Discrete Event Dynamic Systems manuscript No.
(will be inserted by the editor)

Diagnosability of Repairable Faults

Eric Fabre · Loïc Hélouët · Engel Lefaucheux · Hervé Marchand

Received: date / Accepted: date

Abstract The diagnosis problem for discrete event systems consists in deciding whether some fault event occurred or not in the system, given partial observations on the run of that system. Diagnosability checks whether a correct diagnosis can be issued in bounded time after a fault, for all faulty runs of that system. This problem appeared two decades ago and numerous facets of it have been explored, mostly for permanent faults. It is known for example that diagnosability of a system can be checked in polynomial time, while the construction of a diagnoser is exponential. The present paper examines the case of transient faults, that can appear and be repaired. Diagnosability in this setting means that the occurrence of a fault should always be detected in bounded time, but also before the fault is repaired, in order to prepare for the detection of the next fault or to take corrective measures while they are needed. Checking this notion of diagnosability is proved to be PSPACE-complete. It is also shown that faults can be reliably counted provided the system is diagnosable for faults and for repairs.

1 Introduction

The diagnosis problem for discrete event systems appeared two decades ago [22]. In its standard version the problem assumes a dynamic system A with runs of two types : some runs are safe (e.g. they contain no fault event), and the others are faulty. More generally, one may assume a regular property P on runs of A. This property P is absorbing, in the sense that once P is satisfied by some partial run (like the fact of being faulty) it remains true in all extensions of that run. System A is supposed to perform some hidden run u, which is partially observed by an external supervisor : only some events of the hidden run u are visible, possibly through some filtering operation, and the other events of u are silent. The problem then consists in deciding whether the hidden run u satisfies the property P of interest given the observed sequence and the model of A. Specifically, assuming that u satisfies P at some instant t, one would like

INRIA Rennes, Campus de Beaulieu, 35042 Rennes cedex, France name.surname@inria.fr

to detect that P holds in bounded time after t. If this is feasible for all runs satisfying P, the system is declared diagnosable.

Property P can be seen as an abstraction on the behaviors of A, which are partially observed. This is certainly the simplest possible abstraction, as it separates runs of A into two (regular) classes. Diagnosability then amounts to detecting when P holds with bounded delay. A dual version of the problem relates to opacity [1,6]: one would like to ensure that property P (a "secret") is never detectable by an external observer. Diagnosability and opacity are thus two dual facets of the same problem: the observability of a hidden bit of information in a run of a dynamic system. Beyond its simple statement, the diagnosis problem thus readily has numerous implications in terms of security and of safety. But moslty, it is the simplest milestone toward the analysis of observability problems for more complex abstractions over runs of A.

Numerous facets of the diagnosis problem have been explored since its first introduction. A wide variety of models have been considered, covering automata [21], Petri nets [7,11], concurrent systems [2], visibly pushdown automata [13]... It has been extended to stochastic systems [25,3,4], to infinite runs [12], to decentralized and distributed settings. We refer the reader to [27] for a thorough review of these different settings.

In this article, we examine the case of non-persistent properties P, or non-persistent faults, *i.e.* P may hold only on segments of the hidden run u. Diagnosing P thus means being able to detect that P holds in bounded time after it becomes true, and in any case *before P vanishes*, *i.e.* before the fault is repaired or becomes irrelevant. A first natural motivation for this choice is that a repair action should be triggered in time, so detection must not be late. In terms of opacity, fault repair corresponds to a secret disclosure, *i.e.* declaring that the secret part of a run is no longer relevant, so it becomes useless for an attacker to guess this outdated secret. More importantly, the choice of a timely detection prepares the ground for a counting of faults, which is central to evaluate the reliability of a system. Indeed, being able to count occurrences of some events or faults allows one to detect whether a system works properly *i.e.* using metrics such as error rates. This can be a way to detect security breaches such as service denial attacks. In a similar way, counting the number of times a secret is disclosed provides a measure of the system's opacity. In the same spirit, making elements of the system observable at a regular pace gives opportunities to create covert information flows [17,19].

The contributions of the paper are the following: We first formalize the notions of non permanent properties such as repairable faults, and the notion of diagnosability for repairable faults, named T-diagnosability. The systems considered in this paper are partially observed automata, where states are tagged with the truth status of a property P (for simplicity, one can consider this property as a *fault*). T-diagnosability is defined as a property of runs of the automaton: a system is T-diagnosable if runs ending with a fault can not be extended in such a way that they remain observationally equivalent to non-faulty runs for an arbitrary long time, or until the fault is repaired. While the standard setting to decide diagnosability for persistent faults has a quadratic complexity [14], being able to track non-persistent faults is surprisingly much more complex: deciding T-diagnosability is PSPACE-complete. We then propose a T-diagnosability test based on a product between the automaton of the

system and the determined version of its observation, and on the characterization of some particular runs on this machine. T-diagnosability makes sense only in a context where transitions from a non-faulty status to a faulty one (and conversely) can be observed. We hence impose that the considered systems do not contain *vanishing* faults or repairs, i.e. that successive changes of the status of the considered property P necessarily occur along runs that contain an observation. We show that detecting whether a system has vanishing faults or repair is in NLOGSPACE, and propose a technique to transform a system with vanishing faults into a new system in which observations occur between status changes.

T-diagnosability is a first step towards counting fault occurrences. We show that surprisingly, being able to detect the occurrence of faults before they vanish is not sufficient to enable counting them! Counting needs more, namely that both faults are repairs are T-diagnosable. We formalize the ability of counting fault occurrences, and show that deciding whether faults of a system without vanishing faults can be counted is an NLOGSPACE problem. This result is obtained by a twin-plant construction [14], that memorizes in addition to a pair of followed path the difference between the number of faults that have occurred. If this differences exceeds 1, then one can not count fault occurrences in this system.

T-diagnosability is then compared to previous contributions on the topic of non-persistent faults, and in particular to the notion of P-diagnosability introduced by [9]. P-diagnosability is slightly relaxed version of our notion of T-diagnosability, as it does not impose that detection takes place before the fault vanishes. We highlight differences between the notions of P-diagnosability and T-diagnosability. P-diagnosability was already known to be in PSPACE [9], but no lower bound was known for this problem. We close this gap by showing that deciding P-diagnosability is also a PSPACE-complete problem.

The paper is organized as follows. Section 2 briefly recalls standard results about the classical notion of diagnosability. Section 3 introduces a notion of T-diagnosability for repairable faults, shows that deciding T-diagnosability is PSPACE-complete, and gives solutions to detect and suppress vanishing faults. Section 4 addresses the counting of faults in a partially observed run. Section 5 relates these results to previous contributions on the topic. Section 6 shows possible extensions of this work, and concludes.

This paper extends a communication at WODES'16 [10]. New results cover in particular the detection and the removal of vanishing faults and repairs in a model, a detailed comparison of T-diagnosabiliy and P-diagnosability, the proof of the PSPACE completeness of deciding T-diagnosability, and proofs in Section 4 for the counting of faults.

2 Setting and known results

2.1 Diagnosis and diagnoser

Let Σ be a finite alphabet. A *word* over Σ is an element $v \in \Sigma^*$. We denote by ε the empty word. The length of a word $v \in \Sigma^*$ is denoted $|v|$. Given a pair of words

$v, v' \in \Sigma^*$, we will write $v \leq v'$ when v is a prefix of v', i.e. $v' = vv$". We will also write $v < v'$ when $v \leq v'$ and $|v| < |v'|$. A finite automaton over alphabet Σ is a tuple $A = (S, \Sigma, T, s_0)$, where S is a finite set of states, $s_0 \in S$ is the initial state, and $T \subseteq S \times \Sigma \times S$ is a set of transitions. Transitions take the form $t = (s, \alpha, s')$ and we denote by $s^-(t) = s$ the origin of transition t, $\sigma(t) = \alpha$ the label attached to transition t, and by $s^+(t) = s'$ the target state of transition t. *Paths* of A are finite sequences of transitions $u = t_1 ... t_n$ such that $s^+(t_i) = s^-(t_{i+1})$, and *runs* of A are paths rooted at s_0: $s^-(t_1) = s_0$. Functions s^+, s^-, σ extend from transitions to paths : we denote $s^-(u) = s^-(t_1), s^+(u) = s^+(t_n)$, and $\sigma(u) = \sigma(t_1)...\sigma(t_n)$ the sequence of labels associated to a path u. A path $u = t_1 ... t_n$ is *reachable* iff there exists a run u' of A such that $s^+(u') = s^-(u)$. A path u is a *cycle* iff $s^+(u) = s^-(u)$. The *language* of A is the set of label sequences produced by runs of A : $L(A) = \{\sigma(u),\ u \text{ run of } A\}$. An automaton is *deterministic* iff $\forall s, \alpha, (s, \alpha, s') \in T \wedge (s, \alpha, s'') \in T \Rightarrow s' = s''$.

Our starting point for the diagnosis problem, and without loss of generality, is a *deterministic* automaton A. Let us partition states of A in two subsets $S = S_N \uplus S_F$, and let us name S_N normal (or safe) states and S_F faulty states, to help intuition. The *faulty language* of A is derived from *faulty runs*, *i.e.* runs that terminate in a faulty state: $L_F(A) = \{\sigma(u),\ u \text{ run of } A, s^+(u) \in S_F\}$. The *normal (safe) language* of A is denoted $L_N(A)$, and is defined in similar way as $L_N(A) = \{\sigma(u),\ u \text{ run of } A, s^+(u) \in S_N\}$. Obviously, we have $L_N(A) \subseteq L(A)$ and $L_F(A) \subseteq L(A)$. As A is deterministic, σ establishes a one to one correspondence between runs of A and words of its language $L(A)$, so $L_N(A) \cap L_F(A) = \emptyset$, or $L_N(A) \uplus L_F(A) = L(A)$. In this section, we assume that faults are permanent in A. Namely, there is no reachable path u in A such that $s^-(u) \in S_F$ and $s^+(u) \in S_N$. Equivalently, the faulty language of A is *saturated* in $L(A)$: $L_F(A)\,\Sigma^* \cap L(A) = L_F(A)$.

