

# Decentralized Diagnosis of Discrete Event Systems using Unconditional and Conditional Decisions

Yin Wang, Tae-Sic Yoo, and Stéphane Lafortune

**Abstract**—The past decade has witnessed the development of a body of theory, with associated applications, for fault diagnosis of dynamic systems that can be modeled in a discrete event systems framework. This paper first discusses the dual problem of diagnosing the absence of faults in centralized and decentralized settings. The paper then develops new definitions of decentralized diagnosis in the context of a general decentralized architecture that allows for the use of “conditional decisions” by local diagnosers. The properties of these new definitions of decentralized diagnosability are presented and their relationship with existing work discussed. Corresponding verification algorithms are also described.

## I. INTRODUCTION

Fault diagnosis in Discrete Event Systems (DES) consists of detecting unobservable fault events occurring in a system by performing model-based inferencing driven by sequences of observable events; see [1–3] and the references therein. Decentralized and distributed diagnosis protocols become necessary to deal with fault diagnosis in systems where the information is decentralized. In *decentralized* architectures, there are several local “sites” where sensors report their data and diagnosers run at each site processing the local observations and performing model-based inferencing on the basis of the projection of the system model on the locally observable events; see, e.g., [4]. Local diagnosers then report their decisions about system faults; these decisions may or may not be fused at a coordinating site, according to the properties of the architecture. Generally speaking, *distributed* architectures for fault diagnosis differ from decentralized ones in terms of the local models used at the different sites for model-based inferencing and in terms of the ability for local diagnosers to communicate among each other in real-time. Distributed and decentralized diagnosis problems have received a lot of attention recently; see [5–14].

In this paper, we are interested in decentralized architectures where diagnosers at local sites operate independently (namely, without communicating among each other) and where local decisions about (potential) system faults are merged by simple memoryless Boolean operations, in the spirit of the so-called Protocol 3 in [4]. Namely, in Section IV and V, we consider “unconditional architectures” where

there is essentially no need for a coordinating site; i.e., the decisions of the respective diagnosers will not require to be merged other than trivially. In Section VI, we consider “conditional architectures” where diagnosers can issue conditional decisions about fault detection and isolation such as the decision “Fault if no other site says No Fault.” Conditional decisions have to be combined at a coordinating site, but the fusion rule will be simple and memoryless. Our approach builds on the results in [4] regarding Protocol 3 and is inspired by recent work in [15, 16] on decentralized control of DES, where conditional decisions are used to obtain more powerful control architectures and relax the condition of coobservability that arises in the necessary and sufficient conditions for supervisor existence. The use of conditional diagnosis decisions differentiates our approach from that used in [4] to improve upon Protocol 3, namely our results are different in nature from Protocols 1 and 2 in [4] which employ fusion rules based on *diagnoser state intersections* (with memory in the case of Protocol 1).

The paper begins with a brief review of the concept of diagnosability in Section II, followed by new results on the diagnosis of the absence of faults in Section III. The main results on decentralized diagnosis are then presented in Sections IV to VI.

## II. PRELIMINARIES

The system is modeled as a finite state automaton  $G = (Q, \Sigma, \delta, q_0)$ , where  $Q$  is the state space,  $\Sigma$  is the set of events,  $\delta$  is the partial transition function, and  $q_0$  is the initial state. The model  $G$  accounts for normal and faulty behavior of the system. The behavior of the system is described by the prefix-closed language  $\mathcal{L}(G)$  generated by  $G$ , denoted often by  $L$  hereafter for the sake of simplicity. The event set is partitioned as  $\Sigma = \Sigma_o \cup \Sigma_{uo}$  for observable and unobservable events, respectively. Let us first assume there is only one fault event  $f \in \Sigma_{uo}$ . We will see later that extension to multiple fault events is straightforward. A string or a trace  $s \in L$  is called *faulty* if it contains  $f$ , i.e., if there exist  $u, v \in \Sigma^*$  such that  $s = ufv$ .  $\bar{s}$  denotes the set of all prefixes of trace  $s$ . We denote by  $L/s$  the post-language of  $L$  after  $s$ , i.e.,  $L/s = \{t | st \in L\}$ .

Given  $P$  the standard projection operation from  $\Sigma^*$  to  $\Sigma_o^*$  that erases unobservable events, we have that  $P^{-1}(s) := \{t \in \Sigma^* : P(t) = s\}$ . We introduce the notation  $\mathcal{E}(s) = P^{-1}P(s) \cap L$  to denote the set of “estimate traces”, assuming  $s$  is executed by the system and  $P(s)$  is observed. Thus  $t \in \mathcal{E}(s)$  iff  $t \in L$  and  $P(t) = P(s)$ . Therefore,  $\mathcal{E}(s)$  is the

This research is supported in part by NSF grants CCR-0082784 and CCR-0325571, by ONR grant N00014-03-1-0232, and by a grant from the Xerox University Affairs Committee.

Y. Wang and S. Lafortune are with the Department of Electrical Engineering and Computer Science, The University of Michigan, 1301 Beal Avenue, Ann Arbor, MI 48109-2122, U.S.A., {yinw, stephane}@eecs.umich.edu

T. Yoo is with Idaho National Laboratory, Idaho Falls, ID 83403-2528, U.S.A., YOOTS@inel.gov

estimate of the behavior of the system consistent with the model  $L$  after  $P(s)$  has been observed.

For the sake of simplicity, we make the following standard assumptions:

**A1**  $\mathcal{L}(G)$  is live;

**A2** Every cycle of  $G$  must contain at least one observable event.

**A1** can be relaxed easily at the expense of extra statements regarding the diagnosability of terminating traces. **A2** ensures that the system will not generate arbitrarily long sequences of unobservable events, which of course would prevent diagnosis within bounded delays.

The following well-known definition [2, 3] is the starting point for our development.

*Definition 1:* Language  $L$  is said to be diagnosable, F-DIAG for short, w.r.t.  $f$  and  $P$  if the following is true:

$(\exists k \in \mathbb{N})(\forall s \in L \text{ s.t. } s \text{ is faulty})(\forall t \in L/s \text{ s.t. } |t| \geq k)(\forall u \in \mathcal{E}(st)) u \text{ is faulty.}$

This definition means the following. Let  $s$  be a faulty trace and  $t$  be a sufficiently long continuation of  $s$  in  $L$ . Then any trace in  $L$  indistinguishable from  $st$  is also faulty. F-DIAG implies that all possible estimate traces of a sufficiently long faulty trace are faulty. Therefore, it is possible to diagnose the fault in  $s$  after observing  $P(st)$ .

