Diagnosis of Discrete Event Systems Using Decentralized Architectures[*]

Yin Wang
Dept. of EECS
University of Michigan
Ann Arbor, MI 48109-2122
yinw@eecs.umich.edu

Tae-Sic Yoo
Idaho National Laboratory
Idaho Falls, ID 83403-2528
Tae-Sic.Yoo@inl.gov

Stéphane Lafortune
Dept. of EECS
University of Michigan
Ann Arbor, MI 48109-2122
stephane@eecs.umich.edu

Abstract

Decentralized diagnosis of discrete event systems has received a lot of attention to deal with distributed systems or with systems that may be too large to be diagnosed by one centralized site. This paper casts the problem of decentralized diagnosis in a new hierarchical framework. A key feature is the exploitation of different local decisions together with appropriate rules for their fusion. This includes local diagnosis decisions that can be interpreted as "conditional decisions". Under this new framework, a series of new decentralized architectures are defined and studied. The properties of their corresponding notions of decentralized diagnosability are characterized and their relationship with existing work described. Corresponding verification algorithms are also presented and on-line diagnosis strategies discussed.

1 Introduction

Model-based diagnosis of Discrete Event Systems (DES) consists of detecting unobservable significant events (such as faults) that occur in a dynamic system modeled as a DES by performing inferencing driven by sequences of observable events. A number of approaches have been proposed by both the Artificial Intelligence (AI) and the control engineering research communities; see [10, 11, 15, 17, 18] and the references therein.

Decentralized and distributed diagnostic protocols become necessary to deal with diagnosis in distributed systems where the information is decentralized. Approaches using decentralized models, called *communicating automata*, can be found in the AI literature [1, 12]. While decentralized models could potentially reduce the state space exponentially, the actual complexity of the diagnosis algorithms rely on the partition of the system model and the selection of communicating events between local models. These are intricate problems without effective algorithms. On the other hand, works on decentralized diagnosis in the control engineering literature such as [5, 19] employ a global system model. The global model is built from component models automatically via synchronous or asynchronous composition. Diagnosability verification in this approach suffers from the state explosion problem. After off-line verification however, online diagnosis decisions can be computed on-the-fly and a global model is not necessary. Our approach belongs to the latter category.

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

In most works on decentralized diagnosis of DES, there are several local "sites" where sensors report their data. 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., [5]. Local diagnosers then report their decisions about the events to be diagnosed. These decisions may or may not be fused at a coordinating site, according to the properties of the architecture. Generally speaking, distributed architectures for 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. Recently, distributed and decentralized diagnosis problems have received a lot of attention: see, e.g., [6, 3, 2, 7, 8, 20, 22, 21, 13].

Debouk *et al.* [5] developed several communication protocols for decentralized diagnosis. The one termed Protocol 3 is particularly relevant to the work in this paper. Under Protocol 3, diagnosers at local sites operate independently (namely, without communicating among each other) and local decisions about the occurrence of significant events in the system are merged by simple memoryless Boolean operations (disjunction). Qiu and Kumar [13] revisited this idea and developed a polynomial-time verification algorithm for checking whether a system can be diagnosed under Protocol 3. Following the assumption of no communication among sites, Sengupta and Tripakis [20] examined the extreme case where the global decision-maker could be any arbitrary memoryless function and local decisions may not belong to a finite set. They called this situation *joint diagnosability* and they proved that joint diagnosability is undecidable.

In this paper, we are interested in decentralized architectures that lie in the range between Protocol 3 of [5] and joint diagnosability of [20]. One of the contributions of this work is the development of a general hierarchical framework for decentralized diagnosis that incorporates Protocol 3 at the very bottom and joint diagnosability at the very top. It will be shown that this novel framework leads to a hierarchy of architectures and encompasses many existing decentralized architectures for diagnosis. Another contribution of this work is the precise characterization of a set of new architectures that all generalize Protocol 3. After presenting our general framework in Section 3, we consider in Section 4 a conjunctive fusion rule as the global decision-maker, a dual to the disjunctive function used in Protocol 3. In Section 5, we consider more general architectures where local sites can issue an additional decision, which can be interpreted as a "conditional decision", about the event to be diagnosed. One such decision is: "Positive if no other site says Negative." The diagnosability properties of architectures with two decisions are carefully analyzed in Section 5 and revisited in Section 7. Architectures with multiple (more than two) decisions are studied in Section 6. We emphasize that all the architectures discussed in this paper do not require a coordinating site, i.e., they can be implemented in a fully distributed environment such as sensor networks. This will become clear as the architectures are presented. Furthermore, polynomial verification algorithms of diagnosability under these architectures are developed.

Our approach builds on the results in [5] regarding Protocol 3 and is inspired by recent work in [25, 27] 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 [5] to improve upon Protocol 3, namely our results are different in nature from Protocols 1 and 2 in [5] 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 2. The general decentralized diagnosis framework is presented in Section 3. The main results are then presented in the following sections. Preliminary and partial versions of this work were presented in [23, 24]. We note that results related to some of those in this paper have been developed independently in [9].

2 Diagnosing Unobservable Events

The system is modeled as a finite state automaton $G = (Q, \Sigma, \delta, q_0)$, where Q is the state space, Σ is the set of events, δ is the partial transition function, and q_0 is the initial state. The model G accounts for all possible behaviors of the system. The behavior of the system is described by the prefix-closed language $\mathcal{L}(G)$ generated by G, often denoted 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 significant unobservable event $e_d \in \Sigma_{uo}$ whose occurrences must be diagnosed, i.e., detected by model-based inferencing using observed events only. We will see later that extension to inferencing multiple events is straightforward. A string or a trace $s \in L$ is called *positive* if it contains e_d, i.e., if there exist $u, v \in \Sigma^*$ such that $s = ue_dv$. Otherwise the string is called *negative*. The set of all prefixes of trace s is denoted by $\overline{s}$. We denote by L/s the post language of L after s, i.e., $L/s = \{t | st \in L\}$. We refer the reader to [4] for further explanations of the above notations.

Given $\mathcal{P}$ the standard projection operation from Σ^* to Σ_o^* that erases unobservable events[1], we have that $\mathcal{P}^{-1}(s) := \{t \in \Sigma^* : \mathcal{P}(t) = s\}$. We introduce the notation $\mathcal{E}(s) = \mathcal{P}^{-1}\mathcal{P}(s) \cap L$ to denote the set of "estimate traces", assuming s is executed by the system and $\mathcal{P}(s)$ is observed. Thus $t \in \mathcal{E}(s)$ *iff* $t \in L$ and $\mathcal{P}(t) = \mathcal{P}(s)$. Therefore, $\mathcal{E}(s)$ is the estimate of the behavior of the system consistent with the model L after $\mathcal{P}(s)$ has been observed.

We further introduce the notation $\mathcal{E}^{pre,k}(u) := \{s \mid \exists t, |t| \geq k, st \in \mathcal{E}(u)\}$ for the prefixes of $\mathcal{E}(u)$. In words, $\mathcal{E}^{pre,k}(u)$ is the estimate of the system behavior at least k events ago when $\mathcal{P}(u)$ has been observed. We drop the k in the superscript hereafter since we always use symbol k and it is always a fixed constant throughout the paper.

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

A1 $\mathcal{L}(G)$ is live, i.e., there is at least one transition defined at each state of G.

Assumption **A1** can be relaxed easily at the expense of extra statements regarding the diagnosability of terminating traces.

The following definition of diagnosability is the starting point of our development.

Definition 1 *Language L is said to be diagnosable w.r.t. e_d and $\mathcal{P}$ if there exists a function $h : \Sigma_o^* \to$ {positive, negative}, s.t.*
1. $(\exists k \in \mathbb{N})(\forall st \in L$ s.t. s is positive and $|t| \geq k)$, $h(\mathcal{P}(st)) =$ positive*;*
2. $\forall u \in L$ s.t. u is negative, $h(\mathcal{P}(u)) =$ negative.

In above definition, the first condition says that all positive traces can be diagnosed within bounded delay. The second condition guarantees that there is no false positive. When e_d has occurred without a sufficiently long extension, either decision is allowed, i.e., we allow temporary false negatives.

Given the above definition of diagnosability, we are interested in finding out when there exists a function h that makes a given system diagnosable. This task seems to be prohibitive as function h is arbitrary in Definition 1. However, the following equivalent language-based definition provides insight into the problem and leads to polynomial verification algorithms.

Definition 2 *[17, 18] Language L is said to be positive-diagnosable w.r.t. e_d and $\mathcal{P}$ if the following is true:*
$(\exists k \in \mathbb{N})(\forall st \in L$ s.t. s is positive and $|t| \geq k)(\forall u \in \mathcal{E}(st))$ u is positive.

[1]We use $\mathcal{P}$ for projection since the letter P will be used later to denote "Positive" in the constructions of verifiers.

The above definition means the following. Let s be a positive trace containing e_d and t be a sufficiently long continuation of s in L. Then any trace in L indistinguishable from st is also positive. Positive-diagnosability implies that all possible estimate traces of a sufficiently long positive trace are positive. Therefore, it is possible to diagnose the event e_d in s after observing $\mathcal{P}(st)$. Polynomial verification and on-line diagnosis algorithms for positive-diagnosability can be found in [26, 17].

Theorem 1 *Diagnosability* $\Leftrightarrow$ *Positive-diagnosability.*

Proof: Violation of positive-diagnosability implies that there exists an arbitrarily long positive trace st with an indistinguishable negative trace $u \in \mathcal{E}(st)$. Then $h(\mathcal{P}(st)) = h(\mathcal{P}(u))$ would always give a wrong diagnosis result for either st or u.

If a system is positive-diagnosable, we construct h as $h(\mathcal{P}(s)) =$ *positive* if and only if $\mathcal{E}(s)$ contains positive traces only. Thus, all sufficiently long positive traces will be diagnosed according to the definition. On the other hand, for a negative trace u, as $u \in \mathcal{E}(u)$, $h(\mathcal{P}(u)) =$ *negative*. ∎

To facilitate the reasoning in decentralized settings, we present the following dual and equivalent definition to positive-diagnosability.

Definition 3 *Language L is said to be negative-diagnosable w.r.t. e_d and $\mathcal{P}$ if the following is true:*
$(\exists k \in \mathbb{N})(\forall u \in L$ *s.t.* u *is negative*$)(\forall s \in \mathcal{E}^{pre}(u))$ s *is negative.*

The above definition means the following. Let u be a negative trace in L. Then any trace that is indistinguishable from u must have a negative prefix before its last k events. Negative-diagnosability implies that if e_d has not occurred, we are always able to infer that *some events ago*, the system *was* negative.

Theorem 2 *Diagnosability* $\Leftrightarrow$ *Negative-diagnosability.*

Proof: Violation of negative-diagnosability implies that there exists a negative trace u with an arbitrarily long indistinguishable trace $st \in \mathcal{E}(u)$, where s is positive. Then $h(\mathcal{P}(st)) = h(\mathcal{P}(u))$ would always give a wrong diagnosis result for either st or u.

If a system is negative-diagnosable, we construct h as $h(\mathcal{P}(u)) =$ *negative* if and only if $\mathcal{E}^{pre}(u)$ contains negative traces only. Thus all negative traces would be diagnosed correctly according to the definition. On the other hand, if a positive trace s with a sufficiently long extension t happens, $\mathcal{E}^{pre}(st)$ contains a positive trace as $s \in \mathcal{E}^{pre}(st)$; thus $h(\mathcal{P}(st)) =$ *positive*. ∎

3 Decentralized Architecture for Diagnosis

Assume a system is jointly observed by many sites, where each site can only observe a subset of the observable events executed by the system. The problem of decentralized diagnosis can be paraphrased as follows: How can these sites jointly discover the occurrence of event e_d?

