Using Model-Checking Techniques for Diagnosability Analysis of Intermittent Faults - A Railway Case-Study

Abderraouf Boussif Mohamed Ghazel
Univ. Lille Nord de France, F-59000 Lille, France
IFSTTAR, Cosys/Estas, F-59650 Villenveuve d'Ascq, France
{*abderraouf.boussif, momahed.ghazel*}*@ifsttar.fr*

This paper addresses formal verification of intermittent fault diagnosability in Discrete Event Systems (DESs). The system is modeled by a Finite State Automaton and intermittent faults are defined as faults that can automatically recover once they have occurred. Two definitions of diagnosability, regarding the detection of fault occurrences within a finite delay and the detection of fault occurrences before their recovery, are discussed. The diagnosability is analyzed on the basis of the twin-plant structure, which is formally modeled as a Kripke structure, while diagnosability conditions are formulated using LTL temporal logic. We focus on a practical application of this approach, namely a case-study from the railway control field, will serve as a benchmark to illustrate the various developed mechanisms and to assess the scalability of the technique.

Discrete Event Systems, Formal verification, Diagnosability analysis, Intermittent faults, Model-Checking.

1. INTRODUCTION

Fault diagnosis is a crucial and challenging task in large and complex dynamic systems. This problem has been extensively studied by both Artificial Intelligence (AI) and Control Engineering communities. In particular, an increasing amount of work has been devoted to fault diagnosis of DES over the last two decades as witnessed by the survey work in (Zaytoon and Lafortune 2013).

One of the main issues in fault diagnosis of DES, is diagnosability analysis. In simple terms, diagnosability refers to the ability to infer accurately, from partially observed executions, about the faulty behavior within a finite delay after a possible occurrence of a fault. The formal definition of diagnosability was first introduced in the seminal work (Sampath et al. 1995), where a systematic method to check diagnosability based on a diagnoser construction in an *event-based* context was developed. A similar work based on a diagnoser construction in a *state-based* context was proposed in (Zad et al. 2003).

Improvements in terms of complexity, based on the verifier and the twin-plant structures have been introduced in (Jiang et al. 2001; Schumann and Pencolé 2007; Yoo and Lafortune 2002), where the basis idea was to build an intermediate structure by performing a parallel composition of the system model with itself. Then diagnosability problem can be addressed by analyzing every pair of executions that share the same observation.

Model-Checking techniques (Clarke et al. 1999), which have been developed for efficiently verifying complex dynamic systems, have been exploited to deal with diagnosis issues. For instance, (Cimatti et al. 2003) addressed the formal verification of diagnosability using CTL symbolic Model-Checking. In this work, verifying diagnosability is reduced to a reachability analysis problem in the twin-plant structure where the condition of diagnosability is expressed as a CTL formula. Some further reformulations were also given in (Bourgne et al. 2009; Boussif and Ghazel 2015). Several algorithms based on symbolic techniques, have been proposed in (Grastien 2009) to test diagnosability with fairness properties. In (Peres and Ghazel 2014) a novel approach to deal with diagnosability using a μ-calculus logical framework was proposed. The Boolean Satisfiability Problem (SAT), which is a dual technique of Model-Checking has been also used to deal with some fault diagnosis issues (Grastien and Anbulagan 2007; Grastien 2008).

All the DES framework discussed above assume that the failures are permanent, which means once a fault occurs, the system remains indefinitely faulty; hence the terminology "*failure*" is often used for permanent faults.

In many systems, faulty behavior often occurs intermittently, which can be depicted as a failure

event followed by its corresponding "*reset*" event, followed by new occurrences of failure events, and so forth (Contant 2005). Indeed, intermittent faults are defined as faults that can automatically recover once they have occurred. Such faults are predominant in many real life systems, like, for example, electrical contacts, overheating of chips, noise measurements in hardware systems, exceptions, interrupts, and bugs in software systems.

The methodologies referenced above for permanent faults are no longer adequate in the context of intermittent faults. Since the case of intermittent faults shows some subtle configurations compared to the case of permanent failures. Consequently, some DES based frameworks have been proposed to handle intermittent faults. One of the first contributions was made by (Jiang et al. 2003), where a *state-based* DES modeling for the so-called "*repeated faults*" was introduced. Various notions of diagnsability were discussed and polynomial algorithms for checking these properties were provided. Some improvements have been introduced in (Yoo and Garcia 2004; Zhou and Kumar 2009).
These works focus on determining how many times a failure has occurred, but do not address the system status determination. Dealing with diagnosability of intermittent faults in this sense was first discussed in series of works (Contant 2002, 2005). In these studies, an FSA *event-based* approach is used, i.e. the faults and their recovering are considered as unobservable events. The purpose of these works is to determine which failures are present in the system and which failures have occurred and been recovered, This work represents an extension of the seminal work on diagnosability of permanent failures (Sampath et al. 1995) with some modifications regarding the failure modeling and the diagnoser construction in order to cater for intermittent failures. A similar work is reported in (Correcher et al. 2003) with an illustration through an industrial process.

In (Soldani et al. 2007), intermittent fault diagnosis in an FSA framework was reported. Particularly, only the normal behavior of the system is considered and failure are modeled as the occurrence of an extra event or as the lack of a specific event. A diagnoser is then established for each event type. An extension to Petri Net framework was given in (Soldani et al. 2006).

An extension of the *state-based* DES framework, introduced in (Zad et al. 2003), was proposed in (Biswas 2012) to deal with intermittent failures. Two notions of diagnosability were introduced, one for detecting the occurrence of a fault, and the other for detecting its recovery. The diagnoser is constructed in the same way as in (Zad et al. 2003) with the same time-complexity. Necessary and sufficient conditions for each notion have been developed, and an algorithm to verify the diagnosability conditions has been provided.

A new way for modeling failures, which includes permanent and intermittent faults, was proposed in (Guanqian et al.). Diagnosability of both permanent and intermittent failures were revisited, and an approach to discriminate between these fault types was discussed. Illustrative examples to demonstrate the proposed approach and analysis results were presented.

In this paper, we propose an approach for diagnosability analysis of intermittent fault using model-checking techniques. The approach is based on the twin-plant structure (Jiang et al. 2001), and the reformulation of the diagnosability issues as temporal logic formulas that are workable with Model-Checking.

We first discuss two definitions of diagnosability of intermittent faults, regarding the detection of fault occurrences within a finite delay and the detection of fault occurrences before their recovery. Then, necessary and sufficient conditions for each notion are developed based on the twin-plant construction, and reformulated as linear temporal logic (LTL) formulas in order to use model-checking for actual verification. A railway case-study is used to illustrate the various concepts discussed and also to assess the efficiency and the scalability of the approach.

The paper is organized as follows: Section 2 introduces the considered system model and the modeling of intermittent faults as well as some related notions and notations. In section 3, different definitions of diagnosability are discussed. Section 4 discusses the twin-plant construction and gives the necessary and sufficient conditions for each definition. Formulation of diagnosability of intermittent faults as a model-checking issue, and necessary and sufficient conditions as LTL formulas are established in Section 5. We illustrate the discussed concepts through a railway case-study (a level crossing benchmark) in Section 6. Finally, Section 7 draws some concluding remarks and points some future directions.