### III. DIAGNOSING THE ABSENCE OF FAULTS

F-DIAG characterizes the ability to detect the occurrence of the fault event  $f$  using on-line observations and based on the system model. If we are interested in recognizing traces *not* containing fault event  $f$ , i.e., diagnosing the *absence* of  $f$ , the concept of no-fault diagnosis, NF-DIAG for short, needs to be developed. There are many variations of this concept; see the results in [11, 17]. We choose the following definition for our development because it is equivalent to F-DIAG and has nice properties when it is generalized to the decentralized setting [17].

*Definition 2:* Language  $L$  is said to be NF-DIAG w.r.t.  $f$  and  $P$  if the following is true:

$(\exists k \in \mathbb{N})(\forall s \in L \text{ s.t. } s \text{ is not faulty})(\forall t \in L/s \text{ s.t. } |t| \geq k \text{ and } st \text{ is not faulty})(\forall uv \in \mathcal{E}(st) \text{ s.t. } P(u) = P(s)) u \text{ is not faulty.}$

In words, let  $s$  be a fault-free trace in  $L$  and  $t$  be a sufficiently long fault-free extension of  $s$ . Then any trace that is indistinguishable from  $st$  must be fault-free right after its observed prefix  $P(s)$ . NF-DIAG implies that if the system is running without faults, we are always able to infer that *some events ago*, the system *was* not faulty.

*Example 1:* Consider the system described by the language  $a^*fab^*$ . The only unobservable event is  $f$ . The system is F-DIAG because  $f$  happens *iff*  $b$  is observed at most two events after  $f$ . It is also NF-DIAG. The only fault-free trace is  $st = a^n$ , resulting in  $\mathcal{E}(st) = \{a^n, a^n f, a^{n-1} f a\}$ ,  $u \in \{a^n, a^n f, a^{n-1} f a\}$ . Take  $k = 2$ , thus  $s = a^{n-2}$ . Since  $P(s) = P(u)$ ,  $u$  must be  $a^{n-2}$  as well, a fault-free trace. ■

The above example demonstrates the interesting property that we are only able to infer the absence of faults *in the past*. We have found that the other variations of NF-DIAG

considered in [11, 17] are not able to capture this property and result in strictly smaller language classes.

*Theorem 1:* Language  $L$  is NF-DIAG w.r.t. fault event  $f$  and projection  $P$  if and only if it is F-DIAG w.r.t.  $f$  and  $P$ .

*Proof:*  $\neg$ NF-DIAG  $\Rightarrow$   $\neg$ F-DIAG. Violation of NF-DIAG implies there exists a trace  $uv \in \mathcal{E}(st)$ , where  $u$  is faulty and  $P(u) = P(s)$ . Then  $P(v) = P(t)$  and  $|t| \geq k$ , where integer  $k$  could be arbitrarily large. Since there is no unobservable cycle, both  $v$  and  $t$  can be arbitrarily long. Therefore,  $u$  is faulty with an arbitrarily long extension  $v$ ,  $P(uv) = P(st)$ , where  $st$  is fault-free. Hence the system is not F-DIAG.

The other direction is similar and omitted. ■

Since NF-DIAG is equivalent to F-DIAG, verification algorithms for F-DIAG, including *diagnosers* [2] and *verifiers* [18], can be used to verify NF-DIAG as well. We are particularly interested in the verifier approach because it has polynomial computational complexity and can be easily generalized to decentralized settings [11, 19]. Online diagnosis of the absence of faults can be done by diagnosers. Details of these results are in [17].

### IV. DECENTRALIZED DIAGNOSIS

Let us consider the decentralized architecture depicted in Fig. 1 where there are  $n$  local sites jointly diagnosing the system  $G$  by observing subsets of the set of observable events  $\Sigma_o$ , denoted by  $\Sigma_{o,1}, \dots, \Sigma_{o,n}$ , respectively. The blocks  $P_1, \dots, P_n$  in the figure denote the projection operations from  $\Sigma^*$  to  $\Sigma_{o,i}^*$ . The decision fusion block in Fig. 1 is assumed to be a simple memoryless Boolean function that merges the diagnosis decisions of the local sites. As was mentioned in the introduction, we do *not* consider more complicated decision fusion blocks such as “coordinators” that would receive state estimates from local sites and process them in order to compute online the overall diagnosis decisions (cf. Protocols 1 and 2 of [4]). In contrast, our objective is to study the properties of decentralized architectures with the simplest possible types of fusion of local, possibly conditional, decisions.

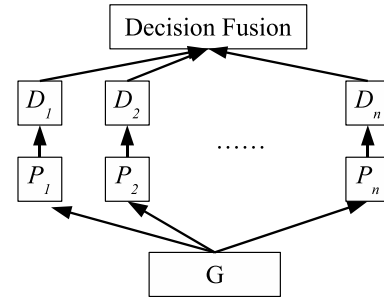


Fig. 1. Decentralized Architecture

The notions of projection and estimate set are extended to the above decentralized setting in a natural way.  $P_i^{-1}(s) := \{t \in \Sigma^* : P_i(t) = s\}$ ,  $\mathcal{E}_i(s) = P_i^{-1}P_i(s) \cap L$ .

The following definition of decentralized diagnosis is the starting point for our development.