The diagnosis problem assumes partially observed systems, so we partition the label set Σ into two disjoint sets of observable and unobservable labels: $\Sigma = \Sigma_o \uplus \Sigma_u$. The *projection* on observable labels $\Pi : \Sigma^* \to \Sigma_o^*$ is defined as the monoid morphism generated by $\Pi(\alpha) = \alpha$ whenever $\alpha \in \Sigma_o$ and $\Pi(\alpha) = \varepsilon$ otherwise. Given $v \in \Sigma^*$, we denote by $|v|_o$ the number of observable events in v i.e. $|v|_o = |\Pi(v)|$.

The *observable (or visible) language* of A is defined as $L_o(A) = \{\Pi(v) \mid v \in L(A))\}$. For technical reasons commented later, we define the inverse projection Π^{-1} as follows :

$$\forall w \in \Sigma_o^*,\ \Pi^{-1}(w) = \{v \in L(A) : \Pi(v) = w\} \cap \Sigma^* \Sigma_o \tag{1}$$

i.e. we restrict the standard inverse projection to words of $L(A)$ that end with an observable letter[1].

From a run u performed by A, or equivalently from the word $v = \sigma(u)$, one only observes the visible actions *i.e.* the word $w = \Pi(v)$. For convenience, we define the function $\sigma_o = \Pi \circ \sigma : T^* \to \Sigma_o^*$ that associates to every path u the sequence of letters that are observed when u is executed. As a given observed word $w \in \Sigma_o^*$ might be the observation (*i.e.* the projection) of any word in $\Pi^{-1}(w)$, the *diagnosis* consists in

[1] Alternatively, we can define $L(A)$ as words that terminate with a letter of Σ_o, or equivalently by assuming faulty states in A that can only be reached by visible transitions, which does not reduce the generality of the setting.

deciding whether a fault has occurred in system A given this observed sequence w. A *diagnoser* for A can be seen as a function $\Delta : L_o(A) \to \{N,F,U\}$ where

$$\Delta(w) = \begin{cases} N \text{ iff } \Pi^{-1}(w) \subseteq L_N(A) \\ F \text{ iff } \Pi^{-1}(w) \subseteq L_F(A) \\ U \text{ otherwise} \end{cases} \quad (2)$$

Letters N,F,U stand for "normal", "faulty", and "uncertain" (or "ambiguous"), as it clearly appears above.

A diagnoser can be derived from an observer (or state estimator) of A. This observer is built in two steps. The first step is the Σ_o-*closure* of A. The Σ_o-*closure* (to the left) of A is defined as $B = Red_{\Sigma_o}(A) = (S, \Sigma_o, T', s_0)$ where $(s, \alpha, s') \in T'$ iff there exists a path $u = t_1 ... t_n$ in A such that $\sigma_o(t_1 ... t_{n-1}) = \varepsilon$, $\sigma_o(t_n) = \alpha$, $s^-(u) = s$ and $s^+(u) = s'$. Intuitively, there is a transition from s to s' in T' iff there exists a path from s to s' with a single observable action labeling the last transition of the path. The Σ_o-*closure* of A is an ε-*reduction* (to the left) assuming all labels of Σ_u are first replaced by ε in A. The second step needed to build a state estimator for A is the determinization of the resulting B, performed by standard subset construction. Let $D = Det(B) = (Q, \Sigma_o, T'', q_0)$ where $Q = 2^S$, $q_0 = \{s_0\}$ and $t = (q, \alpha, q') \in T''$ iff $q' = \{s' \in S : \exists s \in q,\ (s, \alpha, s') \in T'\}$. Of course, both B and D can be trimmed to their reachable part.

Observe that $L(D) = L(B) = L_o(A)$. D is a state estimator of A in the following sense : let $w \in L_o(A)$, as D is deterministic, there exists a unique path r in D such that $\sigma_o(r) = w$. The final state $q = s^+(r) \in Q$ of path r in D satisfies $q = s^+(\sigma_o^{-1}(w)) \in 2^S$ in A, *i.e.* it contains all states of A that are reachable by runs that produce the observed sequence w *and that stop immediately after the last observable transition*. This last condition explains the specific definition of Π^{-1} and the choice of the Σ_o-closure of A *to the left*. Let us call $q \in Q = 2^S$ a *normal* subset iff $q \subseteq S_N$, a *faulty* subset iff $q \subseteq S_F$, and an *uncertain* (or *ambiguous*) subset otherwise. D yields a diagnoser for A as follows : $\Delta(w)$ is the type of $q = s^+(\sigma_o^{-1}(w))$ in D. By extension, D is often called *the diagnoser of* A : $D = Diag(A) = Det(Red_{\Sigma_o}(A))$.

Due to determinization, D can be exponentially larger than A and should not be used for online diagnosis. One should use instead a recursive state estimation driven by the observed sequence w, which has linear complexity in the size of w and A. D can thus be considered as a precompiled version of the diagnosis for all possible observed sequences.

2.2 Remarks and extensions

Let A_1, A_2 be two automata, with $A_i = (S_i, \Sigma_i, T_i, s_{0,i})$, their *synchronous product* (or simply *product* for short) is the automaton $A_1 \times A_2 = (S_1 \times S_2, \Sigma_1 \cup \Sigma_2, T_1 \otimes T_2, (s_{0,1}, s_{0,2}))$ where transitions in $T_1 \otimes T_2$ are triples $((s_1, s_2), \alpha, (s_1', s_2'))$ such that

$$\begin{aligned} (s_1, \alpha, s_1') \in T_1 \ \wedge\ (s_2, \alpha, s_2') \in T_2 \text{ for } \alpha \in \Sigma_1 \cap \Sigma_2 \\ (s_1, \alpha, s_1') \in T_1 \ \wedge\ s_2 = s_2' \in S_2 \text{ for } \alpha \in \Sigma_1 \setminus \Sigma_2 \\ s_1 = s_1' \in S_1 \ \wedge\ (s_2, \alpha, s_2') \in T_2 \text{ for } \alpha \in \Sigma_2 \setminus \Sigma_1 \end{aligned}$$

Faulty runs in A are often not identified by a partition on states, but rather by the firing of some transition carrying a "fault" label $f \in \Sigma_u$. This can be recast in the previous setting as follows. Consider the deterministic and complete *memory automaton* $M = (\{N,F\},\Sigma,T,N)$ of Figure 1 where (N,f,F) is the unique transition of T producing a state change. The product $A \times M$ does not change the language of A, but performs a state augmentation that keeps track of the firing of a faulty transition in A. The label N or F now attached to states of $A \times M$ defines a partition of the state set that characterizes faulty runs. This technique is present in most event-based papers about diagnosability analysis, including the original ones, under the form of a label propagation attached to states. It was also generalized in [13] to detect/diagnose runs satisfying some regular pattern of labels, rather than the simple firing of a transition labeled by f. This show that the event-based diagnosis problem can be reduced to a state-based diagnosis problem in polynomial time. The converse reduction, from a state-based setting to an event-based setting is also straightforward.

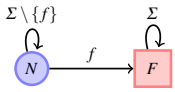

Fig. 1 A fault detection automaton.

When faults are non permanent in A, that is when there exist transitions from S_F to S_N, one may nevertheless be interested in detecting that some transient fault has occurred, i.e. a fault followed by a repair. This can again be handled as a standard diagnosis problem: one can transform A into another automaton A', that adds memory to states of A to propagate the fact that a fault occurred sometime in the past, and then build a diagnoser for A'. With the assumption that A is deterministic, this amounts to saturating the fault language of A: $L_F(A') = L_F(A)\,\Sigma^* \cap L(A)$. A diagnoser for A' detects faulty runs of A', i.e. it also detects runs of A that contain a fault. This idea is a variant of the pattern recognition of [13]. It was used in [15] to track the occurrence of k transient faults. It is also present in [9] under the names of O-diagnosis (detection of the occurrence of a fault) and I-diagnosis (detection of the occurrence of a repair). All these notions are thus variants of the classical diagnosis approach, even if they are recast in the context of transient failures. In [9], the authors propose a "memory automaton" that can be composed with a specification to remember occurrences of faults and repairs. However, even if fault repair is considered, their automaton propagates the information that a fault occurred. Within this setting, past occurrences of faults can be diagnosed, but without guarantee that faults are detected while the system is faulty. In the next section, we consider a different setting, where diagnosis is considered accurate if it detects a fault *before* it is repaired.

2.3 Diagnosability

Let us recall the notion of diagnosability for permanent faults, *i.e.* when A has no transition from S_F to S_N. For simplicity, we assume that A is Σ_o-live : an observable

transition is reachable from any state of A. More formally, an automaton is Σ_o-live iff for every state $s \in S$, there exists a path u such that $s^-(u) = s$ and $\sigma_o(u) \neq \varepsilon$. Intuitively, A is diagnosable iff, whenever it reaches S_F, this is detected/diagnosed after a finite number of extra observations. Formally, A is *diagnosable* iff

$$\begin{array}{c} \forall v_1 \in L_F(A),\ \exists n \in \mathbb{N},\ \forall v_1 v_2 \in L(A), \\ [\ |v_2|_o \geq n\ \Rightarrow\ \Pi^{-1} \circ \Pi(v_1 v_2) \subseteq L_F(A)\ ] \end{array} \tag{3}$$

where $|v_2|_o$ is the length of $\Pi(v_2)$. This expression slightly differs from more frequent ones (for ex. [22]), that count the number of transitions in v rather than the number of observable transitions, (i.e. require $|v_2| \geq n$ instead of $|v_2|_o \geq n$) but remains equivalent in essence. First, Definition (3) counts only visible transitions in $|v_2|_o$, instead of counting all transitions. It makes more sense to have an observable criterion to decide when to collect the diagnosis. And when A has no unobservable cycle, which is generally assumed when one uses $|v|$ instead of $|v|_o$, this rephrasing is harmless. Secondly, one generally assumes a uniform value of n covering all faulty words v_1. Again, taking into account the finiteness of A, this uniform bound comes for free once (3) holds.

