2016 American Control Conference (ACC)
Boston Marriott Copley Place
July 6-8, 2016. Boston, MA, USA

Achieving Fault-tolerance and Safety of Discrete-event Systems through Learning

Jin Dai, Ali Karimoddini, and Hai Lin

***Abstract*— A system is said to be fault-tolerant if it remains functional even after a fault occurs. By describing faults as unpredicted events, we study the active fault-tolerance of discrete-event systems (DES) while ensuring safety requirements. Starting from a finite automaton model of the uncontrolled plant, our proposed control framework consists of nominal supervision, fault diagnosis and active post-fault control reconfiguration. First a nominal supervisor is designed with respect to the nominal mode to ensure the control specification prior to the occurrence of faults. Second, a learning-based algorithm is proposed to compute a diagnoser that can detect the occurrence of a fault. Necessary and sufficient conditions under which a post-fault safety-enforcing control reconfiguration is feasible are explored, and a second learning-based design algorithm for the post-fault supervisor is presented by using the limited lookahead policies. Effectiveness the proposed framework is examined through an example.**

I. Introduction

Sophisticated structures of modern engineering systems have made them more vulnerable to faults [1] that may cause undesired consequences. Inspired by the fact that faults are often modeled as events, discrete-event system (DES) models [2] have been widely adopted for fault diagnosis and fault-tolerance. Fault diagnosis of DES using different decision-making formalisms, such as centralized [3] [4], decentralized [5] and distributed [6] structures, have been studied in literature.

Considerable contributions have also been made to fault-tolerant control problem. The authors of [7] considered the problem of multi-agent cooperative tasking under event failures, the results were expanded in [8] where tolerance of sensor and actuator failures in multi-agent systems were investigated. Wen et al. [9] proposed a framework for the fault-tolerant control of DES where the fault-tolerance and recovery were guaranteed on the basis of language stability. Such technique is classified as "passive" since the designed controller uniformly satisfies nominal and post-fault specifications. In contrast to the passive approaches, Paoli et al. [10] investigated an "active" fault-tolerance [1] of controlled DES and active fault-tolerant supervisor switching mechanism was considered based on diagnostic information. The fault-tolerant control for DES with safety constraints were considered in [11] and state-feedback fault-tolerant control rules were developed. It is worth pointing out that both aforementioned passive and active approaches require a full prior knowledge of the DES plant model before and after the occurrence of faults, which might be unrealistic in some real world applications.

In this paper, we formulate a generic model of a DES plant generating both nominal and faulty behaviors under performance requirements and safety constraints, and propose an active fault-tolerant control architecture such that the supervisor actively reacts to fault in the following manner: (i) design a nominal supervisor enforce the fault-free specification; (ii) design a fault diagnoser that detects the faults with unambiguity; (iii) the system automatically switches to a post-fault supervisor such that the system continues its evolution safely. We derive necessary and sufficient conditions for the existence of appropriate post-fault supervisors that attain the fault-tolerance. Compared with existing approaches, such as [10], our proposed framework assumes only prior knowledge of the non-faulty part of the uncontrolled plant, and the behaviors of the system under faulty mode are confined to a limited lookahead window [12]. Two modified L^*-learning [13] based algorithm are deployed for the design of the nominal and the post-fault supervisors respectively.

The remainder of this paper is organized as follows. Section II introduces basic notations in supervisory control and fault diagnosis of DES and L^* learning, and formulates the active fault-tolerant control problem. Section III provides a learning-based technique to synthesize the nominal supervisor; while Section IV investigates the learning-based diagnoser design problem. Section V studies the problem of post-fault control reconfiguration. Section VI gives a simple example to illustrate the proposed diagnosis-control framework. Concluding remarks are presented in Section VII.

II. Problem Formulation

A. Automata Models

The uncontrolled discrete-event plant [14] [15] is modeled as a deterministic finite automaton (DFA) $G = (Q, \Sigma, q_0, \delta)$, where Q is the finite set of states, Σ is the finite set of events, $\delta : Q \times \Sigma \to Q$ is the (partial) transition function, and $q_0 \in Q$ is the initial state. δ can then be extended to $\delta : Q \times \Sigma^* \to Q$ in a natural way, where Σ^* denotes all the finite-length traces over Σ plus the zero-length trace ϵ. The language generated by G is given by $L(G) = \{s \in \Sigma^* | \exists q_0 \in Q_0, \delta(q_0, s)!\}$ where $\delta(q_0, s)!$ means that $\delta(q_0, s)$ is defined. Given two DFA $G_1 = (Q_1, \Sigma, q_{0,1}, \delta_1, Q_{m,1})$ and $G_2 = (Q_2, \Sigma, q_{0,2}, \delta_2, Q_{m,2})$, G_1 is said to be a subautomaton of G_2, denoted as $G_1 \sqsubseteq G_2$, if either $Q_1 = \emptyset$ or $Q_1 \subseteq Q_2$, and for all $q \in Q_1$ and

Jin Dai and Hai Lin are with the Department of Electrical Engineering, University of Notre Dame, Notre Dame, IN 46556, USA. {jdai1, hlin1}@nd.edu.