*Definition 3:* Language  $L$  is said to be F-codiagnosable, or F-CODIAG, w.r.t.  $f, P_1, \dots, P_n$ , if the following is true:

$(\exists k \in \mathbb{N})(\forall s \in L \text{ s.t. } s \text{ is faulty})(\forall t \in L/s \text{ s.t. } |t| \geq k)(\exists i \in \{1, \dots, n\})(\forall u \in \mathcal{E}_i(st)) u \text{ is faulty.}$

In words, let  $s$  be a faulty trace and let  $t$  be a sufficiently long continuation of  $s$  in  $L$ . Then there must exist at least one local site  $i$  such that any trace in  $L$  indistinguishable from  $st$  at site  $i$  is also faulty. This definition is exactly the same as the definition in [4] of “diagnosability under Protocol 3,” which is revisited in [11] under the name “co-diagnosability.” We adopt here the name “F-codiagnosability” in order to facilitate comparisons between our work and that in [15, 16] for *coobservability* and decentralized control. It is important to note that in F-CODIAG, the only local decision made by diagnosers is “Fault,” and the system is diagnosed to be faulty if and only if there is at least one diagnoser reporting “Fault.” Thus, this architecture is closely analogous to the conjunctive architecture considered in [15, 20] for decentralized control, where “disable” is the only local decision employed and an event is disabled if at least one site disables it. In the next section, we will consider the dual problem of detecting the absence of faults in a decentralized setting and introduce the corresponding notion of NF-codiagnosability.

## V. DECENTRALIZED DIAGNOSIS: ABSENCE OF FAULTS

### A. Notions of Codiagnosability

Let us first look at a motivating example.

*Example 2:* Consider the system described by the language  $(f + a + b)c^*$ , where  $\Sigma_o = \{a, b, c\}$  and  $\Sigma_{uo} = \{f\}$ . There are two local sites,  $n = 2$ ,  $\Sigma_{o,1} = \{a, c\}$  and  $\Sigma_{o,2} = \{b, c\}$ . This system is not F-CODIAG because the arbitrarily long faulty trace  $fc^n$  is indistinguishable from fault-free trace  $bc^n$  at site 1 and indistinguishable from  $ac^n$  at site 2. Recall that in the decentralized architecture corresponding to F-CODIAG, sites are only allowed to issue “Fault” decisions. A faulty trace can therefore be diagnosed only if some site is certain about the occurrence of the fault. In this example, to diagnose faulty trace  $fc^n$ , cooperation between the two sites would be necessary. ■

We observe that the fault-free traces in Example 2 can be detected with certainty by the local sites. For instance, observation of event  $a$  at site 1 is an indication that fault event  $f$  has not occurred. Inspired by this observation, as well as by the notion of “disjunctive architectures” for decentralized supervisory control introduced in [15], we propose the related concept of NF-codiagnosability, which allows local sites to say “No Fault”. This leads to the following definition.

*Definition 4:* Language  $L$  is said to be NF-codiagnosable, or NF-CODIAG, w.r.t.  $f, P_1, \dots, P_n$ , if the following is true:

$(\exists k \in \mathbb{N})(\forall s \in L \text{ s.t. } s \text{ is not faulty})(\forall t \in L/s \text{ s.t. } |t| \geq k \text{ and } st \text{ is not faulty})(\exists i \in \{1, \dots, n\})(\forall uv \in \mathcal{E}_i(st)) P_i(u) = P_i(s) u \text{ is not faulty.}$

This definition is related to the ability to detect the *absence* of a fault, i.e., if trace  $s$  is not faulty, and  $t$  is a sufficiently long fault-free extension in  $L$ , there must exist one local site  $i$  such that any trace in  $L$  indistinguishable from  $st$  at site  $i$  was also fault-free up to the observation of  $P(s)$ .

Definition 4 is the extension to the decentralized setting of NF-DIAG, introduced in Definition 2. We note that based on a variation of NF-DIAG, a similar notion of decentralized diagnosis of absence of faults was independently proposed in [11] and termed “strong codiagnosability”, which results in a stronger notion than that in Definition 4 [17].

It is not difficult to verify that the system in Example 2 above is NF-CODIAG. The fault-free traces with sufficiently long extensions are  $ac^n$  and  $bc^n$ ,  $n \geq 0$ , and each one will unambiguously be detected by sites 1 and 2, respectively.

Consider next the situation where instead of a single fault event  $f$ , there is a set of fault events denoted by  $\Sigma_f \subseteq \Sigma_{uo}$ . Assume there are  $m$  fault events,  $\Sigma_f = \{f_1, \dots, f_m\}$ . A trace  $s$  is called  $f_i$ -faulty if it contains fault event  $f_i$ . Definitions 3 and 4 are extended to this situation in the following manner.

*Definition 5:* Language  $L$  is said to be F-CODIAG w.r.t.  $f_1, \dots, f_m, P_1, \dots, P_n$ , if the following is true:

$(\forall j \in \{1, \dots, m\})(\exists k_j \in \mathbb{N})(\forall s \in L \text{ s.t. } s \text{ is } f_j\text{-faulty})(\forall t \in L/s \text{ s.t. } |t| \geq k_j)(\exists i \in \{1, \dots, n\})(\forall u \in \mathcal{E}_i(st)) u \text{ is } f_j\text{-faulty.}$

*Definition 6:* Language  $L$  is said to be NF-CODIAG w.r.t.  $f_1, \dots, f_m, P_1, \dots, P_n$ , if the following is true:

$(\forall j \in \{1, \dots, m\})(\exists k_j \in \mathbb{N})(\forall s \in L \text{ s.t. } s \text{ is not } f_j\text{-faulty})(\forall t \in L/s \text{ s.t. } |t| \geq k_j \text{ and } st \text{ is not } f_j\text{-faulty})(\exists i \in \{1, \dots, n\})(\forall uv \in \mathcal{E}_i(st) \text{ s.t. } P_i(u) = P_i(s)) u \text{ is not } f_j\text{-faulty.}$

If every fault event in  $\Sigma_f$  is F[NF]-CODIAG, then we say that the system is F[NF]-CODIAG. However, it is possible that some fault events will be F-CODIAG while others will be NF-CODIAG. To account for this situation, we introduce the notion of *codiagnosability*. Inspired by the notion of coobservability in the context of the “general architecture” in [15], we partition  $\Sigma_f$  as  $\Sigma_f = \Sigma_{f,F} \cup \Sigma_{f,NF}$ , where  $\Sigma_{f,F}$  is the set of fault events whose occurrence can be diagnosed and  $\Sigma_{f,NF}$  is the set of fault events whose absence can be diagnosed.

*Definition 7:* Language  $L$  is said to be codiagnosable w.r.t.  $\Sigma_{f,F}, \Sigma_{f,NF}, P_1, \dots, P_n$ , if

1.  $L$  is F-CODIAG w.r.t.  $\Sigma_{f,F}, P_1, \dots, P_n$ ;
2.  $L$  is NF-CODIAG w.r.t.  $\Sigma_{f,NF}, P_1, \dots, P_n$ .

