Diagnosability analysis of distributed discrete event systems

Yannick Pencolé [1]

Abstract. This paper addresses the diagnosability problem of distributed discrete event systems. Until now, the problem of diagnosability has always been solved by considering centralised approaches, monolithic systems. These approaches do not use the fact that real monitored systems are generally modelled in a distributed manner. In this paper, we propose a framework for the diagnosability analysis of such systems: we study the diagnosability of the system using the fact that it is based on a set of communicating components. In the case where the system is not diagnosable, we also want to provide more accurate information in order to better understand the causes.

1 INTRODUCTION

For many years, the problem of fault diagnosis in monitored discrete event system has been receiving an increasing interest from both the model-based diagnosis and the control communities. The results from the control community mainly consist of theoretical properties about diagnosability of such systems [7, 9, 5, 11] and in the meantime, the Model-Based Diagnosis community focuses on the feasibility of diagnosis approaches for discrete event systems in real applications [1, 6, 12]. Until now, the diagnosability analysis has always been based on a central approach: the knowledge about the system is assumed to be a monolithic transition system, an automaton. Dealing with real applications of the field such as telecommunication networks, power distribution networks, the hypothesis is clearly unrealistic because of the size of those applications. Moreover, the diagnosability property is not studied by taking into account the fact that such systems are distributed.

The objective of this paper is to fill two gaps. Firstly, it addresses the problem of diagnosability by taking into account properties of distributed discrete event systems. Secondly, it proposes a way to analyse the diagnosability of the system in a decentralised way without the use of a global model of the system. Moreover, in the case where the system is not diagnosable, we want to provide accurate reasons that make the system not diagnosable.

The paper is presented as follows. Section 2 describes the model formalism used in the paper. Section 3 briefly introduces the diagnosability property we want to check. Section 4 describes the decentralised framework which will serve to analyse the diagnosability of any given system, followed by an algorithm using this framework in section 5. The paper ends with the description of an example (section 6) and some related works (section 7).

[1] The Australian National University, Canberra, Australia email: Yannick.Pencole@anu.edu.au

2 MODEL OF THE SYSTEM

The kind of systems we consider is a set of components which evolve by the occurrence of events. The components can communicate each other by the exchange of events. Such a discrete event system can be modelled as a set of automata [7], each automaton representing the model of component, i.e. a *local model*.

Definition 1 *A local model Γ_i is an automaton $\Gamma_i = (Q_i, E_i, T_i, q_{0i})$ where:*

- *Q_i is a finite set of states;*
- *E_i is the set of events occurring on Γ_i;*
- *$T_i \subseteq Q_i \times E_i \times Q_i$ is the set of transitions;*
- *q_{0i} is the initial state.*

The set E_i can be divided into four disjoint sets.

1. N_i are the *normal* events. If an event $n_i \in N_i$ occurs on the component then the component can change its internal state. The event n_i can only occur on this component.
2. F_i are the *faulty* events. If an event $f_i \in F_i$ occurs on the component then the component becomes faulty and the label of the fault is f_i. The fault f_i can only occur on this component.
3. O_i are the *observable* events. If an event $o_i \in O_i$ occurs on the component then this event is observed by a supervisor of the system. The event o_i can only occur on this component.
4. C_i are the *communication* events. If an event $c_i \in C_i$ occurs on the component then this event occurs at least on another component.

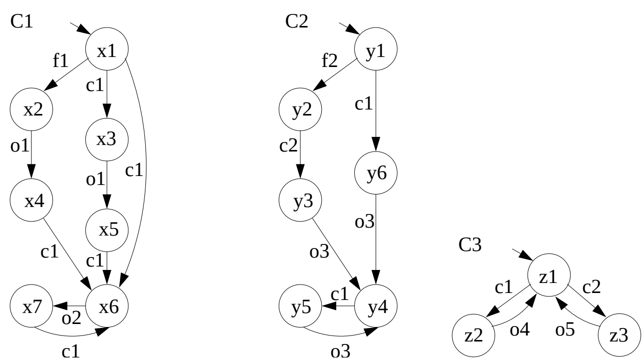

Figure 1. Model of a system defined by three local models

Figure 1 presents a system composed of three local models defined as above. The events *fi* are the fault events, the events *ci* are the communication events and the events *oi* are the observable events. For the sake of simplicity there are not normal events in this example.