2. PRELIMINARIES

2.1. System Model

Discrete models are quite convenient to perform safety analysis of industrial systems in a sufficiently high abstraction level (Cassandras and Lafortune 2008). When systems are abstracted as DESs for diagnosis purposes, the model used is often a finite state automaton (FSA) $G = \langle X, \Sigma, \delta, x_0 \rangle$ where, X is a finite set of states, Σ is a finite set of events, $\delta : X \times \Sigma \to 2^X$ is the partial transition function, and $x_0 \in X$ is the initial state. A triple $(x, \sigma, x') \in X \times \Sigma \times X$ is called a *transition* if $x' \in \delta(x, \sigma)$. The model G accounts for the normal and faulty behaviors of the system. The system behaviors are then described by the prefix-closed language $L \subseteq \Sigma^*$ generated by G, where Σ^* denotes the Kleene-closure of set Σ.

Partial observability is a key issue in fault diagnosis. In this regard, some events in Σ are observable, i.e.,

their occurrence can be observed, while the others are unobservable. Thus, event set Σ can be partitioned as $\Sigma = \Sigma_o \biguplus \Sigma_u$, where Σ_o denotes the set of observable events and Σ_u the set of unobservable events.

In the context of diagnosis of intermittent faults, let $\Sigma_f \subseteq \Sigma_u$ denotes the set of fault events and let $\Sigma_r \subseteq \Sigma_u$ denotes the set of fault reset events. Failures and their recovery are basically represented using unobservable events, since their detection and diagnosis would be trivial if they were observable. Thus, the set of fault events (resp. the set of reset events) can be partitioned as disjoint failure classes $\Sigma_f = \Sigma_{f_1} \biguplus \Sigma_{f_2} \biguplus \cdots \biguplus \Sigma_{f_m}$, where $\Sigma_{f_i} (i = 1, 2, \ldots, m)$ denotes the i^{th} fault class (resp. $\Sigma_r = \Sigma_{r_1} \biguplus \Sigma_{r_2} \biguplus \cdots \biguplus \Sigma_{r_m}$, where $\Sigma_{r_i} (i = 1, 2, \ldots, m)$ denotes the recovering class of faults in Σ_{f_i}).

An event-trace $s = (\sigma_1, \sigma_2, \ldots, \sigma_n)$ is said to be *associated* with state-trace $\pi = (x_1, x_2, \ldots, x_{n+1})$ if $\forall 1 \leq i \leq n, x_{i+1} \in \delta(x_i, \sigma_i)$. We write s^i to indicate the i^{th} event in s and s_f the last event in s (i.e., $s_f = s^{|s|}$). We denote by L/s the post-language of L upon s, i.e., $L/s := \{t \in \Sigma^* \,|s.t \in L\}$. We write $s \leq t$ to denote the fact that s is a prefix of t.

For convenience, we introduce the following particular sets of event-traces:

- $\psi(\Sigma_{f_i}) = \{s = (\sigma_1, \sigma_2, \ldots, \sigma_n) \in L \mid \sigma_n \in \Sigma_{f_i}\}$ is the set of event-traces in L that end with a faulty event in Σ_{f_i}.
- $\psi(\Sigma_{r_i}) = \{s = (\sigma_1, \sigma_2, \ldots, \sigma_n) \in L \mid \sigma_n \in \Sigma_{r_i}\}$ is the set of event-traces in L that end with a reset event in Σ_{r_i}.
- $\overline{\psi}(\Sigma_{f_i}) = \{s = (\sigma_1, \sigma_2, \ldots, \sigma_n) \in L \mid \forall 1 \leq i < n : \sigma_i \notin \Sigma_{f_i} \wedge \sigma_n \in \Sigma_{f_i}\}$ is the set of event-traces in L that have only the last event in Σ_{f_i}.
- $\overline{\psi}(\Sigma_{r_i}) = \{s = (\sigma_1, \sigma_2, \ldots, \sigma_n) \in L \mid \forall 1 \leq i < n : \sigma_i \notin \Sigma_{r_i} \wedge \sigma_n \in \Sigma_{r_i}\}$ is the set of event-traces in L that have only the last event in Σ_{r_i}.

Let us consider $\sigma \in \Sigma$ and $s \in \Sigma^*$, we write $\sigma \in s$ to denote the fact that $\exists\, 1 \leq i \leq |s|$ such that $s^i = \sigma$. By abuse of notation, we write $\Sigma_f \in s$ to denote that a fault event from Σ_f is an event in event-trace s (i.e., $\exists f \in \Sigma_{f_i}$ such that $f \in s$).

To capture the observed behavior of the system, we define the projection operator as a mapping $P : \Sigma^* \to \Sigma_o^*$. In the usual manner, $P(\sigma) = \sigma$ for $\sigma \in \Sigma_o$; $P(\sigma) = \epsilon$ for $\sigma \in \Sigma_u$, and $P(s\sigma) = P(s)P(\sigma)$, where $s \in \Sigma^*$ and $\sigma \in \Sigma$. That is, P simply erases the unobservable events in any event-trace. The inverse projection P_L^{-1} is defined by $P_L^{-1}(y) = \{s \in L(G) : P(s) = y\}$. The projection operator can then be extended to language L by applying the projection to all traces of L. Therefore, if $L \subseteq \Sigma^*$, then $P(L) = \{t \in \Sigma_o^* : (\exists\, s \in L)\,[P(s) = t]\}$.

Let $G_1 = \langle X_1, \Sigma_1, \delta_{G_1}, x_{0_1} \rangle$ and $G_2 = \langle X_2, \Sigma_2, \delta_{G_2}, x_{0_2} \rangle$ denote two finite state automata. The strict synchronous composition of G_1 and G_2 produces an automaton $G_{G_1 \| G_2} = \langle X_1 \times X_2, \Sigma_1 \cap \Sigma_2, \delta_{G_1 \| G_2}, (x_{0_1}, x_{0_2}) \rangle$, where $\delta_{G_1 \| G_2} \subseteq (X_1 \times X_2) \times (\Sigma_1 \cap \Sigma_2) \times (X_1 \times X_2)$ and $(x_1', x_2') \in \delta_{G_1 \| G_2}((x_1, x_2), \sigma)$ if $x_1' \in \delta_{G_1}(x_1, \sigma)$ and $x_2' \in \delta_{G_2}(x_2, \sigma)$.

Finally, we define the non-deterministic automaton $G' = \langle X_o, \Sigma_o, \delta_{G'}, x_0 \rangle$ as the generator of language $L(G') = P(L(G))$. Thus, G' is called "*the constructed generator.*" of G. Elements Σ_o and x_0 are as defined before. $X_o = \{x_0\} \cup \{x \in X : \exists x' \in X, \exists \sigma \in \Sigma_o : x \in \delta(x', \sigma)\}$. The transition relation of G' is given by $\delta_{G'} \subseteq (X_o \times \Sigma_o \times X_o)$ and is defined as follows: $(x, \sigma, x') \in \delta_{G'}$ if $\exists s = (\sigma_1, \sigma_2, \ldots, \sigma_n = \sigma) \in \Sigma^*$ such that $x' \in \delta(x, s)$ and $\forall 1 \leq i \leq n-1, \sigma_i \in \Sigma_u$, $\sigma_n \in \Sigma_o$.