### B. Properties of Codiagnosability

*Theorem 2:* F-CODIAG and NF-CODIAG are incomparable w.r.t. the same fault event and projections  $P_1, \dots, P_n$ .

*Proof:* One part of the theorem is proved by Example 2 above; the other part is proved by Example 3 below. ■

*Example 3:* Consider the system described by the language  $c^*f(a + b)c^*$ , where  $\Sigma_o = \{a, b, c\}$  and  $\Sigma_{uo} = \Sigma_f = \{f\}$ . There are two local sites with  $\Sigma_{o,1} = \{a, c\}$  and  $\Sigma_{o,2} = \{b, c\}$ . The system is F-CODIAG because faulty traces in  $c^*fac^*$  and  $c^*fbc^*$  will be unambiguously detected by sites 1 and 2, respectively. It is not NF-CODIAG because arbitrarily long fault-free trace  $c^n$  is indistinguishable from faulty trace  $fbcn$  at site 1 and indistinguishable from faulty trace  $fac^n$  at site 2. ■

*Theorem 3:* F-CODIAG or NF-CODIAG implies codiagnosability w.r.t. the same set of fault events and projections.

The reverse implication is not true in general.

*Proof:* The first part of the theorem is obvious from the respective definitions; the other part is proved by Example 4 below. ■

*Example 4:* Consider the system  $G$  shown in Fig. 2, where  $\Sigma_o = \{a_1, a_2, b_1, b_2, c_1, c_2\}$ ,  $\Sigma_{uo} = \Sigma_f = \{f_1, f_2\}$ , and where  $\Sigma_f$  is partitioned into two fault events,  $\Sigma_{f,F} = \{f_1\}$ ,  $\Sigma_{f,NF} = \{f_2\}$ . There are two local sites with  $\Sigma_{o,1} = \{a_1, a_2, c_1, c_2\}$  and  $\Sigma_{o,2} = \{b_1, b_2, c_1, c_2\}$ . From Examples 2 and 3, we know that the system is codiagnosable with  $f_1$  F-CODIAG and  $f_2$  NF-CODIAG. It is neither F-CODIAG nor NF-CODIAG for both fault events. ■

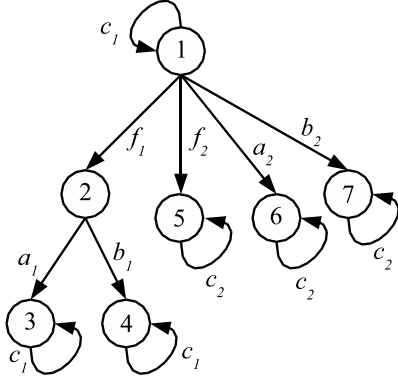


Fig. 2. Codiagnosable but not F(NF)-CODIAG

*Theorem 4:* Codiagnosability w.r.t.  $\Sigma_{f,F}$ ,  $\Sigma_{f,NF}$ ,  $\Sigma_{o,1}$ , ...,  $\Sigma_{o,n}$  implies centralized diagnosability w.r.t. every fault event in  $\Sigma_{f,F} \cup \Sigma_{f,NF}$  and projection corresponding to  $\Sigma_o = \Sigma_{o,1} \cup \dots \cup \Sigma_{o,n}$ . The reverse implication is not true in general.

*Proof:* By definition, fault events that are F-CODIAG are also F-DIAG. Similarly, fault events that are NF-CODIAG are NF-DIAG. Since NF-DIAG equals F-DIAG by Theorem 1, codiagnosability implies diagnosability. The other part is proved by Example 5. ■

*Example 5:* Consider the system described by the language  $abc^* + bac^*$ , where  $\Sigma_o = \{a, b, c\}$  and  $\Sigma_{uo} = \Sigma_f = \{f\}$ . There are two local sites with  $\Sigma_{o,1} = \{a, c\}$  and  $\Sigma_{o,2} = \{b, c\}$ . The system is not codiagnosable because whether  $f$  happens or not, site 1 always observes  $ac^*$  and site 2 always observes  $bc^*$ . In a centralized setting however, it is clearly diagnosable. ■

### C. Verification of Codiagnosability

The verification of codiagnosability (especially NF-CODIAG) can be done by extending verifiers [18] to the decentralized setting and building on the results in [11] for F-CODIAG.

Assume system  $G = (Q, \Sigma, \delta, q_0)$  is to be diagnosed by two local sites (for the sake of simplicity) with observable event sets  $\Sigma_{o,1}$  and  $\Sigma_{o,2}$ , respectively. We construct verifier  $V_{dec} = Acc(Q^{V_{dec}}, \Sigma, \delta^{V_{dec}}, q_0^{V_{dec}})$  for a single fault event  $f$

as follows, where  $Acc$  stands for taking the accessible part.

$$Q^{V_{dec}} = \underbrace{Q \times \{N, F\}}_{s_1} \times \underbrace{Q \times \{N, F\}}_{s_2} \times \underbrace{Q \times \{N, F\}}_s$$

$$q_0^{V_{dec}} = (q_0, N, q_0, N, q_0, N)$$

For the sake of readability, let  $q'_i = \delta(q_i, \sigma)$ . The transition relation  $\delta^{V_{dec}}$  is defined as described below, for all cases where the corresponding transitions are defined:

$$\begin{aligned} &\text{For } \sigma \in \Sigma_{o,1}, \sigma \in \Sigma_{o,2}, \\ &\delta^{V_{dec}}((q_1, l_1, q_2, l_2, q_3, l_3), \sigma) = \{(q'_1, l_1, q'_2, l_2, q'_3, l_3)\} \\ &\text{For } \sigma \in \Sigma_{o,1}, \sigma \notin \Sigma_{o,2}, \\ &\delta^{V_{dec}}((q_1, l_1, q_2, l_2, q_3, l_3), \sigma) = \begin{cases} (q'_1, l_1, q_2, l_2, q'_3, l_3) \\ (q_1, l_1, q'_2, l_2, q_3, l_3) \end{cases} \\ &\text{For } \sigma \notin \Sigma_{o,1}, \sigma \in \Sigma_{o,2}, \\ &\delta^{V_{dec}}((q_1, l_1, q_2, l_2, q_3, l_3), \sigma) = \begin{cases} (q_1, l_1, q_2, l_2, q'_3, l_3) \\ (q'_1, l_1, q_2, l_2, q_3, l_3) \end{cases} \\ &\text{For } \sigma \in \Sigma_{uo} \text{ and } \sigma \neq f, \\ &\delta^{V_{dec}}((q_1, l_1, q_2, l_2, q_3, l_3), \sigma) = \begin{cases} (q'_1, l_1, q_2, l_2, q_3, l_3) \\ (q_1, l_1, q'_2, l_2, q_3, l_3) \\ (q_1, l_1, q_2, l_2, q'_3, l_3) \end{cases} \\ &\text{For } \sigma = f, \\ &\delta^{V_{dec}}((q_1, l_1, q_2, l_2, q_3, l_3), \sigma) = \begin{cases} (q'_1, F, q_2, l_2, q_3, l_3) \\ (q_1, l_1, q'_2, F, q_3, l_3) \\ (q_1, l_1, q_2, l_2, q'_3, F) \end{cases} \end{aligned}$$

The verifier simulates three traces  $s_1$ ,  $s_2$  and  $s$ , where  $s$  indicates the trace the system actually executes and  $s_i$ ,  $i = 1, 2$ , represents the trace that site  $i$  estimates. It satisfies  $P_1(s_1) = P_1(s)$  and  $P_2(s_2) = P_2(s)$ . The construction of the transition rules is such that it captures all possible trace triples  $(s_1, s_2, s)$  that satisfy  $P_1(s_1) = P_1(s)$  and  $P_2(s_2) = P_2(s)$ . A verifier state  $(q_1, l_1, q_2, l_2, q_3, l_3)$  is called a  $(l_1, l_2, l_3)$ -state. For example, the initial state  $q_0^{V_{dec}}$  is an  $(N, N, N)$ -state. A cycle is called an  $(l_1, l_2, l_3)$ -cycle if every state in the cycle is an  $(l_1, l_2, l_3)$ -state.

The above construction can be extended to  $n$  local sites naturally. Basically, we need to simulate  $n+1$  traces and thus the state has  $n+1$  components; there are  $2^{n+1} \times |Q|^{n+1}$  states at most. At each state, event  $\sigma$  has at most  $n+1$  transitions by the transition rules, resulting in  $2^{n+1} \times |Q|^{n+1} \times |\Sigma| \times (n+1)$  transitions at most. So the size of the verifier is polynomial in the number of system states and exponential in the number of local sites. For the case of multiple faults, we build a separate verifier for each fault.

Testing of F-CODIAG or NF-CODIAG using the verifier is based on the following theorem.

*Theorem 5:*  $\mathcal{L}(G)$  is not F-CODIAG w.r.t.  $f$  if and only if  $V_{dec}$  of  $G$  has an  $(N, N, F)$ -cycle.  $\mathcal{L}(G)$  is not NF-CODIAG w.r.t.  $f$  if and only if  $V_{dec}$  has an  $(F, F, N)$ -cycle.

*Proof:* Following the same strategy as in the proof of Theorem 1 in [11]<sup>1</sup>, it can be proved that we can extract a trace triple  $(s_1, s_2, s)$  from a path in  $V_{dec}$  by the transition rules. The trace triple reaches state  $(q_1, l_1, q_2, l_2, q_3, l_3)$  in  $V_{dec}$  if and only if:

<sup>1</sup>There is a technical difference in that fault languages instead of fault events are used to characterize faulty behaviors in [11].

1.  $s_1, s_2$  and  $s$  reach states  $q_1, q_2$  and  $q_3$  in  $G$ , respectively;
2.  $s_1$  ( $s_2$  or  $s$ ) is faulty if and only if  $l_1$  ( $l_2$  or  $l_3$ ) =  $F$ ;
3.  $P_1(s_1) = P_1(s)$  and  $P_2(s_2) = P_2(s)$ .

Based on this result, we complete the proof as follows.

(i) and (ii) (N,N,F)-cycle  $\Leftrightarrow$  not F-CODIAG. The proof of this part is similar to the proof of Theorem 1 in [11] and therefore omitted.

(iii) (F,F,N)-cycle  $\Rightarrow$  not NF-CODIAG. In  $V_{dec}$ , an (F,F,N)-cycle means an arbitrarily long path from  $q_0^{V_{dec}}$ . By the above analysis, we know that this implies the existence of three traces  $s_1t_1^n, s_2t_2^n, st^n$ , where trace triple  $(s_1, s_2, s)$  corresponds to the prefix of the path that reaches the cycle from the initial state and  $(t_1, t_2, t)$  corresponds to the cycle. Furthermore,  $st^n$  is fault-free, and  $s_1$  and  $s_2$  are faulty. Since  $P_1(s_1t_1^n) = P_1(st^n)$ ,  $P_2(s_2t_2^n) = P_2(st^n)$ , fault-free trace  $st^n$  cannot be diagnosed by either site.

(iv) Not NF-CODIAG  $\Rightarrow$  (F,F,N)-cycle. Not NF-CODIAG means there is a fault-free trace  $st$ ,  $t$  is arbitrarily long, and faulty traces  $u_1$  and  $u_2$  with extensions  $v_1$  and  $v_2$  such that  $P_1(u_1v_1) = P_1(st)$  and  $P_2(u_2v_2) = P_2(st)$ . By the above result, these three traces should form a path in  $V_{dec}$ . Since  $t$  could be arbitrarily long and  $V_{dec}$  has only a finite number of states, there must be a cycle. Then  $u_1, u_2$  faulty and  $st$  fault-free imply that this cycle is an (F,F,N)-cycle. ■

## VI. DECENTRALIZED DIAGNOSIS WITH CONDITIONAL DECISIONS

In the architecture considered in Section V, each local site makes “Fault” or “No Fault” decisions, and the global decision fusion block simply takes the disjunction of these local decisions. (In fact, no such fusion block is actually needed.) Under this architecture, Example 2 is NF-CODIAG but not F-CODIAG, which means that only fault-free traces can be detected with certainty. To diagnose faults in Example 2, we consider a decentralized diagnosis architecture where local diagnosis engines are allowed to make conditional decisions such as “Fault if nobody says No Fault” and “No Fault if nobody says Fault”. In analogy with [16], this architecture is called the conditional architecture. The global decision fusion block merges decentralized unconditional and conditional decisions. Inspired by the work in [16], we adopt the decision rules indicated in Table I.

