Undecidable Case and Decidable Case of Joint Diagnosability in Distributed Discrete Event Systems

Lina Ye and Philippe Dague
INRIA, Grenoble-Rhône-Alpes, lina.ye@inria.fr
LRI, Univ. Paris-Sud, philippe.dague@lri.fr

***Abstract*—Diagnosability is an important property that determines at design stage how accurate any diagnosis algorithm can be on a partially observable system. Most existing approaches assumed that each observable event in the system is globally observed. Considering the cases where there is no global information, one of our recent work proposed a new framework to check diagnosability in a system where each component can only observe its own observable events to keep the internal structure private in terms of observations. However, we assumed that the local paths in each component can be exhaustively enumerated, which is not suitable in a general case where there are embedded cycles. In this paper, we get some new results about diagnosability in such a system in a general case, i.e., what we call joint diagnosability in a self-observed distributed system. First, we prove the undecidability of joint diagnosability with unobservable communication events by reducing the Post's Correspondence Problem to joint diagnosability problem. We also propose an algorithm to check a sufficient but not necessary condition of joint diagnosability, which is then adapted when the assumption of all communication events being unobservable is relaxed, i.e., communication events could be either observable or unobservable. Then, we discuss about the decidable case where communication events are all observable and develop a new efficient algorithm to test it. Finally, we also provide an important property of joint diagnosability after analyzing its relationship with classical diagnosability.**

Keywords*—*joint diagnosability, self-observed systems, finite state machine, Post's Correspondence Problem, undecidability.

I. Introduction

OVER the latest decades, with the advancement of technologies, systems are becoming more and more complex since more performance requirements are imposed on them, which causes more errors that they are subject to. However, it is not realistic to manually detect faults for complex systems. Automated diagnosis mechanisms are therefore required to monitor large distributed applications such as transportation systems, communication networks, manufacturing systems, web services, spatial systems and power systems. Thus, it is crucial for a complex system to perceive that it is not operating correctly and then without human intervention, to detect and isolate original faults, which will be restored by repair plans to normality.

Generally speaking, diagnosis reasoning is to detect possible faults that can explain the observations. The possibility to achieve such a diagnosis reasoning depends on the diagnosability of the system. Diagnosability is an important property that determines at design stage how accurate any diagnosis algorithm can be on a partially observable system, which has significant economic impact on the improvement of performance and reliability of complex systems. The diagnosability analysis problem has received considerable attention in the literature, both in centralized and distributed ways.

In this paper, we study diagnosability problem for distributed systems where a component cannot observe the observable events in other components, which are called self-observed systems. We make several contributions by extending the conference article [1]. The first one is to extend diagnosability of globally observed systems to what we call joint diagnosability of self-observed systems before proving its undecidability in the case where communication events are unobservable. This is done by reducing the Post's Correspondence Problem to joint diagnosability problem, which is inspired from the undecidability result of joint observability [2]. The second one is to propose an algorithm for testing a sufficient condition of joint diagnosability with unobservable communication events. To do so, we first obtain pairs of local trajectories in the faulty component, such that for each pair, only one trajectory contains the fault but both trajectories have the same enough local observations. Their global consistency is then checked through two phases. We prove that it is a sufficient condition and point out why it is not necessary. Afterwards, we adapt this algorithm when the assumption of communication events being unobservable is relaxed, i.e., communication events could be observable or unobservable. The third one is to discuss about the decidable case where communication events are observable and to develop an efficient algorithm to test it. Finally, we provide an important property of joint diagnosability after analyzing its relationship with classical diagnosability.

This article extends our recent works [1] and [3] in the following aspects.

- We add detailed proofs for Theorem 1 and Lemma 2, which make our approach more convincing in terms of correctness.
- We define a new structure called local twin checker for a normal component to make the presentation more clear and the algorithm more efficient. The reason is that a local twin checker contains no fault information since it is constructed for a normal component without fault. Then we provide new complexity analysis for the proposed sufficient algorithm to test joint diagnosability when communication events are unobservable.
- We generalize the sufficient algorithm to test joint di-

agnosability when the assumption is relaxed such that communication events could be either observable or unobservable, which is presented in Section V-D.
- In Section VI, we provide more details about why joint diagnosability is decidable when communication events are all observable and then develop a new algorithm to test it.
- We provide a new important property of joint diagnosability after analyzing its relationship with classical diagnosability in Section VII.

The paper is organized as follows. In the next section, we talk about related works in the literature. In Section III, we model self-observed distributed systems and recall classical diagnosability and joint diagnosability. Section IV presents undecidability analysis for joint diagnosability when communication events are unobservable. An algorithm to test a sufficient condition is proposed in Section V with complexity analysis, which is then adapted when communication events could be either unobservable or observable. In Section VI, we discuss about decidable case for joint diagnosability, i.e., when communication events are observable, before giving an algorithm to test it. Finally, we compare joint diagnosability and classical diagnosability to find out their relationship in Section VII before giving the conclusion in Section VIII.

II. Related Work

In the literature, three types of systems are under investigation for diagnosis problem: continuous systems, discrete event systems and hybrid systems ([4], [5], [6], [7], [8], [9], [10], [11], [12]). In this paper, the dynamic systems studied are discrete event systems (abbreviated DES hereafter). Given a system, if its state space is naturally described by a discrete set and if state transitions only occur at discrete points in time, we associate these transitions with events and this system is called a DES. The reason why we choose DES for investigation is that most of the man-made systems are DES and that continuous systems can be abstracted to be DES. Nowadays, lots of works have been studied on control of DES, including diagnosis algorithm, diagnosability analysis, predictability analysis, etc. ([13], [14], [15], [16], [17], [18], [19]).

Some existing works analyzed diagnosability problem in a centralized way ([20], [21] and [22]), i.e., the knowledge of the monolithic model of a given system is hypothesized, which is the very powerful information for diagnosability analysis and leads to an unrealistic combinatorial explosion of the search space. This is why very recently distributed approaches for diagnosability began to be investigated ([23], [24] and [25]), relying on local objects. More precisely, in these distributed approaches, original diagnosability information can be obtained from the components where faults may occur and then the global decision is calculated by checking its global consistency. However, all these approaches assumed that each observable event in the system can be observed by all components, i.e., globally observed. However, in reality, there are some cases where it is not possible to obtain global information. For example, networked control systems are characterized by the fact that multiple distributed components possess their own part of available information instead of a global knowledge. Then one of our recent work [26] proposed a new framework to check diagnosability in a system where each component can only observe its own observable events to keep the internal structure private in terms of observations. However, we assumed that the local paths in each component can be exhaustively enumerated, which is not suitable in a general case where there are embedded cycles. In this paper, we generalize this work to get some new results about the diagnosability of what we call self-observed distributed systems, i.e., systems where locally observable events can only be observed by their own component.

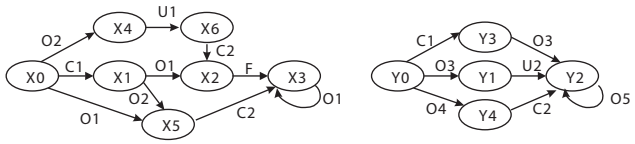

Fig. 1: A system with two components G_1 (left) and G_2 (right).

III. Preliminaries

In this section, we show how to model self-observed distributed DESs and then recall classical diagnosability and joint diagnosability.

A. System Model

We consider a self-observed distributed DES composed of a set of components $\{G_1, G_2,..., G_n\}$ that communicate with each other by communication events. Each component can only observe its own observable events and thus keeps its internal structure private in terms of observations. Such a system is modeled by a set of finite state machines (FSM), each one representing the local model of one component.

Definition 1: (Local Model). The local model of a component G_i is a FSM, denoted by $G_i = (Q_i, \Sigma_i, \delta_i, q_i^0)$, where

- Q_i is the set of states;
- Σ_i is the set of events;
- $\delta_i \subseteq Q_i \times \Sigma_i \times Q_i$ is the set of transitions;
- q_i^0 is the initial state.

The set of events Σ_i is partitioned into four disjoint subsets: Σ_{i_o}, the set of locally observable events that can be observed only by their own component G_i; Σ_{i_u}, the set of unobservable normal events; Σ_{i_f}, the set of unobservable fault events; and Σ_{i_c}, the set of communication events shared by at least one other component, which are the only shared events between components. Figure 1 depicts a self-observed distributed system with two components: G_1 (left) and G_2 (right), where Oi denotes a locally observable event, F denotes an unobservable fault event, Ui denotes an unobservable normal event and Ci denotes a communication event.