We call a *subsystem* any non-empty set of components of the system. The model of a subsystem is defined by the set of the local models of its components and an operation based on the synchronised product of the local models. Given the local models and the synchronisation operation, it is possible to obtain a unique automaton representing the whole subsystem: this automaton is called the *global model* of the subsystem.

Before introducing the global model, we present a general predicate for the synchronisation of a set of transitions. For the reason of clarity in the synchronisation definition, we consider that for each state s of each local model, the transition $s \xrightarrow{\epsilon} s$ exists where ϵ is the *null event*. Such a transition allows the component to stay in its state whereas other components may change their own internal state. Given $(t_1, \ldots, t_m)$ a set of transitions and $\mathcal{E}$ a set of events such that $\epsilon \notin \mathcal{E}$, the predicate $Sync((t_1, \ldots, t_m), \mathcal{E})$ defines the synchronised conditions on $(t_1, \ldots, t_m)$ as follows:

$$Sync((t_1, \ldots, t_m), \mathcal{E}) \equiv$$

$$(\exists j \in \{1, \ldots, m\} : t_j \notin \mathcal{E} \wedge \forall i \in \{1, \ldots, m\} \setminus \{j\} : t_i = \epsilon)$$

$$\vee((\exists j \in \{1, \ldots, m\} : t_j \in \mathcal{E})\wedge$$

$$(\forall i \in \{1, \ldots, m\} : (t_i \in \mathcal{E} \Rightarrow t_i = t_j) \wedge (t_i \notin \mathcal{E} \Rightarrow t_i = \epsilon))).$$

Let n be the number of local models of the system, the global model of any subsystem $\{\Gamma_{i_1}, \ldots, \Gamma_{i_k}\}$ where $\{i_1, \ldots, i_k\} \subseteq \{1, \ldots, n\}, k \geq 1$ is defined as follows.

Definition 2 *The* global model *of the subsystem is the automaton* $\gamma = (Q, E, T, q_0)$ *where:*

- $Q \subseteq Q_{i_1} \times \ldots \times Q_{i_k}$ *is the set of states;*
- $E = \bigcup_{j=i_1}^{i_k} E_j$ *is the set of events;*
- $q_0 = (q_{0i_1} \times \ldots \times q_{0i_k})$ *is the initial state;*
- $T \subseteq T_{i_1} \times \ldots \times T_{i_k}$ *is the set of transitions defined as follows:*

$$(t_{i_1}, \ldots, t_{i_k}) \in T \equiv Sync((t_{i_1}, \ldots, t_{i_k}), \bigcup_{j=i_1}^{i_k} C_j).$$

In the following, a transition $t = (t_{i_1}, \ldots, t_{i_k})$ of a global model will be noted $(q_{i_1}, \ldots, q_{i_k}) \xrightarrow{e} (q'_{i_1}, \ldots, q'_{i_k})$ where $t_{i_j} = q_{i_j} \xrightarrow{e_{i_j}} q'_{i_j}$ and $e = \epsilon$ if $\forall j \in \{1, \ldots, k\}, e_{i_j} = \epsilon$ and $e = e_{i_j}$ if $\exists j \in \{1, \ldots, k\} : e_{i_j} \neq \epsilon$.

The global model of a subsystem containing only a component is the local model of this component. The global model of the system (noted Γ) corresponds to the global model of the biggest subsystem.

The global model Γ represents the behaviour of the whole system upon the hypothesis that communication events that are shared by a set of components, effectively occur on these components at the same time. To end with the characteristics of the considered systems, an assumption is introduced.

Assumption 1 *All the components of the system are free of observation starvation.*

This hypothesis means first that from any state of a component, it will emit observations in the future (the observable language of the component is live) and secondly the first observation will be emitted in a finite delay (the number of events occurring in all the components before the occurrence of this observation is supposed to be finite). This hypothesis has two purposes. Firstly, even if a component is modelled with silent cycles (for compactness reasons), the hypothesis states that the probability that during one execution the system stays in such a silent cycle is null. Secondly, even if a component can emit a infinite sequence of observable events, the hypothesis states that the probability that during one execution of the system such a sequence is emitted, is null.

3 DIAGNOSABILITY OF THE SYSTEM

There are several definitions of diagnosability depending on the kind of systems and the diagnosis approach [7], [3], [2]. The model presented above is for monitored dynamical systems and the methods used to diagnose those systems are on-line diagnosis approaches. In that case, the well-suited diagnosability property to analyse is the one of [7] which states that a system is diagnosable if, given a flow of observations, it will be possible in a finite delay to diagnose all the faults that have occurred on the monitored system.