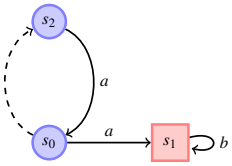

Fig. 2 Normal/faulty states are represented as circle/square boxes. The dashed line represents an unobservable transition. This automaton is diagnosable: after it reaches the faulty state, it can only produce a b which characterizes the occurrence of the fault. Nevertheless, driven by sequence a^n, the diagnoser outputs U^n. So uncertainty can be arbitrarily long.

Def. (3) states that a system is diagnosable iff, when uncertainty appears *after a faulty run*, it does not hold forever. Observe that the diagnosis may nevertheless remain uncertain for an arbitrarily long time, even for a diagnosable system, as long as no fault occurs (see Fig. 2). Conversely, A is *not diagnosable* iff after some faulty run uncertainty may last for an arbitrary long time :

$$\begin{array}{c} \exists v_1 \in L_F(A),\ \forall n \in \mathbb{N},\ \exists v_1 v_2 \in L(A), \\ [\ |v_2|_o \geq n\ \wedge\ \Pi^{-1} \circ \Pi(v_1 v_2) \cap L_N(A) \neq \emptyset\ ] \end{array} \tag{4}$$

The last term in (4) can be rephrased as $\Delta(\Pi(v_1 v_2)) = U$ or equivalently $\exists\, v' \in L_N(A)$, $\Pi(v_1 v_2) = \Pi(v')$. This new formulation expresses that one can find an arbitrary long extension v_2 of some faulty word v_1 which is *observationally equivalent* (or *equivalent* for short) to a safe word v' of A, denoted by $v_1 v_2 \sim_o v'$. As faults are permanent, any prefix of the safe word v' is also safe. Def. (4) thus opens the way to a polynomial test for (non-)diagnosability: one can build a twin-machine that recognizes pairs of

runs made of a faulty one v_1v_2 and an equivalent (w.r.t observation) safe one v', and thus check how long uncertainty can last.

Consider $B = Red_{\Sigma_o}(A)$, the *twin machine* of A is obtained as the product $C = B \times B$. A run in C represents a pair of runs of B that are observationally equivalent. Hence, for every run of C, there exists a pair (v,v') of observationally equivalent words of $L(A)$. The set of states of C is a subset of $S \times S$. So, using the same principle as in D, the diagnoser of A, a state (s,s') of C can be called normal/safe if s and s' both belong to S_N, faulty if s and s' both belong to S_F and uncertain/ambiguous otherwise. An *ambiguous cycle* in C is a reachable cycle that only goes through ambiguous states.

Proposition 1 *A is diagnosable iff its twin machine C has no ambiguous cycle.*

This result was proved in [14]. The only if part is obvious as the presence of an ambiguous cycle allows one to build an arbitrarily long suffix v_2 to a faulty word v_1 by repeating the cycle, while having this faulty word v_1v_2 equivalent to a safe one v'. This proves non-diagnosability. The if part uses the finiteness of A applied to (4): a long enough suffix v_2 necessarily contains a (faulty) cycle of B that is observationally equivalent to a safe/normal cycle of B.

The original version of Proposition 1 actually relied on a twin machine directly built from A and not from $B = Red_{\Sigma_o}(A)$. Building a basis for cycles in a graph can be done in polynomial time w.r.t the size of the original graph [26]. As C is at most of quadratic size w.r.t. the size of A, Proposition 1 clearly yields a polynomial test for the diagnosability of A.

3 Diagnosability of repairable faults

3.1 Diagnosis and T-diagnosability

We still consider a Σ_o-live deterministic automaton A, and now assume that some faults in A can be repaired, i.e. A contains transitions from S_F to S_N, or equivalently that the fault language $L_F(A)$ is not saturated. The diagnosis of an observed sequence $w = \sigma_o(u)$ produced by some run u of A is defined as in (2). However, we reinforce the diagnosability criterion for A by requiring that, when some fault occurs, it is still detected in finite time, but also *before it is repaired*.

Let us first introduce some notation. We denote by $L_F^{min} = \{v\alpha \in L_F(A) \mid v \notin L_F(A) \wedge \alpha \in \Sigma\}$ the set of minimal faulty words of A, *i.e.* words that correspond to a run ending with a transition from a normal state to a faulty one in A. For a word $v_1 \in L_F(A)$, let $v_1 \to v_1v_2 \in L_F(A)$ denote the continuous presence of a fault along v_2. Formally, $v_1 \to v_1v_2 \in L_F(A)$ iff $\forall v_2' \leq v_2,\ v_1v_2' \in L_F(A)$, where $\leq$ denotes the prefix relation on words.

Formally, an automaton A is *timely diagnosable* (*T-diagnosable* for short) iff

$$\begin{array}{c} \forall v_1 \in L_F^{min}(A),\ \exists n \in \mathbb{N},\ \forall v_1v_2 \in L(A), \\ [\ |v_2|_o \geq n \ \Rightarrow\ \exists v_2' \leq v_2 :\ v_1 \to v_1v_2' \in L_F(A) \\ \wedge\ \Pi^{-1} \circ \Pi(v_1v_2') \subseteq L_F(A)\ ] \end{array} \tag{5}$$

T-diagnosability differs from Def. (3) mainly by requiring that the fault that appears in v_1 remains for the whole execution of prefix v_2'. This notion is illustrated in Fig. 3, that depicts several observationally equivalent runs, and shows observation times at which a correct diagnosis/detection can be produced (before repair). Observe that if faults are not repairable, $v_1 \in L_F^{min}(A)$ implies that $v_1 \to v_1 v_2' \in L_F(A)$ for every v_2', and Def. (5) reduces to Def. (3) (condition $\forall v_1 \in L_F(A)$ in Def. (3) can equivalently be replaced by $\forall v_1 \in L_F^{min}(A)$). So, in a setting of permanent faults, T-diagnosability is equivalent to diagnosability.

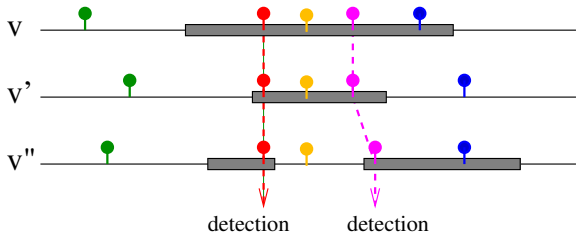

Fig. 3 A faulty word v and two equivalent words v', v''. The observed labels are represented as pins, and the faulty zones as grey rectangles. Detections correspond to times (in number of observations) where all equivalent words are faulty.

Fig. 4 illustrates the notion of T-diagnosability. Safe (resp. faulty) states are represented as circles (resp. boxes). One has $\Sigma = \{a,b,c,d\}$ and $\Sigma_o = \{a\}$. Ignoring the dashed transitions at the bottom, the automaton is T-diagnosable as after the observation of sequence a a fault occurred in both runs at the top, and this fault is each time detected before it is repaired since $\Delta(a) = F$. By adding the bottom part, T-diagnosability is lost : once a has been observed, one knows for sure that a fault occurred, but no detection can take place before repair, in all runs, as now $\Delta(a) = U$.

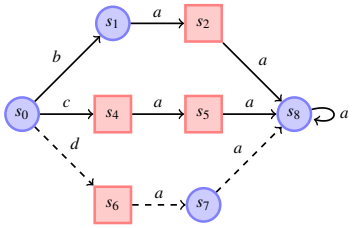

Fig. 4 A T-diagnosable system, when the path at the bottom is ignored.

3.2 Vanishing faults and repairs

T-diagnosability seems to be a reasonable first step towards the ability to count fault occurrences. Unfortunately, this is not the case as it is already apparent in Fig. 3 :

an automaton with such equivalent runs can be T-diagnosable, and nevertheless the same observed sequence matches a run with one fault (top) and one with two faults (bottom). The situation is even worse. Let us call a *vanishing fault* a fault that occurs and is repaired silently in A (between two observations), *i.e.* a path $u = t_1 \dots t_k$ where $\sigma_o(t_1 \dots t_{k-1}) = \varepsilon$ and $s^-(t_1) \in S_N, s^+(t_1) \in S_F, s^+(t_k) \in S_N$ (the last event can be visible). Similarly, let us call a *vanishing repair* the silent occurrence of a repair followed by a new fault, *i.e.* a path u as above where S_N and S_F are interchanged. An automaton A can exhibit runs with an arbitrary number of vanishing faults and repairs without losing its T-diagnosability. This is illustrated by the example in Fig. 5, with $\Sigma = \{a,b\}$, $\Sigma_o = \{a\}$. In this automaton A, one has $\Delta(a) = F$. A vanishing repair appears between the last two b of word abb, and a vanishing fault appears between the last two b of word $abbb$. Nevertheless, T-diagnosability holds : for $v_1 = a \in L_F^{min}(A)$ one gets immediate fault detection ($v_2 = \varepsilon$ works), for $v_1 = ab^2 \in L_F^{min}(A)$ one has $\Pi^{-1}(\Pi(v_1)) = \{a\} \subseteq L_F(A)$ so again the fault detection is "immediate" with $v_2 = \varepsilon$, and similarly for $v_1 = ab^4 \in L_F^{min}(A)$.

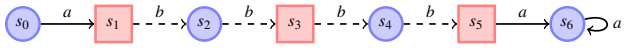

Fig. 5 An arbitrary number of vanishing faults and repairs may exist in a T-diagnosable automaton.

It is quite counter-intuitive that the "immediate" detection of the fault occurring at $v_1 = ab^2$ actually relies on the detection of the fault that took place previously, at $v_1 = a$. This phenomenon is due to the fact that T-diagnosability, just as diagnosability, only refers to runs that stop at a visible transition. Everything that happens between observations is almost ignored. For permanent faults, this is harmless: it only shifts the detection by one observation. For repairable faults, it introduces odd phenomena like the ones mentioned above. A natural way to make fault detection causal (and to open the way to a counting of faults) is thus to forbid the existence of vanishing repairs

$$\begin{aligned} \nexists v = v_1 v_2 \alpha \in L(A) \,:\, & v_1 \in L_F(A) \,\wedge\, v_1 v_2 \in L_N(A) \\ & \wedge\, \alpha \in \Sigma \,\wedge\, v_1 v_2 \alpha \in L_F(A) \,\wedge\, \Pi(v_2) = \varepsilon \end{aligned} \tag{6}$$

and of vanishing faults

$$\begin{aligned} \nexists v = v_1 v_2 \alpha \in L(A) \,:\, & v_1 \in L_N(A) \,\wedge\, v_1 v_2 \in L_F(A) \\ & \wedge\, \alpha \in \Sigma \,\wedge\, v_1 v_2 \alpha \in L_N(A) \,\wedge\, \Pi(v_2) = \varepsilon \end{aligned} \tag{7}$$

Under these assumptions, at most one transition from S_F to S_N or from S_N to S_F can take place between two visible events. Let us say that fault detection is causal when observations following the fault enable its detection, so the detection does not depend on observations that occurred strictly before the fault as in the pathological cases above. Such a causality notion can then be expressed as follows.

