How to Be Sure a Faulty System Does Not Always Appear Healthy?

Lina Ye[1(✉)], Philippe Dague[2], Delphine Longuet[2], Laura Brandán Briones[3], and Agnes Madalinski[4]

[1] LRI, Univ. Paris-Sud, CentraleSupélec, Univ. Paris-Saclay, Orsay, France
lina.ye@lri.fr
[2] LRI, Univ. Paris-Sud, CNRS, Univ. Paris-Saclay, Orsay, France
{philippe.dague,delphine.longuet}@lri.fr
[3] Universidad Nacional de Córdoba, Córdoba, Argentina
[4] Otto-von-Guericke-University Magdeburg, Magdeburg, Germany

Abstract. Fault diagnosis is a crucial and challenging task in the automatic control of complex systems, whose efficiency depends on the diagnosability property of a system. Diagnosability describes the system property allowing one to determine with certainty whether a given fault has effectively occurred based on the available observations. However, this is a quite strong property that generally requires a high number of sensors. Consequently, it is not rare that developing a diagnosable system is too expensive. In this paper, we analyze a new discrete event system property called manifestability, that represents the weakest requirement on observations for having a chance to identify on line fault occurrences and can be verified at design stage. Intuitively, this property makes sure that a faulty system cannot always appear healthy, i.e., has at least one future behavior after fault occurrence observably distinguishable from all normal behaviors. Then, we prove that manifestability is a weaker property than diagnosability before proposing an algorithm with PSPACE complexity to automatically verify both properties. Furthermore, we prove that the problem of manifestability verification itself is PSPACE-complete. The experimental results show the feasibility of our algorithm from a practical point of view. Finally, we compare our approach with related work.

1 Introduction

Fault diagnosis is a crucial and challenging task in the automatic control of complex systems, whose efficiency depends on a system property called diagnosability. Diagnosability is a system property describing whether one can distinguish with certainty fault behaviors from normal ones based on sequences of observable events emitted from the system. In a given system, the existence of two infinite behaviors with the same observations, where exactly one contains the considered fault, violates diagnosability. The existing work concerning discrete event systems (DESs) searches for such ambiguous behaviors, both in centralized

M. F. Atig et al. (Eds.): VECoS 2018, LNCS 11181, pp. 114–129, 2018.
https://doi.org/10.1007/978-3-030-00359-3_8

and distributed ways [10,12–14,20]. However, in reality, diagnosability turns out to be a quite strong property that generally requires a high number of sensors. Consequently, it is often too expensive to develop a diagnosable system.

To achieve a trade-off between the cost, i.e., a reasonable number of sensors, and the possibility to observe a fault manifestation, we recently introduced a new property called manifestability [21], which is borrowed from philosophy "...which I shall call the "manifestability of the mental", that if two systems are mentally different, then there must be some physical contexts in which this difference will display itself in differential physical consequences" [11]. In the domain of diagnosis, similarly, the manifestability property describes the capability of a system to manifest a fault occurrence in at least one future behavior. This should be analyzed at design stage on the system model. Under the assumption that no behavior described in the model has zero probability, the fault will then necessarily show itself with nonzero probability after enough runs of the system. In other words, given a system, if this property holds, this system cannot always appear healthy when a fault occurs in it, i.e., at least one future behavior observably distinguishes from normal behaviors. In all cases, manifestability is the weakest property to require from the system to have a chance to identify the fault occurrence. Differently, for diagnosability, all future behaviors of all fault occurrences should be distinguishable from all normal behaviors, which is a strong property and sensor demanding. Obviously one has to continue to rely on diagnosability for online safety requirements, i.e., for those faults which may have dramatic consequences if they are not surely detected when they occur, in order to trigger corrective actions. But for all other faults that do not need to be detected at their first occurrence (e.g., whose consequence is a degraded but acceptable functioning that will require maintenance actions in some near future), manifestability checking, which is cheaper in terms of sensors needed, is enough under the probabilistic assumption above.

We have several contributions in this paper. First, we define (strong) manifestability before proving that it is weaker than diagnosability. Second, we provide a sufficient and necessary condition for manifestability with a formal algorithm based on equivalence checking and prove that the manifestability problem itself is a PSPACE-complete problem. Third, the algorithm's efficiency is shown by our experimental results before comparing our approach with related work.

2 Motivating Example

In this section, we explain why it is worth analyzing the manifestability property with a motivating example.

Example 1. Figure 1 shows a modified version of a HVAC system from [13], which is a composite model that captures the interactions between the component models, i.e., a pump, a valve, and a controller. In this system, the initial state is q^0, the events $Valve\_open$, $Pump\_start$, $Pump\_stop$, $Valve\_close$ are observable and the fault event $Pump\_failed$ is not observable. Once fault event occurs, the system enters and always stays in an abnormal state.

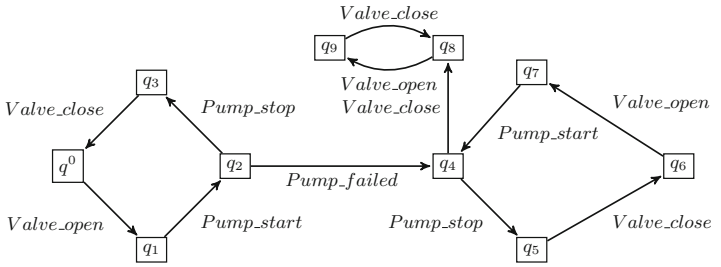

Fig. 1. A simplified HVAC system.

