Decentralized Failure Diagnosis of Discrete Event Systems

Wenbin Qiu, *Student Member, IEEE,* and Ratnesh Kumar, *Senior Member, IEEE*

***Abstract*— By decentralized diagnosis we mean diagnosis using multiple diagnosers, each possessing it's own set of sensors, without involving any communication among diagnosers or to any coordinators. We formalize the notion of decentralized diagnosis by introducing the notion of *codiagnosability* that requires that a failure be detected by one of the diagnosers within a bounded delay. We give algorithms of complexity polynomial in the size of the system and the non-fault specification for (i) testing codiagnosability, (ii) computing the bound in delay of diagnosis, (iii) off-line synthesis of individual diagnosers, and (iv) on-line diagnosis using them. The notion of codiagnosability and the above algorithms are initially presented in a setting of a specification language (violation of which represents a fault), and are later specialized to the case where faults are modeled as the occurrences of certain events. We also introduce the notion of strong-codiagnosability to capture the ability of being certain about both the failure as well as the non-failure conditions in a system within bounded delay.**

***Index Terms*— Discrete event systems, Failure diagnosis, Decentralized diagnosis, Diagnosability, Codiagnosability.**

I. Introduction

In this paper, we study the decentralized failure diagnosis of discrete event systems (DESs)—systems with discrete states that change when certain events occur. A failure is a deviation from an expected or desired behavior. Due to its importance in large complex systems, the problem of failure diagnosis has received considerable attention in the literature. Various approaches have been proposed including fault-trees, expert systems, neural networks, fuzzy logic, Bayesian networks, and analytical redundancy [20]. These are broadly categorized into non-model based (where observed behavior is matched to known failures), and model based (where observed behavior is compared against model predictions for any abnormality). For discrete event systems a certain model based approach for failure diagnosis is proposed in [26], and extended in [25], [11], [12], [10], [6], [35]. The application of DES failure diagnosis includes heating, ventilation, and air conditioning systems [27], transportation systems [16], [7], communication networks [3], [1], [17], manufacturing systems [5], [19], digital circuits [15], [31], and power systems [8].

Failure diagnosis in DESs requires that once a failure occurred, it be detected and diagnosed within bounded "delay" (bounded number of transitions). This is captured by the notion of failure diagnosability introduced in [26]. Polynomial tests for diagnosability are given in [9], [34]. In [25], the notion of active failure diagnosis was introduced where control is exercised to meet given specifications while satisfying diagnosability. In [5], [19], a template based approach was developed for failure diagnosis in timed discrete event system. [24] also studied failure diagnosis in timed DESs.

The above approaches can be thought to be "event-based" as failure is modeled as execution of certain "faulty events". An equivalent "state-based" approach was considered in [15], [35], where the occurrence of a failure is modeled as reaching of certain "faulty states". A theory for failure diagnosis of repeatedly-occurring/intermittent failures was introduced in [12]. The notion of diagnosability was extended to $[1, \infty]$-diagnosability to allow diagnosis of a failure each time it occurred. Polynomial complexity algorithms for testing $[1, \infty]$-diagnosability as well as for off-line diagnoser synthesis were presented in [12]. Algorithms of complexity that are an order lower have been reported in [33]. To facilitate generalization of failure To facilitate generalization of failure specifications, linear-time temporal logic (LTL) based specification and diagnosis of it's failure was proposed in [11]. LTL can be used to specify violations of safety as well as liveness properties, allowing diagnosis of failures that have already occurred (safety violations) as well as prognosis of failures that are inevitable in future (liveness failures). [10] extended the use of LTL based specifications for representing and diagnosing repeatedly-occurring/intermittent failures.

The above mentioned work dealt with *centralized* failure diagnosis, where a central diagnoser is responsible for failure detection and diagnosis in the system. Many large complex systems, however, are physically distributed which introduces variable communication delays and communication errors when diagnosis information collected at physically distributed sites are sent to a centralized site for analysis. Consequently, although all diagnosis information can be gathered centrally, owing to the delayed/corrupted nature of the data, a centralized failure diagnosis approach may not always be appropriate for physically distributed systems, and instead diagnosis may need to be performed decentrally at sites where diagnosis information is collected.

[6], [23], [28], [2], [29] studied distributed diagnosis in which diagnosis is performed by either diagnosers communicating with each other directly or through a coordinator and thereby pooling together the observations. In [28], communication exists among local diagnosers which is assumed to be lossless and in order, and a notion of "decentralized diagnosis" was formulated, which was proved to be undecidable. The problem we study is what if the diagnosis decision are

Manuscript received ; revised . The research was supported in part by the National Science Foundation under the grants NSF-ECS-0218207, NSF-ECS-0244732, NSF-EPNES-0323379, and NSF-ECS-0424048, and a DoD-EPSCoR grant through the Office of Naval Research under the grant N000140110621.

The authors are with the Department of Electrical and Computer Engineering, Iowa State University, Ames, IA 50011 USA (email: wqiu@iastate.edu; rkumar@iastate.edu).

made completely locally without combining the information gathered.

The problem of *decentralized* diagnosis was first considered as one special case of *distributed* diagnosis in [6]. In that paper, "lack of fully ambiguous traces" was stated as a sufficient condition for decentralized diagnosis to be equivalent to that of centralized one, and an algorithm was presented for verifying the "lack of fully ambiguous traces". The algorithm was based upon structural properties of global (centralized) and local (decentralized) diagnosers, which has an exponential complexity in the size of the system owing to the exponential size of the diagnosers.

We make the notion of decentralized diagnosis involving no communication among diagnosers precise by introducing the notion of codiagnosability that requires that the occurrence of any failure be diagnosed within bounded delay by at least one local diagnoser using it's own observations of the system execution. Thus we don't attempt to capture a condition under which decentralized diagnosis is equivalent to centralized diagnosis, and in fact the property of codiagnosability is stronger than that of diagnosability (under the aggregated observations). In other words, it is possible that a system is centrally diagnosable under the aggregated observations, but not decentrally diagnosable. However, when a centralized diagnosis is not possible (due to the physically distributed nature of the underlying system), the system must satisfy the stronger property of codiagnosability to allow for the detection/diagnosis of each failure by some local diagnoser.

Our setting of diagnosis with no communication can also be shown to be useful for distributed diagnosis involving communication among diagnosers. For example for distributed diagnosis under a bounded delay communication (see [30] for a formulation), we show in a recent work [21] that one can define a new observation mask for each local observer that combines the effect of it's own observation and the bounded-delay communication received by other diagnosers. Thus although our formalism does not explicitly allow communication, it is powerful enough to allow capturing the effect of communication.

A failure may be specified in different ways. In this paper we represent a failure as violation of a specification represented as a language, and also as the execution of certain failure events. We study codiagnosability first in specification language framework and later specialize it to the failure event framework. The specification language framework can be viewed as a generalized state/event-based framework (in those frameworks the model generating the specification language is a subautomaton of the system model). We present algorithms of complexity polynomial in the size of the system and the non-fault specification for (i) testing codiagnosability, (ii) computing the delay bound of diagnosis, (iii) off-line synthesis of diagnosers, and (iv) on-line diagnosis using them. As far as the dependence of the complexity on the number of diagnosers involved is concerned, it is exponential for the first two algorithms, and linear for the last two algorithms. Thus our algorithms are computationally efficient compared to the ones given in [6] (which were exponential in the size of the system), and also no algorithm for the computation of the delay bound was given in [6]. Further we do not require that the underlying system be either deadlock free, or unobservable-event-cycle free, which were assumed in [6].

The diagnosis approach proposed in our paper is different from the approach in [26]. In our paper, *nondeterministic* diagnosers are constructed *off-line* and the diagnosis is performed *on-line* using them. The on-line diagnosis system maintains a single $Reach$ set and updates it upon each observation. The cardinality of $Reach$ set is linear in the size of system and non-fault specification, and the update also has the same linear complexity as it only involves a reachability computation. Further one diagnoser is constructed for each particular type of fault, which keeps the complexity also linear in the number of faults.

The computation of the delay bound is important for the following reason. After a failure is detected/diagnosed, a failure recovery procedure ought to be initiated. There may be requirements on how late such recovery procedures may be initiated from the time the failure occurred. If delay of diagnosis is longer than the delay by which the recovery procedures are to be initiated, satisfaction of diagnosability is of little use. For centralized diagnosis, [26] presented a method to compute the delay bound. The method is based on the construction of a diagnoser and has exponential complexity. In [34], the authors presented a method of polynomial complexity for determining the delay bound for centralized diagnosis. In this paper, a notion of delay associated with decentralized diagnosis is introduced, and a polynomial algorithm for computing it is provided.

The notion of codiagnosability guarantees that occurrence of any failure is detected within finite delay by one of the diagnosers, but there is no guarantee that the non-occurrence of failure is unambiguously known. To capture the capability of being certain about the failure as well as non-failure conditions in a system within bounded delay, we introduce the notion of strong-codiagnosability. The corresponding notion of strong-diagnosability can also be defined for the centralized setting. We illustrate that a diagnosable (resp., codiagnosable) system need not be strongly-diagnosable (resp., strongly-codiagnosable).

