12th IFAC/IEEE Workshop on Discrete Event Systems
Cachan, France. May 14-16, 2014

Computation of Supervisors for Fault-Recovery and Repair for Discrete Event Systems ⋆

Ayşe Nur SÜLEK * **Klaus Werner SCHMIDT** **

* *Department of Electronic and Communication Engineering, Çankaya University, Ankara, Turkey, (e-mail: anursulek@cankaya.edu.tr)*
** *Department of Mechatronics Engineering, Çankaya University, Ankara, Turkey, (e-mail: schmidt@cankaya.edu.tr)*

Abstract: In this paper, we study the fault-recovery and repair of discrete event systems (DES). To this end, we first develop a new method for the fault-recovery of DES. In particular, we compute a fault-recovery supervisor that follows the specified *nominal* system behavior until a fault-occurrence, that continues its operation according to a *degraded* specification after a fault and that finally converges to a desired *behavior after fault*. We next show that our method is also applicable to system repair and we propose an iterative procedure that determines a supervisor for an arbitrary number of fault occurrences and system repairs. We demonstrate our method with a manufacturing system example.

Keywords: Discrete event systems, supervisory control, failure-recovery, repair.

1. INTRODUCTION

Fault-tolerant and *failure-recovery* control allow continuing the system operation after a fault occurrence while fulfilling a potentially degraded specification. In this paper, we develop a new approach for fault-recovery of discrete event systems (DES) that is also suitable for handling system repair. We consider DES that allow modeling the occurrence of faults and we use three language specifications to conveniently represent the desired system behavior: the desired non-faulty behavior is given by a *nominal* specification; the desired continuation of the system behavior after a fault-occurrence is given by a *degraded specification*; the desired faulty closed-loop behavior that should finally be achieved is represented by a more restrictive *faulty specification*. As an important feature of our formulation, it is not assumed that the closed-loop system must obey each specification starting from the initial plant state but it has to partially fulfill each specification depending on the presence of a fault.

In order to solve the fault-recovery problem, we propose an algorithm for finding a nonblocking fault-recovery supervisor based on the *interleaving composition* operation (Hoare, 2004 [1995]) and using language convergence (Willner and Heymann, 1995). We further show that the developed method can as well be applied to handle system repair. Then, it is desired to finally achieve the nominal specification after performing a system repair. Finally, an iterative application of our method allows computing a fault-recovery supervisor for an arbitrary number of fault occurrences and system repairs. We apply the described method to a manufacturing system example.

Several approaches in the literature address fault-recovery without an explicit consideration of system repair and the re-occurrence of faults (Saboori and Hashtrudi-Zad, 2005; Paoli et al., 2011; Kumar and Takai, 2012; Wittmann et al., 2012; Wen et al., 2008). Saboori and Hashtrudi-Zad (2005) consider that the system enters a transient mode upon fault occurrence and a recovery mode upon fault detection. The work by Paoli and Lafortune (2008); Paoli et al. (2011) proposes to detect faults using a diagnoser and switch to a different supervisor before the nominal system behavior is violated. Kumar and Takai (2012) derive necessary and sufficient conditions for the controller reconfiguration in case of faults. The use of *fault-accommodating models* is proposed by Wittmann et al. (2012). Similar to our approach, this method allows integrating the nominal and faulty system behavior and system specification into a single model. The work in Wen et al. (2008) uses a notion of convergence similar to our approach. However, Wen et al. (2008) define fault-tolerance in the sense that the system behavior should converge to the nominal system behavior instead of a desired faulty behavior. A specific version of system repair is considered in our previous work in Sülek and Schmidt (2013), where it is assumed that certain plant events are no longer possible if a fault occurs and the plant can return to its *nominal* operation after repair.

The remainder of the paper is organized as follows. Section 2 introduces the basic notation regarding DES and supervisory control. The fault-recovery control problem considered in this paper is stated in Section 3 and a solution is developed in Section 4. Section 5 considers the case of system repair and multiple faults and Section 6 gives conclusions.

⋆ This work was supported by the Scientific and Research Council of Turkey (TÜBITAK) [Carrier Award 110E185].