Formally, the decentralized architecture we consider is depicted in Fig. 1. In that figure, there are n local sites jointly diagnosing the system G by observing subsets of the set of observable events Σ_o, denoted by $\Sigma_{o,1}, \dots, \Sigma_{o,n}$, respectively. Each block $\mathcal{P}_i$ in the figure denotes the projection operations from Σ^* to $\Sigma_{o,i}^*$. The notions of projection and estimate set are extended to the above decentralized setting in a natural way. $\mathcal{P}_i^{-1}(s) := \{t \in \Sigma^* : \mathcal{P}_i(t) = s\}$, $\mathcal{E}_i(s) = \mathcal{P}_i^{-1}\mathcal{P}_i(s) \cap L$ and $\mathcal{E}_i^{pre}(u) := \{s \mid \exists t, |t| \geq k, st \in \mathcal{E}_i(u)\}$. The blocks $D_1, \dots, D_n$ in the figure denote the local decision makers. D_i is a function $h_i : \Sigma_{i,o}^* \to LD$, where LD is the set of local decisions. For example, D_i might be a Moore automaton where $\Sigma_{o,i}$ is the event set or input alphabet and LD is the output alphabet. Local decisions are fused to obtain the global decision. They may be fused in a distributed way so the global decision block, denoted by the dashed box, is optional. To facilitate reasoning, we think of the dashed block as a centralized function $H : LD^n \to \{$positive, negative$\}$, but all global functions considered in this paper allow distributed implementations.

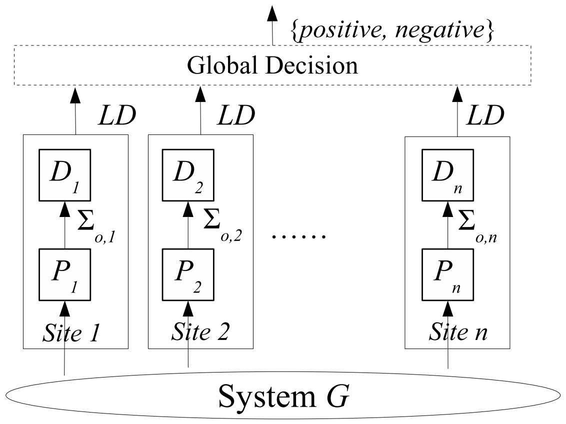


Figure 1: Decentralized Architecture

Within the context of the above architecture, we would like to design local diagnosis algorithms and communication protocols such that local sites can jointly diagnose occurrences of event e_d. Obviously, if each site reports every event observed under zero communication delay or with a globally synchronized timestamp, then the problem can be solved by a centralized diagnoser. The time constraint can be relaxed as long as the messages are globally ordered. In Protocol 1 of [5], each site encapsulates its observations and inferences in the structure called "extended diagnoser state" such that the global coordinator can recover the centralized diagnoser state. In Protocol 2 of [5], sites communicate less information—diagnoser state—to the coordinator and the coordinator cannot exactly recover the centralized diagnoser state. As a result, only a subset of systems diagnosable by Protocol 1 can be diagnosed. In this paper, for the sake of scalability, we would like to diagnose the system in a "distributed" fashion using much simpler rules. For this reason, we introduce the following assumption.

A2 The global fusion rule is memoryless.

Assumption **A2** means that the global decision is completely based on one snapshot of all local decisions. Note that we do not require global ordering of local decisions, but when a global decision is requested, we need to know the latest decision of every local site.

We can extend Definition 1 to our decentralized setting.

Definition 4 *Consider local projections $\mathcal{P}_i$ and the global decision function H, as described in the architecture of Fig. 1. Language L is H-codiagnosable if there exist local decision functions h_i, $i = 1, \ldots, n$, s.t. the two following conditions hold:*
1. $(\exists k \in \mathbb{N})(\forall st \in L$ *s.t.* s *is positive and* $|t| \geq k)$, $H(h_1(\mathcal{P}_1(st)), \ldots, h_n(\mathcal{P}_n(st))) =$ positive;
2. $\forall u \in L$ *s.t.* u *is negative,* $H(h_1(\mathcal{P}_1(u)), \ldots, h_n(\mathcal{P}_n(u))) =$ negative.

The above general definition of codiagnosability means that every sufficiently long positive trace is correctly diagnosed globally, and there is no false positive. The terminology "codiagnosable" is used to emphasize that the sites operate as a team and is consistent with the terminology used in decentralized control of discrete event systems (coobservability).

With Definition 4, the notion of joint diagnosability of [20] can be stated within our framework by setting the h_i functions to be identity functions and H to be arbitrary.

Definition 5 *[20] Language L is said to be jointly diagnosable w.r.t. e_d and local projections $\mathcal{P}_1, \ldots, \mathcal{P}_n$ if the following is true:*

$(\exists k \in \mathbb{N})(\forall st \in L$ *s.t.* s *is positive and* $|t| \geq k)(\forall u \in L$ *s.t.* u *is negative*$)(\exists i \in \{1, \ldots, n\})(\mathcal{P}_i(st) \neq \mathcal{P}_i(u))$.

Definition 5 means that a sufficiently long positive trace and a negative trace must be distinguishable at at least one local site. Obviously this is a necessary condition for local sites to jointly diagnose the system under a memoryless global function. On the other hand, if every pair of sufficiently long positive and negative traces looks different at some site, let every site output the whole string it has observed so far. One can always design a memoryless global function H that makes the system H-codiagnosable. Therefore, joint diagnosability represents exactly the set of systems that can be diagnosed under our decentralized framework. Unfortunately, the verification of joint diagnosability has been shown to be undecidable in [20]. Hence, our objective is to identify stronger notions of diagnosability that are decidable, while keeping Assumption **A2**. Relaxing Assumption **A2** to allow (limited) memory in the global decision block leads to an entirely different framework worthy of research but beyond the scope of this paper.

We further assume that the number of local decisions, i.e., $|LD|$, is finite, and restrict the global function H by the following assumption.

A3 The global decision block does not know the source of a local decision, nor can it count the number of sites issuing the same local decision.

Assumption **A3** implies that local decisions are symmetric, i.e., it does not matter if a given local decision is issued by one site or another site, or by both. Furthermore, the absence of counting rules out functions such as majority (voting) and parity. We enforce this assumption because we believe it is one of the most restrictive and simplest assumptions. It allows the architecture to work in extreme environments, especially fully distributed and symmetric environments with limited computation power, such as sensor networks.

If $|LD| = k$ and there are n local sites, under **A2**, the size of the input domain of the global function H is reduced from infinite (historical input) to k^n (memoryless) possibilities. Under both **A2** and **A3**, it is further reduced to 2^k, because each local decision can only be either "present" or "not present". In the next two sections, we will analyze the possible inputs for the cases of $k = 1$ and $k = 2$.

4 Decentralized Diagnosis with One Local Decision

4.1 Definitions

If $|LD| = 1$, i.e., there is only one local decision, denoted as "A", at the global fusion block, then there are only two different inputs. One is that all sites are silent, the other is that some site reports "A"; note that due to Assumption **A3**, when two or more sites report "A", we are still in the second case. The fusion block can assign either "Positive" or "Negative" to the two inputs, resulting in $2^2 = 4$ different fusion rules, as described in Table 1. In that table, "Nothing" means that all sites are silent and "A" means that at least one site reports "A".

(input cases represented by local decision received)		Global function	
Nothing	A		
(output for the two input cases)			
Negative	Negative	H_1^1	(not viable)
Negative	Positive	H_2^1	(DISJ-CODIAG)
Positive	Negative	H_3^1	(CONJ-CODIAG)
Positive	Positive	H_4^1	(not viable)

Table 1: Fusion Rules with One Local Decision

Functions H_1^1 and H_4^1 are not viable because the output is always positive or negative. However, functions H_2^1 and H_3^1 are viable and will correspond to some classes of languages that can be diagnosed in the context of the general Definition 4, if the global function H there is instantiated by H_2^1 or H_3^1, respectively, leading to the notions of "H_2^1-codiagnosable" and "H_3^1-codiagnosable". Specifically, H_2^1 means that the global decision is positive if and only if some site reports "A". If we interpret "A" as "positive" in this case, then H_2^1 means that the overall decision is positive if and only if at least one site reports positive. On the other hand, H_3^1 in Table 1 means that the global decision is positive if and only if no site says "A". If we interpret "A" as "negative" in this case, H_3^1 means that the overall decision is positive if and only if no site says negative.

It is convenient for the discussion to follow to introduce the following terminology:

$$\text{disjunctive-codiagnosable or DISJ-CODIAG} \Leftrightarrow H_2^1\text{-codiagnosable} \tag{1}$$

$$\text{conjunctive-codiagnosable or CONJ-CODIAG} \Leftrightarrow H_3^1\text{-codiagnosable} \tag{2}$$

We adopt here the names "disjunctive-codiagnosability" and "conjunctive-codiagnosability" in order to facilitate comparisons between our work and that in [25, 27] for *coobservability* in decentralized control. It is important to note that in CONJ-CODIAG, the only local decision made by diagnosers can be interpreted as "negative," and the system is diagnosed to be positive if and only if there is no diagnoser that reports negative. Thus, this architecture is closely analogous to the conjunctive architecture considered in [16, 25] for decentralized control, where "disable" is the only local decision employed and an event is enabled if no site disables it. Similarly, in DISJ-CODIAG, the system is diagnosed to be positive if and only if at least one diagnoser reports "positive," which is closely analogous to the disjunctive architecture [25] for decentralized control, where an event is enabled if at least one site enables it.

4.2 Properties

The notions of decentralized diagnosability termed DISJ-CODIAG and CONJ-CODIAG introduced in the preceding section characterize classes of diagnosable systems (languages) in terms of decision rules. Our objective is to precisely characterize these classes of diagnosable languages. Interestingly, there are language-based definitions of decentralized diagnosability that are equivalent to the rule-based DISJ-CODIAG and CONJ-CODIAG. These can be obtained by building on the centralized Definitions 2 and 3.

Definition 6 *Language L is said to be positive-codiagnosable w.r.t. e_d, $\mathcal{P}_1, \ldots, \mathcal{P}_n$ if the following is true:*

$(\exists k \in \mathbb{N})(\forall st \in L$ *s.t.* s *is positive and* $|t| \geq k)(\exists i \in \{1, \ldots, n\})(\forall u \in \mathcal{E}_i(st))$ u *is positive.*

The above definition means the following. Let s be a positive trace and let t be a sufficient 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 positive. This definition is exactly the same as the definition in [5] of "diagnosability under Protocol 3," which is revisited in [13] under the name "co-diagnosability" and in [24] under the name "F-codiagnosability".

Definition 7 *Language L is said to be negative-codiagnosable w.r.t. e_d, $\mathcal{P}_1, \ldots, \mathcal{P}_n$, if the following is true:*

$(\exists k \in \mathbb{N})(\forall u \in L$ *s.t.* u *is negative*$)(\exists i \in \{1, \ldots, n\})(\forall s \in \mathcal{E}_i^{pre}(u))$ s *is negative.*

The above definition means the following. If trace u is negative, then there must exist one local site i such that any trace in L indistinguishable from u at site i has a negative prefix before its last k events, i.e., site i is sure that the system *was negative k events ago*.

Next we establish relationships between the above rule-based and language-based definitions.

Theorem 3 DISJ-CODIAG $\Leftrightarrow$ *Positive-codiagnosability.*

Proof: Violation of positive-codiagnosable implies that there exists an arbitrarily long positive trace st s.t. $\forall i, \exists u_i \in \mathcal{E}_i(st)), u_i$ is negative. Suppose that the system can be diagnosed by H_2^1. Then if st happens, some site, say i, would report A. Since $\mathcal{P}_i(st) = \mathcal{P}_i(u_i)$, site i would still report A if u_i had occurred, thus resulting in a wrong diagnosis decision. Therefore, the system is not DISJ-CODIAG.