As can be seen from Cases 3-8 in Table I, the conditional decisions “Fault if nobody says No Fault” and “No Fault if nobody says Fault” can be interpreted as “Fault” and “No Fault” decisions, respectively, but with lower priority. Namely, these conditional decisions take effect only if the other sites are silent. The unconditional decisions “Fault” and “No Fault” override conditional decisions. There is a diagnosis conflict if and only if contradictory decisions of the same priority occur, i.e., contradictory unconditional decisions or contradictory conditional decisions. The properties of conditional diagnosability introduced in the next section will, by their very definitions, ensure that no such diagnosis conflicts occur.

### A. Notions of Conditional Codiagnosability

To draw parallels with the previous section and the results in [16], we start by considering diagnosability properties associated with two special cases of the conditional architecture described in Table I: *conditional F-codiagnosability* for the so-called conditional F-architecture and *conditional NF-codiagnosability* for the so-called conditional NF-architecture.

Under the conditional F-architecture, local sites have three types of decisions to choose from: “Fault”, “No Fault”, and “Fault if nobody says No Fault”. The fusion rules correspond to cases 1, 2, 3, 4, 5 and 9 in Table I.

*Definition 8:* Language  $L$  is said to be conditionally F-codiagnosable, or COND-F-CODIAG, w.r.t.  $f, P_1, \dots, P_n$ , if the following is true:

$$(\exists k \in \mathbb{N})(\forall s \in L \text{ s.t. } s \text{ is faulty})(\forall t \in L/s \text{ s.t. } |t| \geq k)(\exists i \in \{1, \dots, n\})(\forall uv \in \mathcal{E}_i(st) \text{ s.t. } P_i(u) = P_i(s) \text{ and } uv \text{ is not faulty})(\exists j \in \{1, \dots, n\})(\forall xy \in \mathcal{E}_j(uw) \text{ s.t. } P_j(x) = P_j(u)) \text{ } x \text{ is not faulty.}$$

In words, this definition means the following. For each sufficiently long faulty trace  $st$ , there is a site  $i$  for which  $st$  might have the same projection as fault-free trace  $uv$ , but for every such fault-free trace  $uv$  that belongs to site  $i$ 's estimate, there is a site  $j$  that can ensure that the system was fault-free up to its observation of  $u$ . That is, site  $i$  can infer that if a fault-free trace  $u$ , instead of  $s$ , has happened, there is another site,  $j$ , that can recognize fault-free trace  $u$  with certainty. Therefore, site  $i$  can use the “Fault if nobody says No Fault” decision and site  $j$  will issue the “No Fault” decision overriding site  $i$  if  $u$  was the trace that the system actually executed.

Under the dual conditional NF-architecture, local sites have three types of decisions to choose from: “No Fault”, “Fault”, and “No Fault if nobody says Fault”. The fusion rules correspond to cases 1, 2, 6, 7, 8 and 9 in Table I.

*Definition 9:* Language  $L$  is said to be conditionally NF-codiagnosable, or COND-NF-CODIAG, w.r.t.  $f, P_1, \dots, P_n$ , if the following is true:

$$(\exists k \in \mathbb{N})(\forall s \in L \text{ s.t. } s \text{ is not faulty})(\forall t \in L/s \text{ s.t. } |t| \geq k \text{ and } st \text{ is not faulty})(\exists i \in \{1, \dots, n\})(\forall uv \in \mathcal{E}_i(st) \text{ s.t. } P_i(u) = P_i(s) \text{ and } u \text{ is faulty})(\exists j \in \{1, \dots, n\})(\forall w \in \mathcal{E}_j(uw)) \text{ } w \text{ is faulty.}$$

Here, for each sufficiently long fault-free trace  $st$ , there is a site  $i$  for which  $st$  might have the same projection as trace  $uv$ , where  $u$  is faulty. But for every such faulty trace  $u$  that belongs to site  $i$ 's estimate, there is a site  $j$  that can ensure that  $uv$  is faulty. That is, site  $i$  can infer that if faulty trace  $u$ , instead of  $s$ , has happened, there is another site,  $j$ , that can recognize faulty trace  $u$  with certainty. Therefore, site  $i$  can use the “No Fault if nobody says Fault” decision and site  $j$  will issue the “Fault” decision overriding site  $i$  if  $u$  has actually happened.

The two preceding definitions can be extended in a straightforward manner to the case of multiple faults, as was done in Definitions 5 and 6. We omit these definitions here and proceed directly to the case of conditional codiagnosability, the conditional version of Definition 7. Let us again

Case	Local Decision 1	Local Decision 2	Global Decision	Architecture
1	Fault	Nothing	Fault	F-CODIAG
2	No Fault	Nothing	No Fault	NF-CODIAG
3	Fault if nobody says No Fault	Nothing	Fault	COND-F-CODIAG
4	Fault if nobody says No Fault	Fault	Fault	
5	Fault if nobody says No Fault	No Fault	No Fault	COND-NF-CODIAG
6	No Fault if nobody says Fault	Nothing	No Fault	
7	No Fault if nobody says Fault	Fault	Fault	
8	No Fault if nobody says Fault	No Fault	No Fault	
9	Nothing	Nothing	Nothing	
10	Fault	No Fault	Diagnosis-conflict	
11	Fault if nobody says No Fault	No Fault if nobody says Fault	Diagnosis-conflict	

TABLE I

LOCAL DECISIONS AND THEIR FUSION IN DIFFERENT ARCHITECTURES

partition  $\Sigma_f$  as  $\Sigma_f = \Sigma_{f,F} \cup \Sigma_{f,NF}$ , where  $\Sigma_{f,F}$  is the set of fault events whose occurrence can be diagnosed and  $\Sigma_{f,NF}$  is the set of fault events whose absence can be diagnosed.