In the remainder of this paper, we consider a finite state automaton $G = \langle X, \Sigma, \delta, x_0 \rangle$ as the model of the system to be analyzed. For the sake of clarity, here only one class of fault event Σ_f and its corresponding class Σ_r of reset events will be considered.

2.2. Modeling of Intermittent Faults

In the literature pertaining to diagnosis of DES, faults are said to be intermittent when they are non-permanent, in the sense that each occurrence of fault is followed by its reset to the recovery behavior of the system within a finite delay. Such faults may be activated or deactivated by some external disturbance. Regarding the system status, an intermittent failure takes the system from a normal state to a faulty state (by the occurrence of the corresponding fault event), and then the system is taken again to a recover state within a finite delay (by the occurrence of the corresponding reset event).

In order to capture these changes in the system status, due to the various types of events, we use the supervision pattern Ω (Carvalho et al. 2012), shown in Figure 1, which is a label automaton that models the dynamic behavior of the system regarding intermittent faults. One can note that automaton Ω plays the role of the label function, which is usually used in fault diagnosis (Sampath et al. 1995).

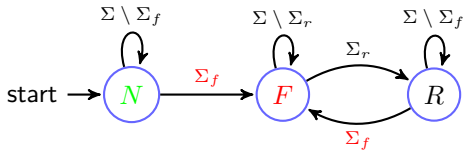

Figure 1: The label automaton Ω

Actually, when label automaton Ω is in state N (N for normal status), this means that the system executes a normal behavior, which indicates that no event from Σ_f has occurred yet. However, when a fault event occurs,

the label automaton Ω moves to state F (F for faulty status), and remains in that state for as long as the system executes a faulty behavior. When the fault is recovered, by the occurrence of a reset event, Ω switches to state R (for recovery status), where it stays as long as the system continues to execute a non-faulty behavior. As we deal with intermittent faults, the system can execute again a fault event. Then the label automaton Ω can return to state F.

In order to keep track of the occurrence of faults and their corresponding resets along the system's evolution, we compute automaton G_ℓ as the parallel composition of automata G and Ω ($G_\ell = G \parallel \Omega$). In fact, the states of G_ℓ are the states of automaton G enriched with labels N, F, or R. The following example illustrates these notions.

Example 1 *Consider automaton G, shown in Figure 2(a) and taken from (Contant et al. 2004). The sets of observable and unobservable events are $\Sigma_o = \{a, b, c, d\}$ and $\Sigma_u = \{f, r\}$, respectively. In addition, $\Sigma_f = \{f\}$ and $\Sigma_r = \{r\}$. Automaton $G_\ell = G \parallel \Omega$ is depicted in Figure 2 (b).*

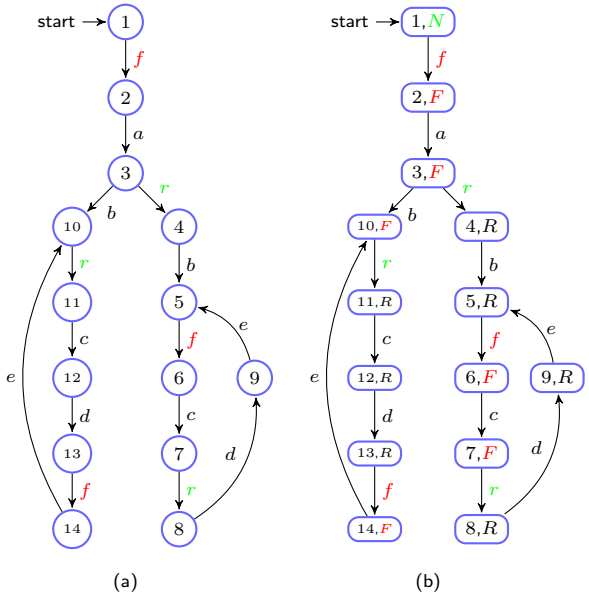

Figure 2: (a) Automaton G, (b) automaton G_ℓ of Example 1.

Regarding the diagnosis of intermittent faults, one can infer that the states of automaton G_ℓ can be partitioned into three subsets: 'Normal', 'Faulty' and 'Recovered', which can be identified using *fault-assignment function*:

$$\Psi : X \to \{N, F, R\}.$$

3. DIAGNOSABILITY OF INTERMITTENT FAULTS

3.1. Assumptions

Besides the well-known assumptions considered in the diagnosis of permanent faults (Sampath et al. 1995), i.e., that language $L(G)$ is live and no cycle of only unobservable events exists in G, the following assumptions are considered:

1. Each fault event σ_f has its corresponding reset event σ_r. Recall that both events are unobservable.

2. At least, one observable event exists between the occurrence of any fault event σ_f and its corresponding reset event σ_r and between the occurrence of any reset event σ_r and a new occurrence of fault event σ_f.

3. Each occurrence of fault event σ_f is followed by the occurrence of its corresponding reset event σ_r within a finite delay, and each occurrence of a reset event σ_r is followed by a new occurrence of the fault event σ_f within a finite delay. This assumption implies that fault and reset events occur with some regularity (pseudo-periodicity). These notions are called the $\Sigma_f - recurrence$ and $\Sigma_r - recurrence$, as introduced in (Contant 2005).

3.2. Diagnosability Definitions

Intermittent failures are dynamic (Contant 2005), that is, they can repeatedly occur and reset, and thus, the fault status evolves along with the system evolution. Consequently, several notions of diagnosability can be introduced, according to the properties and the specifications one may need to investigate. For example, one may want to ensure the detection of any fault occurrence or its corresponding recovery. Another definition would require checking the presence of each fault before its recovery or checking the recovery of any fault before a new occurrence of this fault. Determining accurately the finite delays in which the fault or its recovery can be diagnosed can also be of interest in practice. Obviously, the choice between these considerations greatly depends on the application nature of the system and the objectives assigned to the diagnosis activity.

In this section, we discuss two definitions of diagnosability: F-diagnosability, which ensures the detection of fault occurrence within a finite delay after its occurrence, and F_r-diagnosability, which ensures the detection of fault occurrences before their recovery. Note that, these definitions do not take into account the identification of the system status (i.e., whether the faulty status of the system is precisely known or not, when the fault is diagnosed).