If the system is positive-codiagnosable, we construct the local decision functions as follows:

$$h_i(s) = \begin{cases} A & \textbf{if } \mathcal{E}_i(s) \text{ contains positive traces only} \\ \text{nothing} & \textbf{otherwise.} \end{cases} \tag{3}$$

Then, if an arbitrarily long positive trace st occurs, according to positive-codiagnosability, some site i satisfies the condition that $\mathcal{E}_i(st)$ contains positive traces only. If a negative trace u occurs, since $u \in \mathcal{E}_i(u)$, no site reports A and the system is diagnosed as negative. ■

Theorem 4 CONJ-CODIAG $\Leftrightarrow$ *negative-codiagnosable.*

Proof: Violation of negative-codiagnosability implies that there exists a negative trace u s.t. $\forall i, \exists s_i t_i \in \mathcal{E}_i(u)), s_i$ is positive. Suppose that the system can be diagnosed by H_3^1. Then if u happens, some site, say i, would report A. Since $\mathcal{P}_i(u) = \mathcal{P}_i(s_i t_i)$, site i would still report A if $s_i t_i$ occurs, thus resulting in a wrong diagnosis decision. Therefore, the system is not CONJ-CODIAG.

If a system is negative-codiagnosable, we construct h_i as follows:

$$h_i(s) = \begin{cases} A & \textbf{if } \mathcal{E}_i^{pre}(s) \text{ contains negative traces only} \\ \text{nothing} & \textbf{otherwise.} \end{cases} \tag{4}$$

Then, all negative traces would be diagnosed correctly according to the definition of negative-codiagnosability. On the other hand, if a positive trace s with a sufficiently long extension t happens, since $s \in \mathcal{E}_i^{pre}(st)$ for all i, no site reports A and the system is diagnosed as positive. ■

Note that $\mathcal{E}_i(s)$ and $\mathcal{E}_i^{pre}(s)$ are both computable [4], thus Formulae (3-4) can be used to diagnose positive traces of a DISJ(CONJ)-CODIAG system online. The detailed local diagnoser synthesis algorithm for DISJ-CODIAG can be found in [5]. For CONJ-CODIAG, the synthesis algorithm is technical and beyond the scope of this paper.

Theorem 5 DISJ-CODIAG *and* CONJ-CODIAG *are incomparable w.r.t. the same event* e_d *and projections* $\mathcal{P}_1, \ldots, \mathcal{P}_n$.

Proof: The theorem is proved by Examples 1 and 2. ∎

Example 1 Consider the system G shown in Fig. 2, where $\Sigma_o = \{a, b, c\}$ and unobservable event e_d is to be diagnosed. There are two local sites, $n = 2$, $\Sigma_{o,1} = \{a, c\}$ and $\Sigma_{o,2} = \{b, c\}$. The system is DISJ-CODIAG with the following local decision functions. Site 1(2) reports A if and only if it has observed $a(b)$. There is no local function such that the system is CONJ-CODIAG. This is because to diagnose negative trace c^n, at least one site, say 1, should report A; as $\mathcal{P}_1(c^n) = \mathcal{P}_1(e_d bc^n)$, site 1 reports A with positive trace $e_d bc^n$ as well, resulting in a wrong global decision. ∎

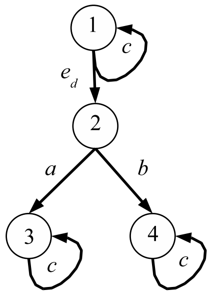


Figure 2: DISJ-CODIAG but not CONJ-CODIAG

Example 2 Consider the system G shown in Fig. 3, where the event to be diagnosed and the local observations are the same as in Example 1. The system is CONJ-CODIAG if each site keeps silent as long as it observes event c only. It is not DISJ-CODIAG though because at least one site, say 1, has to say A if positive trace $e_d c^n$ happens; but then negative trace bc^n will be diagnosed wrong since $\mathcal{P}_1(e_d c^n) = \mathcal{P}_1(bc^n)$. ∎

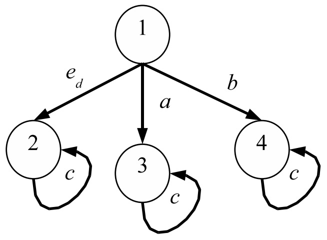


Figure 3: CONJ-CODIAG but not DISJ-CODIAG

Theorem 6 DISJ-CODIAG *or* CONJ-CODIAG *w.r.t. event e_d, local observations $\Sigma_{o,1}, \ldots, \Sigma_{o,n}$ implies centralized diagnosability w.r.t. event e_d and centralized observation $\Sigma_o = \Sigma_{o,1} \cup \cdots \cup \Sigma_{o,n}$. The reverse implication is not true in general.*

Proof: If a system is not (centrally) diagnosable, then there exist two arbitrarily long indistinguishable traces, where one is positive and the other is negative. These two traces are indistinguishable to each local site as well, thus there are no local functions that can diagnose both traces correctly.

The other part is proved by Example 3. ∎

Example 3 Consider the system G shown in Fig. 4, where $\Sigma_o = \{a, b, c\}$ and $\Sigma_{uo} = \{e_d\}$. There are two local sites, $n = 2$, $\Sigma_{o,1} = \{a, c\}$ and $\Sigma_{o,2} = \{b, c\}$. The system is not DISJ-CODIAG or CONJ-CODIAG because site 1 always observes ac^* and site 2 always observes bc^*, no matter whether e_d has occurred or not. ∎

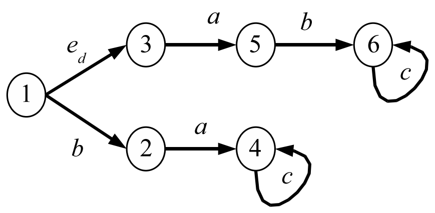


Figure 4: Diagnosable but not codiagnosable

Figure 5 summarizes the relationship among the various notions of codiagnosability discussed in this section.

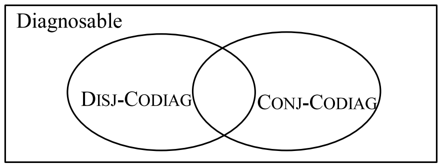


Figure 5: Relationship among notions of diagnosability

4.3 Verification

Verification refers to the problem of deciding whether a system is diagnosable w.r.t. some definition of diagnosability. In the case of DISJ(CONJ)-CODIAG, verification means that there exist functions $h_1, \ldots, h_n$ s.t. the system can be diagnosed by the disjunctive (conjunctive) global rule. This can be solved by extending verifiers [26] to the decentralized setting and building on the results in [13] for the case of DISJ-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 the *one-level verifier* $V_1 = (Q^{V_1}, (\Sigma \cup$

$\{\varepsilon\})^3, \delta^{V_1}, q_0^{V_1})$ as follows.

$$Q^{V_1} = \underbrace{Q \times \{N,P\}}_{s_1} \times \underbrace{Q \times \{N,P\}}_{s_2} \times \underbrace{Q \times \{N,P\}}_{s}$$
$$q_0^{V_1} = (q_0, N, q_0, N, q_0, N)\ .$$

where s_1, s_2 and s are traces in L and N (respectively, P) indicates that the corresponding trace is negative (respectively, positive). For the sake of readability, let $q_i' = \delta(q_i, \sigma)$. The transition function δ^{V_1} is defined as described below, for all cases where the corresponding transitions are defined:

$$\begin{array}{lcl}
\text{For } \sigma \in \Sigma_{o,1}, \sigma \in \Sigma_{o,2}, & & \\
\delta^{V_1}((q_1,l_1,q_2,l_2,q_3,l_3),\sigma\sigma\sigma) & = & (q_1',l_1,q_2',l_2,q_3',l_3) \\
\text{For } \sigma \in \Sigma_{o,1}, \sigma \notin \Sigma_{o,2}, & & \\
\delta^{V_1}((q_1,l_1,q_2,l_2,q_3,l_3),\sigma\varepsilon\sigma) & = & (q_1',l_1,q_2,l_2,q_3',l_3) \\
\delta^{V_1}((q_1,l_1,q_2,l_2,q_3,l_3),\varepsilon\sigma\varepsilon) & = & (q_1,l_1,q_2',l_2,q_3,l_3) \\
\text{For } \sigma \notin \Sigma_{o,1}, \sigma \in \Sigma_{o,2}, & & \\
\delta^{V_1}((q_1,l_1,q_2,l_2,q_3,l_3),\varepsilon\sigma\sigma) & = & (q_1,l_1,q_2',l_2,q_3',l_3) \\
\delta^{V_1}((q_1,l_1,q_2,l_2,q_3,l_3),\sigma\varepsilon\varepsilon) & = & (q_1',l_1,q_2,l_2,q_3,l_3) \\
\text{For } \sigma \in \Sigma_{uo} \text{ and } \sigma \neq e_d, & & \\
\delta^{V_1}((q_1,l_1,q_2,l_2,q_3,l_3),\sigma\varepsilon\varepsilon) & = & (q_1',l_1,q_2,l_2,q_3,l_3) \\
\delta^{V_1}((q_1,l_1,q_2,l_2,q_3,l_3),\varepsilon\sigma\varepsilon) & = & (q_1,l_1,q_2',l_2,q_3,l_3) \\
\delta^{V_1}((q_1,l_1,q_2,l_2,q_3,l_3),\varepsilon\varepsilon\sigma) & = & (q_1,l_1,q_2,l_2,q_3',l_3) \\
\text{For } \sigma = e_d, & & \\
\delta^{V_1}((q_1,l_1,q_2,l_2,q_3,l_3),e_d\varepsilon\varepsilon) & = & (q_1',P,q_2,l_2,q_3,l_3) \\
\delta^{V_1}((q_1,l_1,q_2,l_2,q_3,l_3),\varepsilon e_d\varepsilon) & = & (q_1,l_1,q_2',P,q_3,l_3) \\
\delta^{V_1}((q_1,l_1,q_2,l_2,q_3,l_3),\varepsilon\varepsilon e_d) & = & (q_1,l_1,q_2,l_2,q_3',P)\ .
\end{array}$$

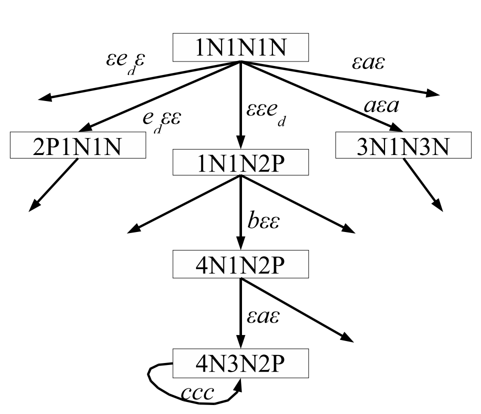
The intention of the construction of one-level verifier is summarized by the following proposition.

Proposition 7 *[13] The transition rule of V_1 guarantees that when there is a path from $q_0^{V_1}$ to state $(q_1, l_1, q_2, l_2, q_3, l_3)$, if we let s_1, s_2 and s be the traces formed by the 1st, 2nd and 3rd components, respectively, of the transitions along the path, then we have:*
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 positive if and only if l_1 (l_2 or l_3) $= P$;
3. $\mathcal{P}_1(s_1) = \mathcal{P}_1(s)$ and $\mathcal{P}_2(s_2) = \mathcal{P}_2(s)$.
On the other hand, if the above three conditions are satisfied, there must be a path in V_1 from $q_o^{V_1}$ to $(q_1, l_1, q_2, l_2, q_3, l_3)$ (not necessarily unique).