Ali Karimoddini is with the Department of Electrical and Computer Engineering, North Carolina A&T University, Greensboro, NC 27411, USA. akarimod@ncat.edu

$\sigma \in \Sigma$, it holds that $\delta_1(q,\sigma)! \Rightarrow \delta_1(q,\sigma) = \delta_2(q,\sigma)$. For traces $s, t \in \Sigma^*$, s is said to be a prefix of t, denoted as $s \leq t$, if there exists $w \in \Sigma^*$ such that $t = sw$. For a language $K \subseteq \Sigma^*$, $\bar{K} = \{s \in \Sigma^* | \exists t \in K : s \leq t\}$ denotes the prefix-closure of K, and K is said to be *prefix-closed* if $\bar{K} = K$. We use the notion $K \setminus t := \{s \in \Sigma^* | ts \in K\}$ to denote the set of all extensions that can be executed in K after the occurrence of t, and $K_1 \setminus K_2 = \{t \in \Sigma^* | \exists s \in K_2 \text{ such that } st \in K_1\}$.

For control purposes, the event set Σ of G is partitioned into sets of controllable events Σ_c and uncontrollable events Σ_{uc}. A language K is said to be *controllable* if $\bar{K}\Sigma_{uc} \cap L(G) \subseteq \bar{K}$. Furthermore, Σ is also partitioned as $\Sigma = \Sigma_o \cup \Sigma_{uo}$, where Σ_o represents the set of observable event and Σ_{uo} represents the set of unobservable events. The natural projection $P : \Sigma^* \to \Sigma_o^*$ and the inverse projection $P^{-1} : \Sigma_o^* \to 2^{\Sigma^*}$ are defined accordingly [2]. A language K is said to be observable if $\forall s, t \in \bar{K}, \sigma \in \Sigma : P(s) = P(t), s\sigma \in \bar{K}, t\sigma \in L(G) \Rightarrow t\sigma \in \bar{K}$. A supervisor S is another finite automaton that operates in parallel with G, and the closed-loop system is denoted as $S||G$, where $||$ stands for the parallel composition operator [2]. It has been shown that given a nonempty and prefix-closed language $K \subseteq L(G)$, there exists a supervisor S such that $L(S||G) = K$ if and only if K is controllable and observable with respect to $L(G)$ [2].

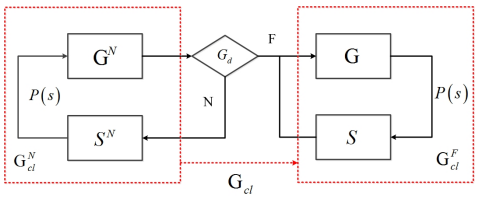


Fig. 1: Active fault-tolerant control framework.

Potential faults of the system are considered at this point. Let $\Sigma_f \subseteq \Sigma$ denote the set of failure events which may occur in the system. Without loss of generality, we assume that $\Sigma_f \subseteq \Sigma_{uc} \cap \Sigma_{uo}$ since a controllable or observable fault events can easily be diagnosed and be disabled before its occurrence. For simplicity of presentation, we consider a single fault event f. We use $\Psi(f) = \{sf \in L(G)\}$ to denote all traces generated by the plant G that end with f.

Let $G^N = (Q^N, \Sigma, q_0^N, \delta^N, Q_m^N) \sqsubseteq G$ denote the non-faulty part of G; clearly, $L(G^N) \cap \Psi(f) = \emptyset$. The active fault-tolerant control framework is shown in Fig. 1. Given G^N, prefix-closed nominal specification $K^N \subseteq L(G^N)$ and prefix-closed safety requirement $K^S \subseteq L(G)$, the proposed framework needs to solve the following three problems.

1) *Nominal supervision* Design a nominal supervisor S^N such that: $L(G_{cl}^N) := L(S^N||G^N) = K^N$.
2) *Fault diagnosis* A diagnoser G_d is designed to monitor behaviors of the real system $G_{cl} := S^N||G$; the diagnoser should accomplish diagnosis before the system violates K^S, where prior knowledge of G is not necessarily given.
3) *Safety-enforcing control reconfiguration:* Once G_d detects the fault, we stop the operation of the nominal supervisor S^N and reconfigure the supervisor S such that $L(G_{cl}^F) := L(S||G) \subseteq K^S \setminus K^N$.

