ must satisfy Item 3 of the theorem, that is, $L(S_\phi/G/1)\Sigma_c \subseteq L(S_\phi/G/0)$ $(= K)$. In addition, this behaviour of $L(S_\phi/G/1)$ should be controllable with respect to $L(G)$, which gives Item 2 of the theorem. The result follows through induction by replacing $L(S_\phi/G/0)$ by $L(S_\phi/G/1)$, and $L(S_\phi/G/1)$ by $L(S_\phi/G/2)$ in the above argument, and repeating, as often as necessary. ■

In the next result, we prove that if $K \subseteq L_m(G)$ is non-blocking with respect to $L_m(G)$, then K_i is also non-blocking with respect to $L_m(G)$ $\forall i \in \{0, 1, \ldots, k_r\}$.

Lemma 1: $\forall i \in \{0, 1, \ldots, k_r\}$: $(\overline{K} \cap L_m(G) = K) \Rightarrow (\overline{K}_i \cap L_m(G) = K_i)$.

Proof: Since $K_i \subseteq K \subseteq L_m(G)$ and $K_i \subseteq \overline{K}_i$ we have $K_i \subseteq (\overline{K}_i \cap L_m(G))$. Now, all strings in $\overline{K}_i$ satisfy Items 2 and 3 in Theorem 2 and $K_i \subseteq K$. Hence, every string in $\overline{K}_i$ that is also in K belongs to K_i. Therefore, $\overline{K}_i \cap K \subseteq K_i$ which implies $\overline{K}_i \cap K = K_i$ as $\overline{K}_i \cap K \supseteq K_i$ is trivially true.

$$\begin{aligned}
&(\omega \in \overline{K}_i \cap L_m(G)) \Leftrightarrow ((\omega \in \overline{K}_i) \wedge (\omega \in L_m(G))) \\
&\Rightarrow (\omega \in \overline{K} \cap L_m(G)) \wedge (\omega \in \overline{K}_i) \quad (\text{as } \overline{K}_i \subseteq \overline{K}) \\
&\Rightarrow (\omega \in K) \wedge (\omega \in \overline{K}_i) \quad (\text{as } \overline{K} \cap L_m(G) = K) \\
&\Rightarrow (\omega \in K_i) \qquad (\text{as } \overline{K_i} \cap K = K_i) \\
&\Rightarrow (\overline{K}_i \cap L_m(G)) \subseteq K_i
\end{aligned}$$

■

Next we discuss computation of the sets K_j. Given a specification $K(= K_0)$ that is controllable with respect to $L(G)$, item 3 in Theorem 2 requires the sublanguage K_1 to be such that the occurrence of a (single) controllable event following a string in $\overline{K}_1$ should result in a string in $\overline{K}_0$. Let $K_1^1 \subseteq K_1$ be the supremal sublanguage of K_0 that satisfies the specification in Item 3. We can use operator Ω with respect to K_0 and Σ_c to obtain a characterization of K_1^1. From Equation 3 and 4:

$$\begin{aligned}
K_{0,1} &= K \\
K_1^1 &= \Omega(K_{0,1}, \Sigma_c)
\end{aligned} \tag{6}$$

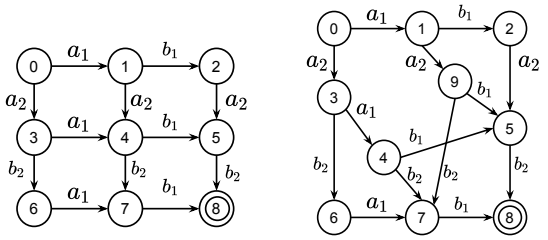
where,

$$\Omega(K_0, \Sigma_c) = \{t : t \in K_0 \text{ such that } \bar{t}\Sigma_c \cap L(G) \subseteq \overline{K}_0\}$$

The set K_1 is the largest controllable sublanguage of K_1^1, $(K_1^1)^\uparrow$, which can be obtained through the iterative procedure described by Equation 4. K_2 can be obtained by repeating the previous steps by replacing K_0 by K_1. That is, first finding the sublanguage K_2^1 as $\Omega(K_1, \Sigma_c)$ and then, computing the supremal controllable subset of K_2^1 which would give us K_2. Sets $K_3, \ldots, K_{k_r}$ can be calculated in a similar manner through recursion. That is, for $j \in \{0, \ldots k_r - 1\}$:

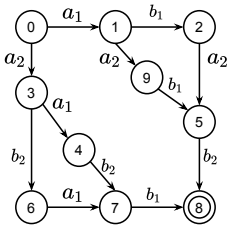
$$\begin{aligned}
K_0 &= K \\
K_{j+1}^1 &= \Omega(K_j, \Sigma_c) \\
K_{j+1} &= (K_{j+1}^1)^\uparrow
\end{aligned} \tag{7}$$