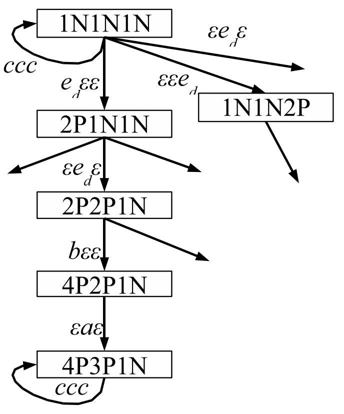
The proof can be found in [13] and thus it is omitted here. The proposition says that if s is the trace the system actually executes, then s_i, $i = 1, 2$, represents the trace that site i conjectures might have been executed. The construction of V_1 guarantees that all possible trace triples (s_1, s_2, s) that satisfy $\mathcal{P}_1(s_1) = \mathcal{P}_1(s)$ and $\mathcal{P}_2(s_2) = \mathcal{P}_2(s)$ are captured.

A one-level verifier state $(q_1, l_1, q_2, l_2, q_3, l_3)$ is called an (l_1, l_2, l_3)-state. For example, the initial state $q_0^{V_1}$ is an (N,N,N)-state. A strongly connected component (SCC) is called an (l_1, l_2, l_3)-SCC if every state in the SCC is an (l_1, l_2, l_3)-state. Figures 6(a) and 6(b) show parts of the one-level verifiers of Examples 2 and 1, respectively. There is an (N,N,P)-SCC in Fig. 6(a) and an (P,P,N)-SCC in Fig. 6(b).

The above construction can be extended to n local sites in a natural manner. 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


1N1N1N
$\varepsilon e_d \varepsilon$
$\varepsilon a \varepsilon$
$e_d \varepsilon\varepsilon$
$\varepsilon\varepsilon e_d$
$a \varepsilon a$
2P1N1N
1N1N2P
3N1N3N
$b\varepsilon\varepsilon$
4N1N2P
$\varepsilon a \varepsilon$
4N3N2P
ccc

(a) One-level Verifier of Example 2


1N1N1N
$\varepsilon e_d \varepsilon$
ccc
$e_d \varepsilon\varepsilon$
$\varepsilon\varepsilon e_d$
1N1N2P
2P1N1N
$\varepsilon e_d \varepsilon$
2P2P1N
$b\varepsilon\varepsilon$
4P2P1N
$\varepsilon a \varepsilon$
4P3P1N
ccc

(b) One-level Verifier of Example 1

Figure 6: One-level Verifier Examples.

each state, event σ 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 one-level verifier is polynomial in the number of system states and exponential in the number of local sites. For the case of diagnosing multiple events, we build a separate verifier for each event. Although the construction requires exponential time in the number of states in the worst case, the number of local sites is usually small in many application areas of interest. As the analogous problem in control (coobservability) is known to be PSPACE-complete [14], we probably need more restrictions on the system in order to handle a large number of local sites.

Testing of DISJ-CODIAG (positive-codiagnosability) or CONJ-CODIAG (negative-codiagnosability) using the one-level verifier is based on the following two theorems which provide verification algorithms.

Theorem 8 *[13] The language generated by system G is not positive-codiagnosable if and only if the one-level verifier V_1 of G has an (N,N,P)-SCC, where there exists an edge $\sigma_1\sigma_2\sigma$ such that $\sigma \neq \varepsilon$.*

The proof of the above theorem can be found in [13][2], where it is proved that a system is DISJ-CODIAG (termed co-diagnosable there) iff there is an (N,N,P)-cycle. We use strongly connected components here instead of cycles for the sake of consistency with the following new result.

Theorem 9 *The language generated by system G is not negative-codiagnosable if and only if the one-level verifier V_1 of G has an (P,P,N)-SCC, where for each site i, there exists an edge $\sigma_1\sigma_2\sigma$ such that $\sigma_i \neq \varepsilon$.*

Proof: (i) (P,P,N)-SCC with corresponding non-ε edges $\Rightarrow$ not negative-codiagnosable. Based on Proposition 7, we can induce an arbitrarily long trace triple $s_1t_1^n, s_2t_2^n, st^n$ from the (P,P,N)-SCC, where trace triple (s_1, s_2, s) corresponds to the prefix of the path that reaches the SCC from the initial state and (t_1, t_2, t) corresponds to the edges within the SCC. We know st^n must be negative, while s_1 and s_2 are positive. Furthermore, we can select t_1 and t_2 such that $t_1, t_2 \neq \varepsilon$. Then $s_1t_1^n, s_2t_2^n, st^n$ violates the definition of negative-codiagnosability.

[2]There is a technical difference in that sub-languages instead of events of the system are to be diagnosed in [13].

(ii) Not negative-codiagnosable $\Rightarrow$ (P,P,N)-SCC with corresponding non-ε edges. Not negative-codiagnosable implies there are negative trace u and positive traces s_1 and s_2 with arbitrarily long extensions t_1 and t_2 such that $\mathcal{P}_1(u) = \mathcal{P}_1(s_1t_1)$ and $\mathcal{P}_2(u) = \mathcal{P}_2(s_2t_2)$. By Proposition 7, these three traces should form a path in V_1. Since t_1 and t_2 could be arbitrarily long and V_1 has only a finite number of states, there must be a SCC. It is an (P,P,N)-SCC as s_1 and s_2 are positive and u is negative. Furthermore, t_1 (t_2) not equal to non-ε means there is an edge $\sigma_1\sigma_2\sigma$ in the SCC such that σ_1 (σ_2) $\neq \varepsilon$. ∎

Figure 6(a) has an (N,N,P)-SCC where there is a non-ε edge for the "P" part so the system is not DISJ-CODIAG. Figure 6(b) has a (P,P,N)-SCC where the edge ccc is non-ε for the two "P" parts, thus the system is not CONJ-CODIAG.

5 Decentralized Diagnosis with Two Local Decisions

5.1 Definitions

With only one local decision, the class of systems that can be diagnosed is characterized by the DISJ-CODIAG and CONJ-CODIAG architectures considered in the preceding section. To enhance the diagnosis capabilities in the context of the the general architecture in Fig.1, we can enlarge the set of local decisions. In this section, we discuss architectures with two local decisions, i.e., $|LD| = 2$. We denote these two decisions by A and B. In this case, because of earlier Assumptions **A2** and **A3**, there are only four different inputs at the global decision block to consider: (i) nothing, i.e., no site reports to the fusion block; (ii) some site says A and no site says B; (iii) some site says B and no site says A; and (iv) some site says A and another site says B. We assign either global decision "positive" or "negative" to the four inputs, resulting in $2^4 = 16$ global fusion rules as described in Table 2.

(input cases represented by local decisions received)					
Nothing	A	B	A and B	Global function	
(output for the four input cases)					
Negative	Negative	Negative	Negative	H_1^2	(not viable)
Negative	Negative	Negative	Positive	H_2^2	
Negative	Negative	Positive	Negative	H_3^2	(COND-DISJ-CODIAG)
Negative	Negative	Positive	Positive	H_4^2	(=DISJ-CODIAG)
Negative	Positive	Negative	Negative	H_5^2	(=H_3^2)
Negative	Positive	Negative	Positive	H_6^2	(=H_4^2)
Negative	Positive	Positive	Negative	H_7^2	
Negative	Positive	Positive	Positive	H_8^2	(=DISJ-CODIAG)
Positive	Negative	Negative	Negative	H_9^2	(=CONJ-CODIAG)
Positive	Negative	Negative	Positive	H_{10}^2	
Positive	Negative	Positive	Negative	H_{11}^2	(=H_{13}^2)
Positive	Negative	Positive	Positive	H_{12}^2	(=H_{14}^2)
Positive	Positive	Negative	Negative	H_{13}^2	(=CONJ-CODIAG)
Positive	Positive	Negative	Positive	H_{14}^2	(COND-CONJ-CODIAG)
Positive	Positive	Positive	Negative	H_{15}^2	
Positive	Positive	Positive	Positive	H_{16}^2	(not viable)

Table 2: Fusion Rules with Two Local Decisions

Clearly, H_1^2 and H_{16}^2 are not viable. However, the remaining functions in Table 2 are potentially viable and will correspond to some classes of languages that can be diagnosed in the context of the general Definition 4, if the global function H there is instantiated by some viable H_i^2, leading to the notion of "H_i^2-codiagnosable". Let us examine the potentially viable rules in more detail. Rules $H_4^2, H_6^2, H_8^2, H_9^2, H_{11}^2$ and H_{13}^2 are not new. H_4^2 means that the system is positive if and only if some site says B, while decision A has the same effect as keeping silent. This is equivalent to H_2^1 in Table 1, i.e., DISJ-CODIAG. H_6^2 is symmetric with H_4^2 by exchanging A and B. H_8^2 means that the system is positive if and only if somebody says A or B, which is again equivalent to DISJ-CODIAG. Similarly, H_9^2, H_{11}^2 and H_{13}^2 are equivalent with CONJ-CODIAG. However, the remaining global functions define some new language classes. H_2^2, H_7^2, H_{10}^2 and H_{15}^2 will be discussed in Section 7. Here, we focus on H_3^2 (symmetric with H_5^2) and H_{14}^2 (symmetric with H_{12}^2).

Specifically, H_3^2 means that the global decision is positive if and only if somebody says B and nobody says A. To understand this rule, let us interpret B as the conditional decision "Positive if nobody says Negative" and A as the unconditional decision "Negative". Now H_3^2 can be summarized as Cases 1-4 in Table 3. The other rule of interest, H_{14}^2, means that the global decision is positive if and only if nobody says B or somebody says A. In this case, we interpret B as the conditional decision "Negative if nobody says Positive" and A as the unconditional decision "Positive". H_{14}^2 is summarized as Cases 5-8 in Table 3.

	Case	*Local Site 1*	*Local Site 2*	*Global Decision*
	1	Nothing	Nothing	Negative
H_3^2	2	A (Negative)	Nothing	Negative
(COND-DISJ	3	B (Positive if nobody says Negative)	Nothing	Positive
CODIAG)	4	B (Positive if nobody says Negative)	A (Negative)	Negative
	5	Nothing	Nothing	Positive
H_{14}^2	6	A (Positive)	Nothing	Positive
(COND-CONJ	7	B (Negative if nobody says Positive)	Nothing	Negative
CODIAG)	8	B (Negative if nobody says Positive)	A (Positive)	Positive

Table 3: Local decisions and their fusion in the conditional architecture

As can be seen from Table 3, the conditional decisions "Positive if nobody says Negative" and "Negative if nobody says Positive" can be explained as "Positive" and "Negative" decisions, respectively, but with lower priority. The unconditional decisions "Positive" and "Negative" override conditional decisions. Namely, these conditional decisions take effect if unconditional decisions are not present. In analogy with [27], we say that these rules result in *conditional architectures*. H_3^2 corresponds to the conditional disjunctive architecture, for *conditional disjunctive codiagnosability*; H_{14}^2 corresponds to the conditional conjunctive architecture, for *conditional conjunctive codiagnosability*

Before studying the properties of H_3^2 and H_{14}^2, we introduce intuitive terminology as was done in Section 4:

$$\text{COND-DISJ-CODIAG} \Leftrightarrow H_3^2\text{-codiagnosable} \tag{5}$$

$$\text{COND-CONJ-CODIAG} \Leftrightarrow H_{14}^2\text{-codiagnosable.} \tag{6}$$

5.2 Properties

For better understanding of COND-DISJ(CONJ)-CODIAG and easier development of verification algorithms, we introduce the following equivalent language-based definitions related to those introduced in [24].