3.1 Definitions

The diagnosability property we want to check is closer from the one of [7] and can be informally described as follows.[2]

Definition 3 *A fault f is diagnosable iff there is a finite number l of observations after the occurrence of f, so that we are sure that f has effectively occurred.*

More formally, let p_f be a path of transitions in Γ from q_0 ending with the occurrence of f to the state q_f and let s_f be a path of transitions in Γ from q_f. Let $Obs(p)$ be the observable sequence produced by any transition path p from q_0 and let $Obs^{-1}(p)$ be the set of paths p' from q_0 in Γ such that $Obs(p') = Obs(p)$. f is diagnosable iff:

$$\forall p_f, s_f, \exists l \in \mathbb{N} : |Obs(s_f)| \geq l \Rightarrow$$

$$(\forall p \in Obs^{-1}(p_f s_f), f \text{ occurs in } p).$$

Definition 4 *A system is diagnosable iff every fault event f is diagnosable.*

3.2 Diagnosability for distributed DES

This section presents the diagnosability property of a system taking into a account the fact that the system is modelled as a set of automata.

Definition 5 *A fault f occurring on a subsystem is* locally diagnosable *iff there is a finite number l of observations from the subsystem after the occurrence of f, so that we are sure that f has effectively occurred on the subsystem.*

This diagnosability definition is similar to the one presented above. The difference is that only the observations from the subsystem are considered. Let γ be a subsystem, let $Obs_\gamma(p)$ be the observable sequence obtained by the projection of $Obs(p)$ (see above) to the observable events of γ and let $Obs_\gamma^{-1}(p)$ be the set of paths p'

2 The definition is not exactly the same as in [7] because the given hypotheses on the model are not exactly the same. Nevertheless, the main idea is kept.

in Γ such that $Obs_\gamma(p') = Obs_\gamma(p)$, then f is locally diagnosable iff:

$$\forall p_f, s_f, \exists l \in \mathbb{N} : |Obs_\gamma(s_f)| \geq l \Rightarrow$$
$$(\forall p \in Obs_\gamma^{-1}(p_f s_f), f \text{ occurs in } p).$$

Definition 6 *A subsystem is* locally diagnosable *iff every fault occurring on that subsystem is locally diagnosable in the subsystem.*

By definition, if the subsystem γ is the system itself, the local diagnosability definition is equivalent to the diagnosability definition.

Proposition 1 *Under the assumption 1, if f is locally diagnosable on a subsystem then f is diagnosable.*

Proof: f is locally diagnosable in the subsystem γ, so there is a finite number l of observations from the subsystem γ after the occurrence of f so that the occurrence of f is sure. Because of the hypothesis 1, each of the l observations from the subsystem γ occurs after a finite number of observations from the other components. It follows that the number of observations after the occurrence of the l^{th} local one is finite: f is diagnosable.

4 DECENTRALISED FRAMEWORK

On this section, we present a framework for checking the diagnosability of a fault F occurring on a component i. The framework is decentralised in the sense that the computation of the global model Γ of the system is not needed.

Definition 7 *The* non-deterministic local F-diagnoser Δ_i^F *is a finite-state machine $\Delta_i^F = (Q_{\Delta_i^F}, E_{\Delta_i^F}, T_{\Delta_i^F}, q_{0_{\Delta_i^F}})$ where:*

- $Q_{\Delta_i^F} \subseteq Q_i \times \{\{F\}, \emptyset\}$ *is the finite set of states;*
- $E_{\Delta_i^F} = O_i \cup C_i$;
- $T_{\Delta_i^F} \subseteq Q_{\Delta_i^F} \times E_{\Delta_i^F} \times Q_{\Delta_i^F}$ *is the set of transitions;*
- $q_{0_{\Delta_i^F}} = (q_{0i}, \emptyset)$ *is the initial state.*

The transitions of $T_{\Delta_i^F}$ are the transitions $(q, f) \xrightarrow{e} (q', f')$ reachable from the initial state $q_{0_{\Delta_i^F}} = (q_{0i}, \emptyset)$ such that: there exists a transition path $q \xrightarrow{uo_1} q_1 \dots q_{m-1} \xrightarrow{uo_m} q_m \xrightarrow{e} q'$ in the model Γ_i with $uo_j \notin O_i \cup C_i, \forall j \in \{1, \dots, m\}$ and $f' = f \cup (\{F\} \cap \{uo_1, \dots, uo_m\})$.