We denote the synchronized FSM of components $G_1, ..., G_n$ by $\|(G_1, ..., G_n)$, where the synchronized events are the shared events between components and each one of them always occurs simultaneously in all components that define it. The

state space of the synchronized FSM is the Cartesian product of the state spaces of components. The monolithic model (also called global model in the following) of the entire system is implicitly defined as the synchronized FSM of all components based on their shared events, i.e., communication events. However, the global model will not be calculated in this paper since in a self-observed distributed system, the global occurrence order of observable events is not accessible. In the following, we call the synchronization of a subset of components of G, i.e., $\|(G_{s_1}, ..., G_{s_m})$, as subsystem of G, where $\{s_1, ...s_m\} \subseteq \{1, ...n\}$. Note that one component or the entire system can also be considered as a subsystem.

Given a system model $G = (Q, \Sigma, \delta, q^0)$, the set of words produced by the FSM G is a prefix-closed language $L(G)$ that describes the normal and faulty behaviors of the system. Formally, $L(G) = \{s \in \Sigma^* | \exists q \in Q, (q^0, s, q) \in \delta\}$, where the transition δ has been extended from events to words. In the following, we call a word of $L(G)$ a **trajectory** in the system G and a sequence $q_0\sigma_0q_1\sigma_1...$ a **path** in G, where $\sigma_0\sigma_1...$ is a trajectory and for all i, we have $(q_i, \sigma_i, q_{i+1}) \in \delta$. Given $s \in L(G)$, we denote the post-language of $L(G)$ after s by $L(G)/s$ and denote the projection of s to observable events of G (resp. G_i) by $P(s)$ (resp. $P_i(s)$). For example, if $s = O1.U2.O3^*$, then we have $P(s) = O1.O3^*$, where Oi denotes an observable event. These notations are also appropriate for local components. We adopt the assumption described in [23], i.e., the projection of the global language on each local model is observable live, in particular there is no unobservable cycle in any component. For the sake of simplicity, our approach is illustrated by dealing with only one fault, which can be extended to the case with multiple faults.

B. Diagnosability and Joint Diagnosability

We now recall classical diagnosability for centralized DESs. Informally speaking, the existence of two indistinguishable behaviors, i.e., holding the same enough observations with exactly one of them containing the given fault F, violates diagnosability property. The diagnosability approaches consist in checking the existence of such indistinguishable behaviors. In other words, a fault F is diagnosable in a system G iff its occurrence is determinable when enough events are observed from G after the occurrence of F, which is formally defined as follows [20], where s^F denotes a trajectory ending with F.

Definition 2: (Diagnosability). A fault F is diagnosable in a system G iff

$$\forall s^F \in L(G), \exists k \in N, \forall t \in L(G)/s^F, |t| \geq k \Rightarrow (\forall p \in L(G),\ P(p) = P(s^F.t) \Rightarrow F \in p).$$

The above definition states that if F is diagnosable, then for each trajectory s^F in G, for each t that is an extension of s^F with sufficient events, every trajectory p in G that is observation equivalent to $s^F.t$ should contain F. We call a pair of trajectories p and $p\prime$ satisfying the following conditions as a **critical pair**:

- p contains F and $p\prime$ does not;
- p has arbitrarily long events after the occurrence of F;
- $P(p) = P(p\prime)$.

It has been proved that the existence of critical pairs violates Definition 2 and thus witnesses non-diagnosability [21]. For distributed systems, the analysis of the above classical diagnosability requires global observations, which is not suitable for self-observed distributed systems where observable events can only be observed by their own component. Now we give the definition of joint diagnosability that only requires local observations without considering their global occurrence order [26].

Definition 3: (Joint diagnosability). A fault F is jointly diagnosable in a self-observed distributed system G composed of components $\{G_1, ...G_n\}$, iff

$$\forall s^F \in L(G), \exists k \in N, \forall t \in L(G)/s^F, (\forall i \in \{1, ..., n\}, |P_i(t)| \geq k) \Rightarrow (\forall p \in L(G) (\forall i \in \{1, ..., n\}, P_i(p) = P_i(s^F.t)) \Rightarrow F \in p).$$

Joint diagnosability of a fault F means that for each trajectory s^F in G, for each t that is an extension of s^F with enough locally observable events in all components, every trajectory p in G that is equivalent to $s^F.t$ for local observations in each component should contain in it F. In a self-observed system, we call a pair of trajectories p and $p\prime$ satisfying the following conditions an (global) **indeterminate pair**:

- p contains F and $p\prime$ does not;
- p has arbitrarily long local observations in all components after the occurrence of F;
- $\forall i \in \{1, ..., n\}$, $P_i(p) = P_i(p\prime)$.

Here arbitrarily long local observations can be considered as infinite local observations. Now we have the following theorem [26].

Theorem 1: Given a self-observed distributed system G, a fault F is jointly diagnosable in G iff there is no (global) indeterminate pair in G.

Proof:

($\Rightarrow$) Suppose that F is jointly diagnosable in a system G and that there exists an indeterminate pair p and $p\prime$ with only p containing the fault F. Now let s^F denote the subpart of p that is ending with F and let t denote the rest part of p, i.e., $p = s^F.t$. Since p and $p\prime$ are an indeterminate pair, from its definition, we have that p has arbitrarily long local observations for each component G_i after the occurrence of F, and that for each component G_i, p and $p\prime$ have the same local observations, i.e., $P_i(p) = P_i(p\prime)$. However, $p\prime$ does not contain F. This contradicts the definition of joint diagnosability, where any trajectory with the same enough local observations in each component as $s^F.t$ should also contain F. So F is not jointly diagnosable in G, which contradicts the assumption.

($\Leftarrow$) Now suppose that there is no indeterminate pair in G and F is not jointly diagnosable in G. From the non joint diagnosability of F and Definition 3, we deduce that $\exists s^F \in L(G)$, such that there exists at least one extension t with enough local observations (represented by infinite paths with cycles) in all components, for which there must exist at least one other trajectory p containing the same local observations as $s^F.t$ for each component but without fault. Since the enough local observations of $s^F.t$ and p come from infinite paths with cycles, $s^F.t$ and p can be prolonged arbitrarily long. It follows that $s^F.t$ and p are an indeterminate pair since they satisfy three conditions of the definition of an indeterminate pair,

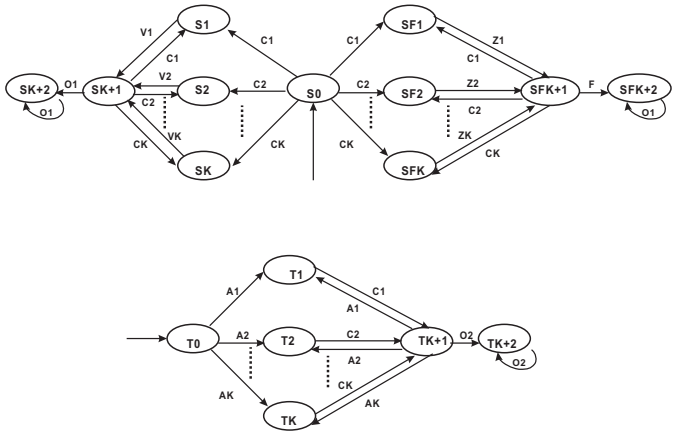

Fig. 2: A system with two components G_1 (top) and G_2 (bottom) for proving undecidability of joint diagnosability.

which contradicts the assumption that there is no indeterminate pair. ■

Another important notation that will be used in joint diagnosability analysis is reconstructibility [27], which is rephrased as follows.

Definition 4: (Reconstructibility). Given a system G that is composed of several subsystems, i.e., $G = \|(G_{s_1}, ..., G_{s_m})$, let ρ be a trajectory in G, the set of trajectories obtained by projecting ρ on these subsystems, i.e., denoted by $\rho_1 = P'_{G_{s_1}}(\rho), ..., \rho_m = P'_{G_{s_m}}(\rho)$, where $P'_{G_{s_i}}(\rho)$ is the projection of ρ on the subsystem G_{s_i}, is said to be reconstructible with respect to G.

Lemma 1: In a system G, given two subsystems G_S and G'_S, if $\Sigma_{S_c} \cap \Sigma_{S'_c} = \emptyset$, then $\forall(s, s\prime), s \in L(G_S), s\prime \in L(G'_S)$, s and $s\prime$ are reconstructible.

Lemma 1 means that if there is no common communication event between two subsystems, then any trajectory in one subsystem and any one in the other one are reconstructible.

IV. Undecidable Case

To discuss about joint diagnosability, we consider two cases: communication events being unobservable and observable. In this section, we first consider the case where all communication events are unobservable.

Theorem 1 implies that checking joint diagnosability boils down to check the existence of indeterminate pairs that witnesses non joint diagnosability. Before discussing about how to test this, it is very important to check whether the problem is decidable or not. Inspired from [2], where undecidability of joint observability is proved by reducing the Post's Correspondence Problem (PCP) to an observation problem, we now prove that joint diagnosability is not decidable with communication events being unobservable.

Theorem 2: Given a self-observed distributed system where communication events are unobservable, checking joint diagnosability of a given fault is undecidable.