The relationship between languages of G^N, G_{cl}^N, G_{cl} and K^S are depicted in Fig. 2: the nominal supervisor S^N is designed with respect to G^N to achieve the nominal specification $K^N \subseteq L(G^N)$ when no fault occurs; after the fault occurs, undesirable behaviors embedded in $L(G_{cl})$ may emerge and may violate the safety constraint K^S, and towards this end, we require that we stop the operation of S^N and switch to a safety-enforcing post-fault supervisor.

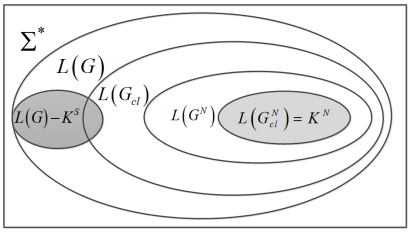


Fig. 2: Specifications for active fault-tolerance of supervised DES.

Three additional assumptions are made to Σ and G_{cl}^N.

(I) G_{cl}^N is deadlock-free [2], i.e., for all $q \in Q$, there exists an event $\sigma \in \Sigma$ such that $\delta(q,\sigma)!$.
(II) G_{cl}^N contains no cycle of unobservable events, i.e., $\exists n_0 \in \mathbb{N}$ such that $\forall ust \in L(G_{cl}^N)$, $s \in \Sigma_o^* \Rightarrow ||s|| \leq n_0$.
(III) All the controllable events must be observable, i.e., $\Sigma_c \subseteq \Sigma_o$.

B. L^* Learning

The L^* learning algorithm learns an unknown regular language U over an alphabet Σ and to produce a deterministic finite automaton accepting it by interacting with a Teacher that answers membership and conjecture queries. L^* processes at most $(k+1)(n+m(n-1))n$ membership queries and $n-1$ conjectures in order to build M recognizing U, where $k = |\Sigma|$, n stands for the minimal size of U, and m is the maximal length of a counterexample. Interested readers are referred to [13] for details of L^*.

III. Learning-based Design of the Nominal Supervisor

We proceed to the design of the nominal supervisor S^N. The proposed algorithm includes two steps: first, we synthesize a full-observation supervisor S^C such that $L(S^C||G^N) = \sup C_{G^N}(K^N)$ by using a modified L^* [13] algorithm L_C^* [16] (termed as L_{LS}^*); next, we modify S^C to be S^N such that $L(S^N||G^N) = \Omega(K^N)$, a prefix-closed, controllable and observable sublanguage of K^N.

The correctness and convergence of L_C^* are guaranteed in [8]. The complexity of constructing S^N is a polynomial of n (n stands for the number of states in the composite of

G^N and K^N), indicating that L_C^* has a higher complexity than conventional algorithms (see e.g. [2]) but is suitable for online implementations.

Next, we consider the construction of S^N based on $S^C = (Q_c, \Sigma, q_{0,c}, \delta_c)$. For each $q \in Q_C$, we define $Act_C(q) = \{\sigma \in \Sigma | \delta_c(q, \sigma)!\}$ as the set of active events in S^C at state q. For any subset $Q \subseteq Q_c$, the set of *σ-reachable* states from Q with an observable event $\sigma \in \Sigma_o$ is defined as $OR(Q, \sigma) = \{q \in Q_S | (\exists q' \in Q) q = \delta_c(q', \sigma)\}$ and the *unobservable reach* is defined as $UR(Q) = \{q \in Q_c | (\exists q' \in Q)(\exists s \in \Sigma_{uo}^*) q = \delta_c(q', s)\}$.

The nominal supervisor S^N is then a DFA $S^N = trim(Q_N, \Sigma_o, \delta_N, Q_{0,N})$ over the observable event set Σ_o, where $trim$ stands for the trim automaton [2], and $Q_N = 2^{Q_c}$, $Q_{0,N} = UR(\{q_{0,s}\})$. Moreover, for all $Q \in Q_N$, $Act_N(Q) = \bigcap_{q \in Q} Act_C(q)$, and for $\sigma \in \Sigma_o$ and $Q \in Q_N$, $\delta_N(Q, \sigma) = UR(OR(Q, \sigma))$. Let $L(G_{cl}^N) := \Omega(K^N)$, it can be shown that $\sup CN_{G^N}(K^N) \subseteq \Omega(K^N) \subseteq \sup C_{G^N}(K^N)$.

The complexity of transforming a full observation supervisor S^C to a partial supervisor S_2^N is above bounded by $O(|Q^N||Q_N|)$.

