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, 10] 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, 11]. 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. In that respect, we introduce the notion of *local diagnosability* defined on a component and its neighbourhood in the distributed system and define a relationship with the global diagnosability of the system. With the help of the local diagnosability, we propose to analyse the diagnosability of the system in a decentralised way. The main idea consists in checking the diagnosability locally with the help of a *local verifier*. If no conclusion can be provided, the local verifier is successively merged with the verifiers of the neighbourhood until we can conclude. The second objective of this analysis is about the conclusion in itself: instead of providing a yes/no answer, we want to provide accurate reasons in the case where the system is 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 with each other by exchanging events. Such a discrete event system can be modelled as a set of automata, each automaton representing the model of a component, i.e. a *local model*. The formalism presented below is equivalent to the one of [7].

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 three disjoint sets.

1. F_i are the *faulty* events. A fault $f_i \in F_i$ can only occur in Γ_i.
2. O_i are the *observable* events. An event $o_i \in O_i$ can only occur in Γ_i.
3. C_i are the *communication* events.

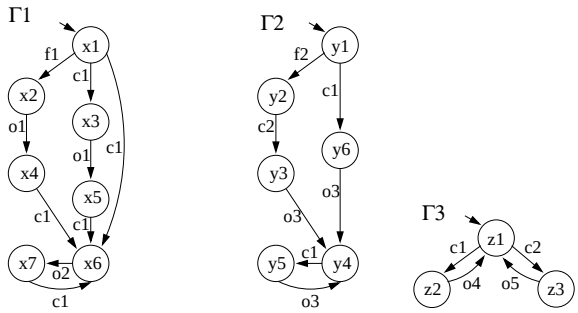

Figure 1. Model of a system defined by three local models

Figure 1 presents a system composed of three local models defined as above. The events f_i are the fault events, the events c_i are the communication events and the events o_i are the observable events.

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 component 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 m-uple of transitions and $\mathcal{E}$ a set of *synchronised* events (s.t. $\epsilon \notin \mathcal{E}$), the predicate $Sync((t_1, \ldots, t_m), \mathcal{E})$ defines the synchronised conditions on $(t_1, \ldots, t_m)$ as follows. Let $ev(t)$ be the event labelling t:

$$Sync((t_1, \ldots, t_m), \mathcal{E}) \equiv (\exists j \in \{1, \ldots, m\} :$$
$$ev(t_j) \notin \mathcal{E} \wedge \forall i \in \{1, \ldots, m\} \setminus \{j\} : ev(t_i) = \epsilon)$$
$$\vee((\exists j \in \{1, \ldots, m\} : ev(t_j) \in \mathcal{E}) \wedge (\forall i \in \{1, \ldots, m\} :$$
$$(ev(t_i) \in \mathcal{E} \Rightarrow ev(t_i) = ev(t_j)) \wedge (ev(t_i) \notin \mathcal{E} \Rightarrow ev(t_i) = \epsilon))).$$

Roughly speaking, $Sync((t_1, \ldots, t_m), \mathcal{E})$ defines a synchronous behaviour for $t_1, \ldots, t_m$ for every event of $\mathcal{E}$ and an asynchronous behaviour otherwise.

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:*

- $Q \subseteq Q_{i_1} \times \cdots \times Q_{i_k}$; $E = \bigcup_{j=i_1}^{i_k} E_j$;
- $q_0 = (q_{0i_1} \times \cdots \times q_{0i_k})$ *is the initial state;*
- $T \subseteq T_{i_1} \times \cdots \times T_{i_k}$ *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. 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 assumption means first that from any state of a component, it will emit observations in the future (the observable language of the component is live) in all executions of the system 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 assumption has two purposes. Firstly, even if a component is modelled with silent cycles (for compactness reasons), the assumption states that the probability that during one execution the system stays in such a silent cycle is null. Secondly, the assumption states that a component will only emit a finite sequence of observable events between the occurrence of two observable events from other components even if the component is modelled with cycles of observable events.

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 check is similar to the one of [7].

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.*

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 f is diagnosable.*

3.2 Diagnosability for distributed DES

This section presents the diagnosability property of a system taking into 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' 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 assumption 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

In 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:*

- $Q_{\Delta_i^F} \subseteq Q_i \times \{\{F\}, \emptyset\}$; $E_{\Delta_i^F} = O_i \cup C_i$;
- $T_{\Delta_i^F} \subseteq Q_{\Delta_i^F} \times E_{\Delta_i^F} \times Q_{\Delta_i^F}$ *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 Γ_1 (see Figure 1).

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

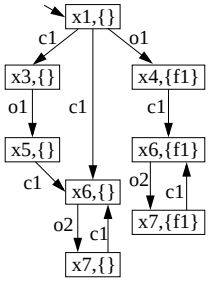

Figure 2. *f1*-diagnoser based on the local model of the component C1.

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

- $Q_{\mathcal{V}_i^F} \subseteq Q_{\Delta_i^F} \times Q_{\Delta_i^F}$; $E_{\mathcal{V}_i^F} = O_i \cup (C_i \times \{\text{left}, \text{right}\})$;
- $q_{0_{\mathcal{V}_i}^F} = (q_{0_{\Delta_i^F}}, q_{0_{\Delta_i^F}})$ *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:* $(t_l, t_r) \in T_{\mathcal{V}_i^F} \equiv Sync((t_l, t_r), O_i)$.