This local F-diagnoser is inspired from the non-deterministic global diagnoser from [5]. This diagnoser is able to diagnose the fault F given the sequence of observable and communication events produced by the component. Figure 2 shows the *f1*-diagnoser computed from the local model of the component C1 (see Figure 1).

The purpose of the F-diagnoser is to be synchronised with itself to obtain the F-verifier as follows.

Definition 8 *The* local F-verifier $\mathcal{V}_i^F$ *is the automaton* $(Q_{\mathcal{V}_i^F}, E_{\mathcal{V}_i^F}, T_{\mathcal{V}_i^F}, q_{0_{\mathcal{V}_i^F}})$

- $Q_{\mathcal{V}_i^F} \subseteq Q_{\Delta_i^F} \times Q_{\Delta_i^F}$ *is the finite set of states;*
- $E_{\mathcal{V}_i^F} = O_i \cup (C_i \times \{\text{left}, \text{right}\})$;
- $q_{0^F_{\mathcal{V}_i}} = (q_{0_{\Delta_i^F}}, q_{0_{\Delta_i^F}})$ *is the initial state;*
- $T_{\mathcal{V}_i^F} \subseteq T_{\Delta_i^F} \times T_{\Delta_i^F}$ *is the set of transitions defined as follows:*

$$(t_l, t_r) \in T \equiv Sync((t_l, t_r), O_i).$$

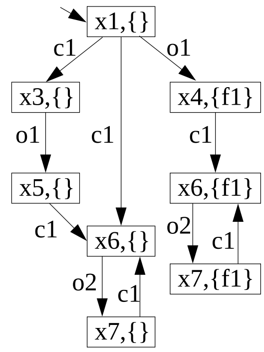

Figure 2. *f1*-diagnoser based on the local model of the component C1.

The F-verifier is obtained by synchronising the observable events of two identical F-diagnoser (a *left* one and a *right* one). It follows that the F-verifier contains non-synchronised communication events from both diagnosers. A communication event c from the left diagnoser is noted (c, left) and a communication event c from the right diagnoser is noted (c, right). Figure 3 shows a part of the *f1*-verifier based on the *f1*-diagnoser from Figure 2. In the figure, a communication event (c, left) is abbreviated by (c, l) and (c, right) is abbreviated by (c, r).

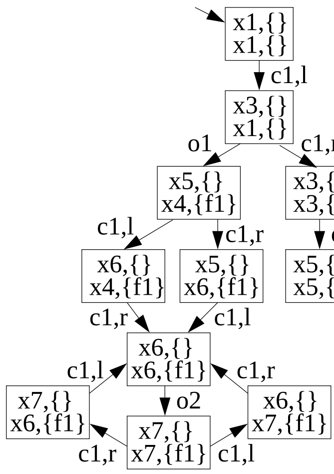

Figure 3. Part of the *f1*-verifi er based on the *f1*-diagnoser.

A transition path of the F-verifier represents the parallel execution (communicating events) of two identical components which produce the same observable sequence regarding the potential occurrence of the fault F. A state $((x_1, f_1), (x_2, f_2))$ of the F-verifier such that $f_1 \neq f_2$ is said to be F-*ambiguous*. Such a state means that there exists at least two different executions of the same component which produce the same observable sequence, one is faulty and the other is not.

Proposition 2 *If the F-verifier $\mathcal{V}_i^F$ has no cycle of F-ambiguous states containing at least an observable event, then F is locally diagnosable with respect to the component where F occurs.*

Proof: This result derives from [5]. If the F-verifier has no cycle of F-ambiguous states with an observable inside, it means that the occurrence of F becomes certain after the occurrence of a finite set of observable events from the component.

If there is such a cycle of F-ambiguous states in the F-verifier, it does not mean that F is not locally diagnosable. Indeed, such a cycle may exist in $\mathcal{V}_i^F$ based on the hypothesis that every communication event in the component Γ_i can effectively occur. In order to be sure that a fault is not locally diagnosable, we need to check if those communication events are valid with respect to the components which share the events. Moreover, a communication event in a component can be the cause of observable events from other components, so such an event can be discriminating and F can be locally diagnosable in a bigger subsystem containing the component where it might occur.

In the following, we present new data structures which help to detect if F is effectively locally diagnosable on a bigger subsystem. We consider Γ_j a component different from Γ_i, where F does not occur (F can only occur on Γ_i).