IV. Learning-based Fault Diagnosis

This section studies fault diagnosis of G_{cl}, which is not necessarily known *a priori* and can only be partially observed. The approach is to develop a diagnoser that can actively learn the faulty situations in the system. For this purpose, we modify the L^* algorithm to L_D^* to effectively and actively learn the diagnoser G_d. In this case, the L_D^* asks the Teacher the following minimum queries:

- *Membership queries*: in which the algorithm asks whether a string $s \in \Sigma_o^*$ belongs to $P(L(G_{cl}))$.
- *Conjecture queries*: in which the algorithm asks whether $L(G_d) = P(L(G_{cl}))$. If not, the oracle returns the string s as a counterexample.

With these queries, the algorithm acquires information about a finite collection of strings over Σ_o, and classifies them as either members or non-members of $P(L(G_{cl}))$. Then, this information will be used to create a series of observation tables to incrementally record and maintain the membership information. The observation table (S, E, T^d) is a 3-tuple where $S \subseteq \Sigma_o^*$ is a non-empty finite set of prefixes, $E \subseteq \Sigma_o^*$ is a non-empty set of suffixes, and T^d is the membership Teacher to be defined. The i-th observation table T_i can be depicted as a 2-dimensional array whose rows are labeled by strings $s \in S \cup S\Sigma_o$, and whose columns are labeled by symbols $\sigma \in E$. The membership queries in L_D^*, T^D, is a function from $\{s\sigma | s \in S \cup S\Sigma_o, \sigma \in E\}$ to the label set $\Sigma_L = \{0, \mathcal{F}, \mathcal{N}, \mathcal{U}\}$: $T^d(s\sigma) = 0$ if $s\sigma \notin P(L(G_{cl}))$; $T^d(s\sigma) = \mathcal{F}$ if $s\sigma \in P(L(G_{cl}))$ is faulty; $T^d(s\sigma) = \mathcal{N}$ if $s\sigma \in P(L(G_{cl}))$ is not faulty; and $T^d(s\sigma) = \mathcal{U}$ if $s\sigma \in P(L(G_{cl}))$ is undecidable. By "faulty" we mean that $\forall t \in S\Sigma^*, P(t) = P(s\sigma), f \in t$[1], and by "undecidable" we mean that there exist $t, t' \in S\Sigma^*$ such that

[1]The notion $f \in t$ formally states that $\bar{t} \cap \Psi(f) \neq \emptyset$

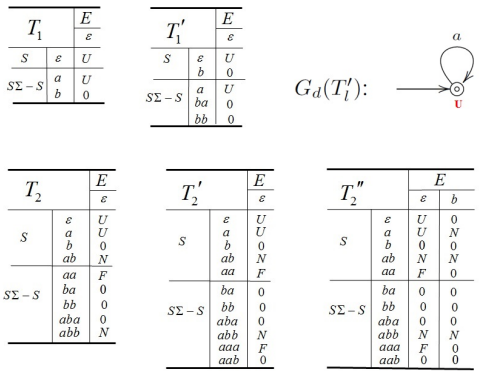


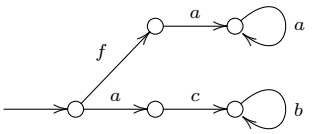
Fig. 3: Construction of the diagnoser automaton.

$P(t) = P(t') = P(s\sigma)$, $f \in t$ but $f \in t'$. The row function $row : (S \cup S\Sigma_o)E \to \Sigma_L^{|E|}$ denotes the table entries in rows.

After making the observation table T_i closed and consistent [13], we can construct a DFA, $G_d(T_i) = CoAc(Q_d, \Sigma_o, \delta_d, q_{d,0}, Q_{d,m})$, where $Q_d = \{row(s) : s \in S\}$, $q_{d,0} = row(\epsilon)$, $Q_{d,m} = \{row(s) : (s \in S) \wedge T^d(s) \neq 0\}$, $\delta(row(s), \sigma) = row(s\sigma)$, and $CoAc$ is an operator that removes the states from which there does not exist a path to marked states. Associated to each state of the resulting DFA, a label can be defined $l(s) = T^d(s\epsilon)$. Then we use the obtained automaton, $G_d(T_i)$, as the diagnoser for the plant G_{cl}. This diagnoser keeps running until a counter-example, $cex \in (P(L(G_{cl})) - L(G_d(T_i))) \cup (L(G_d(T_i)) - P(L(G_{cl})))$, is detected by the conjecture Teacher. In this case, cex and all its prefixes will be added into S. The table should be updated for the newly added elements.