2. PRELIMINARIES

2.1 Discrete Event Systems

For a finite alphabet Σ, the set of all finite strings over Σ is denoted as $\Sigma^\star$, whereas the empty string is $\epsilon \in \Sigma^\star$. For any string $s \in \Sigma^\star$, $|s|$ denotes the length of s. A *language* over Σ is a subset $L \subseteq \Sigma^\star$. Writing $s_1 \leq s$ if there is a $t \in \Sigma^\star$ s.t. $s = s_1 t$, a language L is denoted as *prefix-closed* if $L = \overline{L} := \{s_1 \in \Sigma^\star | \exists s \in L \text{ s.t. } s_1 \leq s\}$. In addition, for a language $L \in \Sigma^\star$ and a string $s \in \overline{L}$, we write $L/s := \{t \in \Sigma^\star | st \in L\}$ for the set of suffixes of s in L.

We model a DES by a deterministic *finite automaton* $G = (X, \Sigma, \delta, x_0)$ with the *states* X, the *alphabet* Σ, the partial *transition function* $\delta : X \times \Sigma \to X$ and the *initial state* x_0. We recursively extend δ to strings such that for $x \in X$, $s \in \Sigma^\star$ and $\sigma \in \Sigma$, $\delta(x, \epsilon) = x$ and $\delta(x, s\sigma) = \delta(\delta(x, s), \sigma)$ if $\delta(x, s)$ exists. Then, we define the *closed language* $L(G) := \{s \in \Sigma^\star | \delta(x_0, s) \text{ exists}\}$ of G and assume that the *synchronous composition* $G_1||G_2$ of two automata G_1 and G_2 is given in the usual way (Wonham, 2010).

2.2 Supervisory Control

In a supervisory control context, we write $\Sigma = \Sigma_\mathrm{c} \dot\cup \Sigma_\mathrm{u}$ for *controllable* (Σ_c) and *uncontrollable* (Σ_u) events. We say that $S = (Q, \Sigma, \nu, q_0, Q_\mathrm{m})$ is a *supervisor* for G with Σ_u if S only disables events in Σ_c. That is, for all $s \in L(G) \cap L(S)$ and $\sigma \in \Sigma_\mathrm{u}$ with $s\sigma \in L(G)$ also $s\sigma \in L(S)$.

A language $K \subseteq L_\mathrm{m}(G)$ is said to be controllable for $L(G)$ and Σ_u if $\overline{K}\Sigma_\mathrm{u} \cap L(G) \subseteq \overline{K}$, and there exists a marking supervisor S such that $L_\mathrm{m}(G||S) = K$ if and only if K is controllable for $L(G)$ and Σ_u Ramadge and Wonham (1987). In case K is not controllable for $L(G)$ and Σ_u, we employ the *supremal controllable sublanguage* of K, denoted as $SupC(K, L(G), \Sigma_\mathrm{u})$ and use the marking supervisor S such that $L_m(S||G) = SupC(K, L(G), \Sigma_\mathrm{u})$. It is ensured that such supervisor is nonblocking and maximally permissive if $SupC(K, L(G), \Sigma_\mathrm{u}) \neq \emptyset$.

2.3 Interleaving Composition

We recall the *interleaving composition* from Hoare (2004 [1995]) [page 99]. Given two languages $K_1, K_2 \subseteq \Sigma^\star$ over the same alphabet, it defines a language that contains all possible interleavings of strings from K_1 and K_2. We reformulate the interleaving composition in our notation.

Definition 1. Let Σ be an alphabet and $K_1, K_2 \subseteq \Sigma^\star$ be two languages. The *interleaving composition* $K_1|||K_2$ of K_1 and K_2 is defined such that

$$s \in K_1|||K_2 \Leftrightarrow s = s_1^1 s_1^2 \cdots s_k^1 s_k^2 \text{ for some } k \in \mathbb{N} \text{ and } s_1^j s_2^j \cdots s_k^j \in K_j \text{ for } j = 1, 2.$$

2.4 Language Convergence

We employ the notion of *language convergence* as introduced by Willner and Heymann (1995). For a string $s \in \Sigma^\star$, we write $\mathrm{suf}_i(s)$ for the string obtained by deleting the first i events from s. Specifically, $\mathrm{suf}_0(s) = s$ and $\mathrm{suf}_{|s|}(s) = \epsilon$. Now consider two languages $M, K \subseteq \Sigma^*$. M is said to *converge* to K, denoted by $K \Leftarrow M$, if there is an integer $n \in \mathbb{N}_0$ such that for each $s \in M$, there exists an $i \leq n$ such that $\mathrm{suf}_i(s) \in K$. The least possible n is called the *convergence time*.