Definition 9 *The* local diagnoser Δ_j *of the component* Γ_j *is a finite-state machine* $\Delta_j = (Q_{\Delta_j}, E_{\Delta_j}, T_{\Delta_j}, q_{0_{\Delta_j}})$ *where:*

- $Q_{\Delta_j} \subseteq Q_j$ *is the finite set of states;*
- $E_{\Delta_j} = O_j \cup C_j$*;*
- $T_{\Delta_j} \subseteq Q_{\Delta_j} \times E_{\Delta_j} \times Q_{\Delta_j}$ *is the set of transitions;*
- $q_{0_{\Delta_j}} = q_{0j}$ *is the initial state.*

The transitions of T_{Δ_j} are the transitions $q \xrightarrow{e} q'$ reachable from the initial state $q_{0_{\Delta_j}} = q_{0i}$ such that: there exists a transition path $q \xrightarrow{uo_1} q_1 \ldots q_{m-1} \xrightarrow{uo_m} q_n \xrightarrow{e} q'$ in the model Γ_j with $uo_j \notin O_j \cup C_j, \forall j \in \{1, \ldots, m\}$.

This diagnoser is able to follow the state of a component given the sequence of observable and communication events produced by this component. The only difference in the building of Δ_j and Δ_i^F is that Δ_j does not contain any failure information.[3]

As for the F-diagnoser, we use Δ_j to compute the local verifier $\mathcal{V}_j$ defined as follows.

Definition 10 *The* local verifier $\mathcal{V}_j$ *is the automaton* $(Q_{\mathcal{V}_j}, E_{\mathcal{V}_j}, T_{\mathcal{V}_j}, q_{0_{\mathcal{V}_j}})$

- $Q_{\mathcal{V}_j} \subseteq Q_{\Delta_j} \times Q_{\Delta_j}$ *is the finite set of states;*
- $E_{\mathcal{V}_j} = O_j \cup (C_j \times \{\text{left}, \text{right}\})$*;*
- $q_{0_{\mathcal{V}_j}} = (q_{0_{\Delta_j}}, q_{0_{\Delta_j}})$ *is the initial state;*
- $T_{\mathcal{V}_j} \subseteq T_{\Delta_j} \times T_{\Delta_j}$ *is the set of transitions defined as follows:*

$$(t_l, t_r) \in T \equiv Sync((t_l, t_r), O_j).$$

The computation of $\mathcal{V}_j$ based on Δ_j is strictly identical to the computation of $\mathcal{V}_i^F$ given Δ_i^F. A transition path of a local verifier represents the parallel execution (communicating events) of two identical components which produce the same observable sequence.

[3] The presented structure is the minimal one. If we need to have a deeper analysis of the F diagnosability by comparing with the occurrence of other faults and detecting if F is not diagnosable because of another fault, we can compute the diagnoser with fault information. This will result as an increasing of the complexity.

Merge of the verifiers In order to check the diagnosability of F, the idea basically consists in merging the verifier $\mathcal{V}_i^F$ with the verifiers $\mathcal{V}_j, j \in \{1, \ldots, n\} \setminus i$. The merge operation of two verifiers is based on a synchronised product where the synchronisations are on the communication events (the *left* communication events of $\mathcal{V}_i^F$ with the *left* ones of $\mathcal{V}_j$ and the *right* communication events of $\mathcal{V}_i^F$ with the *right* ones of $\mathcal{V}_j$). Given a path of $\mathcal{V}_i^F$ and a path of $\mathcal{V}_j$, the result is a set of paths: each path represents the parallel execution of the subsystem $\{\Gamma_i, \Gamma_j\}$ with itself which produces an identical observable sequence from the subsystem $\{\Gamma_i, \Gamma_j\}$. The result of this merging is a verifier for F in the subsystem $\{\Gamma_i, \Gamma_j\}$. To detect if F is locally diagnosable in this new verifier, it suffices to detect if there is no cycle of ambiguous states which contains at least an observable event from Γ_i and an observable event from Γ_j. If such a cycle does not exist, it means that after a finite number of observations from $\{\Gamma_i, \Gamma_j\}$ (necessarily coming from both Γ_i and Γ_j because of the assumption 1) the verifier is in a non-ambiguous state, so F is locally diagnosable in $\{\Gamma_i, \Gamma_j\}$. We can reiterate the same kind of reasoning to merge another verifier $\mathcal{V}_k$ with the one of $\{\Gamma_i, \Gamma_j\}$ until we get the F verifier for the whole system.

The next section describes an algorithm for checking the local diagnosability of F using this framework. The main idea of the algorithm is to detect if F is locally diagnosable as soon as possible.