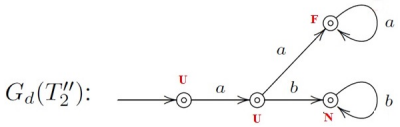
L_D^* constructs a minimal G_d with computational time bounded by a polynomial in $|Q_d|$ and the number of events contained in the longest counterexample provided by the Teacher when answering equivalence queries under the single fault circumstance. However, without full knowledge of G_{cl}, building up the diagnoser for multi-fault cases remains to be elusive, since there might exist non-unique traces containing different fault events but resulting in the same projected values.

Example 1: Consider the closed-loop plant G_{cl}:



with $\Sigma = \{a, b, c, f\}$, $\Sigma_o = \{a, b\}$, $\Sigma_{uo} = \{c, f\}$, and $\Sigma_f = \{f\}$. Assume that $L(G_{cl}^N) = K^N = \overline{acb^*}$, G_{cl} is unknown and only the strings of observable events of the system are available to gradually and incrementally construct the diagnoser G_d. To form the observation table, initially, the sets S and E are set to ϵ, and the observation table , T_1, shown in Fig. 3, is initialized using membership queries for every string in $(S \cup S\Sigma_o) \times E$. The table T_1 is not closed and we make it closed and consistent as shown in T_1'. The corresponding $G_d(T_1')$ is shown in Fig. 3. However,

a counterexample $cex = ab \in L(G_d(T_1')) - P(L(G_{cl}))$ is detected, and a and bb are added to S so that a new observation table is made. After respectively making T_2 to be closed and consistent, the closed and consistent observation table T_2'' is obtained, which turns out to generate no more counterexamples. Hence, the diagnoser is given by $G_d = G_d(T_2)$.



V. Enforcement of Post-fault Safety via Limited Lookahead Policies

This section investigates the problem of reconfiguring the supervisor to achieve active fault-tolerance using diagnostic information, provided that $L(G_{cl})$ is diagnosable [3].

A. Existence of Safety-enforcing Post-fault Supervisors

If $L(G_{cl})$ is diagnosable [3], the fault f can always be diagnosed by G_d within a finite delay after its occurrence. To ensure the safety specification K^S of the system, one shall require that diagnosis of f should be accomplished before the system G_{cl} executes any unsafe behaviors. Such requirement is characterized as the following property called "safe diagnosability".

Definition 1: A prefix-closed L satisfying Assumptions (I) and (II) is said to be safe diagnosable with respect to the fault event f, prefix-closed safety specification K^S and P if

$$(\exists n \in \mathbb{N})(\forall s \in \Psi(f))(\forall t \in L \setminus s)(||t|| \geq n \Rightarrow \mathcal{SD})$$

where the safe diagnosability condition $\mathcal{SD}$ is stated as follows:

$$\exists v \in \overline{st} \cap K^S \Rightarrow \forall u \in P^{-1}P(u) \cap L, f \in u$$

Definition 1 implies that a language is safe diagnosable if it is diagnosable and the unambiguous diagnosis decision should be made before execution of any unsafe behaviors. We define the class of "first observed" F-certain states in G_d as follows.

Definition 2: The set of first observed F-certain states, denoted as $\mathcal{FOF}(G_d)$, is a subset of $\mathcal{F}(G_d)$ such that

$$(\forall q_d \in \mathcal{FOF}(G_d))(\exists \sigma \in \Sigma_o, q' \in Q_d)(q_d = \delta_d(q', \sigma))$$
$$\Rightarrow (q' \notin \mathcal{FOF}(G_d))$$

The safe diagnosability of $L(G_{cl})$ can then be captured by the first observed F-certain states in G_d, indicating that safety specifications can be guaranteed before the observation of an F-certain state in G_d.

Proposition 1: The controlled plant language G_{cl} is safe diagnosable with respect to a fault event f and projection P if

(i) $L(G_{cl})$ is diagnosable, i.e., no F-indeterminate cycles exist in G_d, and
(ii) $\forall q_d \in \mathcal{FOF}(G_d)$ with $q_d = \delta_d(q_{d,0}, s)$, $P^{-1}(\overline{s}) \cap L(G_{cl}) \subseteq K^S$.

If the system G_{cl} is safe diagnosable, G_d can thus always detect the fault f before G_{cl} executes any traces violating K^S. Upon the detection and diagnostic information of f, supervisor reconfiguration actions including stopping the nominal supervisor should be enforced or otherwise unsafe behaviors may emerge. On the other hand, it is still possible that even though we detect the occurrence of the fault while obey the safety constraints, there is no acceptable supervisor by which we can guarantee the plant's behaviors to be safe. The following proposition studies this concern and proposes necessary and sufficient conditions for the existence of a feasible post-fault supervisor.