It was shown in Theorem 1 (Proposition 2.2 in [13]) that for arbitrary languages the characterization Ω provides

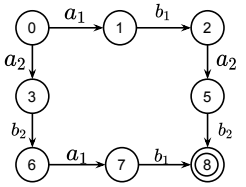


(a) Minimal Automaton, H, for the concurrency model

(b) Automaton G



(c) Automaton G_1 for the specification K_0



(d) Automaton for K_1

Fig. 1: Automatons for illustrating fault-tolerant control of the concurrency model

only an outer approximation to the supremal controllable sublanguage of a given language. Therefore, for arbitrary languages, the recursion described above will provide an outer approximation to the sets K_1 to K_{k_r}. However, from Theorem 1, for regular languages, the sequence of sets obtained through iterating over the operator Ω converge to the supremal controllable sublanguage of the given language, which is also regular. This gives the following result:

Theorem 3: If $L(G)$ and K are regular, then K_j will be regular $\forall j = 0, \ldots, k_r$.

Proof: Follows from Theorem 3.1 in [13]. ■

If a language is regular then it can be realized as a finite state automaton. If the automaton for K refines the automaton for G, then the finite automatons generating the languages $\{K_j\}_{j=0}^{k_r}$ can be obtained from the automaton of G by simply removing the states that do not satisfy the *active event constraint* given in Equation 5 (Lemma 6.1 of [13]). We get the following result:

Theorem 4: Let S_ϕ be a supervisor for a non-blocking DES G, under the influence of fault-event ϕ, with tolerance k_r such that the CDES is compliant to a specification $K (= K^\uparrow = \overline{K}) \neq \emptyset$. Let G_1 and G_2 be two automata such that $L(G_1) = L(S_\phi/G/k)$ and $L(G_2) = L(S_\phi/G/k-1)$. If the automaton for $L(S_\phi/G/0)$ refines the automaton for $L(G)$, then $G_1 \sqsubseteq G_2$.

In the next section, we illustrate the results using an example.

IV. Illustrative Example

We illustrate the problem and results through the example of a database concurrency control model (example 3.3 in [12]). Consider the following two transactions whose execution from the initial state 0 can be modeled by the (minimal) automaton shown in Figure 1a.

$$T_1 = a_1 b_1 \quad \text{and} \quad T_2 = a_2 b_2$$

Suppose all the events a_1, a_2, b_1 and b_2 are controllable and that the only admissible strings are those which take the system to a marked state and where:

$$(a_1 \text{ precedes } a_2) \Leftrightarrow (b_1 \text{ precedes } b_2).$$

The automaton G is obtained by splitting state 4 of H into states 4 and 9. The supervisory policy that disables event b_2 (b_1) at state 9 (resp. state 4) enforces the desired language specification. It can be verified that the automaton G_1 in Figure 1c is the CDES that generates the desired language specification. Theorem 4 requires the specification automaton to be a subautomaton of the plant automaton. For our purpose, we would consider G as the plant automaton.

Suppose the fault-event ϕ with $\Sigma_f = \{a_1, a_2, b_1, b_2\}$ occurs when the system is at state 0. Let $k_r = 1$, that is the rectification event happens after a single detection of events affected by ϕ.

If the system generates the string $a_1 b_1 a_2 b_2$ or $a_2 b_2 a_1 b_1$, then ϕ goes undetected because although b_1 and b_2 were affected by the fault, their occurrence in these strings was permitted by the supervisor.

Intuitively, we see that the fault-tolerant supervisor should prevent the system from reaching state 9 (state 4) because then the uncontrollable occurrence of b_2 (resp. b_1) will result in an inadmissible string. This observation can be formalized as follows. The set of strings generated by the CDES G_1 is $K = \{a_1 b_1 a_2 b_2, a_1 a_2 b_1 b_2, a_2 b_2 a_1 b_1, a_2 a_1 b_2 b_1\}$. $K_1^1 \subseteq K$ is the set of strings in K from where a single firing of controllable transitions will result a string in K. We have $a_1 b_1 \in \overline{K}$ and $a_1 b_1 b_2 \notin \overline{K}$. Similarly, $a_2 b_2 \in \overline{K}$ and $a_2 b_2 b_1 \notin \overline{K}$. Therefore, we get $K_1^1 = \{a_1 a_2 b_1 b_2, a_2 a_1 b_2 b_1\}$. Since all events were originally controllable, $K_1 = (K_1^1)^\uparrow = K_1^1$. The automaton for K_1 can be constructed from the automaton of K by removing the states that do not satisfy the active event constraint of Equation 5. h is an identity function for this example. We see that in G_1: $\Gamma(9) = \{b_1\}$ but in G: $\Gamma(h(9)) \cap \Sigma_f = \{b_1, b_2\} \not\subseteq \{b_1\}$. Therefore, the (sub)automaton for K_1 will not have state 9. Similarly it will not have state 4 as well.