The correct behavior of this system is $(Valve\_open\ Pump\_start\ Pump\_stop\ Valve\_close)^{\omega}$, where ω denotes the infinite concatenation. After the unobservable faulty event $Pump\_failed$, the system has two possibilities: either continue the execution with the same observations as the correct behavior or go to the states q_8 and q_9. Thus, this system is not diagnosable since at least one infinite future behavior of the fault occurrence is indistinguishable from the correct behavior, which is $Valve\_open\ Pump\_start\ Pump\_failed\ (Pump\_stop\ Valve\_close\ Valve\_open\ Pump\_start)^{\omega}$. Considering real faulty scenarios, with an assumption of nonzero probability, at one moment in the future the system will go to q_8, in which case the fault manifests itself and thus can be diagnosed. The original diagnosability property is not suitable to handle such situations. If we consider manifestability, this fault is effectively manifestable since its occurrence has at least one future that is distinguishable from the correct behavior. The manifestability property is the minimal requirement for the system to allow one to establish a diagnostic mechanism. If a fault is not manifestable, then it is totally useless to try to design a diagnoser for the system.

3 Manifestability for DESs

We now present our system model, recall diagnosability, and introduce (strong) manifestability, before giving a formal sufficient and necessary condition for this property to hold. We demonstrate that (strong) manifestability is a weaker property than diagnosability.

3.1 Models of DESs

We model a DES as a Finite State Machine (FSM), i.e., an automaton, denoted by $G = (Q, \Sigma, \delta, q^0)$, where Q is the finite set of states, Σ is the finite set of events, $\delta \subseteq Q \times \Sigma \times Q$ is the set of transitions (the same notation will be kept for its natural extension to words of Σ^*), and q^0 is the initial state. The set of events Σ is divided into three disjoint parts: $\Sigma = \Sigma_o \uplus \Sigma_u \uplus \Sigma_f$, where Σ_o is the set of observable events, Σ_u the set of unobservable normal events and Σ_f the set of unobservable fault events.

Example 2. The left part of Fig. 2 shows an example of a system model G, where $\Sigma_o = \{o1, o2, o3\}$, $\Sigma_u = \{u1, u2\}$, and $\Sigma_f = \{F\}$. Notice that for diagnosis problem, fault is predefined as an unobservable event in the model. This is different from testing, where faulty behaviors are judged against a specification.

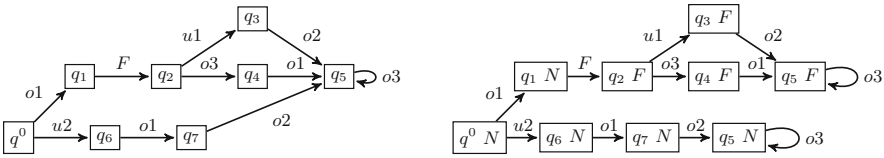

Fig. 2. A system example (left) and its diagnoser (right).

Similar to diagnosability, the manifestability algorithm that we will propose has exponential complexity in the number of fault types. To reduce it to linear complexity, as in [12,14], we consider only one fault type at a time. However, multiple occurrences of faults are allowed. The other types of faults are processed as unobservable normal events. This is justified as the system is manifestable if and only if (iff) it is manifestable for each fault type. Thus, to check the manifestability of a system with several faults, one can check its manifestability with respect to each fault type in turn. In the following, $\Sigma_f = \{F\}$, where F is the currently considered fault.

Given a system model G, its prefix-closed language $L(G)$, which describes both normal and faulty behaviors of the system, is the set of words produced by G: $L(G) = \{s \in \Sigma^* | \exists q \in Q, (q^0, s, q) \in \delta\}$. Those words containing (resp. not containing) F will be denoted by $L_F(G)$ (resp. $L_N(G)$). In the following, we call a word from $L(G)$ a trajectory in the system G and a sequence $q_0\sigma_0 q_1\sigma_1 \ldots$ a path in G, where $q_0 = q^0$ and, for all i, $(q_i, \sigma_i, q_{i+1}) \in \delta$, whose label $\sigma_0\sigma_1 \ldots$ is a trajectory in G. Given $s \in L(G)$, we denote the post-language of $L(G)$ after s by $L(G)/s$, formally defined as: $L(G)/s = \{t \in \Sigma^* | s.t \in L(G)\}$. The projection of the trajectory s to observable events of G is denoted by $P(s)$, the observation of s. This projection can be extended to $L(G)$, i.e., $P(L(G)) = \{P(s) | s \in L(G)\}$, whose elements are called observed trajectories. Traditionally, we assume that each state of Q has a successor, so that $L(G)$ is live (any trajectory has a continuation, i.e., is a strict prefix of another trajectory) and that G has no unobservable cycle, i.e., each cycle contains at least one observable event. This makes it feasible to check the infiniteness of a trajectory. We will need some infinite objects. We denote by Σ^ω the set of infinite words on Σ and by $\Sigma^\infty = \Sigma^* \cup \Sigma^\omega$ the set of words on Σ, finite or infinite. We define in an obvious way infinite paths in G and thus $L^\omega(G)$ the language of infinite words recognized by G in the sense of Büchi automata [6]. As all states of G are considered as final states, those infinite trajectories are just the labels of infinite paths, and the concept of Büchi automaton coincides with that of Muller automaton, which can be determinized, according to the McNaughton theorem. We can conclude from this that $L^\omega(G)$

is the set of infinite words whose prefixes belong to $L(G)$ and that two equivalent system models, i.e., such that $L(G_1) = L(G_2)$, define the same infinite trajectories, i.e., $L^\omega(G_1) = L^\omega(G_2)$. Particularly, we use $L_F^\omega(G) = L^\omega(G) \cap \Sigma^* F \Sigma^\omega$ for the set of infinite faulty trajectories, and $L_N^\omega(G) = L^\omega(G) \cap (\Sigma \setminus \{F\})^\omega$ for the set of infinite normal trajectories, where $\setminus$ denotes set subtraction. We denote $L^\infty(G) = L(G) \cup L^\omega(G)$. In the following, we use the classical synchronization operation between two FSMs G_1 and G_2, denoted by $G_1 \parallel_{\Sigma_s} G_2$, i.e. any event in Σ_s should be synchronized while others can occur whenever possible. It is easy to generalize the synchronization to a set of FSMs using its associativity property [7]. To verify manifestability, we define the following basic operation, which is to keep only information about a given set of events, while keeping the same structure. It will be used to simplify some intermediate structures when checking manifestability without affecting the validity of the result obtained.