Proof:

1) PCP: given a finite alphabet Σ, two sets of words $v_1, v_2, ..., v_k$ and $z_1, z_2, ..., z_k$ over Σ, then a solution to PCP is a sequence of indices $(i_m)_{1\leq m\leq n}$ with $n \geq 1$ and $1 \leq i_m \leq k$ for all m such that $v_{i_1} v_{i_2} ... v_{i_n} = z_{i_1} z_{i_2} ... z_{i_n}$.

2) Now consider the example depicted in Figure 2, where the system is composed of two components G_1 and G_2. In G_1, each one of $Vi, i \in \{1, ..., k\}$, or each one of $Zi, i \in \{1, ..., k\}$, denotes a sequence of observable events all different from the observable event $O1$, and $C1, ..., Ck$ are unobservable communication events, then F denotes a fault event. In G_2, each one of $Ai, i \in \{1, ..., k\}$, denotes an observable event different from the observable event $O2$, and $C1, ..., Ck$ are unobservable communication events. Then the observations in G_1 can be described as $Vi_1 Vi_2 ... Vi_n O1^*$ without fault or $Zi_1 Zi_2 ... Zi_n O1^*$ with fault, where $\forall i_j, j \in \{1, ..., n\}, i_j \in \{1, ..., k\}$. In G_2, the observations are $Ai_1 Ai_2 ... Ai_n O2^*$.

3) With the observation of $O1$, the local observations are $wO1^+$ for G_1 and $Ai_1 Ai_2 ... Ai_n O2^*$ for G_2, where $w = Vi_1 Vi_2 ... Vi_n$ when there is no fault or $w = Zi_1 Zi_2 ... Zi_n$ when there is a fault. Clearly, if PCP has a solution, i.e., $\exists(i_m)_{1\leq m\leq n}$ such that $Vi_1 Vi_2 ... Vi_n = Zi_1 Zi_2 ... Zi_n$, we have two trajectories p and $p\prime$ such that the observations of p in G_1 are $Vi_1 Vi_2 ... Vi_n O1^+$, which is a trajectory without fault, while the observations of $p\prime$ in G_1 are $Zi_1 Zi_2 ... Zi_n O1^+$, which is a trajectory with a fault. And both p and $p\prime$ have the same observations for G_2, i.e., $Ai_1 Ai_2 ... Ai_n O2^*$. Thus we get that p and $p\prime$ have the same observations for both G_1 and G_2, i.e., $Vi_1 Vi_2 ... Vi_n O1^+$=$Zi_1 Zi_2 ... Zi_n O1^+$ for G_1 and $Ai_1 Ai_2 ... Ai_n O2^*$ for G_2, then the fault is not jointly diagnosable.

4) On the other hand, if the fault is not jointly diagnosable, then we obtain at least one indeterminate pair, denoted by p and $p\prime$ such that the projection of p on G_1 is $Ci_1 Vi_1 Ci_2 Vi_2 ... Ci_n Vi_n O1^*$, on G_2 is $Ai_1 Ci_1 Ai_2 Ci_2 ... Ai_n Ci_n O2^*$ and that of $p\prime$ on G_1 is $Cj_1 Zj_1 Cj_2 Zj_2 ... Cj_m Zj_m FO1^*$ and on G_2 is $Aj_1 Cj_1 Aj_2 Cj_2 ... Aj_m Cj_m O2^*$. From the fact that p and $p\prime$ have the same observations for G_2, we get $Ai_1 Ai_2 ... Ai_n O2^* = Aj_1 Aj_2 ... Aj_m O2^*$ and thus we have $m = n$ and $i_1 = j_1, ..., i_n = j_n$. And then from the same observations of p and $p\prime$ on G_1, we get $Vi_1 Vi_2 ... Vi_n O1^* = Zi_1 Zi_2 ... Zi_n O1^*$, i.e., $Vi_1 Vi_2 ... Vi_n = Zi_1 Zi_2 ... Zi_n$, which means that there is a solution for PCP.

5) From above, we get that checking the existence of a solution for PCP is equivalent to checking the existence of an indeterminate pair. Since PCP is an undecidable problem, checking joint diagnosability is also undecidable. ■

There are two major differences between joint diagnosability in our framework and joint observability in [2]. One is that the former assumes that local observers are attached to local components that are synchronized by common communication events to get a global model while the latter separates arbitrarily the observable events in the global model into several sets. The other one is that joint diagnosability consists in separating infinite trajectories while joint observability consists in separating finite ones. Thus, if any communication event is assumed to be unobservable, joint diagnosability checking boils down to infinite PCP. But this one has been proved to be undecidable [28].

V. Algorithm to Test a Sufficient Condition

We have proved that joint diagnosability in self-observed distributed systems with unobservable communication events is undecidable. We can nevertheless propose an algorithm to test a sufficient condition, which is still quite useful in some circumstances. The first step is to construct the local diagnoser for the faulty component, which allows one to get fault information after any local trajectory. Then we show how to build the corresponding local twin plant based on the local diagnoser to obtain the original information about indeterminate pairs (also called local indeterminate pairs in the following). The next step is to check the global consistency, i.e., whether the local indeterminate pairs can be extended into (global) indeterminate pairs, whose existence verifies non joint diagnosability. We give an algorithm to test a sufficient but not necessary condition for global consistency.

A. Original Diagnosability Information

From Theorem 1, we know that joint diagnosability verification consists in checking the existence of indeterminate pairs in the system. In the distributed framework, we use the structure called local twin plant, initially defined in [21], to analyze joint diagnosability. In particular, the considered fault is assumed to only occur in one component, denoted by G_F. Then the local twin plant for G_F contains original information for indeterminate pairs: this twin plant is a FSM that compares every pair of local trajectories to search for the pairs with the same arbitrarily long local observations, but exactly one of the two containing a fault, which are called local indeterminate pairs. First, we define an operation called delay closure with respect to a subset Σ_d of Σ to preserve all information about the events in Σ_d by abstracting away other parts.

Definition 5: (Delay Closure). Given a FSM G = (Q, Σ, δ, q^0), its delay closure with respect to $\Sigma_d \subseteq \Sigma$ is $\mathsf{C}_{\Sigma_d}(G) = (Q, \Sigma_d, \delta_d, q^0)$ where $(q, \sigma, q') \in \delta_d, \sigma \in \Sigma_d$ iff $\exists s \in (\Sigma \backslash \Sigma_d)^*, (q, s\sigma, q') \in \delta$.

We now describe how to construct the local diagnoser for a given component, based on which we build the local twin plant.

Definition 6: (Local Diagnoser). Given a component model G_i, its local diagnoser is the nondeterministic FSM $D_i = (Q_{D_i}, \Sigma_{D_i}, \delta_{D_i}, q^0_{D_i})$, where $Q_{D_i} \subseteq Q_i \times 2^{\Sigma_{i_f}}$ is the set of states, $\Sigma_{D_i} = \Sigma_{i_c} \cup \Sigma_{i_o}$ is the set of events, $\delta_{D_i} \subseteq Q_{D_i} \times \Sigma_{D_i} \times Q_{D_i}$ is the set of transitions, and $q^0_{D_i} = (q^0_i, N)$ is the initial state of the diagnoser. The transitions of δ_{D_i} are those $((q, l), e, (q\prime, l\prime))$ with (q, l) reachable from the initial state $q^0_{D_i}$, and satisfying the following condition: there is a transition path $p = (q \xrightarrow{u_1} q_1 ... \xrightarrow{u_m} q_m \xrightarrow{e} q\prime)$ in G, with $u_k \in \Sigma_{i_u} \cup \Sigma_{i_f}, \forall k \in \{1, ..., m\}$, $e \in \Sigma_{i_o} \cup \Sigma_{i_c}$ and $\ell\prime = \ell \cup (\{u_1, ...u_m\} \cap \Sigma_{i_f})$.

Without loss of generality, we give the definition of local diagnoser for the set of faults in the component G_i, which is perfectly suitable to deal with single fault when we run our algorithm each time for one fault since twin plant method has exponential complexity with the number of faults. A diagnoser constructed as above shows fault information after any sequence of communication events and observable events in the faulty component. Figure 3 shows the local diagnoser for the component G_1 of the system depicted in Figure 1. Given a diagnoser, its corresponding twin plant is obtained by synchronizing this diagnoser with itself, called left and right instances (denoted by D^l_i and D^r_i, respectively), based on the set of observable events. From Definition 6, we know that a diagnoser keeps observable events as well as communication events. However, since the local twin plant is to obtain all pairs with the same observations to search for local indeterminate pairs, only observable events should be synchronized. Since shared events are the only synchronized events, the non-synchronized events are distinguished between the two instances by the prefix L (R): in D^l_i (D^r_i), each communication event $c \in \Sigma_{i_c}$ from D_i is renamed by $L : c$ ($R : c$). The names of all locally observable events remain the same. The definition of local twin plant is shown as follows.