5 ALGORITHM

Given the F-verifier of the component Γ_i where F might occur, the first operation is to delete the states and the transitions that are not the cause of a diagnosability problem. The deletion is done with the procedure $DeleteNonAmbiguity$. This procedure consists in keeping only the paths from the initial state of the verifier to cycles of ambiguous states which contain at least an observable event. Then, the algorithm proceeds as follows. We need to check if those cycles are valid or not according to the other components, so we need to validate or invalidate communication events. Given Γ_j a component which communicates with the component Γ_i, its local verifier $\mathcal{V}_j$ is computed. By synchronisation on communicating events ($Synchronise(\mathcal{V}, \mathcal{V}_j, Sync((t, t_j), C))$ (see the above section) between $\mathcal{V}$ and $\mathcal{V}_j$, a new verifier is obtained. The next operation consists in deleting from this new verifier the communication events that are only shared between components inside this new verifier, they are not useful any more because they cannot invalidate more paths in the future, this operation is interesting because it increases the efficiency of future merging by compacting the current verifier. We apply then the deletion of the states and the transitions that are not the cause of a diagnosability problem. This procedure consists here in keeping only the paths from the initial state of the verifier to cycles of ambiguous states which contain at least an observable event from each component from $componentsOf(\mathcal{V})$. The process is reiterate until $\mathcal{V}$ is empty or there is at least a path in $\mathcal{V}$ with only observable events that contains a cycle of ambiguous states ($notDiagnosable$ predicate). Finally, if $\mathcal{V}$ is empty, it means that there is no cycle of ambiguous states (the last call to $DeleteNonAmbiguity$ has deleted every state, no cycle has been found) that show the fault is locally diagnosable inside the subsystem represented by $\mathcal{V}$. If $\mathcal{V}$ is not empty, then $\mathcal{V}$ contains paths with only observable events. In that case F is not locally diagnosable in the current subsystem, and because there is no more way to invalidate those paths, it means that F is not diagnosable in the whole system.

Algorithm 1 Diagnosability analysis

Input: $\{\Gamma_1, \ldots, \Gamma_n\}$ the decentralised model of the system
Input: F a fault event occurring on the component Γ_i
Compute the F-verifier $\mathcal{V}_i$
$\mathcal{V} \leftarrow DeleteNonAmbiguity(\mathcal{V}_i)$
$\mathcal{C} \leftarrow \{\Gamma_1, \ldots, \Gamma_n\} \setminus \{\Gamma_i\}$
while $\mathcal{V} \neq \emptyset \wedge \mathcal{C} \neq \emptyset \wedge notDiagnosable(\mathcal{V})$ **do**
  Take Γ_j from $\mathcal{C}$ such that Γ_j shares communication events with $componentsOf(\mathcal{V})$
  Compute $\mathcal{V}_j$; Let C be the set of communication events (e, s) with $s \in \{\text{left}, \text{right}\}$ shared by $\mathcal{V}$ and $\mathcal{V}_j$
  $\mathcal{V} \leftarrow Synchronise(\mathcal{V}, \mathcal{V}_j, Sync((t, t_j), C))$
  $\mathcal{V} \leftarrow AbstractCommunicationEvents(\mathcal{V})$
  $\mathcal{V} \leftarrow DeleteNonAmbiguity(\mathcal{V})$
end while
if $\mathcal{V} = \emptyset$ **then**
  return "F is diagnosable on $componentsOf(\mathcal{V})$"
else
  $DeletePathsContainingCommunicationEvents(\mathcal{V})$
  return "F is not diagnosable. Every time a diagnoser will see an observable sequence from $componentsOf(\mathcal{V})$ which is in $\mathcal{V}$, F will not be diagnosable".
end if

6 EXAMPLE RESULTS

This section presents the diagnosability analysis of the model presented in Figure 1.

6.1 Analysis of the fault *f1*

The fault *f1* occurs on the component C1. According to the *f1*-verifier (see Figure 3), there are cycles of ambiguous states containing an observable event. As a consequence, a merge of the verifiers is needed. After the merge of the three verifiers, the result is that *f1* is not diagnosable. The algorithm returns the observable language represented in Figure 4 (the doubled circles are the acceptor states of an automaton representing a language). The considered subsystem is the system itself. If the system produces a sequence observation part of this language then *f1* will not be diagnosed as sure, there is an ambiguity.