Definition 1 *(Delay Closure). Given a FSM $G = (Q, \Sigma, \delta, q^0)$, its delay closure with respect to Σ_d, with $\Sigma_d \subseteq \Sigma$, is $\mathsf{C}_{\Sigma_d}(G) = (Q_d, \Sigma_d, \delta_d, q^0)$, where: (1) $Q_d = \{q^0\} \cup \{q \in Q \mid \exists s \in \Sigma^*, \exists \sigma \in \Sigma_d, (q^0, s\sigma, q) \in \delta\}$; (2) $(q, \sigma, q') \in \delta_d$ if $\sigma \in \Sigma_d$ and $\exists s \in (\Sigma \setminus \Sigma_d)^*, (q, s\sigma, q') \in \delta$.*

3.2 Diagnosability and Manifestability

A fault F is diagnosable in a system model G if it can be detected with certainty when enough events are observed from G after its occurrence. This property is formally defined as follows [13], where s^F denotes a trajectory ending with F and $F \in p$, for p a trajectory, means that F appears as a letter of p.

Definition 2 ***(Diagnosability)**. F is diagnosable in a system model G iff*

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

The above definition states that F is diagnosable iff, for each trajectory s^F in G, for each of its extensions t with enough events, then every trajectory p in G that has the same observations as $s^F t$ should contain F. It has been proved that the existence of two indistinguishable infinite trajectories, i.e., holding the same sequence of observable events, with exactly one of them containing the given fault F, is equivalent to the violation of the diagnosability property [10].

Definition 3 ***(Critical Pair)**. A pair of trajectories s, s' is called a critical pair with respect to F, denoted by $s \nsim s'$, iff $s \in L_F^\omega(G), s' \in L_N^\omega(G)$ and $P(s) = P(s')$.*

Theorem 1. *A fault F is diagnosable in G iff $\nexists s, s' \in L^\omega(G)$, such that $s \nsim s'$.*

The nonexistence of a critical pair w.r.t. F witnesses diagnosability of F. To design a diagnosable system, each faulty trajectory should be distinguished from normal trajectories, which is often very expensive in terms of number of sensors

required. To reduce such a cost and still make it possible to show the fault after enough runs of the system, another property called manifestability has been recently introduced [21], which is much weaker than diagnosability. Intuitively, manifestability describes whether or not a fault occurrence has the possibility to manifest itself through observations. Precisely, if a fault is not manifestable, then we can never be sure about its occurrence no matter which trajectory is executed after it. Thus, the system model should be necessarily revised.

Definition 4 ***(Manifestability)***. *F is manifestable in a system model G iff*

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

F is manifestable iff there exists at least one trajectory s^F in G, and there exists at least one extension t of s^F, such that every trajectory p that is observable equivalent to $s^F t$ should contain F. In other words, manifestability is violated iff each occurrence of the fault can never manifest itself in any future.

Theorem 2. *A fault F is manifestable in a system model G iff the following condition, denoted by $\Im$, is satisfied:*

$$\exists s \in L_F^\omega(G), \nexists s' \in L_N^\omega(G), \textit{such that } s \nsim s'.$$

Proof. $\Rightarrow$ Suppose that F is manifestable in G. Thus from Definition 4, $\exists s \in L_F(G)$ such that $\nexists s' \in L_N(G)$ with $P(s) = P(s')$. By extending s with enough events, which is possible since the language is live, we obtain then $\exists s \in L_F^\omega(G)$, $\nexists s' \in L_N^\omega(G)$, such that $s \nsim s'$.
$\Leftarrow$ Suppose now that F is not manifestable in G and show that the condition $\Im$ is consequently not true. From non-manifestability of F and Definition 4, we have $\forall s^F \in L(G), \forall t \in L(G)/s^F$, $\exists p \in L(G), P(p) = P(s^F t)$, $p \in L_N(G)$. Thus, $\forall s^F t \in L_F(G)$, $\exists p \in L_N(G), P(p) = P(s^F t)$. This can be formulated as equality of the languages of two automata, as it will be seen in Sect. 4. It results that this equality of the languages still holds for infinite words, i.e., $\forall s^F t \in L_F^\omega(G)$, $\exists p \in L_N^\omega(G)$ such that $s^F t \nsim p$, which is $\neg\Im$, i.e., the condition $\Im$ is not true. ■

Manifestability concerns the possibility for the system to manifest at least one occurrence of the fault, i.e., there exists such an occurrence that shows itself in at least one of its futures. Now we propose a strong version of manifestability, which requires that all occurrences of the fault should show themselves in at least one of their futures.

Definition 5 ***(Strong Manifestability)***. *A fault F is strongly manifestable in a system model G iff*

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

F is strongly manifestable iff, for each s^F in G (and not just for only one as in Definition 4) there exists at least one extension t of s^F in G, such that every trajectory p in G that is observable equivalent to $s^F t$ should contain F. Precisely, each occurrence of F should show itself in at least one of its futures. So, in a similar way as Theorem 2, we can prove the following theorem, which provides a sufficient and necessary condition for strong manifestability.