Proposition 2 *Assuming (6) and (7), A is T-diagnosable if and only if*

$$\begin{array}{c}\forall v_1 \in L^{min}_{F,o}(A) \cup L^{min}_{F,u}(A)\Sigma_u^*\Sigma_o,\ \exists n \in \mathbb{N},\ \forall v_1 v_2 \in L(A),\\ [\ |v_2|_o \geq n\ \ \Rightarrow\ \ \exists v_2' \leq v_2 :\ v_1 \rightarrow v_1 v_2' \in L_F(A)\\ \wedge\ v_1 v_2' \in \Sigma^*\Sigma_o\ \wedge\ \ \Pi^{-1} \circ \Pi(v_1 v_2') \subseteq L_F(A)\ ]\end{array} \tag{8}$$

where $L^{min}_{F,o}(A) = L^{min}_F(A) \cap \Sigma^*\Sigma_o$ represents minimal faulty runs that terminate with a visible event, and $L^{min}_{F,u}(A) = L^{min}_F(A) \setminus L^{min}_{F,o}(A)$ represents those that terminate with a silent event.

So if there are no vanishing faults and repairs, fault detection in a T-diagnosable system will occur after the observation that immediately follows (or produced) the fault.

Proof The extra condition $v_1 v_2' \in \Sigma^*\Sigma_o$ requires that the fault detection takes place at the moment one gets an observation. This could have been introduced in (5) without loss of generality, as silent events at the end of v_2' are useless to the criterion $\Pi^{-1} \circ \Pi(v_1 v_2') \subseteq L_F(A)$. So the only novelty lies in the first term. Recall that $L^{min}_F(A) = L^{min}_{F,o}(A) \uplus L^{min}_{F,u}(A)$. Words $v_1 \in L^{min}_{F,o}(A)$ are considered by both (5) and (8). But words $v_1 \in L^{min}_{F,u}(A)$ in (5) are replaced by words $v_1 \in L^{min}_{F,u}(A)\Sigma_u^*\Sigma_o$ in (8). In other words, for faults that occur silently, detection takes places after the next visible event.

Only if part. Assume A is T-diagnosable, and let $v_1 \in L^{min}_{F,u}(A)\Sigma_u^*\Sigma_o$. v_1 decomposes uniquely as $v_1 = v_0 u_3$ where v_0 is the $L^{min}_{F,u}$ part and u_3 the extension in $\Sigma_u^*\Sigma_o$. v_0 further decomposes as $v_0 = u_1 u_2$ where u_2 is the longest silent suffix of v_0. Thanks to (6) and (7), one has that $u_1 \in L_N(A)$, and $u_1 u_2 \rightarrow u_1 u_2 u_3 \in L_F(A)$. As A is T-diagnosable and $v_0 \in L^{min}_F(A)$, let us take any long enough extension $v_2 \geq u_3$ for the fault detection in Def. (5), and let $v_2' \leq v_2$, $v_2' \in \Sigma^*\Sigma_o$ be the detection time. One can not have $v_2' < u_3$ because in that case $\Pi(v_0 v_2') = \Pi(v_0) = \Pi(u_1)$ and $u_1 \in L_N(A)$. So the detection of the fault can not occur before the extra observation lying at the end of u_3. Since $v_2' \geq u_3$, one has $v_2' = u_3 v_2''$ and $v_0 \rightarrow v_0 u_3 v_2'' \in L_F(A)$. This proves the existence of a detection time v_2'' after $v_1 = v_0 u_3$ which satisfies (8).

If part. Assume A satisfies (8) and let $v_1 \in L^{min}_{F,u}(A)$. v_1 decomposes uniquely as $v_1 = u_1 u_2$ where u_2 is the longest silent suffix of v_1. Thanks to (6) and (7), one has $u_1 \in L_N(A)$. Let $v_1 v_2 \in L(A)$, with v_2 long enough, in particular $|v_2|_o \geq 1$. One can write $v_2 = u_3 u_4$ with $u_3 \in \Sigma_u^*\Sigma_o$. Thanks to (6) and (7) again, one has $v_1 \rightarrow v_1 u_3 \in L_F(A)$. As $v_1 u_3 \in L^{min}_{F,u}(A)\Sigma_u^*\Sigma_o$ and u_4 is long enough, there exists a prefix $u_4' \leq u_4$ such that $v_1 u_3 \rightarrow v_1 u_3 u_4' \in L_F(A)$ and $\Delta(\Pi(v_1 u_3 u_4')) = F$. Taking $v_2' = u_3 u_4'$ thus satisfies the conditions of (5). □

Vanishing faults (or repairs) can be considered as design errors in system A, that are either benign and should be disregarded, or conversely that are possibly harmful and should be made visible. Changing the status of such events means modifying the safe and faulty words of A, which is feasible (see later). Meanwhile we show that the detection of vanishing faults/repairs is a simple problem: these events are regular properties along a run, and they can thus be evidenced by standard state augmentation techniques. The construction is given below.

Proposition 3 *Detecting whether an automaton has vanishing faults (resp. repairs) can be done in NLOGSPACE, and in linear time wrt to the size of A.*

Proof An automaton A can produce a vanishing fault iff it contains an accessible path $u = t_1 t_2 \dots t_k$ such that $s^-(t_i) \in S_F$ for every $i \in 2 \dots k$, $s^-(t_1), s^+(t_k) \in S_N$, and $\sigma(t_1 \dots t_{k-1}) = \varepsilon$ (the last transition t_k can be observable or not). Such path will be called a *vanishing fault sequence*.

Starting from an automaton A one can build a non-deterministic automaton $VA = (S \times V, T', \Sigma, (s_0, NO))$ where $V = \{NO, UF, Van\}$ is a label with the following meaning : a state labeled by NO is a state encountered along a run in which the decision to recognize a vanishing fault was not yet taken. A state labeled by UF is a state that is encountered along a run in which a transition from a non-faulty state to a faulty one was met, and no observable action was observed since this transition. A state labeled by *Van* is a state appearing along a run in which a vanishing fault was found. The transition relation T' is defined as follows:

- $((s, NO), \sigma, (s', NO)) \in T'$ if $(s, \sigma, s') \in T$. Label σ can be observable or not. Note that even if s is a non-faulty state and s' a faulty one, the automaton is not forced to recognize a vanishing sequence from this transition. We hence include copies of original transitions in T'.
- $((s, NO), \sigma, (s', UF)) \in T'$ if $(s, \sigma, s') \in T, \sigma \in \Sigma_u, s \in S_N, s' \in S_F$. This transition is where recognition of a potential vanishing sequence starts.
- $((s, UF), \sigma, (s', UF)) \in T'$ if $(s, \sigma, s') \in T, \sigma \in \Sigma_u, s \in S_F, s' \in S_F$. A vanishing sequence recognition has started, and as long as the system remains faulty and does not produce observable actions, the currently followed run can be a vanishing sequence.
- $((s, UF), \sigma, (s', Van)) \in T'$ if $(s, \sigma, s') \in T, s \in S_F, s' \in S_N$. Label σ can be observable or not. This transition occurs as soon as a vanishing sequence is detected: the system was faulty and unobservable from the occurrence of a fault up to the occurrence of the repair.
- $((s, UF), \sigma, (s', NO)) \in T'$ if $(s, \sigma, s') \in T, \sigma \in \Sigma_o, s \in S_F$. An observable transition was reached that was not a repair, so the last fault was not a vanishing fault.

Obviously, automaton A contains a vanishing sequence iff there exists a reachable state of the form (s, Van) in VA. The size of VA is at most $3.|A|$, so VA (restricted to its reachable states) can be built in $O(3.|A|)$, and the search for a vanishing sequence can be performed non-deterministically using a standard reachability algorithm in logarithmic space (in the size of $|A|$), and if all states have to be explored, in time in $O(3.|A|)$. □

Note that one a vanishing fault has been found, there is no need to continue exploration. Hence we create no transition from a state of the form (s, Van). Figure 6 illustrates the construction of an automaton that recognizes vanishing sequences. The top sequence starts recognizing a sequence of unobserved action after a fault, but action σ_k is observable, so the tag attached to the last state reached by the sequence gets back to NO. In the bottom sequence, the system returns to a normal state after observation of the empty word: a vanishing sequence has been detected. A similar construction can be used to detect vanishing repairs, just by moving from a NO state

to an UF state as soon as an unobservable repair occurs, and from an UF state to a *Van* state when a new unobservable fault occurs.

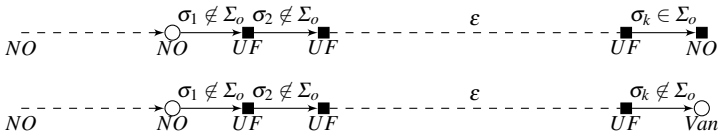

Fig. 6 Construction of automaton *VA* to detect vanishing faults.

In the rest of the paper, we will assume that the considered systems do not contain vanishing faults (repairs). This assumption is not too constraining: a vanishing fault (repair) might not be followed by an observation, hence hindering any chance to diagnose it. If one wants to diagnose faults in a timely way, the assumption that something observable necessarily occurs while the system is faulty is needed. We shall come back to this point later in the paper.