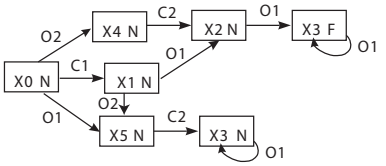

Fig. 3: Local diagnoser for G_1 of the system in Figure 1.

Definition 7: (Local Twin Plant). Given a diagnoser D_i, the corresponding local twin plant is obtained by synchronizing its left instance with its right instance based on the set of observable events, denoted by $T_i = (Q_{T_i}, \Sigma_{T_i}, \delta_{T_i}, q^0_{T_i}) = D^l_i \| D^r_i$. Each state of a local twin plant is a pair of local diagnoser states providing two possible diagnoses with the same local observations. Given a local twin plant state $((q^l, l^l)(q^r, l^r))$, where (q^l, l^l) and (q^r, l^r) are two diagnoser states and each one contains both a system state and a fault label. If the considered fault $F \in l^l \cup l^r$ but $F \notin l^l \cap l^r$, which means that the occurrence of F is not certain up to this state, then this state is called an ambiguous state with respect to the fault F. An ambiguous state cycle is a cycle containing only ambiguous states. In a local twin plant, if a path contains an ambiguous state cycle with at least one locally observable event, then it is called a **local indeterminate path**, which corresponds to a local indeterminate pair. Note that local indeterminate paths contain original diagnosability information and can be obtained only in the local twin plant of the component G_F. If a local indeterminate pair can be extended into a global indeterminate pair, then we say that its corresponding local indeterminate path is globally consistent. Figure 4 presents the left and right instances of the local diagnoser for the component G_1 (top left and top right) as well as part of the corresponding local twin plant (bottom). The gray node represents an ambiguous state since the fault label is only contained in one of two diagnoser states. Thus, we can see that the corresponding path in this local twin plant is a local indeterminate path since it contains an ambiguous state cycle with a locally observable event.

We know that the function of the local twin plant is to obtain the original diagnosability information, i.e., all local indeterminate paths. From Figure 4, we can see that a local twin plant constructed as described above normally has a large

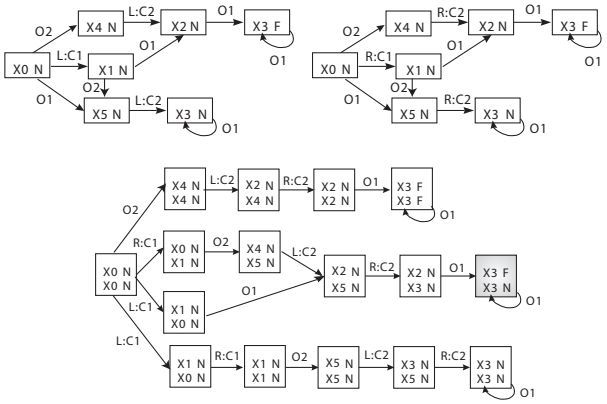

Fig. 4: Two instances of the diagnoser for G_1 (top) and part of the corresponding local twin plant (bottom).

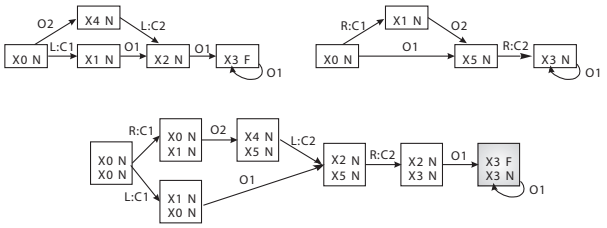

Fig. 5: Two reduced instances of the diagnoser for G_1 (top) and part of the corresponding reduced local twin plant (bottom).

redundant part, which is useless for a global decision, e.g., all paths without ambiguous state cycles. Furthermore, there could be several local indeterminate paths that correspond to the same pair of local trajectories in the local diagnoser. To simplify the local twin plant, we propose one way to reduce the two instances of the local diagnoser: D_i^l is obtained by retaining only paths with at least one fault cycle and D_i^r is reduced by retaining only paths with at least one cycle without fault state. This reduction keeps all necessary original diagnosability information since what we are interested in here are only local indeterminate pairs. Figure 5 shows the left and right reduced instances of the local diagnoser for the faulty component G_1 of Figure 1 (top) as well as part of the reduced local twin plant (bottom) that corresponds to the part of non-reduced local twin plant shown in the bottom part of Figure 4. In this way, the state space of the local twin plant constructed from reduced instances can be considerably reduced, which is called reduced local twin plant.

Lemma 2: The local twin plant corresponds to the same set of local indeterminate pairs as the reduced local twin plant.

Proof:

As described before, the local twin plant is constructed by synchronizing the non-reduced left and non-reduced right instances of the local diagnoser while the reduced one is constructed by synchronizing the reduced left and reduced right instances. For the sake of clarity, in the non-reduced left instance, we denote the set of paths with only fault state cycles by Λ_f^l, the set of paths without fault state cycle by $\Lambda_{\neg f}^l$, and then the set of paths with both fault state cycles and cycles without fault state by Λ_{both}^l. Similarly, in the non-reduced right instance, we denote the set of paths with only fault state cycles by Λ_f^r, the set of paths without fault state cycle by $\Lambda_{\neg f}^r$, and the set of paths with both fault state cycles and cycles without fault state by Λ_{both}^r. Then the local twin plant is constructed by $(\Lambda_{\neg f}^l \cup \Lambda_f^l \cup \Lambda_{both}^l) \parallel (\Lambda_{\neg f}^r \cup \Lambda_f^r \cup \Lambda_{both}^r)$. On the other hand, the reduced left instance retains the paths with at least one fault state cycle and the reduced right instance retains the paths with at least one cycle without fault state. Then the reduced local twin plant is constructed by $(\Lambda_f^l \cup \Lambda_{both}^l) \parallel (\Lambda_{\neg f}^r \cup \Lambda_{both}^r)$. The local twin plant construction can be expressed by the addition of the synchronized results of nine cases: 1) $(\Lambda_{\neg f}^l) \parallel_{\Sigma_{i_o}} (\Lambda_{\neg f}^r)$; 2)$(\Lambda_{\neg f}^l) \parallel_{\Sigma_{i_o}} (\Lambda_f^r)$; 3)$(\Lambda_{\neg f}^l) \parallel_{\Sigma_{i_o}} (\Lambda_{both}^r)$ 4)$(\Lambda_f^l) \parallel_{\Sigma_{i_o}} (\Lambda_{\neg f}^r)$; 5) $(\Lambda_f^l) \parallel_{\Sigma_{i_o}} (\Lambda_f^r)$; 6) $(\Lambda_f^l) \parallel_{\Sigma_{i_o}} (\Lambda_{both}^r)$; 7)$(\Lambda_{both}^l) \parallel_{\Sigma_{i_o}} (\Lambda_{\neg f}^r)$; 8) $(\Lambda_{both}^l) \parallel_{\Sigma_{i_o}} (\Lambda_f^r)$ and 9) $(\Lambda_{both}^l) \parallel_{\Sigma_{i_o}} (\Lambda_{both}^r)$. In the same way, the reduced local twin plant construction can be expressed by the addition of the synchronized results of four cases in the above, which are (case 4 + case 6 + case 7+ case 9). So compared to the reduced local twin plant, the local twin plant has five more synchronized results (case 1 + case 2 + case 3 + case 5 + case 8). Now consider case 4 and case 2, which are actually symmetrical. We can see that the part in left instance of case 2 is the same as the part in right instance of case 4 and the part in right instance of case 2 is the same as the part in left instance of case 4. It is easy to prove that the synchronized result of case 2 corresponds to the same set of local trajectory pairs in the local diagnoser as the synchronized result of case 4 does. In the same way, case 6 has the same result as case 8 and case 7 has the same result as case 3. Now the local twin plant has two more synchronized results (case 1 + case 5) than the reduced local twin plant. However, case 1 and case 5 can never get any local indeterminate pair. The reason is that in case 1, any path in $\Lambda_{\neg f}^l$ and in $\Lambda_{\neg f}^r$ has no fault state cycle, then the synchronized result has no ambiguous state cycle, which means that there is no corresponding local indeterminate pair. And in case 5, any path in Λ_f^l and in Λ_f^r has only fault state cycles. So their synchronization cannot obtain local indeterminate pair. Now we can say that the local twin plant corresponds to the same set of local indeterminate pairs as the reduced local twin plant does, which proves Lemma 2. ■

In our approach, we only construct the reduced local twin plant in the following, which is, from now on, called local twin plant for the sake of simplicity.