Theorem 3. *A fault F is strongly manifestable in a system model G iff the following condition, denoted by $\Im^s$, is satisfied:*

$$\forall s^F \in L(G), \exists t \in L^\omega(G)/s^F, \nexists s' \in L_N^\omega(G), \textit{such that } s^F t \nsim s'.$$

Theorem 4. *Given a system model G and a fault F, we have:*

1. *F is diagnosable in G implies that F is strongly manifestable in G.*
2. *F is strongly manifestable in G implies that F is manifestable in G.*

Proof. 1. Suppose that F is not strongly manifestable, then from Theorem 3, we have $\neg\Im^s$, i.e., $\exists s^F \in L(G), \forall t \in L^\omega(G)/s^F, \exists s' \in L_N^\omega(G)$ such that $s^F t \nsim s'$. This implies that there does exist at least one critical pair in the system. From Theorem 1, F is not diagnosable.

2. Suppose that F is not manifestable. From Theorem 2, we have $\forall s \in L_F^\omega(G)$, $\exists s' \in L_N^\omega(G)$, such that $s \nsim s'$. By choosing arbitrarily one $s^F \in L(G)$ and taking all s of prefix s^F, we obtain $\exists s^F \in L(G), \forall t \in L^\omega(G)/s^F, \exists s' \in L_N^\omega(G)$ such that $s^F t \nsim s'$, i.e., $\neg\Im^s$. Hence F is not strongly manifestable. ■

4 Manifestability Verification

Manifestability verification consists in checking whether the condition $\Im$ in Theorem 2 is satisfied for a given system model. In this section, we show how to construct different structures based on a system model to obtain $L_F^\omega(G)$, $L_N^\omega(G)$ as well as the set of critical pairs. The condition $\Im$ can then be checked by using equivalence techniques with these intermediate structures. Precisely, if for each infinite faulty trajectory $s \in L_F^\omega(G)$, there exists a corresponding critical pair, then the considered fault is not manifestable. Otherwise, it is manifestable. For the sake of simplicity, we concentrate on how to check manifestability, which can be extended in a straightforward way to handle strong manifestability. This extension will be explained explicitly in Sect. 4.3.

4.1 System Diagnosers

Given a system model, the first step is to construct a structure showing fault information for each state, i.e., whether the fault has effectively occurred up to this state from the initial state.

Definition 6 *(Diagnoser). Given a system model G, its diagnoser with respect to a considered fault F is the FSM $D_G = (Q_D, \Sigma_D, \delta_D, q_D^0)$, where: (1) $Q_D \subseteq Q \times \{N, F\}$ is the set of states; (2) $\Sigma_D = \Sigma$ is the set of events; (3) $\delta_D \subseteq Q_D \times \Sigma_D \times Q_D$ is the set of transitions; (4) $q_D^0 = (q^0, N)$ is the initial state. The transitions of δ_D are those $((q, \ell), e, (q', \ell'))$, with (q, ℓ) reachable from q_D^0, such that there is a transition $(q, e, q') \in \delta$, and $\ell' = F$ if $\ell = F \vee e = F$, otherwise $\ell' = N$.*

The right part of Fig. 2 shows the diagnoser for the system depicted in the left part, where each state has its own fault information. Precisely, given a system state q, if the fault has occurred on the path from q^0 to q, then the fault label for q is F. Such a state is called fault (diagnoser) state. Otherwise, the fault label is N and the state is called normal (diagnoser) state. Diagnoser construction keeps the same set of trajectories and splits into two those states reachable by both a faulty and a normal path ($q5$ in the example).

Lemma 1. *Given a system model G and its corresponding diagnoser D_G, then we have $L(G) = L(D_G)$ and $L^\omega(G) = L^\omega(D_G)$.*

In order to simplify the automata handled, the idea is to keep only the minimal subparts of D_G containing all faulty (resp., normal) trajectories.

Definition 7 *(Fault (Refined) Diagnoser). Given a diagnoser D_G, its fault diagnoser is the FSM $D_G^F = (Q_{D^F}, \Sigma_{D^F}, \delta_{D^F}, q_{D^F}^0)$, where: (1) $q_{D^F}^0 = q_D^0$; (2) $Q_{D^F} = \{q_D \in Q_D \mid \exists q'_D = (q, F) \in Q_D, \exists s' \in \Sigma_D^*, (q_D, s', q'_D) \in \delta_D^*\}$; (3) $\delta_{D^F} = \{(q_D^1, \sigma, q_D^2) \in \delta_D \mid q_D^2 \in Q_{D^F}\}$; (4) $\Sigma_{D^F} = \{\sigma \in \Sigma_D \mid \exists (q_D^1, \sigma, q_D^2) \in \delta_{D^F}\}$. The fault refined diagnoser is obtained by performing the delay closure with respect to the set of observable events Σ_o on the fault diagnoser: $D_G^{FR} = \complement_{\Sigma_o}(D_G^F)$.*

The fault diagnoser keeps all fault states as well as all transitions and intermediate normal states on paths from q_D^0 to any fault state. Then we refine this fault diagnoser by only keeping the observable information, which is sufficient to obtain the set of critical pairs. The left (resp. right) part of Fig. 3 shows the fault diagnoser (resp. fault refined diagnoser) for Example 2.

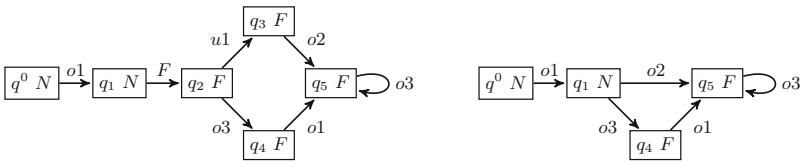

Fig. 3. Fault diagnoser (left) and its refined version (right) for Example 2.