Definition 1 *(F-diagnosability)*

An FSA G is said to be F-diagnosable w.r.t. projection function P, fault class Σ_f and reset event class Σ_r, if the following holds:

$(\exists\, n \in \mathbb{N}) \quad [\forall s \in \psi(\Sigma_f)] \quad (\forall t \in L/s) \quad [\| t \| \geq n \Rightarrow D_F]$
where diagnosability condition D_F is:

$$\forall \omega \in [P_L^{-1}(P(s.t))] \Rightarrow (\Sigma_f \in \omega)$$

F-diagnosability, where 'F' stands for fault occurrences, has the following meaning: for any event-trace s ending with a fault event in Σ_f, and t any continuation of s, then, $n \in \mathbb{N}$ exists such that, after the occurrence of at most n events, it is possible to detect the occurrence of the fault based on the captured observation. This implies that all the event-traces indistinguishable from $s.t$ have experimented, at least one fault from Σ_f.

Example 2 *Let us take automaton G of Example 1 (Figure 2). G contains one fault event f with its corresponding reset event r. Let us consider execution $\rho = 1, f, 2, a, 3, r, 4, b\ (5, f, 6, c, 7, r, 8, d, 9, e)^*$, the infinite event-trace, corresponding to this execution, is noted $s.t$ with, $s = farbf$ (one can see that $s \in \psi(\Sigma_f)$), and $t = (crdef)^*$. The resulting observed event-trace is then $P(s.t) = ab(cde)^*$. The only event-trace in G which shares the same observable event-trace with ρ is $\omega = fab(rcdfe)^*$. One can see that 4 events after executing the faulty event-trace s (2 observable events), it is possible to infer accurately the occurrence of fault f (since f occurs in all the event-traces which share the same observation with $s.t$). Thus, according to Definition 1, G is F-diagnosable (for $n \geq 4$).*

The above-mentioned definition ensures the detection of intermittent fault occurrence within a finite delay. However, it does take into account the detection of each occurrence of the intermittent fault before its recovery. Hereafter, we introduce a strong version of diagnosability that deals with this property.

Definition 2 *(F_r-diagnosability)*
An FSA G is said to be F_r-diagnosable w.r.t. projection function P, fault class Σ_f and reset event class Σ_r, if the following holds:
$[\forall s \in \psi(\Sigma_f)] \quad (\forall t \in L/s \wedge\ t \in \overline{\psi}(\Sigma_r)) \Rightarrow D_{F_r}$
where diagnosability condition D_{F_r} is:

$$(\forall \omega\omega'\omega'' \in L : \omega\omega' \in [P_L^{-1}(P(s))] \wedge \omega'' \in [P_L^{-1}(P(t))]) \\ \Rightarrow [\omega \in \psi(\Sigma_f) \wedge \Sigma_r \notin \omega'] \vee [\Sigma_f \in \omega'']$$

with F stands for 'fault occurrences', and $(_r)$ for 'detection before the fault recovery'.

The above definition means the following: let s be any finite event-trace in L that ends with a faulty event, t be any finite continuation that ends with a reset event. Condition D_{F_r} then requires that any finite event-trace that shares the same observation with $s.t$, shall experiment a faulty behavior between the moment of the fault occurrence (at the end of s) and its recovery (at the end of t), which ensures the fault detection.

Example 3 *Consider again, automaton G of Example 1 (Figure 2). Let us take the finite event-trace $s = farbf$ associated with finite execution $\rho = 1, f, 2, a, 3, r, 4, b, 5, f$. It is clear that s ends with faulty event f. Let event-trace $t = crd$, be the continuation of s until the reset of f. There exists, in automaton G, one event-trace $\omega = fabrcd$ associated to the finite execution $\rho' = 1, f, 2, a, 3, b, 10, r, 11, c, 12, d$ which shares the same observed event-trace with $s.t$, i.e., $P(\omega) = P(s.t) = ab(cd)$. However, according to Definition 2, the occurrence of f cannot be detected, in this case, before its recovery (i.e., no faulty state in ρ' is reached, between the occurrence of f and its reset r). Therefore, G is non-F_r-diagnosable.*

It should be noticed that F_r-diagnosability is stronger than F-diagnosability. Indeed, F_r-diagnosability requires the detection of any fault before its reset. However, F-diagnosability only requires the detection of the fault within a finite delay, without considering the different occurrences or resets of the fault. Thereby, it is straightforward to infer the following,

Proposition 1 *(Relation between definitions)*

- *F_r-diagnosability $\Rightarrow$ F-diagnosability*
- *non-F-diagnosability $\Rightarrow$ non-F_r-diagnosability*

Remark:
As we deal with intermittent faults, each fault occurrence is followed later on by its corresponding reset event. Then, it is also interesting to discuss the diagnosability of the recovery occurrence. Namely, this consists in checking whether we can detect that the system has moved to its recovery behavior after the fault has been recovered (and before a new occurrence of a fault event). These properties will not be discussed in this paper.

4. VERIFICATION OF DIAGNOSABILITY OF INTERMITTENT FAULTS

The procedure we suggest for analyzing diagnosability of intermittent failures will be carried out by combining the twin-plant construction method (Jiang et al. 2001), and some extensions, we develop, of the Model-Checking reformulation of diagnosability in the case of permanent failures as in (Cimatti et al. 2003; Boussif and Ghazel 2015). In this section, we first recall the twin-plant construction, and later we develop the necessary and sufficient conditions for each notion of diagnosability introduced in the previous section.

4.1. Twin-Plant Construction

The twin-plant, firstly introduced in (Jiang et al. 2001), simply consists of two synchronized copies of generator G' of system model G, while the synchronization is performed. Thus, any event-trace in the twin-plant corresponds to a pair of event-traces in the system model that share the same observation. More precisely, a path in the twin-plant corresponds to two *indistinguishable* traces in the system model.

To preserve the label tracking, we use the constructed generator G'_ℓ, instead of the constructed generator G'. Then, in order to generate a reduced state-space of twin-plant (by generating only the behavior of interest for fault diagnosis) we perform the synchronous composition $G'_\ell || G'_{\ell F}$, which is different from that in (Jiang et al. 2001). In fact, $G'_{\ell F}$ depicts only the co-accessible part of generator G'_ℓ from faulty states, i.e. it only contains the faulty event-traces. Thus, $G'_{\ell F} = (X_{oF}, \Sigma_o, \delta_{oF}, x_0)$, where X_{oF} is the set of states in G'_ℓ that are reachable by event-traces which contain at least one fault event. For more details about generating $G'_{\ell F}$, the reader can refer to (Moreira et al. 2011).

Definition 3 *(The reduced twin-plant)*

A reduced twin-plant of model G is an FSA $\mathcal{P} = \langle \mathcal{Q}, \Sigma_o, \Gamma, q_0 \rangle$, where:

- $\mathcal{Q} \subseteq \{(x, x') \,|\, x \in X_o, x' \in X_{oF}\}$ *is the set of states.*
- Σ_o *the set of the (observable) events.*
- $\Gamma \subseteq \mathcal{Q} \times \Sigma_o \times \mathcal{Q}$ *is the partial transition relation s.t.* $(q, \sigma, q') \in \Gamma$, *with* $q = (x_1, x_2)$, *and* $q' = (x'_1, x'_2)$ *if and only if* (x_1, σ, x'_1), $(x_2, \sigma, x'_2) \in \delta_{oF}$.
- $q_0 = (x_0 \times x_0) \in \mathcal{Q}$ *is the initial state.*

It is worthwhile recalling that constructing the twin-plant can be performed in $(\mathcal{O}(|X|^4 \times |\Sigma_o|))$ (Jiang et al. 2001).