Definition 8 *Language L is said to be conditionally positive-codiagnosable w.r.t. e_d and $\mathcal{P}_1, \ldots, \mathcal{P}_n$ if the following is true:*

$(\exists k \in \mathbb{N})(\forall st \in L$ *s.t.* s *is positive and* $|t| \geq k)(\exists i \in \{1, ...n\})(\forall u \in \mathcal{E}_i(st)$ *s.t.* u *is negative*$)(\exists j \in \{1, ...n\})(\forall v \in \mathcal{E}_j^{pre}(u))$ v *is negative.*

In words, this definition means the following. For each sufficiently long positive trace st, there is a site i for which st might have the same projection as negative trace u, but for every such negative trace u that belongs to site i's estimate, there is a site j that can ensure that the system *was negative k events ago*. That is, site i can infer that if a negative trace u, instead of st, has happened, there is another site, j, that can recognize the negative prefix of u with certainty. Therefore, site i can use the "Positive if nobody says Negative" decision; site j will issue the "Negative" decision overriding site i if u is the trace that the system actually executes.

Definition 9 *Language L is said to be conditionally negative-codiagnosable w.r.t. e_d and $\mathcal{P}_1, \ldots, \mathcal{P}_n$ if the following is true:*

$(\exists k \in \mathbb{N})(\forall u \in L$ *s.t.* u *is negative*$)(\exists i \in \{1, ...n\})(\forall st \in \mathcal{E}_i(u)$ *s.t.* $|t| \geq k$ *and* s *is positive*$)(\exists j \in \{1, ...n\})(\forall v \in \mathcal{E}_j(st))$ v *is positive.*

The interpretation of this definition is as follows. For each negative trace u, there is a site i for which u might have the same projection as trace st, where s is positive and t is sufficiently long. But for every such positive trace st that belongs to site i's estimate, there is a site j that can ensure that st is positive. That is, site i can infer that if positive trace st, instead of u, has happened, there is another site, j, that can recognize positive trace st with certainty. Therefore, site i can use the "Negative if nobody says Positive" decision; site j will issue the "Positive" decision overriding site i if actually st has happened.

Before proving the equivalence of the function-based definitions of diagnosability with the language-based definitions, we introduce the following notations. The subset of sufficiently long positive traces in language L is denoted as

$$L^{P,k} = \{st | st \in L \text{ s. t. } s \text{ is positive and } |t| \geq k\} .$$

Again, we will drop superscript k for better readability. The subset of negative traces in language L is denoted as

$$L^N = \{u | u \in L, \text{ s. t. } u \text{ is negative}\} .$$

Similarly, $\mathcal{E}_i^{\mathcal{P}}(s)$ is the subset of sufficiently long positive traces in $\mathcal{E}_i(s)$, and $\mathcal{E}_i^N(s)$ is the subset of negative traces.

Theorem 10 COND-DISJ-CODIAG $\Leftrightarrow$ *Conditionally positive-codiagnosable.*

Proof: If the system is not conditionally positive-codiagnosable, then $\exists st$, s.t. s is positive and t is arbitrarily long, $\forall i, \exists u_i \in \mathcal{E}_i(st), u_i$ is negative, and $\forall j, \exists v_j w_j \in \mathcal{E}_j(u_i), v_j$ is positive, where $i, j \in \{1, \ldots, n\}$ refer to local sites. Supposing st happens, to diagnose it by H_3^2, some site, say i, has to report B. Now if u_i happens, site i would still report B and another site, say j, has to report A to override i. If finally v_jw_j happens, site j would still say A and the system would be incorrectly diagnosed as negative. Thus the system is not H_3^2-codiagnosable.

If the system is conditionally positive-codiagnosable, assuming there are n local sites, we define the local decision functions as follows[3].

$$h_i(s) = \begin{cases} A & \textbf{if } \mathcal{E}_i^{pre}(s) \text{ contains negative traces only} \\ B & \textbf{else if } \mathcal{E}_i^N(s) \cap \left(\bigcap_{j=1,\ldots,n} \mathcal{E}_j(L^P)\right) \text{ is empty} \\ \text{nothing} & \textbf{otherwise.} \end{cases} \tag{7}$$

If a sufficiently long positive trace st has occurred, as positive trace $s \in \mathcal{E}_i^{pre}(st)$, no site would report A (negative). Furthermore, according to the definition of conditional positive-codiagnosability, $\exists i, \forall u \in \mathcal{E}_i^N(st), \exists j$, such that every trace in $\mathcal{E}_j(u)$ contains a negative prefix, i.e., $u \notin \mathcal{E}_j(L^P)$. Thus $u \notin \mathcal{E}_i^N(st) \cap \mathcal{E}_j(L^P)$. As u is an arbitrary negative trace in $\mathcal{E}_i^N(st)$, we have $\mathcal{E}_i^N(st) \cap \left(\bigcap \mathcal{E}_j(L^P)\right) = \emptyset$. Site i would indeed report B (positive if nobody says negative) and the system would be diagnosed as positive.

If a negative trace u is executed and some site reports A, then the system is diagnosed as negative. If otherwise nobody reports A, we have that $\mathcal{E}_i^{pre}(u)$ contains some positive traces, $\forall i$. As a result, $u \in \mathcal{E}_i^N(u) \cap \left(\bigcap \mathcal{E}_j(L^P)\right)$. Thus no site would report B and the system would still be diagnosed as negative. ∎

Theorem 11 COND-CONJ-CODIAG $\Leftrightarrow$ *Conditionally negative-codiagnosable.*

Proof: If the system is not conditionally negative-codiagnosable, then $\exists u$, s.t. u is negative, $\forall i, \exists s_it_i \in \mathcal{E}_i(u)$, s.t. s_i is positive and $|t_i| \geq k$, and $\forall j, \exists v_j \in \mathcal{E}_j(s_it_i), v_j$ is negative, where $i, j \in \{1, \ldots, n\}$ refer to local sites. Supposing u happens, to make the correct decision under H_{12}^2, some site i has to report B. Now if s_it_i happens, site i still reports B and another site, say j, has to say A to override i. If finally v_j happens, site j would still say A and there is no way to diagnose v_j correctly. Thus the system is not COND-CONJ-CODIAG.

If the system is conditionally negative-codiagnosable, define the local decision functions as follows.

$$h_i(s) = \begin{cases} A & \textbf{if } \mathcal{E}_i(s) \text{ contains positive traces only} \\ B & \textbf{else if } \mathcal{E}_i^{\mathcal{P}}(s) \cap \left(\bigcap_{j=1,\ldots n} \mathcal{E}_j(L^N)\right) \text{ is empty} \\ \text{nothing} & \textbf{otherwise.} \end{cases} \tag{8}$$

If a sufficiently long positive trace st has occurred and some site i says A, i.e., $\mathcal{E}_i(st)$ contains positive traces only, then the system is diagnosed as positive under H_{14}^2. If otherwise nobody says A, we know that $\forall j, \mathcal{E}_j(st)$ contains some negative traces. Thus $st \in \mathcal{E}_j(L^N)$, $st \in \mathcal{E}_i^{\mathcal{P}}(st) \cap \left(\bigcap \mathcal{E}_j(L^N)\right)$. As a result, no site reports B and the diagnosis result is still positive.

If a negative trace u happens, since $\forall i, u \in \mathcal{E}_i(u)$, no site reports A. Furthermore, according to the definition of conditional negative-codiagnosability, $\exists i$, for every positive trace $st \in \mathcal{E}_i^{\mathcal{P}}(u), \exists j$, s.t. $\mathcal{E}_j(st)$ contains positive traces only. Thus $st \notin \mathcal{E}_j(L^N)$. Therefore, the intersection of $\mathcal{E}_i^{\mathcal{P}}(s) \cap \left(\bigcap \mathcal{E}_j(L^N)\right) = \emptyset$, site i would say B and the system would be diagnosed as negative. ∎

Note that both local diagnosis functions (7-8) are computable. Thus for systems that have been verified to be COND-DISJ(CONJ)-CODIAG, it is possible to diagnose positive traces online. The specific details regarding the realizations of the local diagnosis functions are beyond the scope of this paper.

[3]The notation used in Equation 7 was partially inspired by the notation used in [9].

Theorem 12 *Either* DISJ-CODIAG *or* CONJ-CODIAG *implies both* COND-DISJ-CODIAG *and* COND-CONJ-CODIAG. COND-DISJ-CODIAG *or* COND-CONJ-CODIAG *does not imply* DISJ-CODIAG *or* CONJ-CODIAG *in general.*

Proof: If the system can be diagnosed by H_2^1 (DISJ-CODIAG), clearly H_3^2 subsumes H_2^1 and the system is COND-DISJ-CODIAG. Let us remap local decisions "nothing" and A to A and B, respectively, i.e., whenever the local decision function outputs "nothing", the site reports A instead, and whenever the function outputs A, it reports B instead. Now, the global diagnosis result is the same under H_{14}^2, i.e., COND-CONJ-CODIAG.

If the system can be diagnosed by H_3^1(CONJ-CODIAG), it is COND-CONJ-CODIAG as H_{14}^2 subsumes H_3^1. In this case we remap local decisions "nothing" and A to A and B, respectively. H_3^2 produces the same global diagnosis results, i.e., COND-DISJ-CODIAG.

The reverse direction that COND-DISJ-CODIAG or COND-CONJ-CODIAG does not imply DISJ-CODIAG or CONJ-CODIAG is proved by Examples 4 and 5 below. ∎

Example 4 Consider the system G shown in Fig. 7, with two local sites, $\Sigma_{o,1} = \{a_1, a_2, c\}$, $\Sigma_{o,2} = \{b_1, b_2, c\}$ and $\Sigma_{uo} = \{e_d\}$. The system is not DISJ-CODIAG because positive trace $b_1 e_d c^n$ is indistinguishable from c^n at site 1 and indistinguishable from $b_1 a_2 c^n$ at site 2. It is not CONJ-CODIAG because negative trace c^n is indistinguishable from $b_1 e_d c^n$ at site 1 and indistinguishable from $a_1 e_d c^n$ at site 2. The system is COND-DISJ-CODIAG however, because positive trace $c^* a_1 e_d c^*$ can be diagnosed this way: site 1 says "positive if nobody says negative" once it sees a_1, and site 2 says "negative" to override site 1 if it sees b_2. Similarly, positive trace $c^* b_1 e_d c^*$ can be diagnosed. ∎

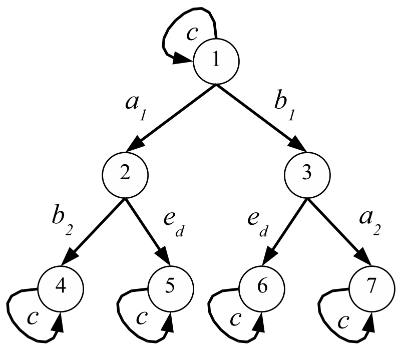


Figure 7: The system of Example 4

Example 5 In Fig. 8, there are two local sites. $\Sigma_{o,1} = \{a_1, a_2, c\}$, $\Sigma_{o,2} = \{b_1, b_2, c\}$ and $\Sigma_{uo} = \{e_d\}$. Similarly with Example 4, the system can be shown to be COND-CONJ-CODIAG but not DISJ-CODIAG or CONJ-CODIAG. ∎

Theorem 13 COND-DISJ-CODIAG *and* COND-CONJ-CODIAG *are incomparable w.r.t. the same event to be diagnosed and the same local projections.*

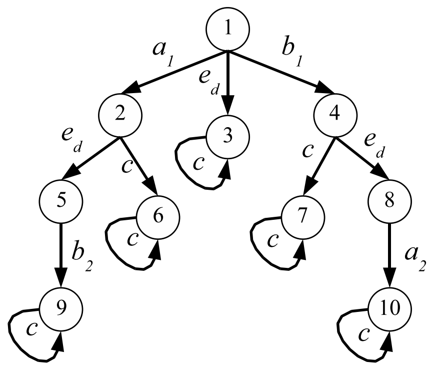


Figure 8: The system of Example 5

Proof: The system in Example 4 is COND-DISJ-CODIAG but not COND-CONJ-CODIAG. The negative trace of concern is c^n; it is indistinguishable from $b_1 e_d c^n$ at site 1 but unfortunately site 2 cannot help on this positive trace since it is indistinguishable from $b_1 a_2 c^n$ at site 2. Similarly c^n cannot be diagnosed by site 2 conditionally.