3.3 A T-diagnosability test

As in Section 2.3, one can consider the converse of (5). Specifically, A is *not T-diagnosable* iff

$$\begin{array}{c} \exists v_1 \in L_F^{min}(A),\ \forall n \in \mathbb{N},\ \exists v_1 v_2 \in L(A) : |v_2|_o \geq n, \\ \forall v_2' \leq v_2,\ v_1 \rightarrow v_1 v_2' \notin L_F(A)\ \vee\ \Pi^{-1} \circ \Pi(v_1 v_2') \not\subseteq L_F(A) \end{array} \tag{9}$$

In words, A is not T-diagnosable whenever it is possible to find a minimal faulty sequence v_1 and arbitrarily long extensions v_2 such that along the longest faulty prefix $v_2' \leq v_2$ of v_2, the detection of the fault can not occur in a timely way, either because repair occurs before any possible detection, or because the extension remains ambiguous.

It is worth noticing that the twin-machine idea used to check the diagnosability of permanent faults is not sufficient to check the T-diagnosability of repairable faults. The main obstacle comes from the fact that T-diagnosability can not be characterized by pairs of equivalent runs. It is rather a global property on classes of equivalent runs in A. This is illustrated in Fig. 7, where unobservable transitions are depicted as dashed arrows ($\Sigma_o = \{a\}$) and faulty states in red. This automaton is not T-diagnosable. However, the twin machine built for this system in Fig. 9 contains no ambiguous cycle. By checking only *pairs* of equivalent runs, one always finds a time where ambiguity disappears. For example, considering only the top and central loops, a^{3n+1} seem to be detection times for the faults that appear in these runs. To reveal that T-diagnosability does not hold, one would have to check triples of equivalent runs here. And it is quite easy to design examples where triples are not sufficient and one needs to escalate to quadruples of equivalent runs to reveal the non T-diagnosability, etc. This suggests a non polynomial complexity of the T-diagnosability test.

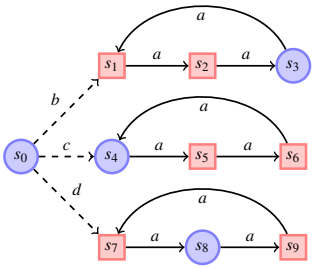

Fig. 7 A system that is not T-diagnosable. Considering only pairs of equivalent runs is not sufficient to characterize non-T-diagnosability.

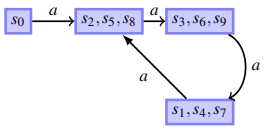

Fig. 8 Diag(A) for the automaton of Figure 7

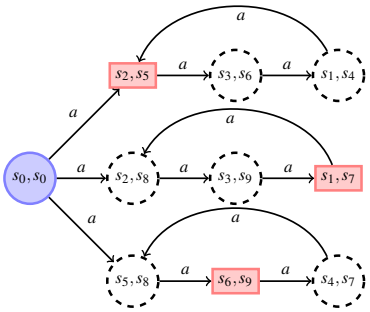

Fig. 9 The twin machine for the system of Fig. 7. Square states represent states of the twim machine with faulty pairs of states. Filled circle are states with non-faulty components, and dashed states are ambiguous states, i.e. in which one component is faulty and the other is safe. Every cycle of this twin machine contains a square faulty state. Though this system does not contain ambiguous cycles, the system is not T-diagnosable.

The idea of the twin-machine construction is to check whether a faulty run can create an ambiguity that can never be resolved. For repairable faults, this ambiguity signal can be directly derived from $Diag(A)$, the diagnoser of A. Consider the (deterministic) automaton $G = A \times Diag(A)$. $Diag(A)$ is a deterministic automaton over alphabet $\Sigma_o \subseteq \Sigma$, and $L(Diag(A)) = L_o(A)$. So $L(G) = L(A)$: the construction of G performs a simple state augmentation on A, without changing its behavior (just like the memory automaton mentioned above). This state augmentation attaches an ambiguity status to each state of A as follows. States of G take the form $(s,q) \in S \times Q$ where $Q = 2^S$. So they can be labeled by elements in $\{N,F\} \times \{N,U,F\}$: for example (s,q) is of type (N,U) iff $s \in S_N$ and q is uncertain. $L_N(A)$ and $L_F(A)$ are easily identifiable in G as words terminating in a state of type $(N,.)$ or $(F,.)$ respectively. A

state (s,q) is said to be *minimally faulty* iff s is the terminal state of a run $v_1 \in L_F^{min}(A)$. Notice that a state (s,q) labeled (F,N) can exist : it characterizes a run that was safe up to the last observation and that later produced (silently) a fault, no yet detectable that the diagnoser. Similarly, a state (s,q) labeled (F,U) characterizes a faulty run that is not yet diagnosed.

Theorem 1 *With the notation above, A is not T-diagnosable if and only if there exists a reachable minimally faulty state $(s,q) \in S \times Q$ in G such that (s,q) is of type (F,N) or (F,U) and either*

1. *there exists a state (s',q') of type (N,N) or (N,U)*
2. *or there exists a cycle of (F,U) states*

that is reachable from (s,q) through a (possibly empty) sequence of (F,N) states followed by a sequence of (F,U) states.

Proof By construction of G, observe that if word $v \in L(A)$ reaches state s in A, then word v reaches state (s,q) in G and $\Delta(\Pi(v))$ is the type of state $q \in Q$, either N,F or U.

For the only if part, consider the witness $v_1 \in L_F^{min}(A)$ of non T-diagnosability in (9), which reaches state (s,q) in G. (s,q) is necessarily of type (F,N) or of type (F,U), as if (s,q) is of type (F,F) then the correct diagnosis is output with $v_2' = \varepsilon$. For a given n, let v_2 be the extension of v_1 satisfying (9), and let v_2' be the longest prefix of v_2 such that $v_1 \rightarrow v_1 v_2' \in L_F(A)$. All along v_2', the correct diagnosis can not be output, so G only crosses states of type (F,N) or (F,U). States of type (F,N) come first (if they exist), then (after the first observable event in v_2') one only crosses states of type (F,U) as at least one faulty run lies in the inverse projection. If there exists $a \in \Sigma$ such that $v_2' a \leq v_2$, then $v_1 v_2' a$ reaches state (s',q') which is either of type (N,N) or of type (N,U). (N,F) is not possible as this would mean that the correct diagnosis was produced for $v_1 v_2'$. This makes point 1 in the theorem. If point 1 never occurs for any n, this means that in the discussion above one always has $v_2' = v_2$. As G is finite, it then contains a cycle with at least one observable event (recall that n counts observations). This cycle is thus made of (F,U) states, which makes point 2 in the theorem.

The if part can be derived in a similar manner, starting from conditions in the theorem and building a witness v_1 and the associated v_2 for every n satisfying (9). □

Back to the example depicted in Figure 7, the automaton $G = A \times Diag(A)$ is given in Figure 10 (left-hand side), whereas its abstract view carrying only labels of composite states is given in Figure 10 (right-hand side) (recall that $\{b,c,d\}$ are unobservable). It is easy to check that G does not fulfill the conditions of Theorem 1. Indeed, G contains a (F,U) state, from which a (N,U) state is reachable (this is highlighted in the figure by a dashed arrow). Thus, as already mentioned, A is not T-diagnosable.

3.4 Complexity of T-diagnosability

Theorem 2 *Deciding whether an automaton A is T-diagnosable is a PSPACE-complete problem.*

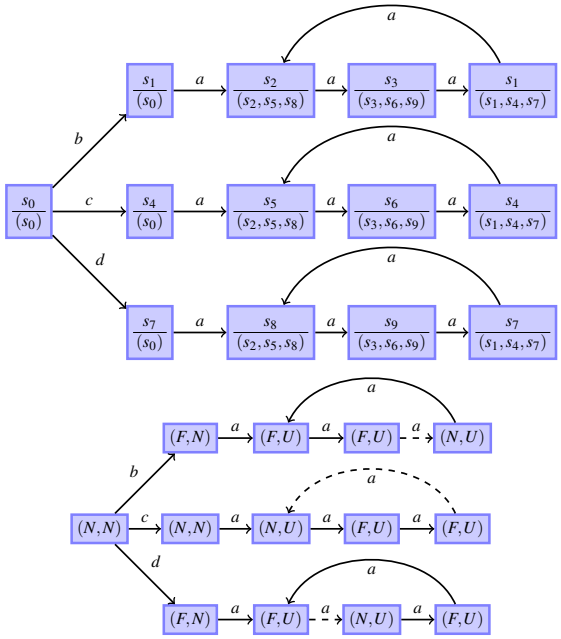

Fig. 10 The augmented machine $G = A \times Diag(A)$ (top) and its abstract version (bottom), for the example given in Fig. 7.

Proof First, we can easily show that T-diagnosability belongs to PSPACE. Following the result of Theorem 1, A is not T-diagnosable iff one can find a witness cycle of type (F,U) or a witness state of type (N,N) or (N,U) reachable after a minimally faulty sequence ending in a state of type (F,N) or (F,U) in G. First of all, the size of G is at most $2^{|A|}.|A|$. To witness a minimally faulty sequence ending in a (F,N) of (F,U) state, one only needs to non-deterministically explore paths of size smaller than $2^{|A|}.|A|$, which can be done with polynomial memory size (to remember current state and whether previous state is faulty). Then, to witness ambiguous cycles or moves to (N,N) or (N,U) states, one can again non-deterministically explore paths of G of size smaller than $2^{|A|}.|A|$ with polynomial memory. Hence, finding witness paths for non-T-diagnosability is a NPSPACE process, and using Savitch's theorem [23], and remembering that PSPACE is closed under complementation, this shows that T-diagnosability is in PSPACE.

The second step of the proof shows hardness of the problem by reduction from a language inclusion problem, which is known to be PSPACE-complete (see [16] and [18]). The problem can be formulated as follows: given $A_1,...,A_n$ some deterministic finite automata, does $\bigcap_{i\in 1..n} L(A_i) = \emptyset$?