As the reduced twin-plant is performed directly on constructed generator G'_ℓ, then label tracking is preserved and therefore, the *fault-assignment* function is extended as follows:

$$\Psi : (X_o, X_o) \rightarrow (\{N, F, R\} \times \{N, F, R\})$$

Hence, different types of states can be distinguished in the reduced twin-plant. Hereafter, only state types, which will be used in the sequel, for developing necessary and sufficient conditions for diagnosability, are defined.

In order to simplify the notations, we introduce labels $\overline{N}, \overline{F}$ and $\overline{R}$ which mean respectively that the label is different from N (i.e., it can be F or R), different from F (i.e., it can be N or R) and different from R (i.e., it can be N or F).

Definition 4 *(Twin-plant state types)*

We define the following state types,

- N*-state (resp.* F*-state,* R*-state): is a state* $q = (x, x') \in \mathcal{Q}$, *such that* $\Psi(q) = (N, N)$ *(resp.* $\Psi(q) = (F, F)$, $\Psi(q) = (R, R)$*).*
- NF*-state (resp: is a state* $q = (x, x') \in \mathcal{Q}$, *such that* $\Psi(q) = (N, F)$. FN*-state is defined similarly.*
- $F\overline{F}$*-state: is a state* $q = (x, x') \in \mathcal{Q}$, *such that* $\Psi(q) = (F, \overline{F})$.
- $F1$*-state (resp.* $R1$*-state,* $N1$*-state): is a state* $q = (x, x') \in \mathcal{Q}$, *such that* $\Psi(q) = (F, \triangle)$ *(resp.* $\Psi(q) = (R, \triangle)$, $\Psi(q) = (N, \triangle)$*),with* $\triangle \in \{N, F, R\}$.
- $\overline{F}1$*-state: is a state* $q = (x, x') \in \mathcal{Q}$, *such that* $\Psi(q) = (\overline{F}, \triangle)$.

One can underline that the twin-plant has an interesting feature, which is the symmetric property. It means that each path in the twin-plant has its symmetric path (e.g., a path containing $\overline{F}F$-states has its symmetric path which contains the symmetric $F\overline{F}$-states, and vice versa). In the following section, we take into account this property for developing the necessary and sufficient conditions.

4.2. Necessary and sufficient conditions for F-diagnosability

In a previous work (Boussif and Ghazel 2015), we have dealt with diagnosability of permanent faults using a twin-plant-based structure in model-checking framework. The necessary and sufficient condition for diagnosability was the absence of *"infinite critical pairs"* in the constructed twin-plant. This means the absence of cycles which are composed only of FN(or NF)-states. In the same way, we formalize a necessary and sufficient condition for the diagnosability of intermittent faults. In order to do so, we need to introduce the following definition,

Definition 5 (F**-confused cycle**)

It is a cycle $\pi = (q_1, q_2, \ldots, q_n, q_{n+1} = q_1)$, *in the twin-plant, s.t.* $\forall\, 1 \leq i \leq n$, q_i *is an* $N1$*-state, and* $\exists\, 1 \leq j \leq n$, *s.t.* q_j *is an* NF*-state.*

An F*-confused cycle* in twin-plant corresponds to two cycles on the system model (automaton G) which generate the same observed event-trace, such that the first one has no fault event (a fault-free cycle) and the second one contains, at least, one fault event (which is depicted by the existence of an NF-state).

Figure 3 shows a path that contains a configuration of an F-confused cycle represented by states q_2, q_3, q_4, q_5.

After having set up the necessary notions, we now establish the necessary and sufficient conditions for F-diagnosability.

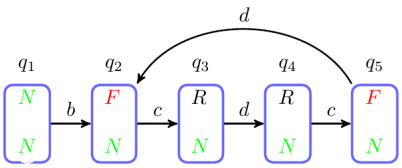

Figure 3: An F-confused cycle

Theorem 1 (F-*diagnosability*)

A system model G is F-diagnosable, w.r.t projection function P, class of fault events Σ_f and its corresponding class of reset events Σ_r, if and only if no F-confused cycle exists in its corresponding twin-plant.

Proof 1 *($\Rightarrow$) Assume that $L(G)$ is F-diagnosable but there exists an F-confused cycle in its corresponding twin-plant: $q_1, \sigma_1, q_2, \ldots, q_n, \sigma_n, q_1$, $n \geq 1$. Such a cycle corresponds to two cycles in G'_ℓ: $c\ell = x_1^1, \sigma_1 x_2^1, \ldots, x_n^1, \sigma_n, x_1^1$ and $c\ell' = x_1^2, \sigma_1, x_2^2, \ldots, x_n^2, \sigma_n, x_1^2$. Let $t = v_1, \sigma_1, v_2, \sigma_2, \ldots, v_n, \sigma_n$ and $t' = v'_1, \sigma_1, v'_2, \sigma_2, \ldots, v'_n, \sigma_n$ be the event-traces that correspond to cyclic executions $c\ell$ and $c\ell'$ in G s.t. $\forall\ i \leq n, v_i, v'_i \in \Sigma_u^*$. (i.e., $P(t) = P(t') = \sigma_1, \sigma_2, \ldots, \sigma_n$).*

By construction of the twin-plant, $\exists\, s_0, s'_0 \in L(G)$, s.t.

$$[P(s_0) = P(s'_0)] \wedge [\delta(x_0, s_0) = x_1^1] \wedge [\delta(x_0, s'_0) = x_1^2] \wedge [\Sigma_f \notin s_0].$$

(the last condition $\Sigma_f \notin s_0$ is due to the fact that x_i^1 is an N-state $\forall 1 \leq i \leq n$). Also, according to Definition 5, $\Sigma_f \in t'$ and $\Sigma_f \notin t$. Thus, one can consider $t' = t'_1.t'_2$ such that $t_1'^{|t'_1|} \in \Sigma_f$ (i.e., t'_1 ends with a fault event). Now, let us consider $s = s'_0.t'_1$, then $s \in \psi(\Sigma_f)$. Thus, for any $n \in \mathbb{N}$ let us take $t''_n = t'_2 t'^n \in L/s$, then $(|t''_n| \geq n)$ and $(\exists \omega_n = s_0.t^{n+1})$ such that $[\omega_n \in P^{-1}P(st''_n)] \wedge [\Sigma_f \notin \omega_n]$.

Therefore, F-diagnosability definition is violated.

($\Leftarrow$) Assume that twin-plant $\mathcal{P}$ is F-confused-cycle-free and suppose that automaton G is non-F-diagnosable. Then,
$(\forall n \in \mathbb{N})(\exists\, s \in \psi(\Sigma_f))\ (\exists\, t \in L(G)/s)$ such that:
$[\![\ t\ ]\!] \geq n] \wedge [(\exists\, \omega \in P^{-1}P(s.t)) \wedge [\Sigma_f \notin \omega]]$
Let us pick any $n \geq |X|^2$, and $\omega \in L(G)$ such that $P(\omega) = P(s.t) = \sigma_1, \sigma_2, \ldots, \sigma_k$, with $k \in \mathbb{N}$. By constructing twin-plant $\mathcal{P}$ of G, we have a path $\pi = q_0, \sigma_1, q_1, \ldots, \sigma_k, q_{k+1}$, $k \leq |s.t|$ that corresponds to executions ω and $s.t$.