*Definition 10:* Language  $L$  is said to be conditionally codiagnosable w.r.t.  $\Sigma_{f,F}, \Sigma_{f,NF}, P_1, \dots, P_n$ , if

1.  $L$  is COND-F-CODIAG w.r.t.  $\Sigma_{f,F}, P_1, \dots, P_n$ ;
2.  $L$  is COND-NF-CODIAG w.r.t.  $\Sigma_{f,NF}, P_1, \dots, P_n$ .

### B. Properties of Conditional Codiagnosability

*Theorem 6:* If language  $L$  is codiagnosable w.r.t.  $\Sigma_{f,F}, \Sigma_{f,NF}, P_1, \dots, P_n$ , then it is COND-F-CODIAG and COND-NF-CODIAG w.r.t.  $f, P_1, \dots, P_n, \forall f \in \Sigma_{f,F} \cup \Sigma_{f,NF}$ . The reverse is not true in general.

*Proof:* The forward direction can be proved by showing that F-CODIAG faults or NF-CODIAG faults are both COND-F-CODIAG and COND-NF-CODIAG.

(i) F-CODIAG implies COND-F-CODIAG by definition, i.e. site  $i$  itself recognizes faulty trace  $st$ .

(ii) F-CODIAG implies COND-NF-CODIAG. F-CODIAG means there is an integer  $k$  such that for every faulty trace  $s$ , extension  $t$ ,  $|t| \geq k$ , there exists site  $j$ , whose estimate  $\mathcal{E}_j(st)$  contains only faulty traces. By assumption, there is no unobservable cycle; let  $d$  be the maximum number of successive unobservable events. To see that the system is COND-NF-CODIAG, let  $uv$  be a fault-free trace,  $|v| \geq nk(d+1)$ . Thus  $v$  contains at least  $nk$  observable events, not necessarily observed by one site though. However, by the Pigeonhole principle, there exists a site  $i$  observing at least  $k$  events of them. So  $P_i(v) \geq k, \forall st \in \mathcal{E}_i(uv), P_i(s) = P_i(u)$  and  $P_i(t) = P_i(v) \geq k, |t| \geq k$ . If  $s$  is faulty,  $st$  must be recognized by a site  $j$  because of F-CODIAG, i.e.,  $\mathcal{E}_j(st)$  contains only faulty traces. Therefore, by definition, the system is COND-F-CODIAG.

(iii) and (iv) NF-CODIAG implies both COND-F-CODIAG and COND-NF-CODIAG. The proof is similar and omitted.

The reverse direction that COND-F-CODIAG or COND-NF-CODIAG do not imply codiagnosability is proved by Examples 6 and 7. ■

*Example 6:* Consider the system  $G$  shown in Fig. 3, with two local sites,  $\Sigma_{o,1} = \{a_1, a_2, c\}$ ,  $\Sigma_{o,2} = \{b_1, b_2, c\}$  and  $\Sigma_{uo} = \Sigma_f = \{f\}$ . The system is not F-CODIAG because faulty trace  $b_1fc^n$  is indistinguishable from  $c^n$  at site 1 and

indistinguishable from  $b_1a_2c^n$  at site 2. It is not NF-CODIAG because fault-free trace  $c^n$  is indistinguishable from  $b_1fc^n$  at site 1 and indistinguishable from  $a_1fc^n$  at site 2. The system is COND-F-CODIAG however, because if faulty trace  $a_1fc^n$  has happened, the estimate by site 1 is  $a_1fc^n$  itself or  $a_1b_2c^n$ , but if  $a_1b_2c^n$  has happened, site 2 would know it for sure. Therefore, the fault can be diagnosed this way: site 1 says ‘‘Fault if nobody says No Fault’’ once it sees  $a_1$ , and site 2 says ‘‘No Fault’’ to override site 1 if it sees  $b_2$ . Similarly, faulty trace  $b_1fc^n$  can be diagnosed. ■

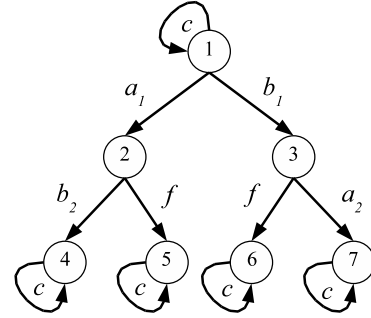


Fig. 3. The system of Example 6

*Example 7:* In Fig. 4, there are two local sites.  $\Sigma_{o,1} = \{a_1, a_2, c\}$ ,  $\Sigma_{o,2} = \{b_1, b_2, c\}$  and  $\Sigma_{uo} = \Sigma_f = \{f\}$ . Similarly with Example 6, the system can be shown to be COND-NF-CODIAG but not F-CODIAG or NF-CODIAG. ■

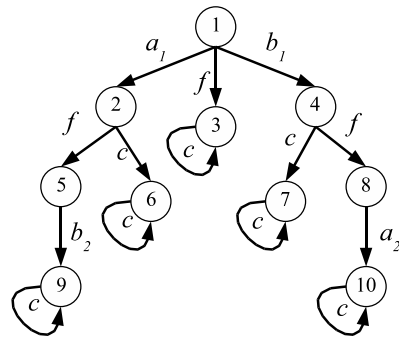


Fig. 4. The system of Example 7

*Theorem 7:* COND-F-CODIAG and COND-NF-CODIAG are incomparable w.r.t. the same fault event and local projections.

*Proof:* The system in Example 6 is COND-F-CODIAG but not COND-NF-CODIAG. The problem fault-free trace is  $c^n$ ; it is indistinguishable from  $b_1fc^n$  at site 1 but unfortunately site 2 cannot help on this faulty trace since it is indistinguishable from  $b_1a_2c^n$  at site 2. Similarly  $c^n$  cannot be diagnosed by site 2 conditionally.

The other part is proved in a similar way by Example 7. ■

*Theorem 8:* COND-F-CODIAG or COND-NF-CODIAG implies conditional codiagnosability with the same fault events and projections. The reverse implication is not true in general.

*Proof:* The forward direction is true by definition. The reverse part can be proved by a counter-example, whose construction is similar with Example 4 and omitted. ■

*Theorem 9:* Conditional codiagnosability w.r.t.  $\Sigma_{f,F}$ ,  $\Sigma_{f,NF}$ ,  $\Sigma_{o,1}, \dots, \Sigma_{o,n}$  implies centralized diagnosability w.r.t. every fault event in  $\Sigma_{f,F} \cup \Sigma_{f,NF}$  and projection corresponding to  $\Sigma_o = \Sigma_{o,1} \cup \dots \cup \Sigma_{o,n}$ . The reverse implication is not true in general.

