ECAI 2016
G.A. Kaminka et al. (Eds.)

doi:10.3233/978-1-61499-672-9-1718

# Fault Manifestability Verification for Discrete Event Systems

**Lina Ye** [1] and **Philippe Dague** [2] and **Delphine Longuet** [3] and **Laura Brandán Briones** [4] and **Agnes Madalinski**[5]

**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 ability to determine whether a given fault has effectively occurred based on the observations. However, this is a very strong property that requires generally high number of sensors to be satisfied. Consequently, it is not rare that developing a diagnosable system is too expensive. To solve this problem, in this paper, we first define a new system property called manifestability that represents the weakest requirement on faults and observations for having a chance to identify on line fault occurrences and can be verified at design stage. Then, we propose an algorithm with PSPACE complexity to automatically verify it.

## 1 Introduction

The diagnosability problem has received considerable attention in the literature. Diagnosability describes the system ability to determine with certainty whether a given fault has effectively occurred based on the observations. In a given system, the existence of two infinite behaviors, with the same observations but exactly one containing the considered fault, violates diagnosability. The existing works search for such ambiguous behaviors both in centralized [5, 3] and distributed [4, 7] ways. The most classical method is to construct a structure called twin plant that captures all pairs of observable equivalent behaviors to directly check the existence of such ambiguous pairs. However, in reality, diagnosability is a very strong property that requires generally 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., the reasonable number of sensors, and the possibility to observe a fault manifestation, we define in this paper a new property called manifestability. This is a property describing the capability of a system to manifest (i.e., be distinguishable from any non faulty behavior) a fault occurrence in at least one context, i.e., one future behavior. This should be analyzed at design stage on the system model. Manifestability is the weakest property to require from the system to have a chance to identify the fault occurrence.

Our contributions in this paper are described as follows: Firstly, we define formally the new manifestability property. Then, we provide a sufficient and necessary condition for manifestability. Finally, we propose an algorithm based on equivalence checking of Finite State Machines (FSMs) with PSPACE complexity.

[1] LRI, CentraleSupélec, France, email: lina.ye@lri.fr
[2] LRI, Univ. Paris-Sud, CNRS, France, email: philippe.dague@lri.fr
[3] LRI, Univ. Paris-Sud, CNRS, France, email: Delphine.Longuet@lri.fr
[4] Universidad Nacional de Córdoba, Argentina
[5] Otto-von-Guericke-University Magdeburg, Germany

## 2 Manifestability for DESs

We model a Discrete Event System (DES) as a FSM, denoted by $G = (Q, \Sigma, \delta, q^0)$, where $Q$ is the finite set of states, $\Sigma$ 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 $\Sigma^*$), and $q^0$ is the initial state. The set of events $\Sigma$ is divided into three disjoint parts: $\Sigma_o$ is the set of observable events, $\Sigma_u$ the set of unobservable normal events and $\Sigma_f$ the set of unobservable fault events. Similar to diagnosability, the manifestability algorithm that we will propose has exponential complexity in the number of fault types. For the sake of reducing this complexity to linear, as in [4, 6], we consider only one fault type at a time but multiple occurrences of faults are allowed.