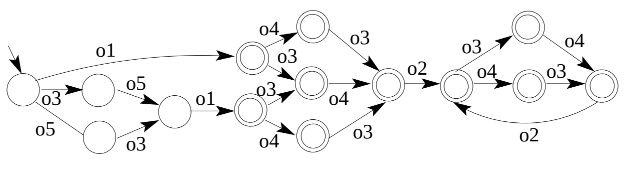

Figure 4. Ambiguous observable language for *f1*.

6.2 Analysis of the fault *f2*

The fault *f2* occurs on the component C2. According to the *f2*-verifier, there are cycles of ambiguous states containing the observable event $o3$. The reason is that the component C2 produces one kind of observable events and *f2* cannot be diagnosed with the only occurrences of *o3*. By merging the *f2*-verifier with the local verifier of C3, the ambiguous cycles disappear. The main reason is that the occurrence of *f2* is detected by the communication event *c2*. In the component C3, *c2* is detected by the occurrence of *o5*. As a consequence *f2* is diagnosed when *o5* occurs. *f2* is therefore locally diagnosable on the subsystem $\{C2, C3\}$, so *f2* is diagnosable.

6.3 Comparison with a centralised approach

This section presents a comparison of the search spaces between the decentralised approach we propose and the centralised approach of [5] for that simple example. In the centralised approach of [5], the global model is needed (it contains 43 states and 76 transitions). Then, the $f1$-diagnoser is computed (without communication events because they are not useful in a centralised approach) which contains 53 states and 100 transitions. And finally, a cycle of ambiguous states is searched in the state space of the $f1$-verifier which contains 99 states and 151 transitions.

Whereas the studied example is very simple, the search spaces of the decentralised approach are clearly smaller. The *f1*-verifier only contains 19 states and 23 transitions and the successive mergings of the *f1*-verifier and the other local verifiers produce transition systems with 25 states and 30 transitions for the first merge and 26 states and 33 transitions for the second one. Due to the fact that we successively merge the verifiers and delete information that are not necessary for the diagnosability checking, our search spaces are smaller than the global model of the system that is the first requirement of any centralised approaches.

This comparison needs to be extended to bigger examples but we think that the decentralised approach will provide promising results and extend the capability for the diagnosability analysis of bigger systems. In particular, we can notice that if a fault is diagnosable, the centralised approach explores the complete F-verifier to detect the diagnosability of the fault whereas the decentralised approach generally finds out a subsystem smaller than the system itself where the fault is diagnosable.

7 RELATED WORKS

The diagnosability definition used in this paper has been introduced in [7]. In order to check the diagnosability of a system, the authors propose a centralised approach based on a global diagnoser. Checking the diagnosability consists in detecting cycles of ambiguous states in this diagnoser. The main problem of this approach is its computation complexity. Firstly, the knowledge of the global model is needed and secondly, the diagnoser computation is exponential to the number of states in the global model.

In [5] and in [11], new algorithms for checking the diagnosability of a system are proposed. These algorithms are based on non-deterministic global diagnosers which allow to make the diagnosability checking with a complexity polynomial in the number of states in the global model of the system (see above).

In [9], the diagnosability is discussed on a system observed by a set of sites. In that framework, a site still knows the global model of the system and is able to only observe a certain type of observable events and is in charge of diagnosing a subset of failure events. A site can exchange messages with the other sites. In this framework, a system is said to be *decentrally* diagnosable if, for every fault, there is at least a site which is able to diagnose this fault without the exchange of messages with the other sites. In our point of view, this notion has a strong relationship with our notion of local diagnosability. In fact, if a site observes a subsystem on which a fault is locally diagnosable then the fault is said to be decentrally diagnosable by the authors of [9].

As a consequence, if we detect for each fault a subsystem on which the fault is locally diagnosable, we can design a set of observation sites that makes the system decentrally diagnosable.

In [3], another diagnosability property is specified: the system is said to be diagnosable if for all relevant observation sequences there is at least one minimal diagnosis [4]. This diagnosability property is checked by the resolution of a process algebra equation. In [10], the same diagnosability property is also analysed by exploiting a set of relations on the system and by analysing their redundancies in order to compute the *diagnosability degree* of the system and to give a way to improve it and to effectiveley get the full diagnosability property. This diagnosability property is more restrictive than the one of [7], it supposes that the set of observations is completely known when the diagnosis reasoning is applied and is not well-suited for monitored systems that are diagnosed on-line.