Let $n \in \mathbb{N}$ and for $1 \leq i \leq n$, let $A_i = (S_i, \Gamma, T_i, q_0^i, F_i)$ be some deterministic finite automaton on alphabet Γ, that recognizes language $L(A_i) = \{\sigma(u) \mid s^-(u) = q_0^i \wedge s^+(u) \in F_i\}$. We build the finite automaton $A = (S, \Sigma, T, q_0)$ (see Figure 11) where:

- $\Sigma = \Gamma \cup \{u_1, \ldots, u_n\} \cup \{f, \sharp, \flat, r\}$

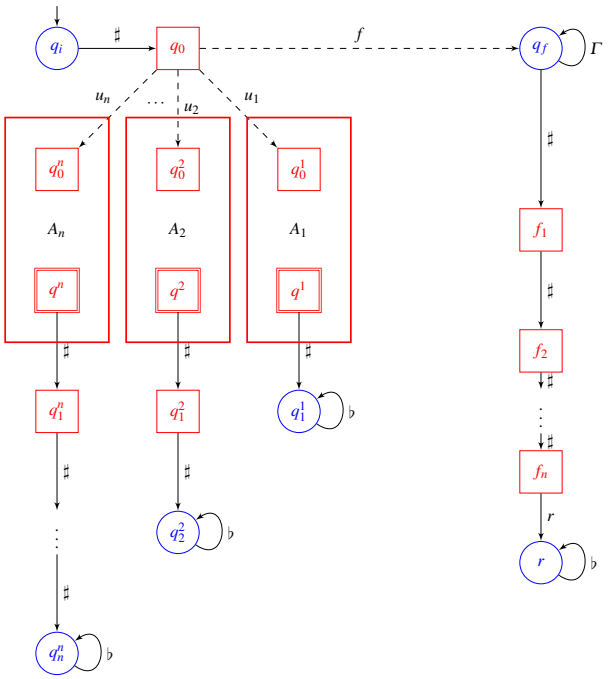

Fig. 11 PSPACE-hardness of T-diagnosability. Red states are faulty, and blue states normal states.

- $S = \{q_i, q_0, q_f, r\} \cup \{f_i \mid i \in 1..n\} \cup \{q_i^j \mid 1 \leq i \leq j \leq n\} \cup S_1 \cup \ldots \cup S_n$
- $T = \bigcup_{1 \leq i \leq n} T_i$
  $\cup \{(q_0, f, q_f), (q_i, \sharp, q_0)(f_n, r, r), (r, \flat, r), (q_f, \sharp, f_1)\}$
  $\cup \{(f_i, \sharp, f_{i+1}) \mid i \in 1..n-1\}$
  $\cup \{(q_f, a, q_f) \mid a \in \Sigma\}$
  $\cup \{(q_0, u_i, q_0^i) \mid i \in 1..n\}$
  $\cup \{(q_i^i, \flat, q_i^i) \mid i \in 1..n\}$
  $\cup \{(q, \sharp, q_1^i) \mid q \in F_i, i \in 1..n\}$
  $\cup \{(q_i^j, \sharp, q_{i+1}^j) \mid 1 \leq i < j \leq n\}$

The set of safe states is $S_N = \{q_i, q_f, r\} \cup \{q_i^i \mid i = 1 \ldots n\}$. The set of faulty states is $S \setminus S_N$. We set $\Sigma_o = \Gamma \cup \{\sharp, \flat, r\}$.

We claim that A is T-diagnosable if and only if $\bigcap_{i \in 1..n} L(A_i) = \emptyset$.

First, remark that after observing $\sharp w \sharp^m$ for $m \leq n$ and $w \in \Gamma^*$, the current run is either in state f_m or in some state q_m^j if $j \geq m$, and automaton A_j accepts w.

Suppose that A is T-diagnosable. Let $w \in \Gamma^*$, and let v_1 be the unique run of A such that $\sigma(v_1) = \sharp f w \sharp$. As v_1 is a minimal faulty run, there exists $m \leq n$ such that the run $v_1 v_2$ with $\sigma(v_1 v_2) = \sharp f w \sharp^m$ verifies $\Pi^{-1} \circ \Pi(v_1 v_2) \subseteq L_F(A)$, because a repair occurs after $\sharp f w \sharp^n$.

According to the construction of A, if automaton A_i accepts w, then for $i \geq m$, there exists also a run of A which observation is $\sharp w \sharp^m$, and that ends in state q_m^i.

Furtermore, as $\Pi^{-1} \circ \Pi(v_1 v_2) \subseteq L_F(A)$, the system can not be in state q_m^m as this state is safe. Hence, A_m does not accept w and w does not belong to the intersection $\bigcap_{i\in 1..n} L(A_i)$. As this is true for every $w \in \Gamma^*$, T-diagnosability of A implies $\bigcap_{i\in 1..n} L(A_i) = \emptyset$.

Conversely suppose that $\bigcap_{i\in 1..n} L(A_i) = \emptyset$. Let v_1 be a minimal faulty run. Only two cases can appear: either v_1 is the word $v_1 = \sharp$ which ends in the faulty state q_0, or v_1 is of the form $v_1 = \sharp f w \sharp$. If $v_1 = \sharp$, then we know that A is in q_0 which is faulty and we can claim the fault. In the second case $s^+(v_1) = f_1$ and $\sigma(v_1) = \sharp f w \sharp$ with $w \in \Gamma^*$. As $\bigcap_{i\in 1..n}(A_i) = \emptyset$, there exists $i \in 1..n$ such that $w \notin L(A_i)$. Consider the run $v_1 v_2$ with $\sigma(v_1 v_2) = \sharp f w \sharp^i$, this run ends in f_i and was not repaired in between. Even if w is recognized by A_k for some $k < i$, no run visiting a state of A_k and with observation $w\sharp^i$ exists. Moreover, for every $j > i$ such that A_j accepts w, runs with the same observation $w\sharp^i$ ends in state q_i^j. As A_i does not accept w and as all states q_i^j for $j > i$ are faulty, $\Pi^{-1} \circ \Pi(v_1 v_2) \in L_F(A)$. Thus the fault can be claimed. As this is true for every minimal faulty run v_1, A is T-diagnosable.

As highlighted by Theorem 2, checking whether an automaton A is T-diagnosable is a *PSPACE*-complete problem. This is not a real surprise: T-diagnosability compares faulty and non-faulty languages, and most of algorithms that use languages comparisons are PSPACE-complete. In practice, *PSPACE*-complete problems are considered as harder than *NP*-complete problems, as one already knows that $NP \subseteq PSPACE$. Moreover, complexity of T-diagnosability should not be considered as a limitation for several reasons. First of all, the complexity of Theorem 2 is a worst-case complexity. It is well known that model checking a simple logic such as LTL is already PSPACE-complete [24]. However, in practice, many model checking tools perform quite well on large structures. Indeed, very often the worst time complexities are met for degenerate cases, that are rarely met in real case studies.

3.5 Removing the vanishing fault hypothesis

It is worth noticing that the verification of the T-diagnosability assumes absence of vanishing faults (resp. repairs). However, as mentioned at the end of section 3.2, whenever there exists a vanishing fault in the system, the diagnoser is not able to detect it, as its verdict is given according to the occurrence of observable events. Thus, if one wants to diagnose faults in a timely way, the assumption that something observable necessarily occurs while the system is faulty is needed. An easy way to guarantee this property is to reduce the unobservable part of the alphabet. Obviously, if $\Sigma_o = \Sigma$, there can be no vanishing fault, but finer modifications of the original systems exist. For instance one can easily reuse the paths of automaton *VA* leading from a fault to a vanishing state to discover the smallest subsets of unobservable actions that should become observable to avoid vanishing faults (the method would be similar to the sensor minimization technique proposed in [8] which aims at minimizing the size of observable events while keeping a system diagnosable).

However, unobservability of some actions is not always a design choice. In a distributed system, for instance, one may observe actions on a limited subset of machines. Another way to avoid vanishing faults is to consider that vanishing and hence

potentially fully unobservable faults are not important failures, but rather design errors producing a specification where the incriminated state change leading to a vanishing fault should not be considered as a fault. So, the next question to study is: how to transform a specification A with vanishing faults into a correct specification A' without vanishing faults such that non vanishing faults that can be timely detected in A are also timely detected in A'?.

Let $A = (S, \Sigma, T, s_0)$ be a deterministic automaton, and VA be the automaton built in the proof of proposition 3 to detect sequences that contain a vanishing fault. We denote by $L_{Van}(A)$ the language of vanishing faults, i.e. the set of words from Σ^* that correspond to sequences with a vanishing fault. Formally,

$$L_{Van}(A) = \{w \in \Sigma^* \mid \exists t_1 \dots t_k \in VA, s^-(t_1) = (s_0, NO), s^+(t_k) \in S \times \{Van\}, \\ \sigma(t_1 \dots t_k) = w\}. \quad (10)$$

To ignore vanishing faults, we use a construction similar to that of automaton VA. Our aim is to design an automaton A' such that $L(A) = L(A')$ and $L_{Van}(A') = \emptyset$ and such that all the faulty sequences that can be diagnosed in A are also diagnosed in A'. We define the set of T-diagnosable sequences of A as follows:

$$L_{diag}(A) = \{v_1 \in L_F^{min}(A) \mid \exists n \in \mathbb{N}, \forall v_1 v_2 \in L(A), \\ [\, |v_2|_o \geq n \;\Rightarrow\; \exists v_2' \leq v_2 : v_1 \to v_1 v_2' \in L_F(A) \\ \wedge\; \Pi^{-1} \circ \Pi(v_1 v_2') \subseteq L_F(A)]\}. \quad (11)$$

Such an automaton is obtained as follows: we let $A' = (S_{A'}, \Sigma, T', (s_0, NO))$, where[2] $S_{A'} = S \times \{NO, UF, Van\}$ and T' is built as follows:

- $((s, NO), \sigma, (s', NO)) \in T'$ if $(s, \sigma, s') \in T$, and $\sigma \in \Sigma_o$
- $((s, NO), \sigma, (s', NO)) \in T'$ if $(s, \sigma, s') \in T$, $\sigma \in \Sigma_{uo}$ and $s' \in S_N$
- $((s, NO), \sigma, (s', NO)) \in T'$ if $(s, \sigma, s') \in T$, and $\sigma \in \Sigma_{uo}$, $s \in S_F$ and $s' \in S_F$
- $((s, NO), \sigma, (s', UF)) \in T'$ if $(s, \sigma, s') \in T$, and $\sigma \in \Sigma_{uo}$, $s \in S_N$ and $s' \in S_F$

- $((s, UF), \sigma, (s', UF)) \in T'$ if $(s, \sigma, s') \in T$, and $\sigma \in \Sigma_{uo}$, and $s' \in S_F$
- $((s, UF), \sigma, (s', NO)) \in T'$ if $(s, \sigma, s') \in T$, and $\sigma \in \Sigma_o$
- $((s, UF), \sigma, (s', Van)) \in T'$ if $(s, \sigma, s') \in T$ and $s' \in S_N$

  Note that one necessarily has $s \in S_F$ in the three previous cases, as a fault has occurred (witnessed by label UF), and is not yet repaired.