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 $q_0\sigma_0 q_1\sigma_1 ...$ a path in $G$, where $\sigma_0\sigma_1 ...$ is a trajectory in $G$ and we have $\forall i, (q_i, \sigma_i, q_{i+1}) \in \delta$. Given $s \in L(G)$, we denote the post-language of $L(G)$ after $s$ by $L(G)/s$, 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 $P(s)$. This projection can be extended to a language $L(G)$, i.e., $P(L(G)) = \{P(s) | s \in L(G)\}$. Traditionally, we assume that the system language is always live (any trajectory has a continuation, i.e., is a strict prefix of another trajectory) without unobservable cycle. We will need some infinite objects. So, inspired from the notation of Büchi automata [1], we denote by $\Sigma^\omega$ the set of infinite words on $\Sigma$ and by $\Sigma^\infty = \Sigma^* \cup \Sigma^\omega$ the set of words on $\Sigma$, finite or infinite. We define in an obvious way $L^\omega(G)$ (infinite words whose all finite prefixes belong to $L(G)$) and $L^\infty(G)$ and thus infinite trajectories and infinite paths. Particularly, we use $L^\omega_F(G) = L^\omega(G) \cap \Sigma^* F \Sigma^\omega$ for the set of infinite faulty trajectories, and $L^\omega_N(G) = L^\omega(G) \cap (\Sigma \setminus \{F\})^\omega$ for the set of infinite normal trajectories, where $\setminus$ denotes set subtraction. We will use the classical synchronization of two FSMs for which only synchronized events should occur simultaneously, denoted by $G_1 \|_{\Sigma_s} G_2$, where $\Sigma_s$ is the set of synchronized events.

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

Delay closure is to keep all information about events in $\Sigma_d$, deleting those not in $\Sigma_d$, while keeping the same structure. It will be used to simplify the system model without affecting the result.

Informally speaking, a fault $F$ is diagnosable in a system $G$ if and only if (iff) it can be determined without ambiguity when enough events are observed from $G$ after its occurrence.

**Definition 2** ***(Critical Pair)****. A pair of trajectories $s, s'$ is called a critical pair with respect to $F$, denoted by $s \nsim s'$, iff $s \in L_N^\omega(G), s' \in L_F^\omega(G)$ and $P(s) = P(s')$.*

The existence of a critical pair w.r.t. $F$ violates diagnosability and thus diagnosability verification consists in checking the nonexistence of such a pair. To design a diagnosable system, each faulty trajectory should be distinguished from all normal trajectories, which is most often expensive in terms of sensors required. To reduce such a cost and still make possible to show the fault after enough runs of the system, we now define another much weaker property, manifestability, as follows, where $s^F$ denotes a trajectory ending with $F$.

**Definition 3** ***(Manifestability)****. A fault $F$ is manifestable in $G$ 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 occurrence $s^F$ in $G$, there exists at least one extension $t$ of $s^F$ in $G$, such that every trajectory $p$ that is observable equivalent to $s^F t$ should contain $F$.

**Theorem 1** *A fault $F$ is manifestable in $G$ iff the following condition $\Im$ is satisfied: $\exists s \in L_F^\omega(G), \nexists s' \in L_N^\omega(G)$, such that $s \nsim s'$.*

## 3 Manifestability Verification

Given a system, manifestability verification consists in checking whether the condition $\Im$ in Theorem 1 is satisfied. Next we show, given a system model, how to construct different structures to obtain $L_F^\omega(G)$, $L_N^\omega(G)$, and critical pairs set. The condition $\Im$ can be checked by using equivalence techniques based on these structures.

Given a system model, the first step that we propose 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 4** *(Diagnoser). Given a system $G$, 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 $\delta_D$ are those $((q,\ell), e, (q',\ell'))$, with $(q,\ell)$ reachable from the initial state $q_D^0$, such that there is a transition $(q, e, q') \in \delta$, and $\ell' = F$ if $\ell = F \vee e = F$, otherwise $\ell' = N$.*

Based on a diagnoser, we define the following two structures to obtain the subparts with only faulty or only normal trajectories.

**Definition 5** *(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) $Q_{D^F} = \{q_D \in Q_D \mid \exists q'_D = (q, F) \in Q_D, \exists s, s' \in \Sigma_D^*, (q_D^0, s, q_D) \in \delta_D^*, (q_D, s', q'_D) \in \delta_D^*\}$; 3) $\delta_{D^F} = \{(q_D^1, \sigma, q_D^2) \in \delta_D \mid q_D^1, q_D^2 \in Q_{D^F}\}$; 4) $\Sigma_{D^F} = \{\sigma \in \Sigma_D \mid \exists (q_D^1, \sigma, q_D^2) \in \delta_{D^F}\}$. Now the fault refined diagnoser, denoted by $D_G^{FR}$, is obtained by $D_G^{FR} = \mathsf{C}_{\Sigma_o}(D_G^F)$.*