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).

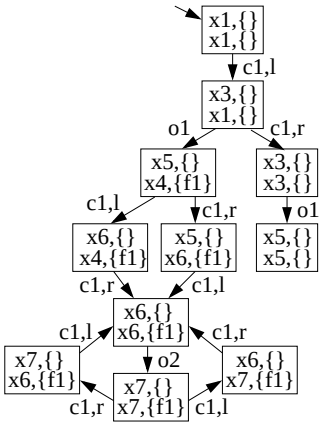

Figure 3. Part of the *f1*-verifier based on the *f1*-diagnoser.

A transition path of the F-verifier represents two possible executions (communicating events) of the same component 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. Such a cycle may exist in $\mathcal{V}_i^F$ based on the assumption 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:*

- $Q_{\Delta_j} \subseteq Q_j$; $E_{\Delta_j} = O_j \cup C_j$;
- $T_{\Delta_j} \subseteq Q_{\Delta_j} \times E_{\Delta_j} \times Q_{\Delta_j}$; $q_{0_{\Delta_j}} = q_{0j}$.

The transitions of T_{Δ_j} are the transitions $q \xrightarrow{e} q'$ reachable from the initial state $q_{0_{\Delta_j}} = q_{0j}$ 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_j with $uo_j \notin O_j \cup C_j, \forall j \in \{1, \dots, m\}$.

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.[2]

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:* $(t_l, t_r) \in T_{\mathcal{V}_j} \equiv Sync((t_l, t_r), O_j)$.

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 two possible executions (communicating events) of the same component which produce the same observable sequence.

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 $\mathcal{V}_j, j \in \{1, \ldots, n\} \setminus \{i\}$ to check the existence of ambiguous cycles. 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 two possible executions of the subsystem $\{\Gamma_i, \Gamma_j\}$ which produce 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 ($AbstractCommunicationEvents$) 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 $compsOf(\mathcal{V})$. 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) which implies that 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 $\mathcal{V}_i^F$
$\mathcal{V} \leftarrow DeleteNonAmbiguity(\mathcal{V}_i^F)$
$\mathcal{M} \leftarrow \{\Gamma_1, \ldots, \Gamma_n\} \setminus \{\Gamma_i\}$
while $\mathcal{V} \neq \emptyset \wedge \mathcal{M} \neq \emptyset \wedge notDiagnosable(\mathcal{V})$ **do**
  Take Γ_j from $\mathcal{M}$ such that Γ_j shares events with $compsOf(\mathcal{V})$
  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 $compsOf(\mathcal{V})$"
else
  $DeletePathsContainingCommunicationEvents(\mathcal{V})$
  return "F is not diagnosable." and **return** $\mathcal{V}$.
end if

6 EXPLANATORY EXAMPLE

This section presents the diagnosability analysis of the model presented in Figure 1. The fault *f1* occurs on Γ_1. According to the *f1*-verifier (see Figure 3), there are cycles of ambiguous states containing an observable event. 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. *f1* cannot be discriminated from the case where the system is not faulty: a sequence containing an infinite number of *o4o3o2* sequences can occur in both cases.

[2] 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 increase of the complexity.

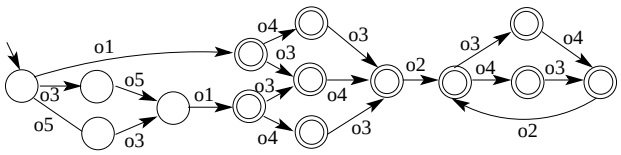

Figure 4. Ambiguous observable language for *f1*.

The fault *f2* occurs on Γ_2. According to the *f2*-verifier, there are cycles of ambiguous states containing the observable event $o3$. By merging the *f2*-verifier with the local verifier of Γ_3, the ambiguous cycles disappear. The main reason is that the occurrence of *f2* is detected by the communication event *c2*. In Γ_3, *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 $\{\Gamma_2, \Gamma_3\}$, so *f2* 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 computational 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. [5] and [10] propose new algorithms based on non-deterministic global diagnosers with a complexity polynomial in the number of states in the global model.

In [9, 4], 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 faults. 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].

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. This diagnosability property is checked by the resolution of a process algebra equation. 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. 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. This framework allows the analysis of bigger systems whose global model is not implementable due to the huge number of states in it. Secondly, the proposed approach focuses on the information to provide if the system is not diagnosable. Our diagnosability analysis provides more concise reasons 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 in a local manner and extend the analysis to its neighbourhood until we find a set of scenarios that are not diagnosable.

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] [11]. 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. 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.

ACKNOWLEDGEMENTS

The author would like to thank the anonymous reviewers and Anika Schumann for their useful advises and comments about this paper.

REFERENCES

[1] P. Baroni, G. Lamperti, P. Pogliano, and M. Zanella, '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, Germany, (2000).
[4] 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), (2002).
[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, Y. Pencolé, and S. Thiébaux, 'Diagnosis of discrete-event systems using binary decision diagrams', in *Proceedings of the International Workshop on Principles of Diagnosis (DX'04)*, (2004).
[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, (1998).
[10] 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).
[11] 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).