The rest of the paper is organized as follows. Section II presents preliminary notions. In Section III, we first introduce a definition of codiagnosability and provide an algorithm for testing codiagnosability in the specification language framework, and then specialize it to the failure event framework. This is followed by the computation of delay of codiagnosability in Section IV. In Section V, we present methods for off-line synthesis of diagnosers and on-line failure diagnosis using them. In Section VI, an extension is made for the case of multiple specification languages and multiple types of failure events. The notion of strong-codiagnosability is introduced in Section VII, which is followed by the conclusion in Section VIII.

II. Notions and Preliminaries

In this section, we give the system model and present some notions necessary for failure diagnosis of DESs. For more details on DESs theory, readers are referred to [22], [13].

Given an event set Σ, Σ^* denote the set of all finite length event sequences over Σ, including the zero length event sequence ε. A member of Σ^* is a trace and a subset of Σ^* is a language. Given a language $L \subseteq \Sigma^*$, it is said to be prefix-closed if $L = pr(L)$, where $pr(L) := \{s \in \Sigma^* | \exists t \in \Sigma^* \text{ s.t. } st \in L\}$. The set of deadlocking traces of L are those traces from which no further extensions exist in L, i.e., $s \in L$ is a deadlocking trace if $\{s\}\Sigma^* \cap L = \{s\}$.

A DES is modeled as a finite state machine (FSM) G and is denoted by $G = (X, \Sigma, \alpha, x_0)$, where X is the set of states, Σ is the finite set of events, $x_0 \in X$ is the initial state, and $\alpha : X \times \overline{\Sigma} \to 2^X$ is the transition function, where $\overline{\Sigma} := \Sigma \cup \{\varepsilon\}$. G is said to be deterministic if $|\alpha(\cdot, \cdot)| \leq 1$ and $|\alpha(\cdot, \epsilon)| = 0$; otherwise, it is called nondeterministic. A path in G is a sequence of transitions $(x_1, \sigma_1, x_2, \cdots, \sigma_{n-1}, x_n)$, where $\sigma_i \in \overline{\Sigma}$ and $x_{i+1} \in \alpha(x_i, \sigma_i)$ for all $i \in \{1, \cdots, n-1\}$. The path is called a cycle if $x_1 = x_n$.

Given a state $x \in X$, the ε-closure of x, denoted by $\varepsilon^*_G(x) \subseteq X$, is recursively defined as:

$$x \in \varepsilon^*_G(x);\ x' \in \varepsilon^*_G(x) \Rightarrow \alpha(x', \varepsilon) \subseteq \varepsilon^*_G(x).$$

In other words, $\varepsilon^*_G(x)$ includes all state that can be reached from state x by zero or more ε transitions. α can be extended from domain $X \times \overline{\Sigma}$ to domain $X \times \Sigma^*$ recursively as follows: $\forall x \in X, s \in \Sigma^*, \sigma \in \Sigma$,

$$\alpha(x, \varepsilon) = \varepsilon^*_G(x);\ \alpha(x, s\sigma) = \varepsilon^*_G(\alpha(\alpha(x, s), \sigma)).$$

The language generated by G is defined as $L(G) := \{s \in \Sigma^* | \alpha(x_0, s) \neq \emptyset\}$, i.e., it includes all traces that can be executed from the initial state of G. States reached by executing deadlocking traces in $L(G)$ are called deadlock states.

Given two automata $G = (X, \Sigma, \alpha, x_0)$ and $H = (Y, \Sigma, \beta, y_0)$, the synchronous composition of G and H is defined as, $G||H = (X \times Y, \Sigma, \gamma, (x_0, y_0))$ such that $\forall (x, y) \in X \times Y, \sigma \in \overline{\Sigma}, \gamma((x, y), \sigma) :=$

$$\begin{cases} \alpha(x, \sigma) \times \beta(y, \sigma), & \text{if } \sigma \neq \varepsilon; \\ (\alpha(x, \varepsilon) \times \{y\}) \cup (\{x\} \times \beta(y, \varepsilon)), & \text{otherwise.} \end{cases}$$

When the system execution is observed by a global observer, events executed by the system are filtered through a global observation mask $M : \overline{\Sigma} \to \overline{\Delta}$ with $M(\varepsilon) = \varepsilon$, where $\overline{\Delta} := \Delta \cup \{\varepsilon\}$ and Δ is the set of observed symbols. The definition of M can be extended from events to event sequences inductively as follows:

$$M(\varepsilon) = \varepsilon;\ \forall s \in \Sigma^*, \sigma \in \Sigma, M(s\sigma) = M(s)M(\sigma).$$

Given an automaton G and mask M, $M(G)$ is the automaton G with each transition (x, σ, x') of G replaced by $(x, M(\sigma), x')$. The local observation masks associated with local observers are defined as $M_i : \overline{\Sigma} \to \overline{\Delta}_i$ $(i \in I_M = \{1, \cdots, m\})$, where m is the number of local observers, $\overline{\Delta}_i := \Delta_i \cup \{\varepsilon\}$ and Δ_i is the set of locally observed symbols.

III. Codiagnosability: Definition and Verification

In this section, we present a definition of codiagnosability and an algorithm to verify the codiagnosability of a system. We study codiagnosability first in the specification language framework and later specialize it to the failure event framework. In the definition below L represents the generated language of a system and is prefix-closed, and $K \subseteq L$ represents a specification language. Since K can capture safety as well as progress properties, it need not be prefix-closed. A failure is said to have occurred if the system executes a trace in $L - pr(K)$ violating the specification. Thus although the "system specification" is K, the "specification for non-failures" is $pr(K)$—a prefix-closed language.

Definition 1: Let L be the prefix-closed language generated by a system and K be the specification language contained in L $(K \subseteq L)$. Assume there are m local sites with observation masks $M_i : \overline{\Sigma} \to \overline{\Delta}_i$ $(i \in I_M = \{1, \cdots, m\})$. (L, K) is said to be codiagnosable with respect to $\{M_i\}$ if

$$(\exists n \in \mathcal{N})(\forall s \in L - pr(K))$$

$$(\forall st \in L - pr(K), |t| \geq n \text{ or } st \text{ deadlocks}) \Rightarrow$$

$$(\exists i \in I_M)(\forall u \in M_i^{-1} M_i(st) \cap L, u \in L - pr(K)).$$

The above definition of codiagnosability has the following meaning. Let s be a trace in the "faulty language" $L - pr(K)$, and t be either a sufficiently long extension in $L - pr(K)$ after s (with at least n transitions), or st be a deadlocking trace. There exists at least one local site i such that any trace in L indistinguishable to st for site i belongs to the faulty language $L - pr(K)$. A local site is *ambiguous* if it's past observations indicate the possible occurrence of a failure but not with complete certainty; otherwise, it is unambiguous. Informally, Definition 1 means that for any faulty trace, there exists at least one local site that can unambiguously detect the occurrence of the failure within finite transitions. It follows from this definition that (L, K) is codiagnosable if and only if $(L, pr(K))$ is codiagnosable.

To facilitate the development of an algorithm for testing codiagnosability, we first present a lemma for the condition of non codiagnosability, which is derived by negating the codiagnosability condition of Definition 1. (The basic idea of our algorithm is to check if there exists a situation that violates the definition of codiagnosability.)

Lemma 1: Let L be the prefix-closed language generated by a system and K be the specification language contained in L $(K \subseteq L)$. Assume there are m local sites with observation masks $M_i : \overline{\Sigma} \to \overline{\Delta}_i$ $(i \in I_M = \{1, \cdots, m\})$. (L, K) is not codiagnosable with respect to $\{M_i\}$ if and only if

$$(\forall n \in \mathcal{N})(\exists s \in L - pr(K))$$

$$(\exists st \in L - pr(K), |t| \geq n \text{ or } st \text{ deadlocks}) \text{ s.t.}$$

$$(\forall i \in I_M)(\exists u_i \in M_i^{-1} M_i(st) \cap L, u_i \in pr(K)).$$

Lemma 1 states that (L, K) is not codiagnosable if and only if there exist a faulty trace $s \in L - pr(K)$ possessing an arbitrarily long extended (or deadlocking) trace st, and for each local site a st-indistinguishable non-faulty trace $u_i \in pr(K)$.

Remark 1: Note that if there exist deadlocks in the system, we can add a self-loop at each deadlocking state on ϵ. In this way, the deadlocking system is converted into a deadlock free system in such a way that the observation generated by

any deadlocking trace t does not change: $M_i(t) = M_i(t\epsilon^*)$. With this observation in mind, in the following discussion the system to be diagnosed is assumed to be deadlock free.