By construction, the sets of faulty trajectories in D_G^F and in G are equal and this is still true for infinite faulty trajectories. This is also the case for infinite faulty trajectories in D_G^{FR} and infinite observed faulty trajectories in G. But

take care that it may exist infinite normal trajectories in D_G^F (resp., D_G^{FR}) if it exists in G a normal cycle in a path to a fault state (e.g., adding a loop in state q_1 of the system model of Example 2).

Lemma 2. *Given a system model G and its corresponding fault diagnoser D_G^F and fault refined diagnoser D_G^{FR}, we have $L_F^\omega(G) = L_F^\omega(D_G^F)$ and $P(L_F^\omega(G)) = L_F^\omega(D_G^{FR})$.*

Similarly, we obtain the subpart of D_G containing only normal trajectories.

Definition 8 *(Normal (Refined) Diagnoser). Given a diagnoser D_G, its normal diagnoser is the FSM $D_G^N = (Q_{D^N}, \Sigma_{D^N}, \delta_{D^N}, q_{D^N}^0)$, where: (1) $q_{D^N}^0 = q_D^0$; (2) $Q_{D^N} = \{(q, N) \in Q_D\}$; (3) $\delta_{D^N} = \{(q_D^1, \sigma, q_D^2) \in \delta_D \mid q_D^2 \in Q_{D^N}\}$; (4) $\Sigma_{D^N} = \{\sigma \in \Sigma_D \mid \exists (q_D^1, \sigma, q_D^2) \in \delta_{D^N}\}$. The normal refined diagnoser is obtained by performing the delay closure with respect to Σ_o on the normal diagnoser: $D_G^{NR} = \complement_{\Sigma_o}(D_G^N)$.*

Lemma 3. *Given a system model G and its corresponding normal diagnoser D_G^N and normal refined diagnoser D_G^{NR}, we have $L_N^\omega(G) = L^\omega(D_G^N)$ and $P(L_N^\omega(G)) = L^\omega(D_G^{NR})$.*

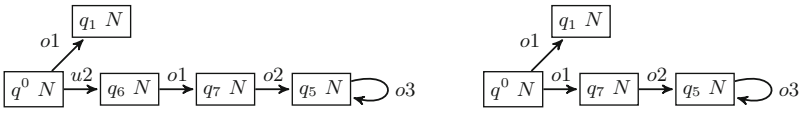

Fig. 4. Normal diagnoser (left) and its refined version (right) for Example 2.

The left (resp. right) part of Fig. 4 shows the normal diagnoser (resp. normal refined diagnoser) for Example 2.

4.2 Manifestability Checking

In this section, we show how to obtain the set of critical pairs based on the diagnosers described in the precedent section. Based on this, equivalence checking will be used to examine the manifestability condition $\Im$ in Theorem 2.

Definition 9 *(Pair Verifier). Given a system model G, its pair verifier V_G is obtained by synchronizing the corresponding fault and normal refined diagnosers D_G^{FR} and D_G^{NR} based on the set of observable events, i.e., $V_G = D_G^{FR} \parallel_{\Sigma_o} D_G^{NR}$.*

To construct a pair verifier, we impose that the synchronized events are the whole set of observable events. Then V_G is actually the product of D_G^{FR} and D_G^{NR} and the language of the pair verifier is thus the intersection of the language of the fault refined diagnoser and that of the normal refined diagnoser. In the pair verifier, each state is composed of two diagnoser states, whose label (F or N) of

the first one indicates whether the fault has effectively occurred in the first of the two corresponding trajectories. If the first of these two states is a fault state, then this verifier state is called ambiguous state since, reaching this state, the first trajectory contains the fault and the second not, while both have the same observations. Infinite trajectories of V_G are thus either normal (all states labels are (N,N)) or ambiguous (all states labels from a certain state are (F,N)), the latter ones being denoted by $L_a^\omega(V_G)$.

Lemma 4. *Given a system model G with its V_G, D_G^{FR} and D_G^{NR}, we have $L_a^\omega(V_G) = L_F^\omega(D_G^{FR}) \cap L^\omega(D_G^{NR})$.*

In the pair verifier depicted in Fig. 5, the gray node represents an ambiguous state.

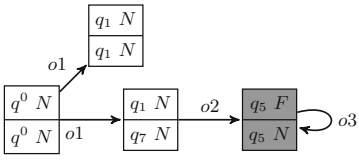

Fig. 5. The pair verifier for the system in Example 2.

Lemma 5. *Given a system model G, a fault F is diagnosable iff $L_a^\omega(V_G) = \emptyset$.*

Proof. $L_a^\omega(V_G) \neq \emptyset \Leftrightarrow L_F^\omega(D_G^{FR}) \cap L^\omega(D_G^{NR}) \neq \emptyset$ (from Lemma 4) $\Leftrightarrow P(L_F^\omega(G)) \cap P(L_N^\omega(G)) \neq \emptyset$ (from Lemmas 2 and 3) $\Leftrightarrow \exists s \in L_F^\omega(G), \exists s' \in L_N^\omega(G) P(s) = P(s') \Leftrightarrow \exists s, s' \in L^\omega(G)\, s \nsim s'$ (from Definition 3) $\Leftrightarrow F$ is not diagnosable (from Theorem 1). ■

Theorem 5. *Given a system model G, a fault F is manifestable iff $L_a^\omega(V_G) \subset L_F^\omega(D_G^{FR})$.*

Proof. $L_a^\omega(V_G) \not\subset L_F^\omega(D_G^{FR}) \Leftrightarrow L_F^\omega(D_G^{FR}) \subseteq L^\omega(D_G^{NR})$ (from Lemma 4) $\Leftrightarrow P(L_F^\omega(G)) \subseteq P(L_N^\omega(G))$ (from Lemmas 2 and 3) $\Leftrightarrow \forall s \in L_F^\omega(G), \exists s' \in L_N^\omega(G)$ $P(s) = P(s') \Leftrightarrow \forall s \in L_F^\omega(G), \exists s' \in L_N^\omega(G)\, s \nsim s'$ (from Definition 3) $\Leftrightarrow \neg\Im \Leftrightarrow F$ is not manifestable (from Theorem 2). ■