Proposition 2: For a controlled plant G_{cl} with a corresponding diagnoser G_d, a post-fault supervisor S exists such that $L(G_{cl}^F) = L(S||G) \subseteq K^S$ if and only if

(i) $L(G_{cl})$ is safe diagnosable with respect to f, K^S and P, and
(ii) For all $q_d \in \mathcal{FOF}(G_d)$ with $q_d = \delta_d(q_0, s)$, $\forall t \in L(G_d) \setminus s \cap \Sigma_{uc}^*$, $P^{-1}(\overline{st}) \cap L(G_{cl}) \subseteq K^S$, provided that Assumption (III) holds.

Proposition 3 implies that a safety-enforcing post-fault supervisor exists if no uncontrollable successive behaviors that drive G_{cl} to unsafe region exist.

B. Synthesis of the Post-fault Supervisor via Limited Lookahead Policies

We now turn to deal with the problem of automatically synthesizing the safety-enforcing post-fault supervisor if the existence conditions presented in the previous subsection are satisfied. In particular, rather than the assumption that the plant model G_{cl} is given in [10] and [9], here we assume that the knowledge of post-fault model can only be confined to a limited lookahead window [12]. Different from [12], the construction of the limited lookahead window in the active fault-tolerant control problem is of the partial observation nature and relies on the diagnoser G_d. Let $L(G_d, N, s)$ denote the set of all traces of observable events with length not longer than N observed by the fault diagnoser G_d, after the occurrence of a trace $t \in \Sigma_o^*$. Constrained by the safe diagonsability property, if the trace t satisfies that $\delta_d(q_{d,0}, s) \in \mathcal{FOF}(G_d)$, then the nominal supervisor S^N should be stopped and a post-fault supervisor S needs to be synthesized with respect to the post-fault safety-enforcing requirement $K^S \setminus K^N$.

To identify an acceptable post-fault supervisor by using the L^* procedure, the membership Teacher must be able to identify whether a given observation s, generated by the underlying system G_{cl} and observed by G_d after encountering a first observed F-certain state, belongs to $P(K^S \setminus K^N)$ (since set inclusion is preserved under projection). In the case where s ends in a controllable event (hence observable due to Assumption 3), the membership of s can be determined unambiguously; however, when s ends in an uncontrollable event, the membership of s can only be "pending" until future behavior observation of G_{cl} is explored. Let $L_u(G_d, N, s)$ denote the set of uncontrollable

traces in $L(G_d, N, s)$ and let $L_c(G_d, N, s)$ denote the controllable traces, both with respect to $P(K^S \setminus K^N)$. The set of pending traces is hence computed as $L_p(G_d, N, s) = L(G_d, N, s) - L_u(G_d, N, s) - L_c(G_d, N, s)$.

Based on the limited lookahead window $L(G_d, N, s)$, we present the following "uncertain" and conditional membership queries to the Teacher in L^* for partial observation case, where the superscript "oc" stantds for "observability and controllability".

$$T^{oc}(t|N,s) = \begin{cases} 0 & \text{if } t \in L_u(G_d, N, s)\Sigma_o^* \\ 1 & \text{if } t \in L_c(G_d, N, s) \\ ud & \text{if } t \in L_p(G_d, N, s) \end{cases}$$

where the entry "ud" means "undecidable". The introduction of entry value ud motivates us to modify the standard definition of closeness and consistency of the observation tables of the L^* procedure.

Definition 3: Let $\mathcal{S} = \{s_i\}$ denote the set of row labels contained in $S \cup S\Sigma_o$. Consider the m-th subset, S_m of $\mathcal{S}$ such thai for any s_i and s_j in $\mathcal{S}$, and for all $e \in E$, the following conditions hold: $T^{oc}(s_i e) \neq *$, $T^{oc}(s_j e) \neq *$, and $T^{oc}(s_i e) = T^{oc}(s_j e)$. The collection S_m is called the m-th aggregated group of the observation table.

An aggregated group is said to be maximal if no more row labels can be added. Based on Definition 5, we can re-define the closeness and consistency as follows.

Definition 4: Let S_i denote the i-th maximally aggregated group for a given observation table T. If for all $s \in S\Sigma - S$, there exists $s_1 \in S$ and i such that $s_1 \in S_i$ and $s \in S_i$, then the observation table as said to be ud-*closed*. Consider all pairs of traces s_1 and s_2 such that for some i, $s_1 \in S_i$ and $s_2 \in S_i$. If for all $\sigma \in \Sigma_o$ there exists j such that either $s_1\sigma \in S_j$ and $s_2\sigma \in S_j$ or $s_1\sigma \in S_j$ and $s_2\sigma \in S_j$ holds, then the observation table is said to be ud-*consistent*.

If an observation table is not ud-closed or ud-consistent, then one can make it ud-closed or ud-consistent by applying similar approaches in L_D^*. We conclude this section by the following theorem.