The following algorithm checks whether codiagnosability is violated. Without loss of generality, assume there are two local sites (i.e., $I_M = \{1, 2\}$), and as mentioned in Remark 1 the system is deadlock-free. The idea behind the algorithm is to construct a testing automaton that tracks a faulty trace, and for each local site a corresponding indistinguishable non-faulty trace. Presence of a cycle involving such a tuple of traces in the testing automaton is equivalent to the violation of codiagnosability. For simplicity of presentation, we assume the generators G and H of L and $pr(K)$, respectively, are both deterministic. However, the algorithm of the paper continues to hold even when G is nondeterministic and either H is a subautomaton of G or H is deterministic.

Algorithm 1: Let $G = (X, \Sigma, \alpha, x_0)$ be a deterministic finite state machine (DFSM) system model and $L = L(G)$ be it's generated language. For the specification language $K \subseteq L$, let $H = (Y, \Sigma, \beta, y_0)$ be the DFSM model with $L(H) = pr(K)$.

Step 1: Construct augmented specification automaton $\overline{H}$

We first augment the state set of H by adding a new state F, which indicates occurrence of a failure. For each $y \in Y$, we add a new transition to the failure state F on each event $\sigma \in \Sigma$ if $\beta(y, \sigma)$ is undefined. At the failure state F, we introduce a self-loop transition for any event $\sigma \in \Sigma$. The resulting automaton is denoted by $\overline{H} = \{\overline{Y}, \Sigma, \overline{\beta}, y_0\}$, where $\overline{Y} = Y \cup \{F\}$ and $\overline{\beta}$ is defined as: $\forall \overline{y} \in \overline{Y}, \sigma \in \Sigma$,

$$\overline{\beta}(\overline{y}, \sigma) := \begin{cases} \beta(\overline{y}, \sigma), & \text{if } [\overline{y} \in Y] \wedge [\beta(\overline{y}, \sigma) \neq \emptyset] \\ F, & \text{if } [\overline{y} = F] \vee [\beta(\overline{y}, \sigma) = \emptyset] \end{cases}$$

It follows from the above construction procedure that all events in Σ are defined at each state in $\overline{H}$. Therefore, $\overline{H}$ generates the language Σ^*, i.e., $L(\overline{H}) = \Sigma^*$. Also, execution of any trace outside $pr(K)$ reaches the failure state F.

Step 2: Construct codiagnosability testing automaton T

The codiagnosability testing automaton $T = (Z, \Sigma^T, \gamma, z_0)$ is defined as follows:

- $Z = X \times \overline{Y} \times Y \times Y$.
- $z_0 = (x_0, y_0, y_0, y_0)$.
- $\Sigma^T = \overline{\Sigma}^3$, where $\overline{\Sigma} = \Sigma \cup \{\varepsilon\}$.
- $\gamma : Z \times \overline{\Sigma}^3 \to Z$ is defined as: $\forall z = (x, \overline{y}, y^1, y^2) \in Z$, $\sigma^T = (\sigma, \sigma^1, \sigma^2) \in \Sigma^T - \{(\varepsilon, \varepsilon, \varepsilon)\}$,

$$\gamma(z, \sigma^T) := (\alpha(x, \sigma), \overline{\beta}(\overline{y}, \sigma), \beta(y^1, \sigma^1), \beta(y^2, \sigma^2))$$

if and only if

$$[M_1(\sigma) = M_1(\sigma^1), M_2(\sigma) = M_2(\sigma^2)] \wedge [\alpha(x, \sigma), \beta(y^1, \sigma^1), \beta(y^2, \sigma^2) \neq \emptyset].$$

The state space of T is $X \times \overline{Y} \times Y \times Y$, and T tracks a triplet of traces $s \in L(G) \cap L(\overline{H}) = L(G), u_1 \in L(H), u_2 \in L(H)$ with the property, $M_1(s) = M_1(u_1), M_2(s) = M_2(u_2)$. The first component of T tracks trace s in G, the second component the trace s in $\overline{H}$, the third component the trace u_1 in H, and the last component the trace u_2 in H. Purpose of tracking s in G and $\overline{H}$ is to determine whether s is a faulty trace belonging to $L \cap (pr(K))^c = L - pr(K)$.

Step 3: Check violation of codiagnosability

We first define a cycle cl^T in the testing automaton T as follows:

$$cl^T := (z_k, \sigma_k^T, z_{k+1}, \cdots, z_l, \sigma_l^T, z_k), (l \geq k \geq 0)$$

where $z_i = (x_i, \overline{y}_i, y_i^1, y_i^2) \in Z$ and $\sigma_i^T = (\sigma_i, \sigma_i^1, \sigma_i^2) \in \Sigma^T (i = k, k+1, \cdots, l)$.

Then, we check if there exists a cycle cl^T in T satisfying

$$\exists i \in [k, l] \text{ such that } (\overline{y}_i = F) \wedge (\sigma_i \neq \varepsilon).$$

If the answer is yes, then (L, K) is not codiagnosable with respect to the observation masks $\{M_i\}$. Otherwise, (L, K) is codiagnosable. The condition $(\overline{y}_i = F)$ indicates that a failure has occurred. Since σ_i is a transition in the system G, the condition $(\sigma_i \neq \varepsilon)$ requires that the system execute at least one event in the cycle cl^T. This requirement originates from the fact G must execute an extension after a failure has occurred in order to allow for it's diagnosis.

Remark 2: If there are no unobservable-event cycles in the system, then the second condition $(\sigma_i \neq \varepsilon)$ is redundant since in absence of unobservable cycles, this condition automatically holds. But the test needs to be strengthened with this condition in the presence of unobservable-event cycles.

The following example illustrates the construction used in the algorithm for testing codiagnosability.

Example 1: A system model G and a specification model H are given in Figure 1 with $L(G) = L$ and $L(H) = pr(K)$. Assume there are two local diagnosers, i.e., $I_M = \{1, 2\}$. The set of events is $\Sigma = \{a, b, c, \sigma_u, \sigma_f\}$, where σ_u and σ_f are two unobservable events, i.e., $\forall i \in I_M, M_i(\sigma_u) = M_i(\sigma_f) = \varepsilon$. The event a can be observed by both diagnosers ($M_1(a) = M_2(a) = a$), the event b can be observed by only the first diagnoser, whereas the event c can be observed by only the second diagnoser ($M_1(b) = b, M_1(c) = \varepsilon, M_2(b) = \varepsilon, M_2(c) = c$).

The augmented specification automaton $\overline{H}$ is constructed according to Algorithm 1 and is shown in Figure 1. Only a part of the testing automaton T is shown in Figure 1 to track a specific transition sequence. The transition sequence "$aaa, \sigma_f \varepsilon b$" causes T to reach the state "5F12". This implies the trace $s = a\sigma_f \in L - pr(K)$ is a failure trace. In Figure 1, T eventually reaches the state "4F44" possessing a self-loop, that violates the codiagnosability condition mentioned in Step 3 of Algorithm 1. Therefore, (L, K) is not codiagnosable with respect to the observation masks $\{M_i\}$.

The following theorem proves the correctness of Algorithm 1.

Theorem 1: Given a prefix-closed system language L and a specification language $K \subseteq L$, let $G = (X, \Sigma, \alpha, x_0)$ and $H = (Y, \Sigma, \beta, y_0)$ be automata with $L(G) = L$ and $L(H) = pr(K)$. Assume there are m local sites with observation masks M_i $(i \in I_M)$. (L, K) is not codiagnosable with respect to $\{M_i\}$ if and only if there exists a cycle $cl^T = (z_k, \sigma_k^T, z_{k+1}, \cdots, z_l, \sigma_l^T, z_k)$ in the testing automaton $T = (Z, \Sigma^T, \gamma, z_0)$ such that:

$$\exists i \in [k, l] \text{ s.t. } (\overline{y}_i = F) \wedge (\sigma_i \neq \varepsilon), \quad (1)$$

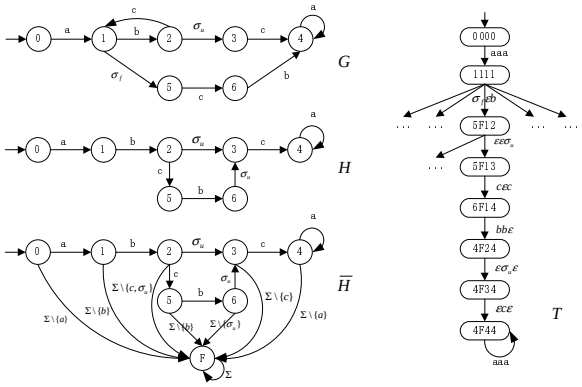

Fig. 1. Algorithm 1 illustrated (G: plant model; H: specification model; $\overline{H}$: augmented automaton H; T: testing automaton)

where $\overline{y}_i$ is the second coordinate of state z_i, and σ_i is the first coordinate of $\sigma_i^T \in \Sigma^T$.