**Definition 6** *(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) $\delta_{D^N} = \{(q_D^1, \sigma, q_D^2) \in \delta_D \mid q_D^1, q_D^2 \in Q_{D^N}\}$; 4) $\Sigma_{D^N} = \{\sigma \in \Sigma_D \mid \exists (q_D^1, \sigma, q_D^2) \in \delta_{D^N}\}$. Now the normal refined diagnoser, denoted by $D_G^{NR}$, is obtained by $D_G^{NR} = \mathsf{C}_{\Sigma_o}(D_G^N)$.*

Now we propose another structure that can capture the set of critical pairs, which can then be used for equivalence checking to examine the manifestability condition $\Im$.

**Definition 7** *(Pair Verifier). Given a system $G$, its pair verifier $V_G$ is obtained by synchronizing the corresponding fault refined diagnoser $D_G^{FR}$ and normal refined diagnoser $D_G^{NR}$ 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. Only in this way, we can guarantee that the language of the pair verifier is the intersection of the languages of the fault refined diagnoser and that of the normal refined diagnoser. In the pair verifier, each state is composed of two diagnoser states, indicating whether the fault has effectively occurred in the first of the two corresponding trajectories.

Once constructed the pair verifier $V_G$, we now give our major result as follows, whose proof with associated lemmas are omitted here due to lack of space.

**Theorem 2** *Given a system $G$ and its pair verifier $V_G$, a fault $F$ is manifestable iff $L^\omega(V_G) \neq P(L_F^\omega(G))$.*

To check manifestability, the complexity of different diagnosers constructions is linear. For the pair verifier, we have to synchronize the fault refined diagnoser and the normal refined diagnoser, which is obviously polynomial with the number of system states. To finally check the manifestability, the equivalence checking should be performed, which is already demonstrated to be PSPACE in the literature [2]. Thus, the total complexity of this algorithm is PSPACE.

## 4 Conclusion and future work

In this paper we addressed the formal verification of manifestability for DESs, inspired from diagnosability checking largely studied in the literature. To bring an alternative to diagnosability analysis, whose satisfaction is very demanding in terms of sensors placement, we defined a new weaker property, manifestability, with a sufficient and necessary condition. Then, we constructed different structures from the system model to be able to check this manifestability condition by using equivalence techniques. One future work is to extend our approach to fault manifestability in at most a fixed number of transitions after the fault occurrence, such as I-diagnosability in [5].

## REFERENCES


[1] J.R. Büchi, 'On a decision method in restricted second order arithmetic', *Z. Math. Logik Grundlag. Math*, **6**, 66–92, (1960).

[2] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introduction To Automata Theory, Languages, and Computation*, Pearson Education, 1979.

[3] S. Jiang, Z. Huang, V. Chandra, and R. Kumar, 'A Polynomial Time Algorithm for Testing Diagnosability of Discrete Event Systems', *Transactions on Automatic Control*, **46**(8), 1318–1321, (2001).

[4] Y. Pencolé, 'Diagnosability Analysis of Distributed Discrete Event Systems', in *Proceedings of the 16th European Conference on Articifial Intelligent (ECAI04)*, pp. 43–47. Nieuwe Hemweg: IOS Press., (2004).

[5] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis, 'Diagnosability of Discrete Event System', *Transactions on Automatic Control*, **40**(9), 1555–1575, (1995).

[6] A. Schumann and J. Huang, 'A Scalable Jointree Algorithm for Diagnosability', in *Proceedings of the 23rd American National Conference on Artificial Intelligence (AAAI-08)*, pp. 535–540. Menlo Park, Calif.: AAAI Press., (2008).

[7] L. Ye and P. Dague, 'Diagnosability Analysis of Discrete Event Systems with Autonomous Components', in *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI-10)*, pp. 105–110. Nieuwe Hemweg: IOS Press., (2010).