Theorem 1: The L^* algorithm with the membership queries T^{oc} defined above synthesizes a post-fault supervisor S such that the post-fault controlled plant $G_{cl}^F = S||G$ is guaranteed to satisfy a controllable and observable sublanguage of $K^S \setminus K^N$, and the synthesis procedure terminates within a finite number of counterexample tests.

VI. An Illustrative Example

We examine the effectiveness of the proposed fault-tolerant control framework by revisiting the example in [10]. Consider the hydraulic system G in Fig. 4 that consists of a water tank T, a pump P, three valves V_1, V_2 and V_r, and associated pipes. The pump is used to move fluid from the tank through the pipe and must coordinate with the valves, and the pipe is equipped with a pressure sensor for monitoring usage. Valves V_1 and V_2 are equipped with a switch such that only one of them can be used at the same time, and by default, V_1 is activated.

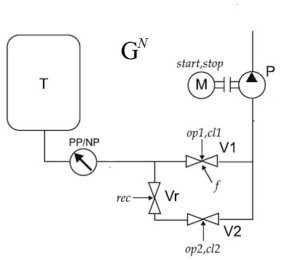


Fig. 4: The hydraulic system.

The events of the system are defined as follows: for the valve V_i for $i = 1, 2$, events op_i and cl_i are used to open and close valve V_i; event rec is used to open the reserved valve V_r; for the pump, event $start$ is used to start the pump while $stop$ to stop it; for the pressure sensor, P means an over-pressure in the pipe and NP means no over-pressure in the pipe. We assume that all these events are controllable and observable. The automaton representation of the nominal behavior G^N is given in Fig. 5 (a). The pump is coordinating with either one of V_1 and V_2 (with the help of V_r). We now consider the fault f that valve V_1 may get stuck closed due to possible malfunction, and the automaton representing G is then given in Fig. 5 (b). Note that we omit events P and NP for space consideration. The nominal specification is given by $K^N = \overline{(op_1.start.stop.cl_1)^*}$. The safety specification requires that the pump should not be working with the closed V_1 , i.e., $L(G) - K^S = \overline{(op_1.start.stop.cl_1)^*}.f.op_1.start$.

Since all the events except for f are controllable and observable, the nominal specification K^N is controllable and observable with respect to G^N. We use L_C^* to help build up the safe nominal specification, and the supervisor S^N as well as the closed-loop plant G_{cl} are depicted respectively in Fig. 6, in red solid lines.

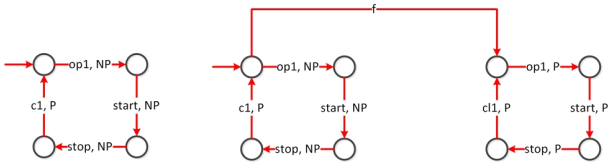


Fig. 6: The nominal supervisor S^N.

Now for closed-loop plant G_{cl}, we apply L_D^* and construct an online fault diagnoser G_d as shown in Fig. 7.

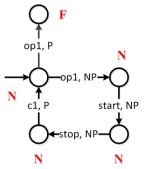


Fig. 7: The fault diagnoser G_d.

From Fig. 7, the fault diagnoser detects the fault f when

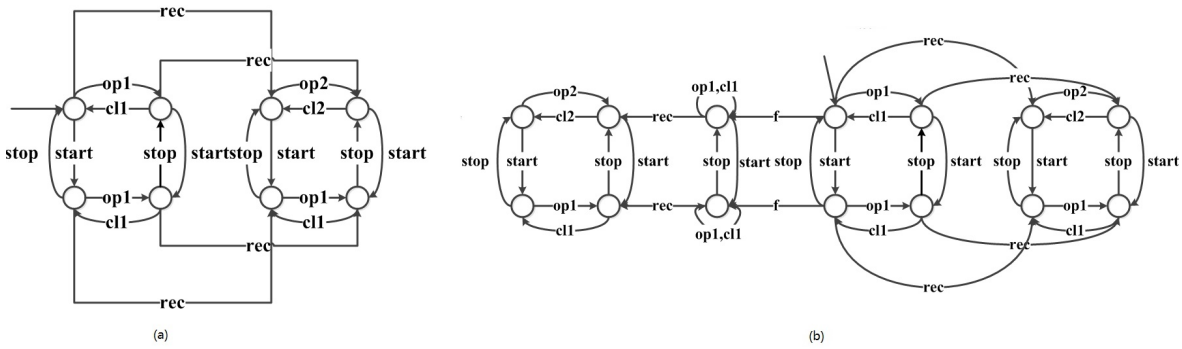


Fig. 5: G^N and G.

observing the trace $s = \overline{(op_1.start.stop.cl_1)}^*(op_1, PP)$. When the fault f is diagnosed, we stop the operation of S^N and consider the generation of post-fault specification K^F. In this example, rather than the malfunctioned V_1, we require that the pump should coordinate with the alternative valve V_2, whose operation recalls the reserved valve V_r. Hence the "partial" post-fault specification is given by $\overline{rec.(op_2.start.stop.cl_2)^*}$, which is obviously controllable and observable, and respects the safety constraint. We set up the length of lookahead window to be 3, and one of the lookahead windows and the post-fault supervisor S that drives the system to satisfy the post-fault performance are both given in Fig. 8.