Proof: ($\Leftarrow$) Suppose there exists a cycle $cl^T = (z_k, \sigma_k^T, z_{k+1}, \cdots, z_l, \sigma_l^T, z_k)$ satisfying condition (1). Let $path$ be a path in T ending with the cycle cl^T:

$$path = (z_0, \sigma_0^T, \cdots, z_{k-1}, \sigma_{k-1}^T, z_k, \sigma_k^T, \cdots, z_l, \sigma_l^T, z_k).$$

Therefore, for any $n \in N$ there exist the following event traces:

$$st = \sigma_0 \cdots (\sigma_k \cdots \sigma_l)^n \in L(G) = L,$$
$$u_1 = \sigma_0^1 \cdots (\sigma_k^1 \cdots \sigma_l^1)^n \in L(H) = pr(K),$$
$$u_2 = \sigma_0^2 \cdots (\sigma_k^2 \cdots \sigma_l^2)^n \in L(H) = pr(K),$$

where $M_1(st) = M_1(u_1)$ and $M_2(st) = M_2(u_2)$.

Since there exists $i \in (k, \cdots, l)$ such that $\overline{y}_i = F$, from the definition of $\overline{H}$, there exists a transition $\beta(y_p, \sigma_p)$ $(0 \leq p < i)$ which is not defined in H. Therefore, $st \notin L(H) = pr(K)$. Further once $\overline{H}$ reaches the failure state F, it remains there, and so $\overline{y}_i = F$ for each $i \in [k, l]$. We can choose $s = \sigma_0 \cdots \sigma_{k-1}$ and $t = (\sigma_k \cdots \sigma_l)^n$. Then since G is deadlock free and $\sigma_i \neq \varepsilon$, it follows that $|t| \geq n$. From Lemma 1, (L, K) is not codiagnosable with respect to $\{M_i\}$.

($\Rightarrow$) Suppose (L, K) is not codiagnosable with respect to $\{M_i\}$. Then from Lemma 1, there exists a faulty trace $s \in L - pr(K)$ and it's extended trace $st \in L - pr(K)$ such that the following holds: $\exists u_1, u_2$,

$$u_1 \in M_1^{-1} M_1(st) \cap pr(K), u_2 \in M_2^{-1} M_2(st) \cap pr(K).$$

Since $s \in L - pr(K)$, according to the definition of $\overline{H}$, $\overline{\beta}(y_0, s) = F$ and $\overline{H}$ remains at the failure state F on any further transition. Let us execute the trace tr in T:

$$tr = \sigma_0^T \cdots \sigma_l^T, \text{ where } \sigma_i^T = (\sigma_i, \sigma_i^1, \sigma_i^2)\ (i \in 0, \cdots, l)$$

such that $st = \sigma_0 \cdots \sigma_l$, $u_1 = \sigma_0^1 \cdots \sigma_l^1$, and $u_2 = \sigma_0^2 \cdots \sigma_l^2$. Let $|Z|$ be the number of states in the testing automaton T. If $|st| > |Z|$, then since T is a finite state machine (FSM), there will be a cycle $cl^T = (z_k, \sigma_k^T, z_{k+1}, \cdots, z_l, \sigma_l^T, z_k)$ along the trace tr, where $l \geq k \geq 0$, such that for some $i \in [k, l]$, $\overline{y}_i = F$ and $\sigma_i \neq \varepsilon$. Therefore, the sufficiency holds. ■

The computation complexity of Algorithm 1 is analyzed as follows.

Remark 3: Let $|X|$ and $|Y|$ be the number of states of G and H, and $|\Sigma|$ be the number of events of G and H. Assume there are m local sites in the system. Table I lists the maximum number of states and transitions of various automata in Algorithm 1. Since $\overline{H}$ and H have same order of states and transitions, i.e., $O(|Y|)$ and $O(|Y| \times |\Sigma|)$ respectively, we do not differentiate the number of states and transitions of H and $\overline{H}$ for complexity analysis. Since there are $m+1$ coordinates in a transition of T, the number of transitions at each state of T is at most $(|\Sigma| + 1)^{m+1}$.

The complexity of Step 1 and Step 2 is linear in the number of states and transitions of H and T respectively. The complexity of Step 3, which is to detect the presence of a certain "offending" cycle in the testing automaton T, is also linear in the number of states and transitions of T. Therefore, the complexity of Algorithm 1 is $O(|X| \times |Y|^{m+1} \times |\Sigma|^{m+1})$.

TABLE I
COMPUTATIONAL COMPLEXITY OF ALGORITHM 1

	number of states	number of transitions
G	$\|X\|$	$\|X\| \times \|\Sigma\|$
H	$\|Y\|$	$\|Y\| \times \|\Sigma\|$
T	$\|X\| \times \|Y\|^{m+1}$	$\|X\| \times \|Y\|^{m+1} \times (\|\Sigma\| + 1)^{m+1}$
complexity	$O(\|X\| \times \|Y\|^{m+1} \times \|\Sigma\|^{m+1})$	

Remark 4: For the special case when the specification model $H = (Y, \Sigma, \beta, y_0)$ is a subautomaton of the system model $G = (X, \Sigma, \alpha, x_0)$, the operation of testing codiagnosability can be performed with less complexity. Since H is a subautomaton of G, we have $Y \subseteq X$, $L(H) \subseteq L(G)$ and for all $s \in L(H)$, $\beta(y_0, s) = \alpha(x_0, s)$. Then for each trace in the testing automaton T, if the system model G reaches a state $x \in X$, then the augmented automaton $\overline{H}$ can only reach either the same state $y = x$ or the failure state $y = F$. Then the first two components of T, namely $G||\overline{H}$, has state size at most $2|X|$ and transition size at most $2|X| \times |\Sigma|$. Thus number of states and transitions in T is at most $2|X|^{m+1}$ and $2|X|^{m+1} \times |\Sigma|^{m+1}$, respectively. The complexity of Algorithm 1 for the special case is $O(|X|^{m+1} \times |\Sigma|^{m+1})$, which is one order less than the general case.

Remark 5: For the centralized case, diagnosability can be defined similarly in the specification language setting by letting $m = 1$ in Definition 1. It follows that diagnosability requires a weaker condition than codiagnosability, i.e., some diagnosable systems may not be codiagnosable. For example, in Example 1, if we set the central observation mask as the "union" of the local observation masks, i.e., $M(a) = a, M(b) = b, M(c) = c$, and $M(\sigma_u) = M(\sigma_f) = \varepsilon$, then we can verify that (L, K) is diagnosable with respect to M. See the automaton T for testing diagnosability shown in Figure 2, which does not contain any offending cycle.

A. Failure As Occurrence of Failure Events

In this subsection, we study codiagnosability in the failure event framework. Let $\Sigma_f \subseteq \Sigma$ be the set of all failure events, $\mathcal{F} = \{f_1, \cdots, f_j\}$ be the set of all failure types, and $\psi : \Sigma \to \mathcal{F} \cup \{\emptyset\}$ be the failure assignment function. Then the

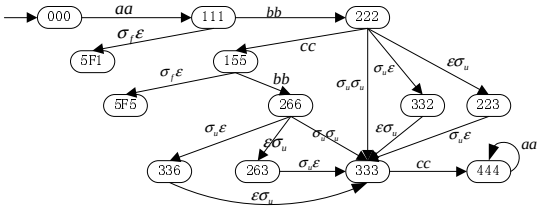

Fig. 2. Testing of diagnosability under a global/central observer

set Σ_f is partitioned into several disjoint sets corresponding to different failure types: $\Sigma_f = \Sigma_{f_1} \cup \cdots \cup \Sigma_{f_l}$, where for all $\sigma \in \Sigma_{f_j} (j \in I_F = \{1, \cdots, l\})$, $\psi(\sigma) = f_j$. Since occurrence of any observable failure event can be immediately diagnosed, we only consider unobservable failure events, i.e.,

$$\forall \sigma \in \Sigma, \psi(\sigma) \neq \emptyset \Rightarrow \forall i \in I_M, M_i(\sigma) = \varepsilon.$$

We first discuss a system with a single failure type f, i.e., $\mathcal{F} = \{f\}$. A system with multiple types of failure events is discussed in Section VI. The following definition defines codiagnosability in the failure event setting.

Definition 2: Given a system model $G = (X, \Sigma, \alpha, x_0)$ with a singleton failure types set $\mathcal{F} = \{f\}$, let $L = L(G)$ be the generated language of G and $\psi : \Sigma \to \mathcal{F} \cup \{\emptyset\}$ be a failure assignment function. Assume there are m local sites with observation masks M_i $(i \in I_M)$. $(L, \mathcal{F})$ is said to be codiagnosable with respect to $\{M_i\}$ if

$$(\exists n \in \mathcal{N})(\forall s \in L, \psi(s_f) = f)$$

$$(\forall st \in L, |t| \geq n \text{ or } st \text{ deadlocks}) \Rightarrow$$

$$(\exists i \in I_M)(\forall u \in M_i^{-1} M_i(st) \cap L)(\exists v \in pr(u), \psi(v_f) = f),$$

where s_f and v_f denote the last events in traces s and v respectively. The system G is said to be codiagnosable if $(L(G), \mathcal{F})$ is codiagnosable.