The proof and the counter-example are similar with those for Theorem 4 and omitted.

In conclusion, the relationship among the different notions of codiagnosability introduced above is shown in Fig. 5, where a directed arc indicates “implies”.

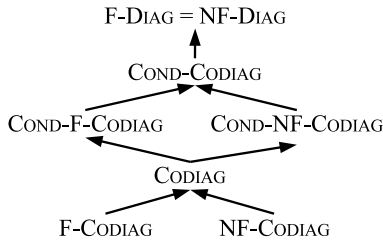


Fig. 5. Relationship among notions of codiagnosability

### C. Discussion

It can be shown that the technique presented in Section V-C for verifying (unconditional) codiagnosability can be extended to develop polynomial time algorithms for testing conditional codiagnosability. The details are omitted due to lack of space. The synthesis of special types of diagnosers to implement conditional decisions is a more intricate problem and is not discussed in this paper.

## VII. CONCLUSION

This paper has outlined the main features of a strategy for performing decentralized diagnosis of DES using architectures where local sites can issue several types of diagnosis decisions about the presence or absence of each fault, including so-called conditional decisions of the type “Fault if nobody says No Fault” and “No Fault if nobody

says Fault”. The use of such decentralized architectures allows for diagnosing larger classes of systems that can be diagnosed under the decentralized architecture corresponding to Protocol 3 in [4]. Moreover, the various notions of codiagnosability that characterize these new architectures are verifiable in polynomial time in the size of the state space of the system.

## REFERENCES

- [1] S. Lafortune, D. Teneketzis, M. Sampath, R. Sengupta, and K. Sinamohideen, “Failure diagnosis of dynamic systems: An approach based on discrete event systems,” in *Proc. 2001 American Control Conf.*, June 2001, pp. 2058–2071.
- [2] M. Sampath, R. Sengupta, K. S. S. Lafortune, and D. Teneketzis, “Diagnosability of discrete event systems,” *IEEE Trans. Automat. Contr.*, vol. 40, no. 9, pp. 1555–1575, September 1995.
- [3] —, “Failure diagnosis using discrete event models,” *IEEE Trans. Contr. Syst. Technol.*, vol. 4, no. 2, pp. 105–124, March 1996.
- [4] R. Debouk, S. Lafortune, and D. Teneketzis, “Coordinated decentralized protocols for failure diagnosis of discrete-event systems,” *Discrete Event Dynamic Systems: Theory and Applications*, vol. 10, no. 1-2, pp. 33–86, Jan. 2000.
- [5] A. Benveniste, S. Haar, E. Fabre, and C. Jard, “Distributed and asynchronous discrete event systems diagnosis,” in *Proc. 41st IEEE Conf. on Decision and Control*, Dec. 2003, pp. 3742–3747.
- [6] R. Boel and J. van Schuppen, “Decentralized failure diagnosis for discrete-event systems with costly communication between diagnosers,” in *Proc. of the 2002 International Workshop on Discrete Event Systems - WODES’02*, Zaragoza, Spain, Oct. 2002.
- [7] R. K. Boel and G. Jiroveanu, “Distributed contextual diagnosis for very large systems,” in *Proc. of the 2004 International Workshop on Discrete Event Systems - WODES’04*, Reims, France, 2004.
- [8] E. Fabre, A. Benveniste, C. Jard, L. Ricker, and M. Smith, “Distributed state reconstruction for discrete event systems,” in *Proc. 39th IEEE Conf. on Decision and Control*, Dec. 2000, pp. 2252–2257.
- [9] S. Genc and S. Lafortune, “A distributed algorithm for on-line diagnosis of place-bordered petri nets,” in *Proc. of 16th IFAC World Congress*, 2005.
- [10] G. Lamperti and M. Zanella, *Diagnosis of active systems: principles and techniques*. Kluwer Academic Publishers, 2003.
- [11] W. Qiu and R. Kumar, “Decentralized failure diagnosis of discrete event systems,” in *Proc. of the 2004 International Workshop on Discrete Event Systems - WODES’04*, Reims, France, 2004.
- [12] R. Sengupta and S. Tripakis, “Decentralized diagnosability of regular languages is undecidable,” in *Proc. 40th IEEE Conf. on Decision and Control*, Dec. 2002, pp. 423–428.
- [13] R. Su, W. Wonham, J. Kurien, and X. Koutsoukos, “Distributed diagnosis for qualitative systems,” in *Proc. of the 2002 International Workshop on Discrete Event Systems - WODES’02*, Zaragoza, Spain, Oct. 2002, pp. 169–174.
- [14] R. Su and W. Wonham, “Distributed diagnosis under global consistency,” in *Proc. 42nd IEEE Conf. on Decision and Control*, Dec. 2004.
- [15] T. Yoo and S. Lafortune, “A general architecture for decentralized supervisory control of discrete-event systems,” *Discrete Event Dynamic Systems: Theory and Applications*, vol. 12, no. 3, pp. 335–377, July 2002.
- [16] —, “Decentralized supervisory control with conditional decisions: supervisor existence,” *IEEE Trans. Automat. Contr.*, vol. 49, no. 11, pp. 1886–1904, Nov. 2004.
- [17] Y. Wang and S. Lafortune, “Decentralized diagnosis of discrete event systems: architectures based on unconditional and conditional decisions,” University of Michigan, Ann Arbor, MI, Tech. Rep. CGR-05-01, Jan. 2005.
- [18] T. Yoo and S. Lafortune, “Polynomial-time verification of diagnosability of partially-observed discrete-event systems,” *IEEE Trans. Automat. Contr.*, vol. 47, no. 9, pp. 1491–1495, September 2002.
- [19] K. Rudie and J. C. Willems, “The computational complexity of decentralized discrete-event control problems,” *IEEE Trans. Automat. Contr.*, vol. 40, no. 7, pp. 1313–1318, July 1995.
- [20] K. Rudie and W. M. Wonham, “Think globally, act locally: decentralized supervisory control,” *IEEE Trans. Automat. Contr.*, vol. 37, no. 11, pp. 1692–1708, November 1992.