In [2], another diagnosability property, similar to the one of [3], is written as a problem of model-checking. In this approach, the system is not diagnosable if there exist two behaviours with the same observable signature such that the system is not in the same behavioural mode (a fault occurs in one behaviour but not in the other one) at the end of the execution of each behaviour separately. The main idea of this paper is to use the efficiency of symbolic model-checking tools to check diagnosability properties. This is a possible way for us to symbolically implement our algorithm, by mixing the efficiency of a decentralised approach and the efficiency of model-checking tools.

8 CONCLUSION

This paper presents a framework for the diagnosability analysis of distributed discrete event systems. Based on the diagnosability definition of [7], the diagnosability checking is done thanks to a set of local verifiers and without the need of a global model of the system. The main purpose of this framework is to allow the analysis of bigger systems whose global model is not implementable due to the huge number of states in it. Secondly, in practice, it is very unprobable that a system is diagnosable if it has been developed without the help of a diagnosability analysis. The proposed approach focuses then on the result to provide in the case that the system is not diagnosable. Our diagnosability analysis provides more concise information about why the system is not diagnosable by giving an observation language for which a fault is not diagnosable. This concision is due to the fact that we analyse the diagnosability of a fault in a local manner and extend the analysis to its neighbourhood until we find a set of scenarios that are not diagnosable.

In that framework, we introduce the notion of local diagnosability. This diagnosability is based on the fact that the system respects the hypothesis 1. Nevertheless, the proposed framework still works to check diagnosability on a system which does not respect this hypothesis. The difference is that we cannot stop the search if we find a verifier where F is considered as locally diagnosable because the local diagnosability of F does not imply the diagnosability of F in that case. The diagnosability check can still be done by merging local verifiers, only the criteria to stop the search are different.

The efficiency of the method has not been developed yet. When dealing with decentralised approach, one main key point is to apply a merging strategy, a reconstruction plan. This can be done in the same way it is done in some diagnosis methods as proposed in [1] [12]. Another way to improve the efficiency of the approach is to use symbolic techniques [8] or symbolic off-the-shelf model-checkers [2].

Two main perspectives could extend this work. The first one is the use of such an approach to help in the synthesis of a diagnosable distributed DES. Given local non diagnosable scenarios, it is easier to modify the system such that the non diagnosable scenario does not occur. Moreover, we think that it is possible to provide an automatic information to fix the diagnosability problem for one given found scenario. The second perspective is about the diagnosis of the DES itself. Given a language of observations which is considered as ambiguous for the diagnosis of one failure, we know that if we observe, at a given time, a sequence of observations that belongs to the language, we have the guarantee that the failure cannot be discriminated and an on-line abstraction of the model is then possible, abstraction which increases the efficiency of the on-line diagnosis of the DES.

REFERENCES

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

[2] A. Cimatti, C. Pecheur, and R. Cavada, 'Formal verification of diagnosability via symbolic model checking', in *Proceedings of the 18th International Joint Conference on Artificial Intelligence IJCAI'03*.

[3] L. Console, C. Picardi, and M. Ribaudo, 'Diagnosis and diagnosibility analysis using pepa', in *Proceedings of the European Conference on Artificial Intelligence (ECAI'2000)*, pp. 131–135, Berlin, Germany, (2000).

[4] L. Console and P. Torasso, 'A spectrum of logical definitions of model-based diagnosis', *Computational Intelligence*, **7**(3), 133–141, (1991).

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

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

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

[8] A. Schumann, *Is symbolic technology applicable to the diagnosis of discrete event systems?*, Master's thesis, Faculty of Business, Administration, Economics and Social Sciences, University of Rostock, Germany, 2003.

[9] R. Sengupta, 'Diagnosis and communication in distributed systems', in *Proceedings of the Workshop on Discrete Event Systems (WODES'98)*, pp. 144–151, Cagliari, Italy, (August 1998).

[10] L. Travé-Massuyès, T. Escobet, and R. Milne, 'Model-based diagnosability and sensor placement application to a frame 6 gas turbine subsystem', in *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI'01*, volume 1, pp. 551–556, (2001).

[11] T. Yoo and S. Lafortune, 'Polynomial-time verification of diagnosability of partially-observed discrete-event systems', *IEEE Transactions of Automatic Control*, **47**(9), 1491–1495, (2002).

[12] Y.Pencolé, M.-O. Cordier, and L. Rozé, 'A decentralized model-based diagnostic tool for complex systems', *International Journal on Artificial Intelligence Tools*, **11**(3), 327–346, (2002).