4.3 Algorithm

Algorithm 1 is the pseudo-code to verify manifestability, which can simultaneously verify diagnosability. Given the input (line 1) as the system model G and the fault F, we first construct the diagnoser (line 2) as described by Definition 6. We then construct fault and normal refined diagnosers (lines 3–4) as defined by Definitions 7 and 8. The next step is to synchronize D_G^{FR} and D_G^{NR} to obtain the pair verifier V_G (line 5). With D_G^{FR} and V_G, we have the following verdicts:

- if $L_a^\omega(V_G) = \emptyset$ (line 6), from Lemma 5, F is diagnosable and thus manifestable from Theorems 1 and 4 (line 7).
- if $L_a^\omega(V_G) = L_F^\omega(D_G^{FR}) \neq \emptyset$ (line 8), we can deduce from Theorem 5 that F is not manifestable. Thus, by Theorem 4 (or directly from Lemma 5), F is not diagnosable (line 9).
- if $L_a^\omega(V_G) \neq \emptyset$ and $L_a^\omega(V_G) \subset L_F^\omega(D_G^{FR})$ (line 10), which can be deduced because of Lemma 4, the former condition means that F is not diagnosable and, by Theorem 5, the latter means that F is manifestable (line 11).

Algorithm 1. Manifestability and Diagnosability Algorithm for DESs

1: INPUT: System model G; the considered fault F
2: $D_G \leftarrow ConstructDiagnoser(G)$
3: $D_G^{FR} \leftarrow ConstructFRDiagnoser(D_G)$
4: $D_G^{NR} \leftarrow ConstructNRDiagnoser(D_G)$
5: $V_G \leftarrow D_G^{FR} \parallel_{\Sigma_o} D_G^{NR}$
6: **if** $L_a^\omega(V_G) = \emptyset$ **then**
7: return "F is diagnosable and manifestable in G"
8: **else if** $L_a^\omega(V_G) = L_F^\omega(D_G^{FR})$ **then**
9: return "F is neither diagnosable nor manifestable in G"
10: **else**
11: return "F is not diagnosable but manifestable in G"
12: **end if**

Note that $L_F^\omega(D_G^{FR}) = L^\omega(D_G'^{FR})$ (resp., $L_a^\omega(V_G) = L^\omega(V_G')$) where $D_G'^{FR}$ is identical to D_G^{FR} (resp., V_G' identical to V_G), except that the final states, for Büchi acceptance conditions, are limited to fault (resp., ambiguous) states. Note also that the condition $L_a^\omega(V_G) = L_F^\omega(D_G^{FR})$ is equivalent to $L^\omega(V_G) = L^\omega(D_G^{FR})$ as the infinite normal trajectories are identical in V_G and in D_G^{FR}.

In Algorithm 1, the complexity of the different diagnosers constructions is linear. Building the pair verifier by synchronizing the fault and the normal refined diagnosers is polynomial with the number of system states. To finally check the manifestability, the equivalence checking (line 8) cannot be avoided, which is already demonstrated to be PSPACE, even for infinite words, in the literature [18]. Thus, the total complexity of this algorithm is PSPACE. Algorithm1 suggests that the manifestability problem is more complex than diagnosability (for which a test of language emptiness is sufficient, which implies a total NLOGSPACE complexity, a result already known), which we will formally prove later.

To verify the strong manifestability, one has to check the condition $\Im^s$ in Theorem 3. Algorithm 1 can be adapted for this with the following modifications:

- For each occurrence of the fault, we construct one fault refined diagnoser. To do this, we assume that the system has a finite number of fault occurrences (excluding thus cycles before a fault occurrence or containing a fault occurrence). To simplify, it is then enough to consider those latest occurrences of

the fault (for which no future contains another occurrence of the fault) since if such occurrence can show itself in one future, then this is the case for all earlier occurrences of the fault in the same trajectory.
- For each fault refined diagnoser, one constructs a pair verifier as described by Definition 9. Then, one has to compare the language defined by each fault refined diagnoser with the language defined by its corresponding verifier. The fault is not strongly manifestable iff there exists at least one such pair verifier and fault refined diagnoser defining the same languages for infinite words, as this violates the condition $\Im^s$ in Theorem 3.

Now we show that the problem of manifestability verification itself is a PSPACE-complete problem by the reduction to it of rational languages equivalence checking. The problem of checking non-deterministic FSM equivalence on infinite words is already proved to be PSPACE-complete [18].

Theorem 6. *Given a system model G and a fault F, the problem of checking whether F is manifestable in G is PSPACE-complete.*

Proof. The complexity of Algorithm 1 is PSPACE. Now we demonstrate that the problem of checking manifestability is PSPACE-hard. Let $G_1 = (Q_1, \Sigma, \delta_1, q_1^0)$ and $G_2 = (Q_2, \Sigma, \delta_2, q_2^0)$ be two arbitrary (non-deterministic) automata on the same vocabulary defining live languages. One can always assume that $Q_1 \cap Q_2 = \emptyset$. Based on G_1 and G_2, one can construct a new FSM, representing a system model, $G = (Q, \Sigma \cup \{F\}, \delta, q_2^0)$, where $Q = Q_1 \cup Q_2$ and $\delta = \delta_1 \cup \delta_2 \cup \{(q_2^0, F, q_1^0)\}$, with $\Sigma_o = \Sigma$, $\Sigma_u = \emptyset$ and $\Sigma_f = \{F\}$. From the construction of G, one has $L^\omega(G_1) = P(L_F^\omega(G))$ and $L^\omega(G_2) = P(L_N^\omega(G))$. From Lemmas 2, 3 and 4, one obtains $L^\omega(V_G) = P(L_F^\omega(G)) \cap P(L_N^\omega(G))$. This implies $L^\omega(G_1) \cap L^\omega(G_2) = L^\omega(V_G)$. From Theorem 5, one has $L^\omega(G_1) \cap L^\omega(G_2) \subset L^\omega(G_1) \iff F$ is manifestable in G, i.e., $L^\omega(G_1) \subseteq L^\omega(G_2) \iff F$ is not manifestable in G. So, rational languages inclusion testing on infinite words boils down to manifestability checking, which gives the result. ■