B. Global Consistency Checking

We obtain the set of local indeterminate paths in the local twin plant for the faulty component. However, up to now, its communication with the neighborhood in the whole system is not yet taken into account. Recall that if a local indeterminate pair can be extended into a global indeterminate pair, its corresponding local indeterminate path is said to be globally

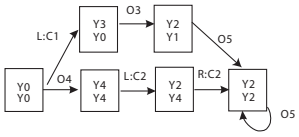

Fig. 6: Part of local twin checker for G_2 of the system in Figure 1.

consistent. In other words, joint diagnosability verification consists in checking the existence of globally consistent local indeterminate paths, whose existence proves non joint diagnosability. To check the global consistency, we consider now two important issues:

- the global consistency of the corresponding left trajectories of the local indeterminate paths in the local twin plant, shortly called left consistency checking in the following;
- the global consistency of the corresponding right trajectories of the local indeterminate paths in the local twin plant, shortly called right consistency checking.

Now we define another structure called local twin checker for a normal component, which is used to be synchronized with local twin plant for checking the global consistency. With similar idea as local twin plant, local twin checker is to obtain all pairs of local trajectories with the same observations, which has no fault information since it is defined for a normal component. Given a normal component, we first perform delay closure to keep only observable events and communication events. Then the left instance of the component is obtained by adding the prefix of L to non-synchronized events, i.e., communication events, and the right instance is acquired by adding the prefix of R to communication events, denoted by G_i^l and G_i^r, respectively.

Definition 8: (Local twin checker). Given a normal component G_i, its local twin checker is the FSM $C_i = G_i^l \| G_i^r$.

Figure 6 shows a part of the local twin checker for the component G_2 of the system depicted in Figure 1, where there is no fault information. To check left consistency and right consistency of local indeterminate paths, we define two consistent structures as follows.

Definition 9: (Left (Right) consistent plant). Given a subsystem G_S composed of $G_{i_1}, ..., G_{i_m}$, including the faulty component, and their corresponding local twin plant and local twin checkers, to obtain a left (right) consistent plant with respect to the subsystem G_S, denoted by T_f^l (T_f^r), we perform the following two steps:

- Distinguish right (left) communication events between local twin plant and local twin checkers by renaming them with the prefix of component ID. For example, $R{:}C2$ ($L{:}C2$) in the local twin checker of G_2 is renamed as $G_2{:}R{:}C2$ ($G_2{:}L{:}C2$).
- Note that observable events do not intersect between components and non-synchronized right (left) communication events are distinguished by the prefix of component ID. Now the renamed local twin plant and local twin checkers are synchronized, where the synchronized events are the common left (right) communication events.

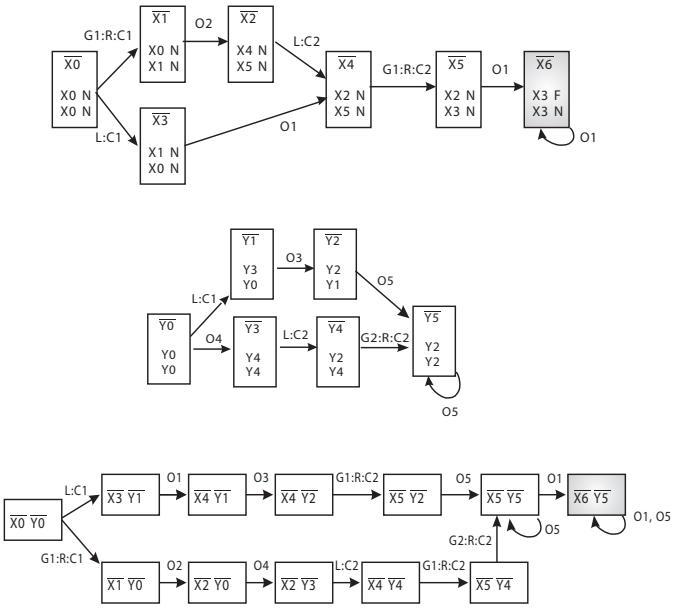

Fig. 7: Part of the renamed local twin plant for G_1 (top), part of renamed local twin checker for G_2 (middle) and part of the left consistent plant T_f^l (bottom).

In the left (right) consistent plant with respect to a subsystem G_S, each path p corresponds to a set of paths $p_{i_1}, ..., p_{i_m}$ in the local twin plant and local twin checkers of all components in G_S such that the set of left (right) trajectories of $p_{i_1}, ..., p_{i_m}$ are reconstructible with respect to G_S. Given a state in a left or right consistent plant, if it contains an ambiguous state in the local twin plant for the faulty component, then this plant state is also called an ambiguous state. For our example, the bottom part of Figure 7 shows a part of the left consistent plant T_f^l, which is obtained by synchronizing the renamed local twin plant of G_1 and the renamed local twin checker of G_2 (top and middle parts of Figure 7) based on the common left communication events. In the bottom part of this figure, the gray node $(\overline{X6}Y5)$ is an ambiguous state of the plant since it contains the local twin plant state $\overline{X6}$ that is an ambiguous one.

C. Algorithm

Algorithm 1 presents the procedure to verify a sufficient condition of joint diagnosability. As shown in the pseudo-code, algorithm 1 performs as follows. Given the set of component models as well as the fault F that may occur in the component G_F as input, we initialize the parameters as empty, i.e., G_S^l, the subsystem for the left consistency checking and G_S^r, the subsystem for the right consistency checking. The procedure of the algorithm can be separated into two parts: left consistency checking and right consistency checking.

Algorithm 1 Algorithm to check a sufficient condition of joint diagnosability

1: INPUT: the system model $G = (G_1, ..., G_n)$; the fault F and the faulty component G_F
2: Initializations: $G_S^l \leftarrow \emptyset$ (subsystem for left consistency checking); $G_S^r \leftarrow \emptyset$ (subsystem for right consistency checking)
3: $T_f^l \leftarrow ConstructLTP(G_F)$
4: $G_S^l \leftarrow G_F$
5: **while** $T_f^l \neq \emptyset$ and $DirectCC(G, G_S^l) \neq \emptyset$ **do**
6: $G_i \leftarrow SelectDirectCC(G, G_S^l)$
7: $C_i \leftarrow ConstructLTC(G_i)$
8: $T_f^l \leftarrow T_f^l \| C_i$
9: $G_S^l \leftarrow Add(G_S^l, G_i)$
10: $T_f^l \leftarrow RetainConsisPaths(T_f^l)$
11: **end while**
12: **if** $T_f^l = \emptyset$ **then**
13: return "F is jointly diagnosable in G"
14: **else**
15: $T_f^r \leftarrow AbstractRight(G_F, T_f^l)$
16: $G_S^r \leftarrow G_F$
17: **while** $T_f^r \neq \emptyset$ and $G_S^l \neq G_S^r$ **do**
18: $G_i \leftarrow SelectDirectCC(G_S^l, G_S^r)$
19: $T_f^r \leftarrow T_f^r \| AbstractRight(G_i, T_f^l)$
20: $G_S^r \leftarrow Add(G_S^r, G_i)$
21: $T_f^r \leftarrow RetainConsisPaths(T_f^r)$
22: **end while**
23: **if** $T_f^r = \emptyset$ **then**
24: return "F is jointly diagnosable in G"
25: **else**
26: return "Joint diagnosability cannot be determined"
27: **end if**
28: **end if**

- Left consistency checking begins with the local twin plant construction of G_F, the subsystem G_S^l being now G_F (line 3-4). As long as both the left consistent plant T_f^l with respect to the current left subsystem G_S^l and $DirectCC(G, G_S^l)$ are not empty (line 5), where $DirectCC(G, G_S^l)$ is the set of directly connected components to the subsystem G_S^l (a directly connected component being one sharing at least one common communication event with the subsystem) the algorithm repeatedly performs the following steps to further check left consistency in an extended subsystem:
  1) Select one directly connected component G_i to the subsystem G_S^l and construct its local twin checker C_i (line 6-7).
  2) Synchronize T_f^l with C_i to obtain left consistent plant for this extended subsystem based on the set of common left communication events (line 8). To do so, the set of non-synchronized right communication events are distinguished by the prefix of component ID.
  3) Update the subsystem G_S^l by adding G_i and reduce the newly obtained T_f^l by retaining only paths with ambiguous state cycles containing observable events for all components in G_S^l, which are called consistent paths with respect to G_S^l (line 9-10).

  If the left consistent plant T_f^l is empty, then there is no local indeterminate path that corresponds to a set of paths in the local twin plant and local twin checkers of all components in the subsystem such that their corresponding left trajectories are reconstructible. This follows that there is no globally consistent local indeterminate path. In this case, joint diagnosability information is returned (line 12-13). Otherwise, if T_f^l is not empty (line 14), then we proceed to check right consistency of the corresponding paths in T_f^l that have already been verified to be left consistent in the whole system.