As $|t| \geq n \geq |X|^2$, it is clear that executions corresponding to $s.t$ and ω contain cycles (Jiang et al. 2001). Thus, $\exists 0 \leq i \leq k'$, with $k' \leq k$ such that $c\ell \in \pi$, with $c\ell = q_i, \sigma_{i+1}, q_i, \ldots, q_{k'-1}, \sigma_{k'}, q_i$ (i.e., a cycle $c\ell$ exists in π).

Since $\Sigma_f \notin \omega$, then $\forall q \in c\ell$, q is an $N1$-state. Moreover, $\Sigma_f \in s$ (since $s \in \psi(\Sigma_f)$). According to assumption 3, the fault event occurs and reset regularly. Then, $\exists\ i \leq k'$ s.t. q_i is an NF-state. Thus, $c\ell$ is an F-confused cycle, according to Definition 3, which contradicts our assumption. □

4.3. Necessary and Sufficient Conditions for F_r-diagnosability

It should be noticed that, as stated in Proposition 1, the non-F-diagnosability implies the non-F_r-diagnosability. Then, it is straightforward that the necessary and sufficient condition for F-diagnosability (i.e, the absence of an F-confused cycle in the twin-plant) is also a necessary condition for F_r-diagnosability. Hereafter, the notion of F-indicating sequence is introduced, such a notion will be used for stating the necessary and sufficient condition of F_r-diagnosability.

Definition 6 (F-**indicating sequence**)

It is a finite path $\pi = (q_1, q_2, \ldots, q_n)$, such that q_1 is an $\overline{F}1$-state, $\forall\, 1 < i < n$, q_i is $F\overline{F}$-state, and q_n $(n > 2)$ is an $R1$-state.

Figure 4 shows a path that contains a configuration of an F-indicating sequence represented by states q_1, q_2, q_3, and q_4.

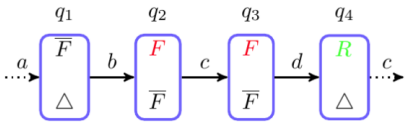

Figure 4: An F-indicating sequence

Theorem 2 (F_r-*diagnosability*)
A system model G is F_r-diagnosable, w.r.t a projection function P, a class of fault events Σ_f and its corresponding class of reset events Σ_r, if and only if no F-indicating sequence exists in its corresponding twin-plant.

Proof 2 *The proof of Theorem 2 is not given, but it is developed in the same manner as for F-diagnosability by adopting an absurd reasoning.*

5. DIAGNOSABILITY VERIFICATION USING MODEL-CHECKING

5.1. Model-Checking

Model-Checking is an automatic formal verification technique that is widely applied for the design of complex dynamic systems (Clarke et al. 1999). It allows for verifying whether the system behavior (modeled by a Kripke Structure) satisfies a given property expressed as a temporal logic formula or not, using efficient algorithms based on exhaustive exploration of the

system state-space. A counter-example is generated if the system does not satisfy the property, which is an interesting feature namely for debugging.

In order to use Model-Checking for verifying diagnosability of permanent faults, (Cimatti et al. 2003) proposed a method to formulate the diagnosability issue as a Model-Checking problem using a CTL/LTL temporal logic formulas and the Kripke structure that corresponds to the twin-plant of the system. Hereafter, we reformulate the necessary and sufficient conditions for the different definitions in the same way as in (Cimatti et al. 2003).

5.2. The Kripke Structure

In simple terms, a Kripke structure is a non-deterministic state/transition system with atomic propositions assigned to the states (or the actions). Each state of the Kripke structure represents some possible configuration of the system, while a labeling function associates with each state the properties holding in it. Thus, in order to formulate a twin-plant as a Kripke structure, one can simply encode states (of the two copies of the system) and the observed events of the twin-plant in the state space of the Kripke structure, i.e., a state in the Kripke structure is defined as a vector (x_1, x_2, σ), where x_1, x_2 are the states of the system copies and σ is a feasible (observable) event from both x_1 and x_2. The labeling function associated with each state in the Kripke structure takes one proposition from $\{N, F, R\} \times \{N, F, R\}$.

5.3. Linear Temporal Logic (LTL)

Temporal logics allow one to formally express properties to be satisfied by a model. LTL is a temporal logic that reasons over linear traces of Kripke structure through time. At each time instant, there is only one real future timeline that is considered. Conventionally, that timeline is defined as starting "now", in the current time step, and progressing infinitely in the future. LTL formulas may contain a finite number of atomic propositions, Boolean connectives $\neg, \wedge, \vee$, and temporal connectives: 'X' that means '$neXt$', 'G: $Globally$','R: $Release$', 'F: *in the Future*','U: $Until$'.

5.4. Diagnosability Conditions as LTL Formulas

In order to formulate the two notions of diagnosability as Model-Checking problems, we first formulate each diagnosability condition as an LTL formula. For the sake of simplicity, we introduce these atomic propositions: $N1$, NF, $\overline{F}1$, $F\overline{F}$, and $R1$, which mean respectively: the state q is an $N1$-state, NF-state, $\overline{F}1$-state, $F\overline{F}$-state, and $R1$-state.

5.4.1. F-diagnosability as a Model-Checking problem: The LTL formula which characterizes each state of an F-confused cycle in the twin-plant is,

$$\boxed{\phi_1 : \; G(\,\mathbf{N1} \wedge F\,\mathbf{NF}\,)}$$

The specification can be read as follows: " *for the considered infinite path in the twin-plant, all states, from the current state, are $N1$-states and at least one state is an NF-state*". Therefore, F-diagnosability can be expressed as follows:

$$\boxed{K_{\mathcal{P}}, S_{\mathcal{P}} \models \neg\, F\, G(\,\mathbf{N1} \wedge F\,\mathbf{NF}\,)}$$

where $K_{\mathcal{P}}$ is the Kripke structure corresponding to the twin-plant $\mathcal{P}$ of G, and $S_{\mathcal{P}}$ is the initial state in $K_{\mathcal{P}}$.

5.4.2. F_r-diagnosability as a Model-Checking problem: The LTL formula which characterizes the first state of any F-indicating sequence in the twin-plant is,

$$\boxed{\phi_1 : \; \overline{F}1 \wedge \mathbf{X}\; F\overline{F} \wedge \mathbf{X}\; [F\overline{F}\;\mathbf{U}\; R1]}$$

The specification can be read as follows: "*for the considered path in the twin-plant, the current state is an $\overline{F}1$-state, the successor state is an $F\overline{F}$-state and the successors states are $F\overline{F}$-states until an $R1$-state* is reached".