The other part is proved by Example 5 in a similar manner. ∎

Theorem 14 *Either* COND-DISJ-CODIAG *or* COND-CONJ-CODIAG *implies centralized diagnosability w.r.t. the projection corresponding to* $\Sigma_o = \Sigma_{o,1} \cup \cdots \cup \Sigma_{o,n}$. *The reverse implication is not true in general.*

The proof and the counter-example are similar with those for Theorem 6 and hence omitted.

In conclusion, the relationship among the different notions of codiagnosability introduced above is shown in Fig. 9, where a directed arc indicates "implies".

$$\begin{tikzcd}
 & \text{Diagnosable} & \\
\textsc{Cond-Disj-Codiag} \arrow[ur] & & \textsc{Cond-Conj-Codiag} \arrow[ul] \\
\textsc{Disj-Codiag} \arrow[u] \arrow[urr] & & \textsc{Conj-Codiag} \arrow[u] \arrow[ull]
\end{tikzcd}$$

Figure 9: Relationship among notions of codiagnosability

5.3 Verification

The verification of COND-DISJ-CODIAG and COND-CONJ-CODIAG can be done by extending one-level verifiers to "two-level" verifiers.

Assume system $G = (Q, \Sigma, \delta, q_0)$ is to be diagnosed by two local sites with observable event sets $\Sigma_{o,1}$

and $\Sigma_{o,2}$, respectively. We construct the *two-level verifier* $V_2 = (Q^{V_2}, (\Sigma \cup \{\epsilon\})^5, \delta^{V_2}, q_0^{V_2})$ as follows.

$$\begin{aligned}
Q^{V_2} &= \underbrace{Q \times \{N, P\}}_{s_1} \times \underbrace{Q \times \{N, P\}}_{s_{1,2}} \times \underbrace{Q \times \{N, P\}}_{s_2} \times \underbrace{Q \times \{N, P\}}_{s_{2,1}} \times \underbrace{Q \times \{N, P\}}_{s} \\
q_0^{V_2} &= (q_0, N, q_0, N, q_0, N, q_0, N, q_0, N) \ .
\end{aligned}$$

where $s_1, s_{1,2}, s_2, s_{2,1}$ and s are traces in $\mathcal{L}(G)$, and N, P indicates that the corresponding trace is negative, positive, respectively. For the sake of readability, let $q_i' = \delta(q_i, \sigma)$. The transition function δ^{V_2} is defined as described below, for all cases where the corresponding transitions are defined:

$$\begin{array}{lcl}
\text{For } \sigma \in \Sigma_{o,1}, \sigma \in \Sigma_{o,2}, & & \\
\delta^{V_2}((q_1, l_1, q_2, l_2, q_3, l_3, q_4, l_4, q_5, l_5), \sigma\sigma\sigma\sigma\sigma) & = & (q_1', l_1, q_2', l_2, q_3', l_3, q_4', l_4, q_5', l_5) \\
\text{For } \sigma \in \Sigma_{o,1}, \sigma \notin \Sigma_{o,2}, & & \\
\delta^{V_2}((q_1, l_1, q_2, l_2, q_3, l_3, q_4, l_4, q_5, l_5), \sigma\varepsilon\varepsilon\varepsilon\sigma) & = & (q_1', l_1, q_2, l_2, q_3, l_3, q_4, l_4, q_5', l_5) \\
\delta^{V_2}((q_1, l_1, q_2, l_2, q_3, l_3, q_4, l_4, q_5, l_5), \varepsilon\sigma\varepsilon\varepsilon\varepsilon) & = & (q_1, l_1, q_2', l_2, q_3, l_3, q_4, l_4, q_5, l_5) \\
\delta^{V_2}((q_1, l_1, q_2, l_2, q_3, l_3, q_4, l_4, q_5, l_5), \varepsilon\varepsilon\sigma\sigma\varepsilon) & = & (q_1, l_1, q_2, l_2, q_3', l_3, q_4', l_4, q_5, l_5) \\
\text{For } \sigma \notin \Sigma_{o,1}, \sigma \in \Sigma_{o,2}, & & \\
\delta^{V_2}((q_1, l_1, q_2, l_2, q_3, l_3, q_4, l_4, q_5, l_5), \varepsilon\varepsilon\sigma\varepsilon\sigma) & = & (q_1, l_1, q_2, l_2, q_3', l_3, q_4, l_4, q_5', l_5) \\
\delta^{V_2}((q_1, l_1, q_2, l_2, q_3, l_3, q_4, l_4, q_5, l_5), \varepsilon\varepsilon\varepsilon\sigma\varepsilon) & = & (q_1, l_1, q_2, l_2, q_3, l_3, q_4', l_4, q_5, l_5) \\
\delta^{V_2}((q_1, l_1, q_2, l_2, q_3, l_3, q_4, l_4, q_5, l_5), \sigma\sigma\varepsilon\varepsilon\varepsilon) & = & (q_1', l_1, q_2', l_2, q_3, l_3, q_4, l_4, q_5, l_5) \\
\text{For } \sigma \in \Sigma_{uo} \text{ and } \sigma \neq e_d, & & \\
\delta^{V_2}((q_1, l_1, q_2, l_2, q_3, l_3, q_4, l_4, q_5, l_5), \sigma\varepsilon\varepsilon\varepsilon\varepsilon) & = & (q_1', l_1, q_2, l_2, q_3, l_3, q_4, l_4, q_5, l_5) \\
\delta^{V_2}((q_1, l_1, q_2, l_2, q_3, l_3, q_4, l_4, q_5, l_5), \varepsilon\sigma\varepsilon\varepsilon\varepsilon) & = & (q_1, l_1, q_2', l_2, q_3, l_3, q_4, l_4, q_5, l_5) \\
\delta^{V_2}((q_1, l_1, q_2, l_2, q_3, l_3, q_4, l_4, q_5, l_5), \varepsilon\varepsilon\sigma\varepsilon\varepsilon) & = & (q_1, l_1, q_2, l_2, q_3', l_3, q_4, l_4, q_5, l_5) \\
\delta^{V_2}((q_1, l_1, q_2, l_2, q_3, l_3, q_4, l_4, q_5, l_5), \varepsilon\varepsilon\varepsilon\sigma\varepsilon) & = & (q_1, l_1, q_2, l_2, q_3, l_3, q_4', l_4, q_5, l_5) \\
\delta^{V_2}((q_1, l_1, q_2, l_2, q_3, l_3, q_4, l_4, q_5, l_5), \varepsilon\varepsilon\varepsilon\varepsilon\sigma) & = & (q_1, l_1, q_2, l_2, q_3, l_3, q_4, l_4, q_5', l_5) \\
\text{For } \sigma = e_d, & & \\
\delta^{V_2}((q_1, l_1, q_2, l_2, q_3, l_3, q_4, l_4, q_5, l_5), e_d\varepsilon\varepsilon\varepsilon\varepsilon) & = & (q_1', P, q_2, l_2, q_3, l_3, q_4, l_4, q_5, l_5) \\
\delta^{V_2}((q_1, l_1, q_2, l_2, q_3, l_3, q_4, l_4, q_5, l_5), \varepsilon e_d\varepsilon\varepsilon\varepsilon) & = & (q_1, l_1, q_2', P, q_3, l_3, q_4, l_4, q_5, l_5) \\
\delta^{V_2}((q_1, l_1, q_2, l_2, q_3, l_3, q_4, l_4, q_5, l_5), \varepsilon\varepsilon e_d\varepsilon\varepsilon) & = & (q_1, l_1, q_2, l_2, q_3', P, q_4, l_4, q_5, l_5) \\
\delta^{V_2}((q_1, l_1, q_2, l_2, q_3, l_3, q_4, l_4, q_5, l_5), \varepsilon\varepsilon\varepsilon e_d\varepsilon) & = & (q_1, l_1, q_2, l_2, q_3, l_3, q_4', P, q_5, l_5) \\
\delta^{V_2}((q_1, l_1, q_2, l_2, q_3, l_3, q_4, l_4, q_5, l_5), \varepsilon\varepsilon\varepsilon\varepsilon e_d) & = & (q_1, l_1, q_2, l_2, q_3, l_3, q_4, l_4, q_5', P) \ .
\end{array}$$

The relationships of traces s_1, $s_{1,2}$, s_2, $s_{2,1}$ and s are explained by the following proposition.

Proposition 15 *Similarly with Proposition 7, there is a path from $q_0^{V_2}$ to state $(q_1, l_1, q_2, l_2, q_3, l_3, q_4, l_4, q_5, l_5)$ if and only if:*
1. s_1, $s_{1,2}$, s_2, $s_{2,1}$ and s reach states q_1, q_2, q_3, q_4 and q_5 in G, respectively;
2. s_1 ($s_{1,2}$, s_2, $s_{2,1}$ or s) is positive iff l_1 (l_2, l_3, l_4 or l_5) $= P$;
3. $\mathcal{P}_1(s_1) = \mathcal{P}_1(s)$, $\mathcal{P}_2(s_1) = \mathcal{P}_2(s_{1,2})$, $\mathcal{P}_2(s_2) = \mathcal{P}_2(s)$ and $\mathcal{P}_1(s_2) = \mathcal{P}_1(s_{2,1})$;
where traces $s_1, s_{1,2}, s_2, s_{2,1}$ and s correspond to each component in the transitions along the path.

The proof is similar to the proof of Proposition 7 and thus omitted. According to the proposition, s_1, s_2 and s play the same role as in the one-level verifier, namely, they correspond to estimate traces by each site and to the trace the system executes, respectively. Trace $s_{i,j}$ is a trace that site j may estimate if s_i happens, namely, $s_{i,j}$ is site i's estimate of site j's estimate. The construction of V_2 guarantees that all possible trace 5-tuples satisfying the above properties are captured.

A two-level verifier state $(q_1, l_1, q_2, l_2, q_3, l_3, q_4, l_4, q_5, l_5)$ is called an $(l_1, l_2, l_3, l_4, l_5)$-state. For example, the initial state $q_0^{V_2}$ is an (N,N,N,N,N)-state. A strongly connected component is called an $(l_1, l_2, l_3, l_4, l_5)$-SCC if all states in the SCC are $(l_1, l_2, l_3, l_4, l_5)$-states.

The above construction can be extended to n local sites in a natural (but tedious) manner. We need to simulate $n^2 + 1$ traces and there are $2^{n^2+1} \times |Q|^{n^2+1}$ states at most. At each state, event σ has at most $n^2 + 1$ transitions by the transition rules, resulting in $2^{n^2+1} \times |Q|^{n^2+1} \times |\Sigma| \times (n^2 + 1)$ transitions at most. So the size of a two-level verifier is polynomial in the number of system states and exponential in the number of local sites.

Testing of COND-DISJ-CODIAG (conditional positive-codiagnosability) or COND-CONJ-CODIAG (conditional negative-codiagnosability) using the two-level verifier is based on the two following theorems which provide verification algorithms.