5 Experimental Results

We have applied our algorithm on more than one hundred examples taken from literature and hand-crafted ones. The latter ones are constructed to show the scalability since the sizes of the former ones are very small. Our experimental results are obtained by running our program on a Mac OS laptop with a 1.7 GHz Intel Core i7 processor and 8 Go 1600 MHz DDR3 of memory.

Table 1 shows part of our experimental results, where verdicts (i.e., Manifes(tability), S(trong)Manifes(tability), Diagno(sability), N(on)Manifes (tability)) show the strongest property satisfied by the system. For example, if it is Manifes, then it is not SManifes nor Diagno. Diagno implies both SManifes and Manifes. We give the number of states and transitions of the system (|S|/|T|), of the pair verifier (|S|/|T|(PV)), as well as the execution time (millisecond is used as time unit). The size of the pair verifier includes all transitions

Table 1. Experimental Results

LitSys	|S|/|T|	|S|/|T|(PV)	Time	Verdict	HCSys	|S|/|T|	|S|/|T|(PV)	Time	Verdict
Ex. 2	8/10	4/4	15	SManifes	h-c1	22/24	18/18	32	SManifes
[14]	16/23	21/23	51	Manifes	h-c2	36/39	74/77	90	Manifes
[12]	16/20	7/9	25	Manifes	h-c3	46/50	105/110	120	Manifes
[9]	3/6	4/6	12	SManifes	h-c4	52/57	160/183	151	SManifes
[20]	18/21	53/57	69	SManifes	h-c5	57/69	32/37	78	SManifes
[15]	9/11	2/1	16	Diagno	h-c6	509/570	79/81	132	Manifes
[13]	12/28	45/51	68	NManifes	h-c7	320/390	1752/1791	323	NManifes

generated from the synchronization of the fault refined diagnoser and the normal refined diagnoser. The examples shown here include Example 2 in this paper with the illustrative examples of other papers that handle similar problems.

To construct the hand-crafted examples (HCSys) from those selected from the literature (LitSys), we are not interested in diagnosable examples. First, diagnosable systems are rare in the literature as well as in the industry. Second, diagnosability implies an empty language of ambiguous infinite words for the pair verifier, which can be verified without equivalence checking. The efficiency cannot be convincing by applying our algorithm on diagnosable examples. When extending the examples from the literature, we keep the same verdict. For example, for a manifestable system, an arbitrary FSM without fault is added in a place such that at least one faulty infinite trajectory can always manifest itself (and obviously critical pairs are preserved).

From our experimental results, the executed time is also dependent on the size of the pair verifier besides that of the system. To achieve a worst case, one way is to employ the example construction in the proof of Theorem 6 by setting $L^\omega(G_1) = L^\omega(G_2)$. The hand-crafted example h-c7 is constructed in such a way.

We can see that the original HVAC system in [13] is not manifestable, i.e., any faulty behavior cannot be diagnosed in all its infinite futures. It is thus necessary to go back to design stage to revise the system model. For other manifestable but not diagnosable systems, one interesting future work is to study bounded-manifestability, making sure to detect the fault in bounded time.

6 Related Work

The first approach to verify the diagnosability of DESs is to construct a deterministic FSM to check the existence of critical pairs [13], which has however exponential complexity in the number of system states. Then the authors of [10] proposed another method called twin plant with polynomial complexity. Here we adapted the twin plant plus equivalence checking to verify manifestability. Note that the existence of critical pairs, that excludes diagnosability, does not exclude manifestability. Intuitively, manifestability is a more complicated problem than diagnosability, which was demonstrated by proving that the problem itself is PSPACE instead of polynomial (actually NLOGSPACE) for diagnosability.

In [16,17], the authors proposed different variants of detectability (e.g., (strong) detectability) about state estimation. The system is detectable (resp. strongly detectable) if, based on a sequence of observations, one can be sure about the state in which is the system for some given trajectory (resp. all trajectories). They proposed a polynomial algorithm for strong detectability, for which two different trajectories with the same observations implies the violation. However, to analyze detectability, they constructed a deterministic observer that has exponential complexity with the number of system states. Our approach can be adapted to handle state estimation by considering an ambiguous state as one that contains different system states. Thus, we can improve their state estimation by using the improved equivalence checking techniques (e.g., the approach of [5] normally constructs a small part of the deterministic automaton). Furthermore, we proved that the problem of manifestability itself is PSPACE-complete.

The authors of [1,8] proposed an approach for weak diagnosability in a concurrent system by using Petri net, i.e., impose a constraint of weak fairness by disallowing the enabled transition to be perpetually ignored. The idea is to make impossible some non-diagnosable scenarios in order to upgrade the diagnosability level. They focused on how to get the more appropriate model, based on which the solution can be polynomial such as that for classical diagnosability.