- $((s, Van), \sigma, (s', NO)) \in T'$ if $(s, \sigma, s') \in T$ and $\sigma \in \Sigma_o$.
- $((s, Van), \sigma, (s', UF)) \in T'$ if $(s, \sigma, s') \in T$, $\sigma \in \Sigma_{uo}$ and $s' \in S_F$.
- $((s, Van), \sigma, (s', NO)) \in T'$ if $(s, \sigma, s') \in T$, $\sigma \in \Sigma_{uo}$ and $s' \in S_N$.

  Note that $s \in S_N$ in the three previous cases, as vanishing faults are detected only after a repair.

[2] where NO, UF, Van have the same meaning as in the automaton VA.

Let $CoaccVan(A',X)$ denote the set of states of A' from which there exists a sequence of transitions that are all unobservable, excepted possibly the last transition, leading to a state of X. Then, we partition $S_{A'}$ into $S'_N \uplus S'_F$, where $S'_F = S_F \times \{UF,NO\} \setminus CoaccVan(A', S \times \{Van\})$, and $S'_N = S' \setminus S_F$. Informally, we consider as faulty the sets of states from which a fault will *necessarily* be followed by an observation before repair, i.e. it will not vanish.

Intuitively, in this construction, automaton A' memorizes if a fault has occurred, if an observable action has occurred since occurrence of a fault, and if a repair occurs before occurrence of an observable action.

We can now prove the following propositions:

Proposition 4 $L(A') = L(A)$

Proof We can show $L(A') \subseteq L(A)$ and $L(A) \subseteq L(A')$ by induction on the length of words. Obviously, $\varepsilon \in L(A') \cap L(A)$. Let us now suppose that inclusion is verified in both directions for words of length at most n. Let w be a words of $L(A)$ and $L(A')$. This word corresponds to a run of A from s_0 to some state s, and to a run from (s_0,NO) to a state (s,l) where $l \in \{NO,UF,Van\}$. Let us assume that $(s,\sigma,s') \in T$. According to the construction rules, if $\sigma \in \Sigma_o$ then regardless of label l, a transition labeled by σ to some state (s',l') exists. Conversely, if w leads A to some state (s,l) and a transition $((s,l),\sigma,(s',l'))$ exists in T', then (s,σ,s') is also a transition of T.□

Proposition 5 $L_{Van}(A') = \emptyset$

Proof Let us assume that there exists a vanishing sequence in A'. This sequence is of the form $u = t_1 \dots t_n$ such that $s^-(t_1) = (s_0,NO) \in S'_N$, $s^+(t_n) \in S'_N$, and there exists $k \in 1...n-1$ such that $(s_{k-1},l_{k-1}) = s^-(t_k) \in S'_N$, $(s_k,l_k) = s^+(t_k) \in S'_F$, $\sigma(t_k \dots t_{n-1}) = \varepsilon$, and $\sigma(t_n) \in \Sigma_o$. Let us first notice that S'_F is a subset of $S_F \times \{UF,NO\}$, and that the transition from (s_{k-1},l_{k-1}) to (s_k,l_k) along the chosen sequence is unobservable. Hence, according to the construction rules for A', we have $l_k = UF$.

Similarly, one has $l_i = UF$ for every $i \in k+1...n-1$ as otherwise, the considered sequence $t_k \dots t_{n-1}$ would be an observable one. Let us now consider l_n. As the last state reached by the sequence is a normal state, but as $l_{n-1} = UF$, we necessarily have $l_n = Van$. This contradicts the rule imposed for the construction of S'_F, that requires that no unobservable path leads from a faulty state in S'_F to a state with label Van . One can handle similarly the case where the last transition t_n is not observable. □

Let us illustrate the construction of A' on the example of Figure 12. In this automaton, $\Sigma_o = \{a,a'\}, \Sigma_{uo} = \{b,c,d\}$. One can immediately notice that the run $v = (q_0,b,q_2)(q_2,b,q_3)(q_3,b,q_5)$ contains a vanishing fault. Figure 13 shows how the automaton A is unfolded and how the resulting states are tagged with NO,UF,Van. Let us highlight the difference between q_2,UF and q_2,NO: q_2,NO identifies state q_2 of the system reached when no unobservable move from a non-faulty sate to a faulty one has been observed. Conversely, state q_2,UF represents state q_2 of the system when an unobservable move from a safe state to a faulty one occurred. Rectangle dashed states in this figure are states of the form (s,l) such that $s \in S_F$. However, as one must avoid vanishing faults, all these dashed states are not considered faulty in A'. The final set

of faulty states of A' excludes the dashed states that are in $CoaccVan(A', \{(q_5, Van\})$: indeed, from (q_2, UF) and (q_3, UF), there exist runs in which fault are repaired before any observation occurs. The final automaton A' computed is depicted in Figure 14. We hence have $S'_F = \{(q_2, NO), (q_3, NO), (q_4, NO), (q_7, UF), (q_7, NO)\}$. Remark that $L(A) = L(A')$: both languages define the prefix closure of the regular language $(bab + cba)(aba'^* + ba'^*) + cbba'^* + da'a'^*$. On this example, one can see that the construction simply adds memory to states of the system, to capture the information on whether a fault occurred, and whether an observable action followed or not.

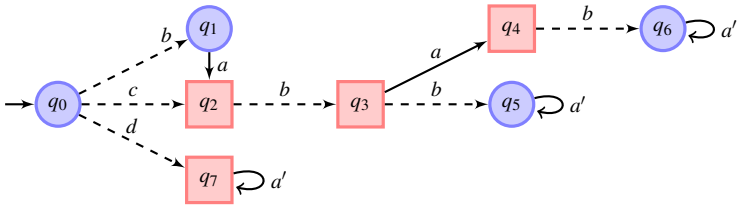

Fig. 12 An example of system with vanishing faults. $\Sigma_O = \{a, a'\}, \Sigma_{uo} = \{b, c, d\}$.

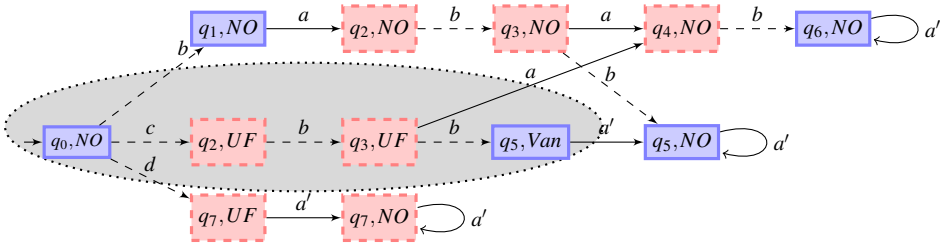

Fig. 13 Adding memory to the system of Figure 12. $\Sigma_{uo} = \{c, b, d\}$. Blue square states are states of the form (s, l) where $s \in S_N$. Red dashed square states are states of the form (s, l) where $s \in S_F$. Note that all these states are not necessarily faulty in A'. States in $CoaccVan(A', \{(q_5, Van)\})$ are $\{(q_0, NO), (q_2, UF), (q_3, UF)\}$, and are represented in the dotted zone.

The last question to address is how diagnosability of a system is affected by vanishing faults removal. Let us first recall that removing vanishing faults and repairs allows one to get back to a setting where the characterization of diagnosability proposed in Theorem 1 applies, and also makes T-diagnosis causal. Note however that removing vanishing faults does not make a system diagnosable: Indeed, even if all faults are eventually followed by an observation before they are repaired, this does not mean that every ambiguity in a system is resolved, and several faults may remain ambiguous forever. A desirable property is that all faults that were in $L_{diag}(A)$ remain in $L_{diag}(A')$. For example, this is the case in Figure 14. Indeed, for this example, we have : $L_F^{min}(A) = \{ba, c, d\}$, and $L_{diag}(A) = \{ba, c, b\}$ because for every fault, immediately after occurrence of an observation, one can claim that a fault has occurred, even

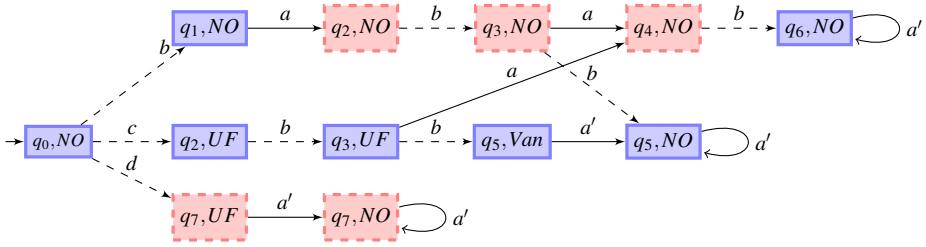

Fig. 14 A new automaton A' without vanishing faults. $\Sigma_{uo} = \{c,b,d\}$. Faulty states are $S'_F = \{(q_2,NO),(q_3,NO),(q_4,NO),(q_7,UF),(q_7,NO)\}$.