Theorem 16 *The language generated by system G is not* COND-DISJ-CODIAG *if and only if the two-level verifier V_2 of G has an (N,P,N,P,P)-SCC, where for each "P" in the 5-tuple, there is at least one transition whose corresponding event is not ε.*

Proof: (i) (N,P,N,P,P)-SCC with corresponding non-ε edges $\Rightarrow$ not conditionally positive-codiagnosable. Based on Proposition 15, we can induce an arbitrarily long 5-tuple trace $s_1 t_1^n, s_{1,2} t_{1,2}^n, s_2 t_2^n, s_{2,1} t_{2,1}^n, st^n$ from the (N,P,N,P,P)-SCC, where $t_{1,2}, t_{2,1}$ and t are non-ε. Now, arbitrarily long positive trace st^n is indistinguishable from negative trace $s_1 t_1^n$ at site 1 and indistinguishable from negative trace $s_2 t_2^n$ at site 2, while site 2's estimate of $s_1 t_1^n$ always contains positive prefix $s_{1,2}$ and site 1's estimate of $s_2 t_2^n$ always contains $s_{2,1}$. Thus it is not conditionally positive-codiagnosable.

(ii) Not conditionally positive-codiagnosable $\Rightarrow$ (N,P,N,P,P)-SCC with corresponding non-ε edges. If the system is not conditionally negative-codiagnosable then there is an arbitrarily long positive trace st, negative traces u_1 and u_2 that have the same projection at site 1 and 2, respectively, and arbitrarily long positive traces $v_1 w_1$ and $v_2 w_2$ such that $\mathcal{P}_2(u_1) = \mathcal{P}_2(v_1 w_1)$ and $\mathcal{P}_1(u_2) = \mathcal{P}_1(v_2 w_2)$. By Proposition 15, these five traces should form an arbitrarily long path in V_2, i.e., an (N,P,N,P,P)-SCC. The non-ε edges in the SCC follow directly from that t, w_1, and w_2 are arbitrarily long (non-ε). ∎

Theorem 17 *The language generated by system G is not* COND-CONJ-CODIAG *if and only if the two-level verifier V_2 of G has an (P,N,P,N,N)-SCC, where for each "P" in the 5-tuple, there is at least one transition whose corresponding event is not ε.*

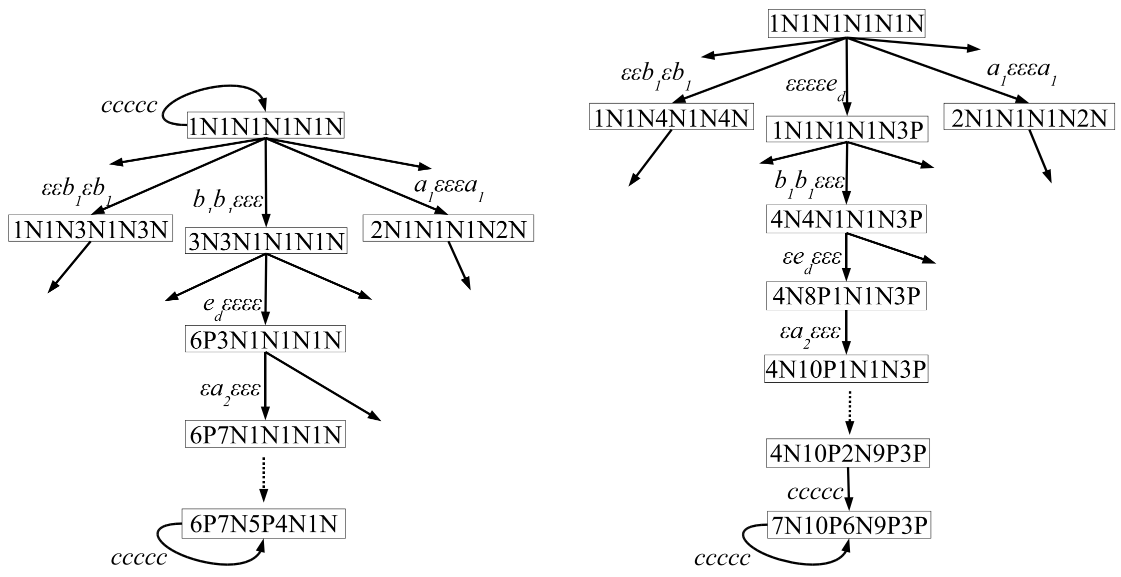
The proof is similar with the proof of Theorem 16 and thus omitted.

Figures 10(a) and 10(b) show parts of the two-level verifiers of Examples 4 and 5, respectively. There is an (P,N,P,N,N)-SCC in Fig. 10(a) and thus the system is not COND-CONJ-CODIAG. There is an (N,P,N,P,P)-SCC in Fig. 10(b) and thus the system is not COND-DISJ-CODIAG.

6 Disjunctive and Conjunctive Architectures with m Decisions

6.1 Definitions

If there are m local decisions, $A_1, \ldots, A_m$, at the global decision block, each local decision can be either present or not present, accounting for 2^m different combinations. Each combination can be mapped to either "positive" or "negative" by the global function, thus totally 2^{2^m} global decision functions are available. Similarly with the case of two local decisions, there are redundancies in the 2^{2^m} functions, as well as functions defining new decentralized architectures.



(a) Two-level Verifier of Example 4 (b) Two-level Verifier of Example 5

Figure 10: Two-level Verifiers.

Inspired by the notions of (conditional) disjunctive and conjunctive codiagnosability, we define m-disjunctive-codiagnosability and m-conjunctive-codiagnosability, or m-DISJ-CODIAG and m-CONJ-CODIAG for short, by stating their global decision functions. Let us first order local decisions as $A_m \succ A_{m-1} \succ \cdots \succ A_1 \succ$*nothing*; the global decision is determined solely by the local decision with *highest order*, as described by Table 4 (note that Table 2 lists the output for the different cases of decisions received while Table 4 lists the output for the highest decision received). For example, if some site says A_2 and no site says A_3 or any other higher-order decision, the system is diagnosed as negative under architecture m-DISJ-CODIAG (or positive under m-CONJ-CODIAG).

(input cases represented by the *highest-order* local decision received)						
Nothing	A_1	A_2	A_3	...	A_m	Definition
(output for the different input cases)						
Negative	Positive	Negative	Positive	...	...	m-DISJ-CODIAG
Positive	Negative	Positive	Negative	...	...	m-CONJ-CODIAG

Table 4: Global decision rules of m-DISJ(CONJ)-CODIAG. It is generalized from DISJ(CONJ)-CODIAG and COND-DISJ(CONJ)-CODIAG. Global decisions are alternating following the local decision order.

From Table 4, we can see that DISJ(CONJ)-CODIAG=1-DISJ(CONJ)-CODIAG and COND-DISJ(CONJ)-CODIAG=2-DISJ(CONJ)-CODIAG.

6.2 Properties

Similarly with COND-DISJ(CONJ)-CODIAG, the notions of m-DISJ(CONJ)-CODIAG, $m \geq 3$, define new classes of language that are incomparable. The theorems and examples below are generalized from the corresponding ones in Section 5.2.

Theorem 18 *Either* $(m$-1)-DISJ-CODIAG *or* $(m$-1)-CONJ-CODIAG *implies both* m-DISJ-CODIAG *and* m-CONJ-CODIAG. m-DISJ-CODIAG *or* m-CONJ-CODIAG *does not imply* $(m$-1)-DISJ-CODIAG *or* $(m$-1)-CONJ-CODIAG *in general.*

Proof: The proof is similar to the proof of Theorem 12.

The global function of m-DISJ-CODIAG subsumes the global function of $(m$-1)-DISJ-CODIAG, and $(m$-1)-CONJ-CODIAG implies m-DISJ-CODIAG by remapping the local decisions "nothing", $A_1, \ldots, A_{m-1}$ to $A_1, A_2, \ldots, A_m$, respectively. By symmetry, $(m$-1)-DISJ(CONJ)-CODIAG implies m-CONJ-CODIAG.

The fact that m-DISJ-CODIAG does not imply $(m$-1)-DISJ-CODIAG or m-CONJ-CODIAG is proved by considering the system in Example 6 below. Let us start with the diagnosis of trace $\varepsilon \in L$. We need $H(h_1(\varepsilon), h_2(\varepsilon)) =$ Negative. Under the m-DISJ(CONJ)-CODIAG architecture in Table 4, the highest-order decision determines the global decision. As the system is symmetric, w.l.o.g., assume $h_1(\varepsilon) \succeq h_2(\varepsilon)$ and $h_1(\varepsilon)$ implies negative. Now, to diagnose $e_d b_1$, since $h_1(\mathcal{P}_1(e_d b_1)) = h_1(\varepsilon)$ implies negative, we need $h_2(\mathcal{P}_2(e_d b_1)) = h_2(b_1) \succ h_1(\varepsilon)$ to override site 1's decision. Similarly we have $h_1(\varepsilon) \prec h_2(b1) \prec h_1(a_2) \prec h_2(b_3)$..., a chain of $m+1$ strict inequalities. Thus, there is no way to diagnose the language with less than m local decisions, i.e., it is not $(m$-1)-DISJ(CONJ)-CODIAG.

The fact that m-CONJ-CODIAG does not imply $(m$-1)-DISJ-CODIAG or m-CONJ-CODIAG can be proved in a similar manner by considering the system in Example 7. ∎

Example 6 Define the system as follows:[4]

$$L = \{\underbrace{...e_d a_5 b_4, a_3 b_4, e_d a_3 b_2, a_1 b_2, e_d a_1}_{m \text{ traces}}, \varepsilon, \underbrace{e_d b_1, b_1 a_2, e_d b_3 a_2, b_3 a_4, e_d b_5 a_4 ...}_{m \text{ traces}}\ .\}$$

There are two local sites with $\Sigma_{o,1} = \{a_1, a_2, ...\}$, $\Sigma_{o,2} = \{b_1, b_2, ...\}$ and $\Sigma_{uo} = \{e_d\}$.

The system is m-DISJ-CODIAG by local functions $h_1(\varepsilon) = h_2(\varepsilon) =$ nothing, $h_1(a_i) = A_i$ and $h_2(b_i) = A_i, i = 1, \ldots, m$. ∎

Example 7

$$L = \{\underbrace{...a_5 b_4, e_d a_3 b_4, a_3 b_2, e_d a_1 b_2, a_1}_{m \text{ traces}}, e_d, \underbrace{b_1, e_d b_1 a_2, b_3 a_2, e_d b_3 a_4, b_5 a_4 ...}_{m \text{ traces}}\ .\}$$

There are two local sites with $\Sigma_{o,1} = \{a_1, a_2, ...\}$, $\Sigma_{o,2} = \{b_1, b_2, ...\}$ and $\Sigma_{uo} = \{e_d\}$. The system is m-CONJ-CODIAG by local functions $h_1(\varepsilon) = h_2(\varepsilon) =$ nothing, $h_1(a_i) = A_i$ and $h_2(b_i) = A_i, i = 1, \ldots, m$. ∎

Theorem 19 m-DISJ-CODIAG *and* m-CONJ-CODIAG *are incomparable w.r.t. the same event to be diagnosed and the same local projections.*

Proof: The system in Example 6 is m-DISJ-CODIAG. In the analysis of Theorem 18, we have $h_1(\varepsilon) \prec h_2(b1) \prec h_1(a_2) \prec h_2(b_3) \cdots$, resulting in $m+1$ strict inequalities. If m-CONJ-CODIAG holds, according to Table 4, these $m+1$ inequalities have to be mapped as $h_1(\varepsilon) =$ nothing, $h_2(b_1) = A_1$, $h_1(a_2) = A_2$, and so on. However, with the assumption that $h_2(\varepsilon) \preceq h_1(\varepsilon)$, the global decision on trace ε is determined by $h_1(\varepsilon) =$ nothing, which results in global decision "Positive" under m-CONJ-CODIAG. Thus, the system is not m-CONJ-CODIAG.

The other part is proved by Example 7 in a similar manner. ∎

[4]In Examples 6, 7 and 9, L is not live. One can add cycles c^* to each trace in L, i.e., $L' = Lc^*$. We omit c^* here for better readability. The analysis on L applies to L' in the same way.