The Model-Checking problem which expresses F_r-diagnosability is:

$$\boxed{K_{\mathcal{P}}, S_{\mathcal{P}} \models \neg\, \mathbf{F}\; (\,\overline{F}1 \wedge \mathbf{X}\; F\overline{F} \wedge \mathbf{X}\; [F\overline{F}\;\mathbf{U}\; R1]\,)}$$

where $K_{\mathcal{P}}$ is the Kripke structure corresponding to the twin-plant $\mathcal{P}$ of G, and $S_{\mathcal{P}}$ is the initial state in $K_{\mathcal{P}}$.

6. A RAILWAY CASE-STUDY

In order to evaluate the efficiency and the scalability of the proposed approach, we experiment a railway case-study: a Level Crossing benchmark (Liu 2014). For verification, we use the symbolic model-checker nuXmv (version 1.0) (Bozzano et al. 2014), which is a symbolic model-checker for analyzing of synchronous finite-state and infinite-state systems. It is an extension of the existing NuSMV model-checker with some new interesting features. Both are originated from the re-engineering, re-implementation and extension of the CMU SMV Tool. Its main advantage is the integration of techniques based on the "*Satisfiability Modulo Theory* (SMT)", implemented through a tight integration with MathSAT5.

6.1. Level Crossing Benchmark

A Level Crossing (LC) is an intersection where a railway line intersects with a road or path at the same level. It is composed of three subsystems: the railway traffic, the LC controller and the barriers, these subsystems are modeled by Labeled Petri Net (LPN). The global system is established using some shared places and transitions between these sub-models.

An interesting feature of this benchmark is that it can be extended to n railway tracks in order to obtain

larger models and assess the scalability of the used techniques. For more details about modeling, function and the development of the benchmark, the reader can refer to (Ghazel and Liu 2016).

The global single-line LC model is depicted in Figure 5. Transitions in green squares are observable and the others are unobservable, where the red one is a fault event from the fault class Σ_f, the yellow one is a reset event from Σ_r corresponding to fault class Σ_f, and the grey one is a normal unobservable event. Actually, the fault event is pertaining to train-sensors along the track and may cause the arrival of the train into the LC intersection zone before the barriers are ensured to be lowered.

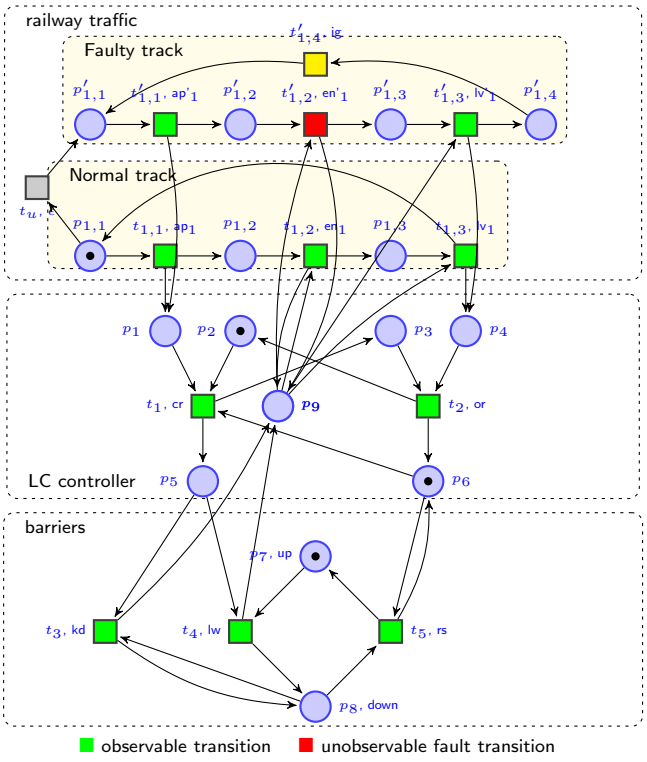

Figure 5: Level Crossing Benchmark

As the LC system is modeled by an LPN, we first generate its reachability graph with the help of TINA Tool (which represents our input automaton G) and then, perform our technique based on the generated reachability graph. In order to assess the scalability, we increase the number of railway track k progressively.

The diagnosability verification will be then performed on the obtained reachability graph. Before proceeding with tests, we construct the twin-plant as a Kripke structure. It is described as a synchronous composition of two copies of a system modules instead of enumerating the whole model. The verification task is conducted as follows, we first check F-diagnosability. If the specification is satisfied by the Kripke structure corresponding to the twin-plant model (which means, that an F-confused cycle exists in the twin-plant and will be directly generated by the model-checker as a counter-example), then the system is not F-diagnosable. In this case, we can directly decide about the F_r-diagnosability (the system is not F_r-diagnosable), without proceeding to check its corresponding specifications, since the non-F-diagnosability implies the non-F_r-diagnosability as stated in proposition 1. In the other case, we proceed by checking F_r-diagnosability as done with F-diagnosability.

6.2. Results and Discussion

All experiments were conducted on a 64-bit PC, Ubuntu 14.04 operating system, an Intel Core i5, 2.5 GHz Processor with 4 cores and 6 GB RAM.

Table 1 shows the obtained results. Columns from left to right correspond to: n: the numbers of railway tracks, G_S: the obtained number of states in automaton G (i.e., the reachability graph of the LPN model); G_T: the number of transitions in G; $\mathcal{P}_S$: the number of obtained states in twin-plant $\mathcal{G}$ at the end of the analysis; $t_{\mathcal{P}}$: the time needed to generate twin-plant (as a Kripke structure); $Diag_F$: the F-diagnosability verdict; t_{Diag_F} : the time needed to conclude about F-diagnosability; $Diag_{F_r}$: the F_r-diagnosability verdict and finally, $t_{Diag_{f_r}}$: the time needed to conclude about F_r-diagnosability.

The analysis of the different notions of diagnosability shows that diagnosability properties are violated by the model whatever the number of tracks, which means that the LC benchmark is neither F-diagnosable nor F_r-diagnosable. Based on the generated counter-examples, one can conclude that this is due to the fact that two scenarios exists in the model that generate the same observation (i.e, the same subsequent firing sequence). these two scenarios correspond to : (1) a train-sensing failure occurs and then the train does never leave the intersection zone, and (2) the train stops indefinitely before reaching the intersection zone (See (Ghazel and Liu 2016)).

n	G_S	G_T	$\mathcal{P}_S$	$t_{\mathcal{P}}$	$Diag_F$	t_{Diag_F}	$Diag_{F_r}$	$t_{diag_{F_r}}$
1	26	39	123	0.1 s	non	0.27 s	non	0.17 s
2	271	683	3536	1.42 s	non	5.80 s	non	11.23 s
3	1938	6771	144210	28,63 s	non	51.29 s	non	83.50 s
4	2542	9293	3053060	40,29 s	non	1274.59 s	non	1846 s

Table 1: Experimental results for n-LC models.