Two definitions for stochastic diagnosability were introduced and analyzed in [19], which are weaker than diagnosability. A-diagnosability requires that the ambiguous behaviors have a null probability. AA-diagnosability admits errors in the provided information which should have an arbitrary small probability. Then four variants of diagnosability (FA, IA, FF, IF) were introduced and studied for different probabilistic system models [3,4]. Different ambiguity criteria were then defined according to different types of runs: for faulty runs only or for all runs; for infinite runs or for finite sub-runs. Among them IF-diagnosability (for infinite faulty runs) is the weakest one. Note that IF-diagnosability of a finite probabilistic system is equivalent to A-diagnosability.

The authors of [2,9] analyzed (safe) active diagnosability by introducing controllable actions for (probabilistic) DESs, where the complexity of these problems were also studied. The idea is to design controllers (resp. label activation strategies for probabilistic version) to enable a subset of actions in order to make it diagnosable (resp. stochastically diagnosable).

7 Conclusion and Future Work

In this paper we addressed the formal verification of manifestability for DESs. To bring an alternative to diagnosability analysis, whose satisfaction is very demanding in terms of sensors placement, we defined (strong) manifestability, a new weaker property. Then, we constructed different structures from the system model to check manifestability by using equivalence techniques. The entailment relations between different properties were proved and demonstrated on examples from the literature. Thus, engineers have a variety of criteria to design systems with optimal trade-off between safety and cost. One interesting future

work is to extend our approach for distributed systems composed of a set of components, each one being modeled as a FSM with synchronization events.

References

1. Agarwal, A., Madalinski, A., Haar, S.: Effective verification of weak diagnosability. In: Proceedings of the 8th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes (SAFEPROCESS 2012), pp. 636–641. IFAC (2012)
2. Bertrand, N., Fabre, É., Haar, S., Haddad, S., Hélouët, L.: Active diagnosis for probabilistic systems. In: Muscholl, A. (ed.) FoSSaCS 2014. LNCS, vol. 8412, pp. 29–42. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54830-7_2
3. Bertrand, N., Haddad, S., Lefaucheux, E.: Foundation of diagnosis and predictability in probabilistic systems. In: 34th International Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2014, 15–17 December 2014, New Delhi, India, pp. 417–429 (2014)
4. Bertrand, N., Haddad, S., Lefaucheux, E.: Diagnosis in infinite-state probabilistic systems. In: 27th International Conference on Concurrency Theory, CONCUR 2016, 23–26 August 2016, Québec City, Canada, pp. 37:1–37:15 (2016)
5. Bonchi, F., Pous, D.: Checking NFA Equivalence with Bisimulations up to Congruence. In: Proceedings of 40th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL-2013), pp. 457–468. ACM (2013)
6. Büchi, J.R.: On a decision method in restricted second order arithmetic. Z. Math. Logik Grundlag. Math **6**, 66–92 (1960)
7. Cassandras, C.G., Lafortune, S.: Introduction To Discrete Event Systems, 2nd edn. Springer, Heidelberg (2008). https://doi.org/10.1007/978-0-387-68612-7
8. Germanos, V., Haar, S., Khomenko, V., Schwoon, S.: Diagnosability under weak fairness. ACM Trans. Embed. Comput. Syst. **14**(4), 69 (2015)
9. Haar, S., Haddad, S., Melliti, T., Schwoon, S.: Optimal constructions for active diagnosis. J. Comput. Syst. Sci. **83**(1), 101–120 (2017)
10. Jiang, S., Huang, Z., Chandra, V., Kumar, R.: A polynomial time algorithm for testing diagnosability of discrete event systems. Trans. Autom. Control **46**(8), 1318–1321 (2001)
11. Papineau, D.: Philosophical Naturalism. Blackwell Publishers, Hoboken (1993)
12. Pencolé, Y.: Diagnosability analysis of distributed discrete event systems. In: Proceedings of the 16th European Conference on Articifial Intelligent (ECAI 2004), pp. 43–47. IOS Press, Nieuwe Hemweg (2004)
13. Sampath, M., Sengupta, R., Lafortune, S., Sinnamohideen, K., Teneketzis, D.: Diagnosability of discrete event system. Trans. Autom. Control **40**(9), 1555–1575 (1995)
14. Schumann, A., Huang, J.: A scalable jointree algorithm for diagnosability. In: Proceedings of the 23rd American National Conference on Artificial Intelligence (AAAI 2008), pp. 535–540. AAAI Press, Menlo Park (2008)
15. Schumann, A., Pencolé, Y.: Scalable diagnosability checking of event-driven system. In: Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI 2007), pp. 575–580. International Joint Conferences on Artificial Intelligence Inc., Menlo Park (2007)
16. Shu, S., Lin, F.: Detectability of discrete event systems with dynamic event observation. Syst. Control Lett. **59**(1), 9–17 (2010)

17. Shu, S., Lin, F.: I-detectability of discrete-event systems. IEEE Trans. Autom. Sci. Eng. **10**(1), 187–196 (2013)
18. Sistla, A.P., Vardi, M.Y., Wolper, P.: The complementation problem for Büchi automata with applications to temporal logic. Theor. Comput. Sci. **49**(2–3), 217–237 (1987)
19. Thorsley, D., Teneketzis, D.: Diagnosability of stochastic discrete-event systems. IEEE Trans. Autom. Control **50**(4), 476–492 (2005)
20. Ye, L., Dague, P.: Diagnosability analysis of discrete event systems with autonomous components. In: Proceedings of the 19th European Conference on Artificial Intelligence (ECAI 2010), pp. 105–110. IOS Press, Nieuwe Hemweg (2010)
21. Ye, L., Dague, P., Longuet, D., Briones, L.B., Madalinski, A.: Fault manifestability verification for discrete event systems. In: Proceedings of the 22nd European Conference on Artificial Intelligence (ECAI 2016), pp. 1718–1719. IOS Press (2016)