Recall the notion of joint diagnosability in Definition 5.

Theorem 20 m-DISJ(CONJ)-CODIAG *implies joint diagnosability and joint diagnosability implies centralized diagnosability, where* $\Sigma_o = \Sigma_{o,1} \cup \cdots \cup \Sigma_{o,n}$. *The reverse implications are not true in general.*

Proof: If the system is not jointly diagnosable, there is a sufficiently long positive trace indistinguishable from a negative trace at every site. It is therefore not possible to diagnose the system with any global function. On the other hand, Example 8 tells us that joint diagnosability does not imply m-DISJ(CONJ)-CODIAG.

The fact that joint diagnosability implies centralized diagnosability follows from their definitions. The reverse direction is proved by Example 3. ∎

In conclusion, we obtain the complete relationship chart presented in Fig. 11.

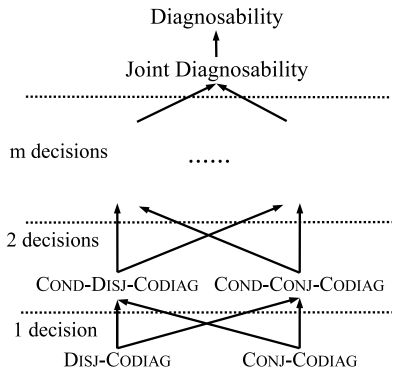


Figure 11: Relationship among notions of codiagnosability

7 Other Global Functions with Two Local Decisions

The notions of COND-DISJ-CODIAG and COND-CONJ-CODIAG studied previously correspond to global functions H_3^2 and H_{14}^2 in Table 2. There are four other functions in the table that have not been discussed, namely, H_2^2, H_7^2, H_{10}^2 and H_{15}^2. Interestingly, new classes of diagnosable languages are defined by these functions. We only discuss H_2^2 in this paper, as its rule appears to be the simplest.

Global decision rule H_2^2 says that the diagnosis result is "positive" if both decisions A and B are present. There is no priority among these two decisions. A system is called H_2^2*-codiagnosable* if it can be diagnosed by H_2^2. The system in Example 8 is an H_2^2-codiagnosable system.

Example 8 System G is shown in Fig. 12; take $\Sigma_{o,1} = \{a_1, a_2, c\}$ and $\Sigma_{o,2} = \{b_1, b_2, c\}$. This system is H_2^2-codiagnosable by local functions $h_1(a_1) = h_2(b_1) = A$ and $h_1(a_2) = h_2(b_2) = B$. ∎

It is unknown at this point whether there is a language-based definition that is equivalent to the notion of H_2^2-codiagnosability. The relationship between H_2^2-codiagnosability and other notions of codiagnosability introduced elsewhere in this paper is captured in the following theorem. The verification of H_2^2-codiagnosability is an open problem at this point. Our conjecture is that this new notion of diagnosability is undecidable.

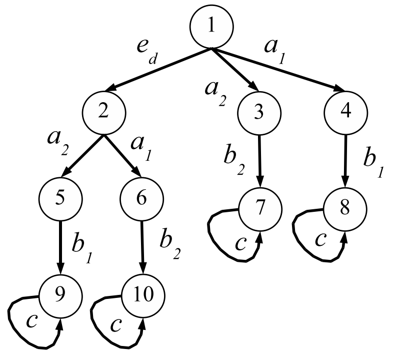


Figure 12: An H_2^2-codiagnosable system

Theorem 21 *H_2^2-codiagnosability is incomparable with m-*DISJ-CODIAG *and m-*CONJ-CODIAG*, $m \geq 2$, w.r.t. the same event to be diagnosed and the same local projections.*

Proof: If the system in Example 8 is diagnosed under the global rules for m-DISJ(CONJ)-CODIAG, an analysis similar to the proof of Theorem 18 gives us the order relation $h_2(b_1) \prec h_1(a_2) \prec h_2(b_2) \prec h_1(a_1) \prec h_2(b_1)$, which is impossible. Thus the system is not m-DISJ(CONJ)-CODIAG.

The other direction is proved by Example 9. If the system is diagnosable under H_2^2, for positive trace e_dab, decisions A and B must both be reported by local sites. As the two decisions are symmetric, w.l.o.g., let $h_1(e_dab) = h_1(a) = A$ and $h_2(\mathcal{P}_2(e_dab)) = h_2(b) = B$. To diagnose trace e_da correctly, as site 1 still says A, site 2 has to say B, i.e., $h_2(\mathcal{P}_2(e_da)) = h_2(\varepsilon) = B$. Similarly $h_1(\varepsilon) = A$. Then trace ε cannot be diagnosed correctly. ∎

Example 9 System $L = \{\varepsilon, e_da, e_db, e_dab\}$, where $\Sigma_{o,1} = \{a, c\}$ and $\Sigma_{o,2} = \{b, c\}$. This system is DISJ-CODIAG by $h_1(\varepsilon) = h_2(\varepsilon) =$ "nothing" and $h_1(a) = h_2(b) = A$. Therefore it is m-DISJ(CONJ)-CODIAG, $m \geq 2$, by Theorem 18. ∎

8 Conclusion

This paper has introduced and analyzed a general hierarchical framework for decentralized diagnosis of DES. The framework is parameterized by the number of different decisions a local site can issue and by the global fusion rule for these local decisions, leading to the sets of functions listed in Tables 1 and 2 and their generalized form in Table 4. Several equivalence results between many notions of decentralized diagnosability and their corresponding architectures in the general framework were established. The cases where local sites issue one or two local diagnosis decisions were studied in detail. It was discovered that in several cases these local decisions could be interpreted as conditional decisions of the type "Positive if nobody says Negative" and "Negative if nobody says Positive". These conditional interpretations were key to identifying equivalent language-based notions of decentralized diagnosability, which in turn enabled the construction of polynomial tests for their verification. Moreover, these conditional interpretations extend to the case of more than two local decisions, albeit the details become more intricate. Although not discussed in this paper, the conditional interpretations are also key to the synthesis of diagnosers for online diagnosis under the conditional architectures.

Another contribution of this work is the discovery of a new decentralized architecture with two local decisions that is not comparable to any other existing notion of decentralized diagnosability. This notion was called H_2^2-codiagnosability in Section 7. The verification of this new property is an open problem.

Overall, the use of decentralized diagnosis architectures of the type studied in this paper allows for diagnosing larger classes of systems that can be diagnosed under prior work, such as the decentralized architecture corresponding to Protocol 3 in [5]. The hierarchical framework that was introduced connects Protocol 3 in [5], whose verification is polynomial, and joint diagnosability in [20], whose verification is undecidable. Moreover, it identifies several decidable classes of diagnosable languages in between these two extreme points. Identifying the boundary between decidability and undecidability in this space would be interesting future work.

9 Acknowledgements

It is a pleasure to acknowledge insightful discussions with Stavros Tripakis, Demosthenis Teneketzis, and William Rounds. The reviewers are also acknowledged for their useful comments and suggestions.

References

[1] P. Baroni, G. Lamperti, P. Pogliano, and M. Zanella. Diagnosis of large active systems. *Artificial Intelligence*, 110:135–183, 1999.

[2] 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, September 2004.

[3] R.K. Boel and J.H. 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, October 2002.

[4] C. G. Cassandras and S. Lafortune. *Introduction to Dsicrete Event Systems.* Kluwer Academic Publishers, 1999.

[5] R. Debouk, S. Lafortune, and D. Teneketzis. Coordinated decentralized protocols for failure diagnosis of discrete-event systems. *Discrete Event Dynamic Systems: Theory and Applications*, 10(1-2):33–86, January 2000.

[6] E. Fabre, A. Benveniste, S. Haar, and C. Jard. Distributed monitoring of concurrent and asynchronous systems. *Discrete Event Dynamic Systems: Theory and Applications*, 15(1):33–84, March 2005.

[7] 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*, pages 2252–2257, December 2000.

[8] 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.

[9] R. Kumar and S. Takai. Inference-based ambiguity management in decentralized decision-making: Decentralized diagnosis of discrete event systems. preprint, 2006.

[10] S. Lafortune, D. Teneketzis, M. Sampath, R. Sengupta, and K. Sinnamohideen. Failure diagnosis of dynamic systems: An approach based on discrete event systems. In *Proc. 2001 American Control Conf.*, pages 2058–2071, June 2001.

[11] G. Lamperti and M. Zanella. *Diagnosis of active systems: principles and techniques.* Kluwer Academic Publishers, 2003.

[12] Y. Pencolé and M.-O. Cordier. A formal framework for the decentralised diagnosis of large scale discrete event systems and its application to telecommunication networks. *Artificial Intelligence,* 164:121–170, 2005.

[13] 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, September 2004.

[14] K. Rohloff, T.-S. Yoo, and S. Lafortune. Deciding coobservability is PSPACE-complete. *IEEE Trans. Automat. Contr.,* 48(11):1995–1999, November 2003.

[15] L. Rozé and M.-O. Cordier. Diagnosing discrete-event systems: extending the "diagnoser approach" to deal with telecommunication networks. *Discrete Event Dynamic Systems: Theory and Applications*, 12(1):43–81, 2002.

[16] K. Rudie and W. M. Wonham. Think globally, act locally: decentralized supervisory control. *IEEE Trans. Automat. Contr.,* 37(11):1692–1708, November 1992.

[17] M. Sampath, R. Sengupta, K. Sinnamohideen S. Lafortune, and D. Teneketzis. Diagnosability of discrete event systems. *IEEE Trans. Automat. Contr.*, 40(9):1555–1575, September 1995.

[18] M. Sampath, R. Sengupta, K. Sinnamohideen S. Lafortune, and D. Teneketzis. Failure diagnosis using discrete event models. *IEEE Trans. Contr. Syst. Technol.*, 4(2):105–124, March 1996.

[19] R. Sengupta. Diagnosis and communication in distributed systems. In *Proc. of the 1998 International Workshop on Discrete Event Systems - WODES'98*, Cagliari, Italy, 1998.

[20] R. Sengupta and S. Tripakis. Decentralized diagnosability of regular languages is undecidable. In *Proc. 40th IEEE Conf. on Decision and Control*, pages 423–428, December 2002.

[21] R. Su and W.M. Wonham. Distributed diagnosis under global consistency. In *Proc. 42nd IEEE Conf. on Decision and Control*, December 2004.

[22] R. Su, W.M. Wonham, J. Kurien, and X. Koutsoukos. Distributed diagnosis for qualitative systems. In *Proc. of the 2002 International Workshop on Discrete Event Systems - WODES'02*, pages 169–174, Zaragoza, Spain, October 2002.

[23] Y. Wang, T.-S. Yoo, and S. Lafortune. New results on decentralized diagnosis of discrete-event systems. In *Proceedings of 2004 Annual Allerton Conference,* 2004.

[24] Y. Wang, T.-S. Yoo, and S. Lafortune. Decentralized diagnosis of discrete event systems using conditional and unconditional decisions. In *Proceedings of the 44th IEEE Conference on Decision and Control*, 2005.

[25] T.-S. Yoo and S. Lafortune. A general architecture for decentralized supervisory control of discrete-event systems. *Discrete Event Dynamic Systems: Theory and Applications*, 12(3):335–377, July 2002.

[26] T.-S. Yoo and S. Lafortune. Polynomial-time verification of diagnosability of partially-observed discrete-event systems. *IEEE Trans. Automat. Contr.*, 47(9):1491–1495, September 2002.

[27] T.-S. Yoo and S. Lafortune. Decentralized supervisory control with conditional decisions: supervisor existence. *IEEE Trans. Automat. Contr.*, 49(11):1886–1904, November 2004.