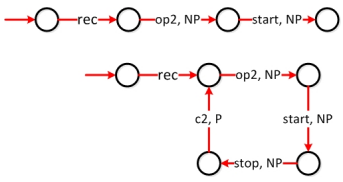


Fig. 8: The post-fault supervisor S.

VII. Conclusion

In this paper, the active fault-tolerance and safety-enforcement of discrete-event systems is studied. The contributions of this paper are as follows. First we present a learning-based synthesis algorithm for the nominal supervisor under the fault-free environment. Next, learning-based design algorithm of the fault diagnoser is proposed. By using the diagnostic information provided by the diagnoser, necessary and sufficient conditions under which a safe post-fault reconfiguration is feasible are presented, and the problem of learning-based post-fault supervisor design via limited lookahead policies is studied. The effectiveness of the proposed fault-tolerant control framework is also examined by an illustrative example. Future research directions may include extending the current work to decentralized and/or distributed discrete-event systems and to multiple possible faults.

Acknowledgement

The first and third author would like to appreciate the support from NSF-CNS-1239222, NSF-EECS-1253488 and NSF-CNS-1446288. The second author would like to appreciate the support from Air Force Research Laboratory and OSD under agreement number FA8750-15-2-0116.

References

[1] M. Blanke and J. Schröder, *Diagnosis and fault-tolerant control*. Springer, 2006, vol. 2.
[2] C. G. Cassandras and S. Lafortune, *Introduction to discrete event systems*. Springer, 2008.
[3] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis, "Diagnosability of discrete-event systems," *Automatic Control, IEEE Transactions on*, vol. 40, no. 9, pp. 1555–1575, 1995.
[4] S. Jiang and R. Kumar, "Failure diagnosis of discrete-event systems with linear-time temporal logic specifications," *Automatic Control, IEEE Transactions on*, vol. 49, no. 6, pp. 934–945, 2004.
[5] W. Qiu and R. Kumar, "Decentralized failure diagnosis of discrete event systems," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 36, no. 2, pp. 384–395, 2006.
[6] R. Su and W. M. Wonham, "Global and local consistencies in distributed fault diagnosis for discrete-event systems," *Automatic Control, IEEE Transactions on*, vol. 50, no. 12, pp. 1923–1935, 2005.
[7] M. Karimadini and H. Lin, "Fault-tolerant cooperative tasking for multi-agent systems," *International Journal of Control*, vol. 84, no. 12, pp. 2092–2107, 2011.
[8] J. Dai and H. Lin, "Learning-based design of fault-tolerant cooperative multi-agent systems," in *American Control Conference, 2015*. IEEE, 2015, pp. 1929–1934.
[9] Q. Wen, R. Kumar, J. Huang, and H. Liu, "A framework for fault-tolerant control of discrete event systems," *Automatic Control, IEEE Transactions on*, vol. 53, no. 8, pp. 1839–1849, 2008.
[10] A. Paoli, M. Sartini, and S. Lafortune, "Active fault tolerant control of discrete event systems using online diagnostics," *Automatica*, vol. 47, no. 4, pp. 639–649, 2011.
[11] S. Shu and F. Lin, "Fault-tolerant control for safety of discrete-event systems," *Automation Science and Engineering, IEEE Transactions on*, vol. 11, no. 1, pp. 78–89, 2014.
[12] S.-L. Chung, S. Lafortune, and F. Lin, "Limited lookahead policies in supervisory control of discrete event systems," *Automatic Control, IEEE Transactions on*, vol. 37, no. 12, pp. 1921–1935, 1992.
[13] D. Angluin, "Learning regular sets from queries and counterexamples," *Information and computation*, vol. 75, no. 2, pp. 87–106, 1987.
[14] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM journal on control and optimization*, vol. 25, no. 1, pp. 206–230, 1987.
[15] ——, "The control of discrete event systems," *Proceedings of the IEEE*, vol. 77, no. 1, pp. 81–98, 1989.
[16] J. Dai and H. Lin, "Automatic synthesis of cooperative multi-agent systems," in *Control and Decision (CDC), 53rd IEEE International Conference on*.