- Right consistency checking begins with the function $AbstractRight(G_F, T_f^l)$ (line 15), which performs delay closure with respect to the set of right communication events and observable events of G_F. In this way, what we obtain does not contain left communication events, which are useless for right consistency checking. Then the subsystem G_S^r is assigned as G_F (line 16). As long as the right consistent plant T_f^r for the current right subsystem G_S^r is not empty and $G_S^l \neq G_S^r$ (line 17), we repeatedly perform the following steps to check right consistency in an extended subsystem (note that since left consistency checking does explore all connected components, during right consistency checking, we only need to consider the subsystem G_S^l instead of the whole system):
  1) Select a directly connected component G_i to G_S^r from G_S^l (line 18).
  2) Perform the function $AbstractRight(G_i, T_f^l)$, which has already been described above, and then synchronize with T_f^r based on the set of common right communication events (line 19). To do this, we rename the right communication events by removing the prefix of component ID, e.g., G_i:R:$C2$ renamed as R:$C2$.
  3) Update the subsystem G_S^r by adding G_i and reduce the newly obtained T_f^r by retaining only paths with ambiguous state cycles containing observable events for all components in G_S^r, i.e., consistent paths (line 20-21).

  If the right consistent plant T_f^r is empty, then there is no local indeterminate path that corresponds to a set of paths in the local twin plant and local twin checkers such that their left trajectories and right trajectories are reconstructible respectively, i.e., there is no globally consistent local indeterminate path. In this case, the algorithm returns joint diagnosability information (line 23-24). Otherwise, if T_f^r is not empty, we cannot determine whether the fault is jointly diagnosable or not. Then the algorithm returns the information about

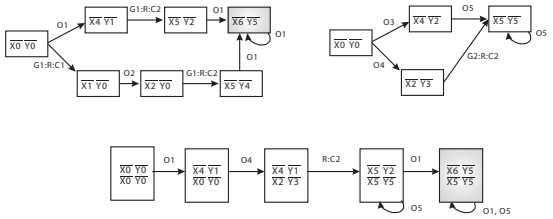

Fig. 8: FSM after delay closure on the left consistent plant (Figure 7) for G_1 (top left) and for G_2 (top right) and part of the right consistent plant (bottom).

the indetermination of joint diagnosability (line 25-26). In other words, empty left consistent plant T_f^l or empty right consistent plant T_f^r is a sufficient but not a necessary condition of joint diagnosability.

Theorem 3: In algorithm 1, if the left consistent plant T_f^l or the right consistent plant T_f^r is empty, then the fault is jointly diagnosable, but the reverse is not true.

Proof:

($\Rightarrow$) Suppose that T_f^l or T_f^r is empty and that the fault is not jointly diagnosable. From non joint diagnosability, it follows that there exists at least one globally consistent local indeterminate path. Since global consistency of a local indeterminate path implies both left consistency and right consistency, from algorithm 1, we know that, after left and right consistency checking, this local indeterminate path must correspond to a path both in T_f^l and in T_f^r. Thus, neither T_f^l nor T_f^r is empty, which contradicts the assumption.

($\Leftarrow$) Now we explain why if both T_f^l and T_f^r are not empty, it does not necessarily imply that the fault is not jointly diagnosable. Suppose that the left consistent plant T_f^l is not empty and that it contains two paths, denoted by $\rho 1$ and $\rho 2$, corresponding to two local indeterminate paths. $\rho 1$ corresponds to a set of paths $\rho_i^1, 1 \leq i \leq n$ in the local twin plant and local twin checkers of all components and $\rho 2$ corresponds to a set of paths $\rho_i^2, 1 \leq i \leq n$. Note that since the two sets of paths come from left consistent plant, then it is only guaranteed that the corresponding left trajectories of each set are reconstructible. Now suppose that the right trajectories of the set of paths $\rho_i^1, 1 \leq i \leq n$ are not reconstructible and the same for that of the set of paths $\rho_i^2, 1 \leq i \leq n$. It follows that the two local indeterminate paths cannot be extended into global indeterminate pairs and thus are not globally consistent. Then we further suppose that the right trajectories of the set of paths $\rho_1^1, ..., \rho_{n-1}^1, \rho_n^2$ are reconstructible and the same for the set of paths $\rho_1^2, ..., \rho_{n-1}^2, \rho_n^1$. From algorithm 1, it follows that finally the right consistent plant T_f^r is not empty. In this case, both T_f^l and T_f^r are not empty but there is no globally consistent local indeterminate paths, i.e., the fault is jointly diagnosable. ■

Now illustrate on our example that this condition is not necessary. The top parts of Figure 8 show the results of performing delay closure with respect to the set of right communication events and locally observable events both for G_1 and G_2 on the left consistent plant depicted in the bottom part of Figure 7. Then, to check right consistency, we rename again the right communication events by removing the component ID such that they can be synchronized. Finally, the bottom part of Figure 8 shows a part of the right consistent plant, which is not empty. Now both left and right consistent plants are not empty, but this does not imply the existence of global indeterminate pairs that witness non joint diagnosability. Actually, the part of the left consistent plant depicted here corresponds to two local indeterminate pairs in G_1 with their corresponding left consistent pairs in G_2, i.e., one local indeterminate pair is $((C1.O1.F.O1^*), (O1.C2.O1^*))$ in G_1 with its left consistent pair $((C1.O3.O5^*), (O3.U2.O5^*))$ in G_2 and the other local indeterminate pair is $((O2.U1.C2.F.O1^*), (C1.O2.C2.O1^*))$ in G_1 with its left consistent pair $((O4.C2.O5^*), (O4.C2.O5^*))$ in G_2. While the right consistent plant shown here corresponds to one local indeterminate pair in G_1, which is $((C1.O1.F.O1^*), (O1.C2.O1^*))$, with its corresponding right consistent pair in G_2, i.e., $((O4.C2.O5^*), (O4.C2.O5^*))$. Thus we can see that the same local indeterminate pair, i.e., $((C1.O1.F.O1^*), (O1.C2.O1^*))$ does not correspond to the same consistent pair in G_2 in the left consistent plant and in the right consistent plant, which means that this local indeterminate pair cannot be extended into a global indeterminate pair. In fact, our algorithm gives indeterminate information for joint diagnosable systems that satisfy the following condition: for any set of paths, i.e., one path in the local twin checker of each component and one in the local twin plant for faulty component being a local indeterminate path, that are left consistent and right consistent, respectively, and their corresponding local trajectories in the components cannot constitute an indeterminate pair through synchronization.

Theorem 4: Algorithm 1 has polynomial complexity with the number of system states, exponential complexity with the number of faults and exponential complexity with the number of components.

Proof: From their construction, for a component G_i, the maximum number of states and transitions of the local diagnoser are $(|Q_i| \times 2^{|\Sigma_{i_f}|})$ and $(|Q_i|^2 \times 2^{2|\Sigma_{i_f}|} \times (|\Sigma_{i_o}| + |\Sigma_{i_c}|))$, respectively, where $|Q_i|$ is the number of the component states, $|\Sigma_{i_f}|$ is the number of faults in G_i and $|\Sigma_{i_o}|$ ($|\Sigma_{i_c}|$) is the number of observable events (communication events) in G_i. The maximum number of states and transitions of its local twin plant (local twin checker) are $(|Q_i|^2 \times 2^{2|\Sigma_{i_f}|})$ and $(|Q_i|^4 \times 2^{4|\Sigma_{i_f}|} \times (|\Sigma_{i_o}| + |\Sigma_{i_c}|))$, respectively. In the worst case, the global consistency checking (both left and right ones) consists in synchronizing local twin plant and local twin checkers of all components. Thus, we can conclude that Algorithm 1 has polynomial complexity with the number of system states, exponential complexity with the number of faults and exponential complexity with the number of components. ■

Note that the exponential complexity with the number of faults is for the case where we handle all faults in one component simultaneously. To reduce the complexity, our algorithm is illustrated by dealing with one fault each time.

D. Relaxation of Assumption

In our approach, we have the assumption that all communication events are unobservable, which is a more difficult case than that where communication events are observable. Now we relax this assumption by dividing the set of communication events into two disjoint parts: $\Sigma_{i_c^o}$, the set of observable communication events and $\Sigma_{i_c^u}$, the set of unobservable communication events. In other words, communication events may be observable or unobservable. Algorithm 1 can be reused for this relaxed case after the following modifications.

- For the local component G_F, we construct its local diagnoser by preserving the information about all observable events as well as all communication events, including observable and unobservable ones, and then append to each retained state with fault information, which is the same as that described in Section V-A.
- In this local diagnoser, for each unobservable communication event, we distinguish it between two instances by adding the prefix of L (for left instance, σ being $L : \sigma$) and R (for right instance, σ being $R : \sigma$). Then we reduce the instances as described before and construct local twin plant by synchronizing the two reduced instances based on the set of observable events and the set of observable communication events.
- For other connected normal components, we construct their local twin checker in the same way except that the prefixes should be added only to unobservable communication events in the left and right instances such that the two instances are synchronized based on the set of observable events and observable communication events.
- During left (right) consistent plant construction, unobservable right (left) communication events are distinguished between the local twin plant and local twin checkers by the prefix of component ID. Then the renamed local twin plant and local twin checkers are synchronized based on unobservable left (right) communication events and observable communication events.