The verification of codiagnosability of $(L, \mathcal{F})$ is performed as follows. Given a failure assignment function $\psi : \Sigma \to \mathcal{F} \cup \{\emptyset\}$, we define a failure event set as $\Sigma_f = \{\sigma \in \Sigma | \psi(\sigma) = f\}$. Next, we construct a single state automaton H_0 with a self-loop labeled by all non-faulty events $\Sigma - \Sigma_f$. Generated language of H_0 is $(\Sigma - \Sigma_f)^*$, i.e., all traces containing no failure events. We define an equivalent specification model H as the synchronous composition of G and H_0. $H := G||H_0$, which generates all traces of G that contain no failure events. Clearly, $(L, \mathcal{F})$ is codiagnosable if and only if $(L, L(H))$ is codiagnosable. As in Algorithm 1, we build a testing automata T and check the presence of "offending" cycles.

Since H_0 has a single state, $H = G||H_0$ is a subautomaton of G. So for a deterministic model G, the complexity of testing codiagnosability of $(L, \mathcal{F})$ is same as the subautomaton case analyzed in Remark 3, i.e., the complexity is $O(|X|^{m+1} \times |\Sigma|^{m+1})$, where m is the number of local diagnosers. If G is nondeterministic, one transition may lead to different states. The number of transitions of G and T is at most $|X|^2 \times |\Sigma|$ and $4|X|^{2(m+1)} \times |\Sigma|^{m+1}$ respectively. Therefore, the complexity of testing codiagnosability for a nondeterministic G is $O(|X|^{2(m+1)} \times |\Sigma|^{m+1})$. It should be noted that even when G is nondeterministic, $H = G||H_0$ is a subautomaton of G and $\overline{H} = G||\overline{H_0}$.

The following example illustrates the testing approach for the failure event case.

Example 2: Consider the system model G introduced in Example 1. The system only has one failure event σ_f, which belongs to the failure type f, i.e., $\Sigma_f = \{\sigma_f\}$ and $\psi(\sigma_f) = f$. Also, there are two diagnosers in the system with observation masks defined as before: $M_1(a) = a, M_1(b) = b, M_1(c) = M_1(\sigma_u) = M_1(\sigma_f) = \varepsilon$, and $M_2(a) = a, M_2(c) = c, M_2(b) = M_2(\sigma_u) = M_2(\sigma_f) = \varepsilon$.

The single state automaton H_0 is shown in Figure 3. Then we construct the equivalent specification model $H = G||H_0$ and build the augmented automaton $\overline{H}$ by adding a failure state F. Figure 3 shows a branch of transitions in T, where a "bad" state "4F44" possessing a self-loop is reached. Therefore, $(L, \mathcal{F})$ is not codiagnosable with respect to $\{M_i\}$. ■

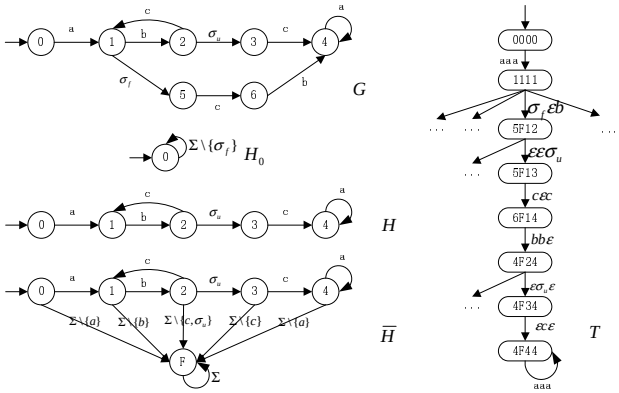

Fig. 3. Automata in Example 2 (G: plant model; H_0: single state automaton; H: specification model; $\overline{H}$: augmented automaton H; T: testing automaton)

IV. Delay of Codiagnosability

In the above section, we discussed codiagnosability and it's verification methods. Codiagnosability guarantees that once a failure occurs, there exists at least one local diagnoser that can unambiguously detect and diagnose the failure within bounded delay. In this section we compute the value of delay and also compute the maximum delay possible. This information is very important because one purpose of failure diagnosis is to detect and isolate failures so as to activate some failure recovery procedures. Even for a codiagnosable system, if the diagnosis result of a failure arrives late, some recovery deadlines may be missed and the failure may cause catastrophic results. Since the case of failure events can be transformed to the case of specification language, we only consider the case of specification language.

The computation of delay of codiagnosability is based on the testing automaton $T = (Z, \overline{\Sigma}^3, \gamma, z_0)$ constructed in Algorithm 1. Once a failure occurs in the system model G, the second coordinate of T reaches the failure state "F" and stays there forever. Define a set of states containing all the "failure" states as:

$$Z_F = \{(x, \overline{y}, y^1, y^2) \in Z | \overline{y} = F\}.$$

From analysis in Section III, we know that if a system (L, K) is codiagnosable, there does not exist any cycle among states in Z_F. Since T is a finite automaton, some deadlocking states will be reached eventually in Z_F. Then we can define the delay of codiagnosability as follows.

Definition 3: The delay of codiagnosability for $z \in Z_F$ is defined by

$$d(z) = \max_{\{(s,u_1,u_2)\in(\overline{\Sigma}^3)^*:\ \gamma(z,(s,u_1,u_2))=\emptyset\}} (EC(s)) + 1,$$

where for $s \in \Sigma^*$, $EC(s)$ is the *event count* of s, i.e., the number of elements in s that are not ε.

$d(z)$ is the number of non-silent transitions in the system that must be executed to lead failure state z to a deadlocking state. Since it is not possible to sustain the ambiguity of failure beyond deadlocking states (as no further execution possible), an extra transition will resolve ambiguity of failure, and so an extra one is added to $EC(s)$ above. Based on Definition 3, we can define the delay of codiagnosability of a system as follows.

Definition 4: Let L be the system language and K $(K \subseteq L)$ be the specification language. The delay of codiagnosability of (L, K) with respect to $\{M_i\}$ is defined as:

$$d(L, K) = \max_{z\in Z_F} d(z).$$

The above definition states that the delay of codiagnosability of (L, K) is the maximum value of all delays of codiagnosability for any state in Z_F. To compute the delay of codiagnosability of (L, K), we present the following algorithm that computes the length of the longest path over Z_F and adds an extra one to it.

Algorithm 2: Given a system language L and a specification language K $(K \subseteq L)$, let G be the system model with $L(G) = L$ and H be the specification model with $L(H) = pr(K)$.

1) Construct a testing automaton $T = (Z, \overline{\Sigma}^3, \gamma, z_0)$ as in Algorithm 1.
2) Initialize the delay of codiagnosability for all states $z \in Z_F$: $d^0(z) = 1$, and a counter $k = 0$.
3) Execute one-step forward search and update the delay of codiagnosability for each state $z \in Z_F$ as follows: $d^{k+1}(z) :=$

$$\begin{cases} \max[d^k(z), d^k(z') + 1], & \text{if } \exists(\sigma, \sigma^1, \sigma^2) \in \Sigma \times (\overline{\Sigma})^2 \\ & \text{s.t. } [z' \in \gamma(z, (\sigma, \sigma^1, \sigma^2))] \\ d^k(z), & \text{otherwise.} \end{cases}$$

4) Repeat Step 3 until $\forall z \in Z_F, d^{k+1}(z) = d^k(z)$, each time incrementing k by 1.
5) Compute the maximum delay of codiagnosability:

$$d(L, K) = \max_{z\in Z_F} d^k(z).$$

The following example illustrates this algorithm.

Example 3: Figure 4 shows a system model G and a specification model H. The system has two local diagnosers with observation masks defined as: $M_1(a) = a, M_1(b) = b, M_1(c) = M_1(\sigma_f) = \varepsilon, M_1(d) = d$, and $M_2(a) = a, M_2(c) = c, M_2(b) = M_2(d) = M_2(\sigma_f) = \varepsilon$.

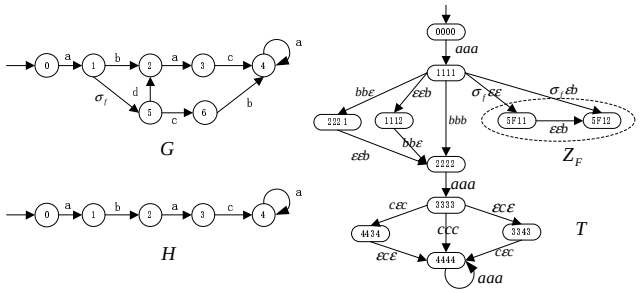

Fig. 4. Delay of codiagnosability (G: system model; H: specification model; T: Testing automaton)

From the testing automaton T of Figure 4, it is easy to verify that the system (L, K) is codiagnosable with respect to the observation masks $\{M_i\}$. The set of failure states Z_F includes two states, (5F11) and (5F12). We first assign an initial value "1" to these two states. Then we perform the one-step forward search recursively until the termination condition is satisfied. Though (5F12) is a successor state of (5F11), the first component of the transition between them is ε. Therefore, state (5F11) remains labeled by "1" and the maximum delay of codiagnosability equals 1. This means that if a failure occurs in the system, the failure will be detected after the system executes at most one more transition.