We can illustrate the evolution of the system under the influence of faults and controlled by a fault-tolerant controller for $k_r = 1$ as follows: $0 \xrightarrow{a_1\ \phi\ a_2\ \rho\ b_1 b_2} 8$. It can be verified that the system is not tolerant to faults if $k_r = 2$. We will get $K_2 = \emptyset$ using the process described above.

V. Discussion

We considered the supervisory control of DES in the presence of a *fault*, denoted by ϕ, that renders an arbitrary subset of controllable events, Σ_f, to be temporarily uncontrollable. The occurrence of ϕ is not directly observed but is inferred from the occurrence of an event $\sigma \in \Sigma_c$ that was disabled by the supervisor. The semantics adopted by this paper supposes that the rectification-event ρ occurs *immediately after* the k_r-th unintended occurrence of a disabled event due to the

fault. We proved the necessary and sufficient condition for the existence of a supervisor tolerant to such faults for a given value of k_r. We used results from [13] to prove that such a supervisor can always be synthesized if the language of the plant and the specification is *regular*.

In addition to discussing the synthesis of controllers for the particular case of faults, we also prove that if the automaton of the specification is a subautomaton of the plant, then the fault semantics can be fit into the framework developed in [15]; where a framework for fault-tolerant supervisory control of DESs in which the plant automaton possesses both faulty and non faulty behavior was discussed.

The supervisory policy proposed in this paper, which ensures compliance to a specification, is agnostic to the subset of controllable events that are affected by the fault. One direction of future research is to customize the supervisory policy when particular, known, subsets of controllable events become uncontrollable due to the fault-event.

The detection algorithm depends on the observations of the occurrence of events that were disabled by the supervisor. The study of simultaneous faults in controllability and observability is also suggested as a future research topic.

REFERENCES

[1] P. J. Ramadge and W. M. Wonham, "The control of discrete event systems," *Proceedings of the IEEE*, vol. 77, no. 1, pp. 81–98, 1989.

[2] F. Lin, "Robust and adaptive supervisory control of discrete event systems," *IEEE Transactions on Automatic Control*, vol. 38, no. 12, pp. 1848–1852, 1993.

[3] A. M. Sánchez and F. J. Montoya, "Safe supervisory control under observability failure," *Discrete Event Dynamic Systems*, vol. 16, no. 4, pp. 493–525, 2006.

[4] J. L. Peterson, *Petri net theory and the modeling of systems*. Prentice Hall PTR, 1981.

[5] F.-S. Hsieh, "Fault-tolerant deadlock avoidance algorithm for assembly processes," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 34, no. 1, pp. 65–79, 2004.

[6] M. Lawley and W. Sulistyono, "Robust supervisory control policies for manufacturing systems with unreliable resources," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 3, pp. 346–359, 2002.

[7] F.-S. Hsieh, "Robustness of deadlock avoidance algorithms for sequential processes," *Automatica*, vol. 39, no. 10, pp. 1695–1706, 2003.

[8] S. Reveliotis and Z. Fei, "Robust deadlock avoidance for sequential resource allocation systems with resource outages," *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 4, pp. 1695–1711, 2017.

[9] G. Liu, P. Li, Z. Li, and N. Wu, "Robust deadlock control for automated manufacturing systems with unreliable resources based on petri net reachability graphs," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2018.

[10] L. Li, C. Hadjicostis, and R. Sreenivas, "Designs of bisimilar petri net controllers with fault tolerance capabilities," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 38, no. 1, pp. 207–217, 2008.

[11] R. Fritz and P. Zhang, "Overview of fault-tolerant control methods for discrete event systems," *IFAC-PapersOnLine*, vol. 51, no. 24, pp. 88–95, 2018.

[12] C. G. Cassandras and S. Lafortune, *Introduction to discrete event systems*. Springer Science & Business Media, 2009.

[13] W. M. Wonham and P. J. Ramadge, "On the supremal controllable sublanguage of a given language," *SIAM Journal on Control and Optimization*, vol. 25, no. 3, pp. 637–659, 1987.

[14] J. E. Hopcroft, *Introduction to automata theory, languages, and computation*. Pearson Education India, 2008.

[15] Q. Wen, R. Kumar, J. Huang, and H. Liu, "A framework for fault-tolerant control of discrete event systems," *IEEE Transactions on Automatic Control*, vol. 53, no. 8, pp. 1839–1849, 2008.