VI. Decidable Case

We have proved the undecidability of joint diagnosability when communication events are unobservable. If we assume their observability, then this problem becomes decidable. The reason is that, when all communication events are observable, in the local twin plant and local twin checkers of all components, we obtain all pairs of local trajectories with the same observations, including the same observable communication events. In other words, each path in the local twin plant or local twin checkers corresponds to a pair of local trajectories with the same sequence of communication events. It follows that to check global consistency of local indeterminate paths, the two separate phases for left and right consistency checking becomes only one phase. While in Algorithm 1, the consistency checking separated into two different phases is the reason why it gives only a sufficient but not necessary condition for joint diagnosability. Actually, the observability of communication events makes joint diagnosability equivalent to classical diagnosability since only one phase of global consistency checking implies the same global occurrence order of observations for global indeterminate pairs.

Algorithm 2 presents the procedure to check joint diagnosability when communication events are assumed to be observable. Taking the system model and the fault occurring in the faulty component as input, the parameter, i.e., the current subsystem G_S, is initialized as empty. Then the algorithm begins with the construction of the local twin plant of the faulty component G_F, the current subsystem becoming G_F (line 3-4). Here we emphasize that, due to the observability of communication events, the local twin plant should be constructed by synchronizing two reduced instances based on the set of observable events and the set of communication events, i.e., here communication events do not need to be distinguished by adding prefixes. As long as both current local twin plant T_f and $DirectCC(G, G_S)$ are not empty (line 5), the following steps are repeatedly performed:

- Select one component G_i directly connected to the current subsystem G_S and then construct its local twin checker C_i, which is obtained by first operating delay closure with respect to the set of communication events and observable events and then by synchronizing the two instances based on all events, i.e., the set of communication events and observable events. (line 6-7)
- Synchronize the current local twin plant T_f and C_i based on the common communication events of G_S and G_i, where the newly synchronized FSM is also called the local twin plant for the extended subsystem. (line 8)
- Update the current subsystem by adding this selected component and keep only the paths in the newly obtained FSM that contain ambiguous state cycles with observations for all involved components. (line 9-10)

During this procedure, if the local twin plant T_f for the current subsystem happens to be empty, which means that there is no path that contains ambiguous state cycle with observations for all concerned components. In this case, there is no local indeterminate path that is globally consistent and the algorithm returns joint diagnosability information (line 12-13). Otherwise, if at the end the final FSM is not empty, it is returned by the algorithm as non joint diagnosability information (line 14-15), where any path in it corresponds to a globally consistent local indeterminate path. The reason is that if the communication events are observable, then any path in the local twin plant or a local twin checker corresponds to a pair of local trajectories with the same observations, including the same communication events. So with the assumption of observability of communication events, joint diagnosability checking becomes decidable.

VII. Comparison and Discussion

In this section, we compare joint diagnosability for self-observed distributed systems with classical diagnosability for distributed systems where observable events can be globally observed. In other words, in the latter one, diagnosability analysis requires the global occurrence order of observable events, which is not the case for the former one. Before this

Algorithm 2 Algorithm for checking joint diagnosability with observable communication events

```
INPUT:
  the system model G = (G_1, ..., G_n);
  the fault F and the faulty component G_F
Initializations:
  G_S ← ∅ (subsystem considered for current checking)
T_f ← ConstructLTP(G_F)
G_S ← G_F
while T_f ≠ ∅ and DirectCC(G, G_S) ≠ ∅ do
    G_i ← SelectDirectCC(G, G_S)
    C_i ← ConstructLTC(G_i)
    T_f ← T_f ‖ C_i
    G_S ← Add(G_S, G_i)
    T_f ← RetainConsisPaths(T_f)
end while
if T_f = ∅ then
    return "F is jointly diagnosable in G"
else
    return T_f
end if
```

comparison, we briefly recall classical diagnosability definition as follows [20].

Definition 10: (Diagnosability). A fault F is diagnosable in a system G iff

$$\forall s^F \in L(G), \exists k \in N, \forall t \in L(G)/s^F, |t| \geq k \Rightarrow (\forall p \in L(G),\ P(p) = P(s^F.t) \Rightarrow F \in p).$$

The above definition states that if F is diagnosable, then for each trajectory ending with the fault s^F in G, for each t that is an extension of s^F in G with sufficient events, every trajectory p in G that is observation equivalent to $s^F.t$ should contain in it F. Informally speaking, the existence of two indistinguishable behaviors, i.e., holding the same enough observations with exactly one of them containing the given fault F, violates diagnosability property. The diagnosability analysis approaches check the existence of such indistinguishable behaviors. With similar idea to indeterminate pairs for joint diagnosability, we call a pair of trajectories p and $p\prime$ satisfying the following conditions as a **critical pair**:

- p contains F and $p\prime$ does not;
- p has arbitrarily long observations after the occurrence of F;
- $P(p) = P(p\prime)$.

The existence of critical pairs violates Definition 10 and thus witnesses non-diagnosability.

We can prove that joint diagnosability is stronger than diagnosability for systems with global observations.

Lemma 3: Given two systems G composed of $G_1, ...G_n$ and $G\prime$ composed of $G'_1, ...G'_n$ such that for each $i \in \{1, ..., n\}$, components G_i and G'_i have the same structure except that all observable events in the component G_i can only be observed by G_i while each observable event in the component G'_i can be observed by all components of $G\prime$. In other words, G is a self-observed distributed system and $G\prime$ is the one with global observations. Then we have the following result: if the fault F is jointly diagnosable in G, then it is diagnosable in $G\prime$.

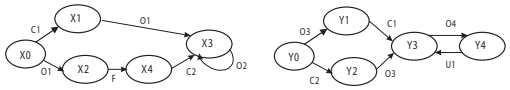

Fig. 9: A simple system model with two components: G_1(left) and G_2(right).

Proof:

Suppose that the fault F is jointly diagnosable in G and that F is not diagnosable in $G\prime$. From the non diagnosability of F in $G\prime$ and $G\prime$ is a system with global observations, we know that there exists at least one global critical pair of trajectories $p1\prime$ and $p2\prime$ in $G\prime$, i.e., $p1\prime$ and $p2\prime$ satisfying three conditions: 1) only one of them contains F, suppose $p1\prime$; 2) $p1\prime$ has enough observations after the occurrence of F in all components; 3) $P(p1\prime) = P(p2\prime)$, the projection of $p1\prime$ to observable events of $G\prime$ is the same as that of $p2\prime$, which means that they have the same observations from a global point of view. Now in the self-observed system G, let $p1$ and $p2$ denote the corresponding trajectories of $p1\prime$ and $p2\prime$. If we do not consider the difference that each observable event in $p1$ and $p2$ can only be observed by its own component while each observable event in $p1\prime$ and $p2\prime$ can be observed by all components, then we have $p1 = p1\prime$ and $p2 = p2\prime$. It follows that the fault F is contained in $p1$ but not in $p2$ and after the occurrence of F, $p1$ has enough local observations in each component. Furthermore, note that $p1$ and $p2$ have the same observations, then from the fact that there is no intersection of observable events between components, we can deduce that $\forall k \in \{1, ..., n\}, P_k(p1) = P_k(p2)$. Clearly, $p1$ and $p2$ are an indeterminate pair and thus F is not jointly diagnosable in G, which contradicts the assumption that F is jointly diagnosable in G.

■

If the fault F is diagnosable in $G\prime$, it is not necessarily jointly diagnosable in G. Actually, if F is diagnosable in $G\prime$, this means that there is no critical pair in $G\prime$. However, this does not imply that there is no indeterminate pair in G. Suppose that there is an indeterminate pair $p1$ and $p2$ in G with $p1$ containing the fault. Then we have $\forall k \in \{1, ..., n\}, P_k(p1) = P_k(p2)$. Now in the system with global observations $G\prime$, let $p1\prime$ and $p2\prime$ denote the corresponding trajectories of $p1$ and $p2$. Then we have $\forall k \in \{1, ..., n\}, P_k(p1\prime) = P_k(p2\prime)$. This does not mean that we can get $P(p1\prime) = P(p2\prime)$, which is however one condition of a critical pair. So there does not necessarily exist a critical pair in $G\prime$. In other words, if two trajectories in G have the same enough local observations in all components, their corresponding trajectories in $G\prime$ may have different observations from global point of view. Thus the existence of indeterminate pairs in G does not imply the existence of critical pairs in $G\prime$.