Remark 6: The complexity of Step 3 of Algorithm 2 is linear in the number of states of Z_F and transitions among them, which in worst case is of order the number of states and transitions in T. Since there are no cycles among states in Z_F for a codiagnosable system, Step 3 may be repeated at most $|Z_F|$ times. Therefore, the complexity of Algorithm 2 is $O(|X|^2 \times |Y|^{2m+2} \times |\Sigma|^{m+1})$.

Remark 7: Failure diagnosis can be viewed as a type of *convergence* or *stability* [4], [18], [14], [32] problem. Informally, the convergence or stability specifies the eventual behavior of a system, i.e., a system is said to be convergent or stable if the system state can eventually converge to a legal set of states regardless of the current state of the system. Codiagnosability of a system requires that once a failure occurs, the testing system state (extended appropriately to evolve over $X \times \overline{Y}^3$ as opposed to $X \times \overline{Y} \times Y^2$) eventually converges to a "diagnosis region" where the failure can be unambiguously diagnosed by at least one local diagnoser. This diagnosis region can be defined over $X \times \overline{Y}^3$ as:

$$Z_D = \{(x, \overline{y}, \overline{y}^1, \overline{y}^2) | \overline{y} = \overline{y}^1 = F \text{ or } \overline{y} = \overline{y}^2 = F\}.$$

Then, the maximum delay of codiagnosability equals one plus the event count associated with the first component of the longest path from Z_F to Z_D, where Z_F is the set of failure states, and Z_D is the set of diagnosis states.

V. Diagnoser Synthesis and On-Line Diagnosis

In the previous sections, we discussed verification of codiagnosability and computation of maximum delay of codiagnosability. In this section, we discuss how to construct local diagnosers that can diagnose the system failures. Since the setting of failure events can be transformed to the setting

of a specification language by constructing the equivalent specification model H as introduced in Section III-A, we only discuss diagnoser synthesis and on-line diagnosis in the setting of specification language.

The local diagnoser at site i is the state machine $G||\overline{H}$ masked by observation mask M_i, which is defined as

$$D_i = M_i(G||\overline{H}) = (X \times \overline{Y}, \Delta_i, \delta_i, (x_0, y_0)).$$

Note that D_i is in general nondeterministic. We use D_i to perform on-line failure diagnosis as follows. At each local site i, we maintain a set of possible current states of D_i reached by the observation sequence executed so far, denoted $Reach_i(\cdot)$. $Reach_i(\cdot)$ is computed recursively upon each observation as follows.

$$Reach_i(\varepsilon) = \varepsilon^*_{D_i}((x_0, y_0));$$

$$Reach_i(\tau\eta) = \varepsilon^*_{D_i}(\delta_i(Reach_i(\tau), \eta)), \tau \in \Delta_i^*, \eta \in \Delta_i.$$

After each update of $Reach_i(\cdot)$, we check all states in $Reach_i(\cdot)$. If all states have second coordinate as "F", then a failure is detected at site i, i.e.,

$$\text{Failure detected by } D_i \Leftrightarrow Reach_i(\cdot) \subseteq X \times \{F\}.$$

If some but not all the states in $Reach_i(\cdot)$ have 2nd coordinate as F, then D_i is said to be ambiguous. For a codiagnosable system, there exists at least one local diagnoser which can detect/diagnose a failure within a bounded delay.

Remark 8: The complexity for local diagnoser construction and on-line failures detection is analyzed as follows. The number of states and transitions of $G||\overline{H}$ is of order $|X| \times |Y|$ and $|X| \times |Y| \times |\Sigma|$ respectively. The complexity of off-line construction of D_i is linear in the number of transitions of $G||\overline{H}$, i.e., for m local diagnosers, the complexity is $O(m \times |X| \times |Y| \times |\Sigma|)$. During on-line failure diagnosis, update of $Reach_i(\cdot)$ is required following each new observation at site i. The complexity of such an update operation is linear in the number of states and transitions of D_i since it involves a certain reachability computation. Therefore, for a system with m local diagnosers, the complexity of performing each step of on-line failure diagnosis is $O(m \times |X| \times |Y| \times |\Sigma|)$, the same as the complexity of constructing local diagnosers.

If the specification model H is a subautomaton of the system model G, then as analyzed before the number of states and transitions in $G||\overline{H}$ is of order $2|X|$ and $2|X| \times |\Sigma|$ respectively. So the complexity of off-line diagnoser construction and of each step of on-line failure diagnosis is $O(m \times |X| \times |\Sigma|)$.

Example 4: Consider the codiagnosable system introduced in Example 3. Figure 5 shows the two local diagnosers $D_i = M_i(G||\overline{H})$ $(i \in \{1, 2\})$, where G is the system model and H is the specification model.

Assume that the system G executes the event trace $s = a\sigma_f cba^n$ $(n \in \mathcal{N})$. Due to the presence of partial observation, the traces observed in local diagnosers are $M_1(s) = aba^n$ and $M_2(s) = aca^n$ respectively. Figure 5 shows all $Reach_i(\cdot)$ computations during the on-line diagnosis procedure. Notice that following the observation of the trace ac by diagnoser D_2, all states in $Reach_2(ac)$ have second coordinate F. So D_2 detects and reports a failure at this point. On the other hand, all along the observation trace aba^n, D_1 remains ambiguous about the occurrence of a fault since in each $Reach_1(\cdot)$ set containing a state with second coordinate F, exists another state with second coordinate different from F. Therefore, the execution of the failure trace $s = a\sigma_f cba^n$ can only be diagnosed by the second diagnoser D_2. Similarly, it can be verified that the execution of failure trace $s = a\sigma_f daca^n$ in the system can be diagnosed by D_1, but not by D_2.

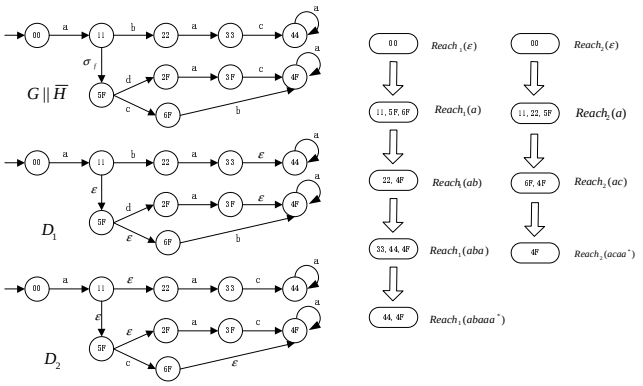

Fig. 5. Off-line diagnoser synthesis and on-line diagnosis

Remark 9: Failure diagnosis introduced in [26] and [6] is based on the construction of centralized/local diagnosers, which are DFSMs (as opposed to NFSMs in our setting). As analyzed in [9], having DFSMs for diagnosers results in an exponential complexity. In contrast, our methods for diagnoser synthesis and on-line failure diagnosis have linear complexity.

VI. Multiple Specifications/Failures-Types

For the sake of simplicity, in the discussion so far we considered systems with only one specification language or only one single failure type. In this section, we extend the above results to multiple sub-specification languages and multiple failure types.

A. Multiple sub-specification languages

In some applications, it is more convenient to capture the desired behavior of a system by a set of sub-specification languages than by one single specification language. The system behavior is said to be faulty if it violates one of the sub-specification languages. In addition to detecting a failure, i.e., detecting the violation of the overall specification, we further need to detect which sub-specification has been violated. Such diagnosis result is helpful to isolate/locate failures and perform appropriate failure recovery operations.

Let L be a language generated by a system. Assume there are l sub-specification languages K_j $(j \in I_K = \{1, \cdots, l\})$, where $K_j \subseteq L$. The nominal behavior K is a conjunct of all prefix closures of sub-specification languages, i.e., $K = \bigcap_{j \in I_K} pr(K_j)$. It follows that K is prefix-closed. Then, the faulty behavior of the system can be captured by the language

$$L - K = L - \bigcap_{j \in I_K} pr(K_j) = \bigcup_{j \in I_K} (L - pr(K_j)).$$

In other words, a system behavior is faulty if it is faulty with respect to any of the sub-specification language K_j. Therefore, instead of detecting the system failures with respect to K, we can detect the system failures with respect to each sub-specification language K_j individually.

Definition 5: Let L be the prefix-closed language generated by a system, and $\mathcal{K}$ be the set of all sub-specification languages K_j, i.e., $\mathcal{K} = \{K_j | j \in I_K = \{1, \cdots, l\}\}$. Assume there are m local sites with observation masks M_i $(i \in I_M = \{1, \cdots, m\})$. $(L, \mathcal{K})$ is said to be codiagnosable with respect to $\{M_i\}$ if

$$\forall j \in I_K,\ (L, K_j) \text{ is codiagnosable with respect to } \{M_i\}.$$

Let $K = \bigcap_{j \in I_K} pr(K_j)$. $(L, \mathcal{K})$ is said to be codetectable if (L, K) is codiagnosable.