Regarding the scalability of the approach, one can observe that the size of the marking graph (G_S and G_T) significantly increases with the dimension of the net, namely with the number of tracks (n) in the benchmark. It is not surprising that the number of the reachable states ($\mathcal{P}_S$) of the Kripke structure highly increases with the size of the reachability graph, since it is very sensitive to combinatorial explosion. Indeed, as we use a *symbolic Model-Checker*, it is difficult to conclude about the evolution of the state-space since it depends on variable and transition orderings and clustering (which is not taken into account in our experiments). It should be stressed that no reduction or optimization techniques have been used to perform the analysis.

Finally, three remarks relatively to the elapsed times for generating the twin plant and verifying diagnosability, can be emphasized:

1. The Model-Checker spends more time in verification than in generating the twin plant.

2. Elapsed times for generating the twin plant and verifying diagnosability stay in the order of seconds until 3-tracks, then it increases significantly.

3. More time elapsed for verifying F_r-diagnosability than F-diagnosability. This result is logical, since the LTL formula that expresses F_r-diagnosability contains more temporal connectives than the LTL formula expressing F-diagnosability (i.e., the model-checking techniques are sensitive to the size of the properties).

7. CONCLUSION

In this paper, we propose an approach to analyze diagnosability of intermittent faults in DESs using a model-checking framework. System modeling, faults modeling, and two notions of diagnosability with their corresponding necessary and sufficient conditions have been discussed. Diagnosability issues were then formulated as LTL Model-Checking problems based on the twin-plant construction. The effectiveness and scalability of the proposed approach are experimentally evaluated through a railway case-study.

This work falls within the scope of our activities on reformulating issues related to fault diagnosis in a model-checking framework. We have already studied the case of permanent failures and we wish, on one hand, to extend our study to deal with more complex classes of faults such as repeated failures, supervision patterns in untimed and timed domains. On the other hand, we intend to evaluate the efficiency of advanced formal verification techniques such as abstraction techniques, SAT analysis, on-the-fly model-checking and also develop optimization techniques for the diagnosability analysis process, in such a way as to be able to deal with large systems.

REFERENCES

J. Zaytoon and S. Lafortune. Overview of fault diagnosis methods for discrete event systems. *Annual Reviews in Control*, 37:308–320, 2013.

M. Sampath, R. Sengupta, and S. Lafortune. Diagnosability of discrete-event systems. *IEEE Transactions on Automatic Control 40(9)*, pages 1555–1575, 1995.

S. H. Zad, R .H. Kwong, and W. M. Wonham. Fault diagnosis in discrete-event systems: Framework and model reduction. *IEEE Transactions on Automatic Control*, 48(7):1199–1212, 2003.

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

A. Schumann and Y. Pencolé. Scalable diagnosability checking of event-driven system. *20th International Joint Conference on Artificial Intelligence*, pages 575–580, 2007.

T. S. Yoo and S. Lafortune. Polynomial-time verification of diagnosability of partially observed discrete-event systems. *IEEE Transactions on automatic control*, 47 (9):1491–1495, 2002.

E. M . Clarke, O. Grumberg, and D. Peled. Model Checking, The MIT Press Cambridge, MA, 1999.

A. Cimatti, C. Pecheur, and R. Cavada. Formal verification of diagnosability via symbolic model checking. *Int. Conference on Artificial Intelligence*, pages 363–369, 2003.

G. Bourgne, P. Dague, F. Nouioua, and N. Rapin. Diagnosability of input output symbolic transition systems. *1st Int. Conference on Advances in System Testing and Validation Lifecycle*, pages 147–154, 2009.

A. Boussif and M. Ghazel. Diagnosability analysis of input/output discrete event system using model checking. *The 5th International Workshop on Dependable Control of Discrete Systems (DCDS'15)*, 48(7):71–78, 2015.

A. Grastien. Symbolic testing of diagnosability. *20th International Workshop on Principles of Diagnosis (DX-09)*, 2009.

F. Peres and M. Ghazel. An operative formulation of the diagnosability of discrete event systems using a single logical framework. *The 8th Int. Workshop on Verification and Evaluation of Computer and communication Systems*, pages 1–11, 2014.

A. Grastien and A. Anbulagan. Diagnosis of discrete-event systems using satisfiability algorithms. *Proceedings of the National Conference on Artificial Intelligence*, 2007.

A. Grastien. Incremental diagnosis of des by satisfiability. *Proceedings of the conference on ECAI*, 2008.

S. Jiang, R. Kumar, and H. E. Garcia. Diagnosis of repeated / intermittent failures in discrete event systems. *IEEE Transactions on Robotics and Automation*, 19(2):310–323, 2003.

T. S. Yoo and H. E. Garcia. Event diagnosis of discrete-event systems with uniformly and nonuniformly bounded diagnosis delays. *American Control Conference*, 6:5102–5107, 2004.

C. Zhou and R. Kumar. Computation of diagnosable fault-Occurrence indices for systems with repeatable-faults. *IEEE Transactions on automatic control*, 54 (7):1477–1490, 2009.

O. Contant. Failure diagnosis of discrete event system: the case of intermittent faults. *International conference on decision and control*, 4:4006–4017, 2002.

O. Contant. On monitoring and diagnosing classes of discrete event systems. *PhD Thesis, University of Michigan*, 2005.

A. Correcher, E. Garcia, F. Morant, and E. Quiles. Intermittent failure diagnosis in industrial processes. 2:723–728, 2003.

S. Soldani, M. Combacau, A. Subias, and J. Thomas. Intermittent fault diagnosis: a diagnoser derived from the normal behavior. *International workshop principles diagnosis*, pages 391–399, 2007.

S. Soldani, M. Combacau, A. Subias, and J. Thomas. Intermittent fault detection through message exchanges: a coherence based approach. *International workshop principles diagnosis*, pages 251–257, 2006.

S. Biswas. Diagnosability of discrete event systems for temporary failures. *Computers & Electrical Engineering*, 38(6):1534–1549, 2012.

D. Guanqian, Q. Jing, L. Guanjun, and L. Kehong. A discrete event systems approach to discriminating intermittent from permanent faults.

C.G. Cassandras and S. Lafortune. *Introduction to discrete event systems*. 2008.

L.K. Carvalho, J.C. Basilio, and M. V. Moreira. Robust diagnosis of discrete event systems against intermittent loss of observations. *Automatica*, 48(9): 2068–2078, 2012.

O. Contant, S. Lafortune, and D. Teneketzis. Diagnosis of intermittent faults. *Discrete Event Dynamic Systems: Theory and Applications*, 14(2):171–202, 2004.

M.V. Moreira, T.C. Jesus, and J.C. Basilio. Polynomial time verification of decentralized diagnosability of discrete event systems. *IEEE Transactions on Automatic Control*, 56(7):1679–1684, 2011.

B. Liu. An efficient approach for diagnosability and diagnosis of DES based on LPN. *PhD thesis, Univ. de Lille1*, 2014.

M. Bozzano, R. Cavada, A. Cimatti, M. Dorigatti, A. Griggio, and A. et al. Mariotti. nuXmv 1.0 User manual. 2014.

M. Ghazel and B. Liu. A customizable benchmark to deal with fault diagnosis issues in DES. *13th International Workshop on Discrete Event System (WODES 2016)*, 2016.