Figure 9 shows a very simple system with two components, G_1 (left) and G_2 (right). Suppose that it is a distributed system with global observations. Intuitively, we can see that for any faulty trajectory, the occurrence of observable event $O1$ is before the occurrence of observable event $O3$. While

for any normal trajectory, the occurrence of $O3$ is before that of $O1$. In other words, there is no critical pair and thus the fault is diagnosable in this system. Now suppose that this system is a self-observed distributed system, i.e., the observable events can only be observed by their own component. It is easy to find an indeterminate pair. Consider the pair of trajectories $\rho = (O3, C1, O1, (O2, O4, U1)^*)$ and $\rho\prime = (O1, F, C2, O3, (O2, O4, U1)^*)$. Only $\rho\prime$ is a faulty trajectory and both of them have the same sufficient observations for each component. Thus the fault is not jointly diagnosable. Here we can see that the global occurrence order of observable events makes the fault F diagnosable. While since joint diagnosability does not require global occurrence order of observations, so the fault is not jointly diagnosable.

VIII. Conclusion

In this paper, we consider self-observed distributed systems where observable events can only be observed by their own components. Clearly, the monolithic model of the system is not required, thus the distributed and private (w.r.t. observation) nature of real systems is taken into account. Then we prove the undecidability of checking joint diagnosability when communication events are unobservable, before proposing an algorithm to test a sufficient condition. To check whether there exist indeterminate pairs in the system, we start from local indeterminate paths in the local twin plant and then we check both in sequence left consistency and right consistency. Note that, due to the observation-privacy, the global occurrence order of observable events between different components is not known, which is taken into account through constructing left and right consistent plants separately. At the opposite, in the approaches for DES with globally observable events, twin plant is constructed by incrementally synchronizing local twin plants via both left and right communication events at the same time, which means that in their case, knowledge of the global occurrence order is required. Similar to the distributed algorithms for classical diagnosability ([23], [25], [29], etc.), our algorithm has to construct some part of a global structure, but much less than that in the centralized approach (in particular the components not involved in the algorithm have their model completely not disclosed) and this is normally unavoidable for off-line diagnosability analysis. Afterwards, we adapt this sufficient algorithm for a more relaxed case, i.e., communication events could be unobservable and observable. Then we discuss the decidable case where communication events are observable by giving a formal algorithm to check joint diagnosability. However, the decidable case in [26] is not the same as that in this paper. The former also deals with unobservable communication events, which becomes decidable because of the assumption of exhaustive enumeration about local paths. While here the decidable case is thanks for the observability of communication events. Finally, we prove that joint diagnosability of self-observed systems is stronger than classical diagnosability of globally observed systems with an illustrated example. We see that there is a gap between the decidable and undecidable cases. Next interesting work is to investigate where is the frontier between these two cases, i.e., to study the decidability of joint diagnosability for partial observability of communication events. Another future work is to study the extension of our approach to deal with predictability [30], i.e., the property of the system to be able to predict the fault with certainty before its occurrence considering that in some critical cases, it is very expensive to recover the system after fault occurrence.

Acknowledgment

This work has been supported by the STIC-AmSud DATE project, which is funded by the regional program STIC-AmSud (No.13STIC-04).

References

[1] L. Ye and P. Dague, *Diagnosability analysis for self-observed distributed discrete event systems*, Proceedings of the 4th International Conference on Advances in System Testing and Validation Lifecycle (VALID-12), 2012, pp. 93-98.

[2] S. Tripakis, *Undecidable problems of decentralized observation and control on regular languages*, Information Processing Letters, vol. 9, no. 1, 2004, pp. 21-28.

[3] L. Ye and P. Dague, *New results for joint diagnosability of self-observed distributed discrete event systems*, Proceedings of the 23rd International Workshop on Principles of Diagnosis (DX-12), 2012.

[4] M. Bayoudh, L. Travé-Massuyès, and X. Olive, *Hybrid systems diagnosability by abstracting faulty continuous dynamics*, Proceedings of the 17th International Workshop on Principles of Diagnosis (DX-06), Burgos, Spain, 2006.

[5] M. Bayoudh, L. Travé-Massuyès, and X. Olive, *Coupling continuous and discrete event system techniques for hybrid system diagnosability analysis*, Proceedings of the 18th European Conference on Artificial Intelligence (ECAI-08), Patras, Greece, 2008, pp. 219-223.

[6] M.-O. Cordier, L. Travé-Massuyès, and X. Pucel, *Comparing diagnosability in continuous and discrete-event systems*, Proceedings of the 17th International Workshop on Principles of Diagnosis (DX-06), 2006, pp. 55-60.

[7] M. Daigle, X. Koutsoukos, and G. Biswas, *An event-based approach to hybrid systems diagnosability*, Proceedings of the 19th International Workshop on Principles of Diagnosis (DX-08), Blue Mountains, Australia, 2008.

[8] E. Fabre, A. Benveniste, C. Jard, L. Ricker, and M. Smith, *Distributed state reconstruction for discrete event systems*, proceedings of the 38th IEEE Control and Decision Conference (CDC-00), Sydney, 2000, pp. 2252-2257.

[9] R. Debouk, S. Lafortune, and D. Teneketzis, *Coordinated decentralized protocols for failure diagnosis of discrete event systems*, Journal of Discrete Event Dynamical Systems: Theory and Application 10 (1-2), 2000, pp. 33-86.

[10] R. Debouk, R. Malik, and B. Brandin, *A modular architecture for diagnosis of discrete event systems*, Proceedings of the 41st IEEE Conference on Decision and Control (CDC-02), Las Vegas, NV, USA, 2002, pp. 417-422.

[11] A. Grastien, J. R. Anbulagan, and E. Kelareva, *Diagnosis of discrete-event systems using satisfiability algorithms*, Proceedings of the 22th American National Conference on Artificial Intelligence (AAAI-07), 2007, pp. 305-310.

[12] S. Haar, A. Benveniste, E. Fabre, and C. Jard, *Partial order diagnosability of discrete event systems using petri nets unfoldings*, Proceedings of the 42nd IEEE Conference on Decision and Control (CDC-03), Hawaii, USA, 2003, pp. 3748-3753.

[13] S. Bavishi and E. Chong, *Automated fault diagnosis using a discrete event systems framework*, Proceedings of the 9th IEEE International Symposium on Intelligent Control, 1994 pp. 213-218.

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

[15] C. G. Cassandras and S. Lafortune, *Introduction To Discrete Event Systems*, Second Edition. Springer, New York, 2008.

[16] R. Cieslak, C. Desclaux, A. Fawaz, and P. Varaiya, *Supervisory control of discrete-event processes with partial observations*, IEEE Transactions on Automatic Control, 33(3), 1988, pp. 249-260.

[17] O. Contant, S. Lafortune, and D. Teneketzis, *Diagnosis of modular discrete event systems*, Proceedings of the 7th International Workshop on Discrete Event Systems (WODES-04), Reims, France, 2004.

[18] O. Contant, S. Lafortune, and D. Teneketzis, *Failure diagnosis of discrete event systems: The case of intermittent faults*, Proceedings of the 41st IEEE Conference on Decision and Control (CDC-02), 2002, pp. 4006-4011.

[19] E. Fabre, A. Benveniste, and C. Jard, *Distributed diagnosis for large discrete event dynamic systems*, Proceedings of the IFAC world congress, 2002, pp. 237-256.

[20] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis, *Diagnosability of discrete event system*, IEEE Transactions on Automatic Control, 40(9), 1995, pp. 1555-1575.

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

[22] A. Cimatti, C. Pecheur, and R. Cavada, *Formal verification of diagnosability via symbolic model checking*, Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-03), 2003, pp. 363-369.

[23] Y. Pencolé, *Diagnosability analysis of distributed discrete event systems*, Proceedings of the 16th European Conference on Articifial Intelligent (ECAI04), 2004, pp. 43-47.

[24] Y. Pencolé, *Assistance for the design of a diagnosable component-based system*, Proceedings of the 17th IEEE International Conference on Tools with Artificial Intelligence (ICTAI05), 2005, pp. 549-556.

[25] A. Schumann and Y. Pencolé, *Scalable diagnosability checking of event-driven systems*, 20th International Joint Conference on Artificial Intelligence (IJCAI-07), 2007, pp. 575-580.

[26] L. Ye and P. Dague, *Diagnosability analysis of discrete event systems with autonomous components*, Proceedings of the 19th European Conference on Artificial Intelligence (ECAI-10), 2010, pp. 105-110.

[27] R. Cori and Y. Métivier, *Recognizable subsets of some partially abelian monoids*, Theoretical Computer Science, vol. 35, 1985, pp. 179-189.

[28] V. Halava and T. Harju, *Undecidability of infinite post correspondence problem for instances of Size 9*, Theoretical Informatics and Applications - ITA, vol. 40, no. 4, 2006, pp. 551-557.

[29] A. Schumann and J. Huang, *A scalable jointree algorithm for diagnosability*, Proceedings of the 23rd AAAI Conference on Artificial Intelligence (AAAI-08), 2008, pp. 535-540

[30] L. Ye, P. Dague, and F. Nouioua, *Predictability analysis of distributed discrete event systems*, Proceedings of the 52nd IEEE Conference on Decision and Control (CDC-13), 2013, pp. 5009-5015.