The above definition requires that a system with a set of sub-specification languages $(L, \mathcal{K})$ is codiagnosable if the system with each individual sub-specification language (L, K_j) is codiagnosable. Since codiagnosability of (L, K) allows detection of a failure unambiguously but it may not allow isolation of the failure, we refer to it as codetectability of $(L, \mathcal{K})$. For the desired system behavior $K = \bigcap_{j \in I_K} pr(K_j)$, we have the following property about the relationship between the codiagnosability of $(L, \mathcal{K})$ and (L, K). The proof can be easily derived from Definitions 1 and 5, and Lemma 1.

Property 1: $(L, \mathcal{K})$ codiagnosable $\Rightarrow$ (L, K) codiagnosable $\Leftrightarrow$ $(L, \mathcal{K})$ codetectable.

Given a system language L with a set of multiple sub-specification languages $\mathcal{K} = \{K_j | j \in I_K\}$, Algorithm 1 can be used to check the codiagnosability of $(L, \mathcal{K})$. To test the codiagnosability of $(L, \mathcal{K})$, we apply Algorithm 1 to each sub-specification language K_j $(j \in I_K)$ individually. Similarly, diagnoser synthesis and failure diagnosis can be performed for each sub-specification language individually. When the system is in operation, if a local diagnoser i $(i \in I_M)$ unambiguously detects a failure that violates a sub-specification language K_j $(j \in I_K)$, then the diagnoser i reports that failure, i.e., the violated sub-specification language K_j.

For the case of multiple sub-specification languages, the complexities of codiagnosability testing, diagnoser synthesis and on-line failure diagnosis are linear in the number of sub-specification languages. That is, for l sub-specification languages, these complexities are $O(l \times |X| \times |Y|^{m+1} \times |\Sigma|^{m+1})$, $O(l \times m \times |X| \times |Y| \times |\Sigma|)$ and $O(l \times m \times |X| \times |Y| \times |\Sigma|)$ respectively. Here $|X|$ is the number of states in the system model and $|Y| = \max_{j \in I_K} |Y_j|$, where $|Y_j|$ is the number of states in the sub-specification model for K_j. For the special case where all automata models of sub-specification languages are subautomata of the system model, these complexity are $O(l \times |X|^{m+1} \times |\Sigma|^{m+1})$, $O(l \times m \times |X| \times |\Sigma|)$ and $O(l \times m \times |X| \times |\Sigma|)$ respectively.

To compute the delay of codiagnosability in a system with multiple sub-specification languages, we can apply Algorithm 2 for each sub-specification language. After computing each delay bound $d(L, K_j)$ $(j \in I_K)$, the maximum value over all sub-specifications is taken as the maximum delay of codiagnosability, i.e.,

$$d(L, \mathcal{K}) = \max_{j \in I_K} d(L, K_j).$$

B. Multiple failure types

When a system has events of multiple failure types, the definition of codiagnosability is given as follows.

Definition 6: Given a system model $G = (X, \Sigma, \alpha, x_0)$ with a set of failure types $\mathcal{F} = \{f_1, \cdots, f_l\}$, let $L = L(G)$ be the generated language of G and $\psi : \Sigma \to \mathcal{F} \cup \{\emptyset\}$ be a failure assignment function. Assume there are m local diagnosers with observation masks M_i $(i \in I_M = \{1, \cdots, m\})$. $(L, \mathcal{F})$ is said to be codiagnosable with respect to $\{M_i\}$ if for all $j \in I_F$, $(L, \{f_j\})$ is codiagnosable with respect to $\{M_i\}$.

To test the codiagnosability of $(L, \mathcal{F})$ as well as for the synthesis of the local diagnosers, we can apply the method developed for each single failure type individually. When we check the codiagnosability of $(L, \{f_j\})$, failure events of other failure types are treated as normal unobservable events. To construct local diagnosers for each failure of type j, we first define an equivalent sub-specification model H_j for each failure type f_j $(j \in I_F)$, and then apply the approach introduced in Section V to (L, H_j). When a system is in operation, all such diagnosers are running concurrently. Each diagnoser is responsible for detecting one type of failure. A f_j-type failure is reported if there exists a diagnoser at a certain local site that is responsible for detecting f_j-type failures and reaches an unambiguous set of states.

The complexities of codiagnosability testing, off-line diagnoser synthesis and on-line failure diagnosis are linear in the number of failure types. That is, for a set $\mathcal{F}$ with l failure types, if the system model G is given as a DFSM, then the complexities of these operations are $O(l \times |X|^{m+1} \times |\Sigma|^{m+1})$, $O(l \times m \times |X| \times |\Sigma|)$ and $O(l \times m \times |X| \times |\Sigma|)$ respectively. If G is given as a NFSM, the complexities are $O(l \times |X|^{2(m+1)} \times |\Sigma|^{m+1})$, $O(l \times m \times |X|^2 \times |\Sigma|)$ and $O(l \times m \times |X|^2 \times |\Sigma|)$ respectively.

Also, after defining the equivalent sub-specification model H_j for failure type f_j, the delay of codiagnosability in a system with multiple failure types can be computed as follows. We first apply Algorithm 2 to compute the delay for each (L, H_j). The delay of codiagnosability with respect to multiple failure types is the maximum value of all these delays, i.e.,

$$d(L, \mathcal{F}) = \max_{j \in I_F} d(L, H_j).$$

VII. Strong-(Co)diagnosability

The notion of codiagnosability guarantees that occurrence of any failure is detected within finite delay, but there is no guarantee that non-occurrence of failure is unambiguously known within bounded delay. The following example illustrates the situation.

Example 5: Consider the system introduced in Example 3, Table II shows three traces st executed in the system model G and their local observations $M_1(st)$ and $M_2(st)$. From the analysis in Example 4, we know that when the system executes either the failure trace $a\sigma_f daca^n \in L(G) - pr(K)$

or $a\sigma_f cba^{n+1} \in L(G) - pr(K)$ $(n \in \mathcal{N})$, there exists one local diagnoser that can unambiguously detect that failure. However, when the system executes the non-faulty trace $abaca^n \in pr(K)$, both local diagnosers cannot unambiguously detect that no failure has happened. This is because regardless of what n is, $abaca^n \in pr(K)$ has the same observation as $a\sigma_f cba^{n+1} \in L(G) - pr(K)$ at the first local site, and as $a\sigma_f daca^n \in L(G) - pr(K)$ at the second local site.

TABLE II
POSSIBLE SYSTEM EXECUTIONS AND THEIR LOCAL OBSERVATIONS

st	$M_1(st)$	$M_2(st)$
$abaca^n$	aba^{n+1}	$aaca^n$
$a\sigma_f daca^n$	ada^{n+1}	$aaca^n$
$a\sigma_f cba^{n+1}$	aba^{n+1}	aca^{n+1}

To capture the additional property of being able to be unambiguous about non-faulty executions, we introduce the notion of strong-codiagnosability. The purpose of this notion is that if a system executes an infinitely long non-faulty trace, then there must exist infinitely many instances when it is known without ambiguity that no failure has occurred. Strong codiagnosability captures the ability of being certain about failure as well as non-failure conditions within bounded delay.

Definition 7: Let L be a system language and K be a specification language ($K \subseteq L$). Assume there are m local diagnosers with observation masks M_i $(i \in I_M = \{1, \cdots, m\})$. (L, K) is said to be strongly-codiagnosable with respect to $\{M_i\}$ if it is codiagnosable and

$$(\exists n \in \mathcal{N})(\forall s \in pr(K))$$

$$(\forall st \in pr(K), |t| \geq n \text{ or } st \text{ deadlocks}) \Rightarrow$$

$$(\exists i \in I_M)(\forall u \in M_i^{-1} M_i(st) \cap L, u \in pr(K)). \quad (2)$$

Note that if an offending trace $s \in pr(K)$ exists, then non-faulty condition of s is not known for ever for some non-faulty extension beyond s. I.e., non-faulty condition for this extension is known at only finitely many instances, namely for some prefixes of s. In addition to the codiagnosability condition, the strong-codiagnosability requires that if s is a non-faulty trace in L and st is a non-faulty trace extended by sufficient number of transitions (or st deadlocks), then at least at one local site all st-indistinguishable traces are non-faulty as well.

To verify the strong-codiagnosability of (L, K), we need to check condition (2) as well as the codiagnosability of (L, K). Let $L(G) = L$ and $L(H) = pr(K)$. To check condition (2), we can construct a testing automaton to track traces $s \in L(H)$, $u_1 \in L(G) - L(H)$ and $u_2 \in L(G) - L(H)$ in the space $Y \times (X \times \overline{Y}) \times (X \times \overline{Y})$ with the properties $M_1(s) = M_1(u_1)$ and $M_2(s) = M_2(u_2)$. Then following the similar approach as in Algorithm 1 condition (2) can be verified by checking the absence of any "offending" cycles. Also, the computational complexity of this verification procedure is polynomial in the number of states and transitions of it's testing automaton.