In the supervisory control context, the *controlled convergence problem* (CCP) is studied. Let G be a plant automaton over the alphabet Σ, $\Sigma_\mathrm{u} \subseteq \Sigma$ be a set of uncontrollable events and $K \subseteq \Sigma^*$ specification. A supervisor S is said to solve the CCP for G, K and Σ_u if $S||G$ is nonblocking, $L(S||G)$ is controllable for $L(G)$ and Σ_u, and $K \Leftarrow L_\mathrm{m}(S||G)$. Assume that X is the state set of G and Y is the state set of a recognizer C such that $L_\mathrm{m}(C) = K$. It is shown by Willner and Heymann (1995) that the solvability of the CCP can be decided by an algorithm with complexity $\mathcal{O}(|X|^2 2^{2|Y|})$. In addition, a possible supervisor is also suggested in that paper.In the sequel, we employ this supervisor whenever we solve the CCP.

3. PROBLEM STATEMENT

In this section, we formulate the fault-recovery problem studied in this paper. We consider that the system is modeled using the alphabets $\Sigma, \Sigma^\mathrm{N}, \Sigma^\mathrm{F}, \Sigma_\mathrm{u}$. Hereby, Σ^F contains fault events whose occurrence indicates the occurrence of a fault, Σ^N contains all events that are not associated to faults and $\Sigma = \Sigma^\mathrm{N} \dot\cup \Sigma^\mathrm{F}$. Σ_u is the set of uncontrollable events. Then, the system behavior is characterized by the plant model $G = (X, \Sigma, \delta, x_0, X_\mathrm{m})$ that includes the potentially faulty system behavior.

The main objective of this research is to synthesize a supervisor $S^\mathrm{F} = (Q^\mathrm{F}, \Sigma, \nu^\mathrm{F}, q_0^\mathrm{F}, Q_\mathrm{m}^\mathrm{F})$ that achieves fault-recovery in the closed loop $G||S^\mathrm{F}$. In order to specify the desired system behavior, we consider three different specifications. First, the nominal specification $K^N \subseteq L_\mathrm{m}(G)$ characterizes the desired system behavior in case no fault is present in the system. That is, the closed-loop behavior without any fault occurrence should be a subset of the nominal specification and nonblocking as stated in the following condition.

P1: $L_\mathrm{m}(G||S^\mathrm{F}) \cap (\Sigma^\mathrm{N})^\star \subseteq K^\mathrm{N}$

Second, we use the degraded specification $K^\mathrm{D} \subseteq \Sigma^\star$ that represents the *admissible* behavior after a fault occurrence. In principle, we want that the system continues its operation after any fault while considering the past system behavior until the fault. That is, a suitable part of the behavior before a fault concatenated with the behavior after a fault should belong to K^D. Formally, we want that

P2: it holds for all $s \in L_\mathrm{m}(G||S^\mathrm{F}) \cap (\Sigma^\mathrm{N})^\star \Sigma^\mathrm{F} (\Sigma^\mathrm{N})^\star$ that there exists a partition $s = s_1^1 s_1^2 \cdots s_k^1 s_k^2 \mathtt{f} s_3$ with $\mathtt{f} \in \Sigma^\mathrm{F}$, $s_i^j \in (\Sigma^\mathrm{N})^\star$ for $i = 1, \ldots, k$ and $j = 1, 2$, $s_1^1 \cdots s_k^1 \in \overline{K^\mathrm{N}}$ and $s_1^2 \cdots s_k^2 s_3 \in K^\mathrm{D}$.

In words, $s_1^1 \cdots s_k^1 \in \overline{K^\mathrm{N}}$ requires that one part of the substring before a fault occurrence belongs to the nominal behavior, whereas $s_1^2 \cdots s_k^2 s_3 \in K^\mathrm{D}$ requires that the remaining substring $s_1^2 \cdots s_k^2$ before the fault occurrence can be continued to a string in K^D. That is, a substring of the non-faulty behavior that should originally fulfill