Remark 10: The notion of strong-diagnosability for centralized case can be defined as in Definition 7 with the local observation mask M_i replaced by the central observation mask M. We provide an example to illustrate that a diagnosable system may not be strongly-diagnosable even in the centralized case. Let $L = \{\varepsilon, ab^*, a\sigma_f c^*\}$ and $K = \{\varepsilon, ab^*\}$, where $M(a) = a$, $M(c) = c$ and $M(b) = M(\sigma_f) = \varepsilon$. It follows that (L, K) is diagnosable owing to the observability of event c. However, since $M(ab^*) = M(a\sigma_f) = a$, (L, K) is not strongly-diagnosable.

VIII. CONCLUSION

In this paper the decentralized diagnosis of discrete event systems is studied. The notion of decentralized failure diagnosis is formalized by introducing the definition of codiagnosability. Algorithms of complexity polynomial in the size of the system and the non-fault specification are provided for (i) testing codiagnosability, (ii) computing the bound in delay of diagnosis, (iii) off-line synthesis of individual diagnosers, and (iv) on-line diagnosis using them. The complexity with respect to the number of diagnosers is exponential for the the first two algorithms and linear for the last two algorithms. It is further shown that an order complexity reduction results when specification language model is given as a subgraph of the system model, which is the case for the setting of failure events/states. A notion of strong-codiagnosability is introduced to capture the ability of being certain about the failure or non-failure conditions in a system within bounded delay. The approach for analyzing strong-codiagnosability is similar to that for analyzing codiagnosability.

REFERENCES

[1] A. Beneveniste, E. Fabre, S. Haar, and C. Jard. Diagnosis of asynchronous discrete-event systems: A net unfolding approach. *IEEE Transactions on Automatic Control*, 48(5):714–727, 2003.

[2] R. K. Boel and J. H. van Schuppen. Decentralized failure diagnosis for discrete-event systems with constrained communication between diagnosers. In *Proceedings of International Workshop on Discrete Event Systems*, 2002.

[3] A. Bouloutas, G. W. Hart, and M. Schwartz. Simple finite-state fault detectors for communication networks. *IEEE Trans. on Communications*, 40(3):477–479, March 1992.

[4] Y. Brave and M. Heymann. On stabilization of discrete event processes. *International Journal of Control*, 51(5):1101–1117, 1990.

[5] S. R. Das and L. E. Holloway. Characterizing a confidence space for discrete event timings for fault monitoring using discrete sensing and actuation signals. *IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans*, 30(1):52–66, 2000.

[6] R. Debouk, S. Lafortune, and D. Teneketzis. Coordinated decentralized protocols for failure diagnosis of discrete event systems. *Discrete Event Dynamical Systems: Theory and Applications*, 10:33–79, 2000.

[7] D. N. Godbole, J. Lygeros, E. Singh, A. Deshpande, and A. E. Lindsey. Communication protocols for a fault-tolerant automated highway system. *IEEE Transactions on Control Systems Technology*, 8(5):787–800, September 2000.

[8] C.N. Hadjicostis and G.C. Verghese. Power system monitoring based on relay and circuit breaker information. In *Proceedings of the 2001 IEEE International Symposium on Circuits and Systems*, volume 2, pages 197–200, May 2001.

[9] S. Jiang, Z. Huang, V. Chandra, and R. Kumar. A polynomial time algorithm for diagnosability of discrete event systems. *IEEE Transactions on Automatic Control*, 46(8):1318–1321, 2001.

[10] S. Jiang and R. Kumar. Diagnosis of repeated failures for discrete event systems with linear-time temporal logic specifications. In *Proceedings of IEEE Conference on Decision and Control*, pages 3221–3226, Maui, Hawaii, 2003.

[11] S. Jiang and R. Kumar. Failure diagnosis of discrete event systems with linear-time temporal logic fault specifications. *IEEE Transactions on Automatic Control*, 49(6):934–945, 2004.

[12] S. Jiang, R. Kumar, and H. E. Garcia. Diagnosis of repeated/intermittent failures in discrete event systems. *IEEE Transactions on Robotics and Automation*, 19(2):310–323, 2003.
[13] R. Kumar and V. K. Garg. *Modeling and Control of Logical Discrete Event Systems*. Kluwer Academic Publishers, Boston, MA, 1995.
[14] R. Kumar, V. K. Garg, and S. I. Marcus. Language stability and stabilizability of discrete event dynamical systems. *SIAM Journal of Control and Optimization*, 31(5):1294–1320, September 1993.
[15] F. Lin. Diagnosability of discrete event systems and its applications. *Discrete Event Dynamic Systems: Theory and Applications*, 4(1):197–212, 1994.
[16] J. Lygeros, D. N. Godbole, and M. Broucke. A fault tolerant control architecture for automated highway system. *IEEE Transactions on Control Systems Technology*, 8(2):205–219, March 2000.
[17] R. E. Miller and A. K. Arisha. Fault identification in networks by passive testing. In *Proceedings of the IEEE Annual Simulation Symposium*, pages 277–284, 2001.
[18] C. M. Ozveren, A. S. Willsky, and P. J. Antsaklis. Stability and stabilizability of discrete event dynamical systems. *Journal of ACM*, 38(3):730–752, July 1991.
[19] D. Pandalai and L. Holloway. Template languages for fault monitoring of timed discrete event processes. *IEEE Transactions on Automatic Control*, 45(5):868–882, May 2000.
[20] A. D. Pouliezos and G. S. Stavrakakis. *Real time fault monitoring of industrial processes*. Kluwer Academic Publishers, Boston, MA, 1994.
[21] W. Qiu and R. Kumar. Distributed failure diagnosis under bounded delay using immediate observation passing protocol. In *Proceedings of 2005 American Control Conference*, Portland, OR, June 2005.
[22] P. J. Ramadge and W. M. Wonham. The control of discrete event systems. *Proceedings of IEEE: Special Issue on Discrete Event Systems*, 77:81–98, 1989.
[23] S. L. Ricker and J. H. van Schuppen. Decentralized failure diagnosis with asynchronous communication between supervisors. In *Proceedings of the European Control Conference*, pages 1002–1006, 2001.
[24] R. H. Kwong S. H. Zad and W. M. Wonham. Fault diagnosis in timed discrete-event systems. In *Proceedings of the 38th IEEE Conference on Decision and Control*, pages 1756–1761, Phoenix, AZ, 1999.
[25] M. Sampath and S. Lafortune. Active diagnosis of discrete event systems. *IEEE Transactions on Automatic Control*, 43(7):908–929, 1998.
[26] M. Sampath, R. Sengupta, S. Lafortune, K. Sinaamohideen, and D. Teneketzis. Diagnosability of discrete event systems. *IEEE Transactions on Automatic Control*, 40(9):1555–1575, September 1995.
[27] M. Sampath, R. Sengupta, S. Lafortune, K. Sinaamohideen, and D. Teneketzis. Failure diagnosis using discrete event models. *IEEE Transactions on Control Systems Technology*, 4(2):105–124, March 1996.
[28] R. Sengupta and S. Tripakis. Decentralized diagnosis of regular language is undecidable. In *Proceedings of IEEE Conference on Decision and Control*, pages 423–428, Las Vegas, NV, December 2002.
[29] R. Su, W. M. Wonham, J. Kurien, and X. Koutsoukos. Distributed diagnosis for qualitative systems. In *Proceedings of International Workshop on Discrete Event Systems*, 2002.
[30] S. Tripakis. Decentralized control of discrete-event systems with bounded or unbounded delay communication. *IEEE Transactions on Automatic Control*, 49(9):1489–1501, 2004.
[31] G. Westerman, R. Kumar, C. Stroud, and J. R. Heath. Discrete event systems approach for delay fault analysis in digital circuits. In *Proceedings of 1998 American Control Conference*, Philadelphia, PA, 1998.
[32] Y. Willner and M. Heymann. Language convergence in controlled discrete-event systems. *IEEE Transactions on Automatic Control*, 40(4):616–627, 1995.
[33] T. Yoo and H. E. Garcia. Event diagnosis of discrete-event systems with uniformly and nonuniformly bounded diagnosis delays. In *Proceedings of 2004 American Control Conference*, pages 5102–5107, Boston, MA, June 2004.
[34] T. S. Yoo and S. Lafortune. Polynomial-time verification of diagnosability of partially observed discrete-event systems. *IEEE Transactions on Automatic Control*, 47(9):1491–1495, 2002.
[35] S. H. Zad, R. H. Kwong, and W. M. Wonham. Fault diagnosis in discrete-event systems: Framework and model reduction. *IEEE Transactions on Automatic Control*, 48(7):1199–1212, 2003.