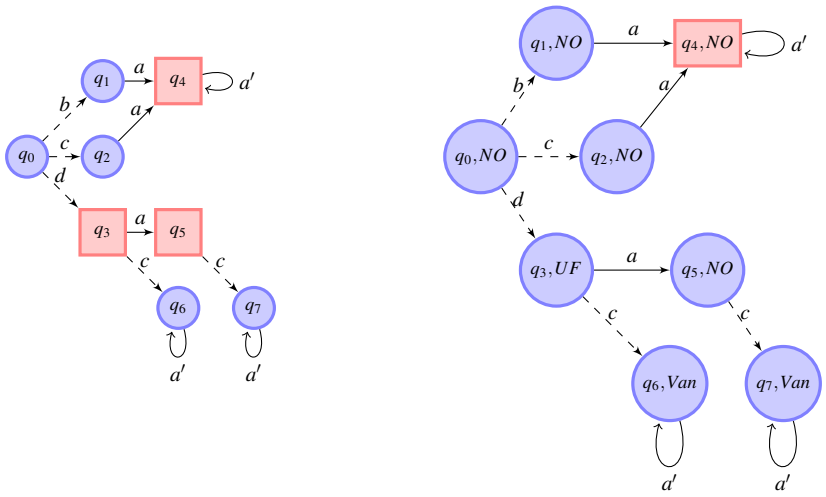

Fig. 15 Removing vanishing fault can make faults non-(T)diagnosable. In this example, state q_3 becomes a non-faulty state after transformation, and then in the obtained automaton, any observation of the form $a.b^*$ corresponds to both faulty and non-faulty runs.

if the exact run followed remain uncertain. We then have $L_F^{min}(A') = \{ba,cba,d\}$, $L_{diag}(A') = \{ba,cba,d\}$. An intersting question is whether faults that appear both in A and A' and that are diagnosable in A remain diagnosable in A'. More formally, if $w \in L_F^{min}(A) \cap L_F^{min}(A')$ and $w \in L_{diag}(A)$, does it implies that $w \in L_{diag}(A')$? Unfortunately, in general, removing vanishing faults does not preserve diagnosability of other faults. The example of Figure 15 shows an automaton A (at the left of the figure) and the automaton A' obtained after removing vanishing faults (at the right of the figure). This example shows a situation where changing the status of a state creates new ambiguity on occurrence of a fault. Indeed, q_3 is faulty in A and non faulty in A', because the fault occurring after action d may vanish. Hence, in A, one can claim that a fault has occurred as soon as action a is observed, and in A' observation of any word of the form $a.a'^*$ may correspond to faulty or non-faulty runs. Hence, word $b.a \in\in L_F^{min}(A) \cap L_F^{min}(A')$ belongs to $L_{diag}(A)$, but not to $L_{diag}(A')$.

Yet, removing vanishing faults and repairs is an interesting refactoring process for a partially observed system. First of all, it is obvious that when status changes within

a system are not followed by some observable event, one can not track the status of the system even with a delay, nor count the number of faults,.... Our interpretation of a vanishing fault (resp. repair) is the following: if a status change is not followed by an observation, then either it is not a noticeable event in the system, or the system is not well designed. Unimportant vanishing events should not be classified as faults (resp. repairs), and as their presence prevents from counting and diagnosing systems, the procedure shown above should be applied to remove these faults. If vanishing faults come from a lack of observation, then designers should impose more observable events.

4 Counting faults

As faults are not permanent, counting the number of faults occurring at runtime is a useful information: even if a system is able to repair all occurrences of faults, a too large number of faults may indicate a major failure. To count faults, an immediate idea is to maintain a fault counter that is incremented each time the diagnoser goes from N to F and from U to F. Even if a diagnosis can be triggered in time, i.e. before the fault is repaired, T-diagnosability is not sufficient to correctly count faults along a trajectory. Fig. 3 reveals that this can not work as counting moves of the diagnoser from $\{N,U\}$ to F in this example would detect two faults, while v has only one fault and v'' has two. Conversely, counting only moves from N to F or from U to F leads to minoring the real number of faults that occurred in some runs. This section considers extra conditions that enable counting. A *fault counter* C of an automaton A is a function from Σ_o^* to $\mathbb{N}$ such that: for every run $v \in L(A)$, letting k_v be the number of faults in v, $C(\Pi(v)) \in \{k_v - 1, k_v\}$. An automaton A is *fault countable* if there exists a fault counter of A.

Given an automaton A, $q, q' \in Q$, $k \in \mathbb{N}$ and $a \in \Sigma_o$, we write $q \rightarrow_A^{a,k} q'$ if there is a path in A from q to q' of unobservable events except for the last transition labeled by a with k faulty transitions.

Proposition 6 *Assuming that there are no vanishing repairs/faults in A, deciding if A is fault countable w.r.t. F is in NLOGSPACE.*

Proof Let $A = (S, \Sigma, T, s_0)$ be an automaton. As there is no vanishing repairs/faults, if $q \rightarrow_A^{a,k} q'$, then $k \leq 1$. We build the following variant of the twin automaton $A_C = (S_C, \Sigma, T_C, \{s_0, s_0, 0\})$ where:

- $S_C = S \times S \times \{-1, 0, 1, \perp\}$,
- $((q_1, q_2, n), a, (q_1', q_2', m)) \in T_C$ iff $q_1 \rightarrow_A^{a,k} q_1', q_2 \rightarrow_A^{a,k'} q_2'$ and
  - if $n = \perp$, then $m = \perp$,
  - if $n \neq \perp$ and $n + k - k' \in \{-1, 0, 1\}$, then $m = n - k + k'$,
  - if $n \neq \perp$ and $n + k - k' \notin \{-1, 0, 1\}$, then $m = \perp$.

This construction is of size at most $4.|A|^2$. Remark that the third component of the automaton keeps the difference of the number of faults by the two followed runs as long as this difference is not strictly greater than 1. It is set to $\perp$ if the difference gets bigger than 1.

We will now show that A is count diagnosable if and only if no state of the form $(q,q',\perp)$ is reachable, which can be tested in NLOGSPACE.

Let us suppose that there exists $q,q' \in S$ such that $(q,q',\perp)$ is reachable. Let v be a run reaching $(q,q',\perp)$. We suppose $(q,q',\perp)$ to be the first state of v which third component is a $\perp$. By construction of our twin product, we can associate v to two runs v_1 and v_2 of A with same observation and such that one of the two runs (say v_1) contains at least two more faults than the other. Consequently, suppose there exists a fault counter C, let k be the number of faults in v_1 (v_2 has thus less than $k-2$ faults), let $c = C(\Pi(v_1))$, then $c \leq k-2$ and $c \geq k-1$ by definition of a fault counter, which is not possible. Therefore A is not fault countable.

Conversely, suppose A is not fault countable. Then there exists $w \in \Sigma_o^*$ such that w is the projection of several runs of A, for which no correct estimation of the number of faults (up to one missing fault occurrence) can be given from observation w. By definition of a fault counter, this means that there exists two runs v_1 and v_2 such that $w = \Pi(v_1) = \Pi(v_2)$ and the number of faults in v_1 is at least two more than the number of fault in v_2. Let v be the run in A_C following the two runs v_1 and v_2. This run ends in a state $(s^+(v_1), s^+(v_2), \perp)$. Thus a state of A_C which third component is $\perp$ is reachable.

Remark that this proof does not immediately give the construction of a fault counter for the automaton. We will say that an automaton is T-Diagnosable w.r.t. N if repairs can be faithfully detected. Intuitively, this property can be checked by inversion of safe and faulty states, an then checking T-diagnosability of the so-obtained system. Consider the Diagnosis function $\Delta : L_o(A) \to \{N,F,U\}$ defined by (2).

We define the function $\sharp^F_\Delta$ from $L_o(A)$ to $\mathbb{N}$ as follows: Let $\mu \in L_o(A)$ and $\rho \in (N+U+F)^*$ the associated sequence of verdict emitted by Δ. Let $\rho' \in (N+F)^*$ be the projection of ρ on the verdicts $\{N,F\}$, then $\sharp^F_\Delta(\mu)$ is the number of occurrences of pairs NF that appear in ρ'. Intuitively, $\sharp^F_\Delta$ is a function that will be used to count the number of faults the diagnoser is able to detect. We can define similarly function $\sharp^N_\Delta$, counting the number of detected repairs, by inverting N and F in the previous definition.

Given a run u of A, $\sharp^F_A(u)$ denotes the number of times A moves from a normal state to a faulty state in u and $\sharp^N_A(\sigma(u))$ denotes the number of times A evolves from a faulty state to a normal state in u. Remark that detecting faults before they are repaired and detecting that a repair have occurred before the next fault are symmetric problems. The first problem is the notion of T-Diagnosis studied in section 3. The second problem can be handled with similar techniques, by considering that one has to detect moves from faulty state to normal ones before the next move from a normal state to a faulty one. We will hence say that A is T-Diagnosable w.r.t. F if one can detect occurrences of faults before they are repaired (A is T-diagnosable), and that A is T-Diagnosable w.r.t. N if one can detect that a system has been repaired before the occurrence of the next fault. We can now state the following proposition:

Proposition 7 If A is T-Diagnosable w.r.t. F and T-Diagnosable w.r.t. N, and has no vanishing faults nor repairs, then $\forall v \in L(A)$ and $\mu = \Pi(v)$

- $0 \leq \sharp^F_A(v) - \sharp^F_\Delta(\mu) \leq 1$.

- $0 \leq \sharp^N_A(v) - \sharp^N_\Delta(\mu) \leq 1$.

Moreover if $\Delta(\mu) = F$ then $\sharp^F_A(v) = \sharp_\Delta(\mu)$ and if $\Delta(\mu) = N$ then $\sharp^N_A(v) = \sharp^N_\Delta(\mu)$.

Proof We proceed by induction. The base case is $v = \varepsilon$. In this case, $\sharp^N_A(v) = 0$ and $\sharp^N_\Delta(\Pi(v)) = 0$ as $s_o \in N$ and $\Delta(\Pi(v)) \in \{N, U\}$ whereas $\sharp^F_A(v) = 0$ and $\sharp^F_\Delta(\Pi(v)) = 0$.

Let us now consider a sequence

$$v = v^N_1 v^F_1 \cdots v^N_k v^F_k \in L(A)$$

such that $\forall i \leq k$,

- $v^N_1 v^F_1 \cdots v^F_{i-1} \longrightarrow v^N_1 v^F_1 \cdots v^F_{i-1} v^N_i \in L_N(A)$, and
- $v^N_1 v^F_1 \cdots v^N_i \longrightarrow v^N_1 v^F_1 \cdots v^F_{i-1} v^N_i v^F_i \in L_F(A)$

and assume that

- $0 \leq \sharp^F_A(v) - \sharp^F_\Delta(\Pi(v)) \leq 1$.
- $0 \leq \sharp^N_A(v) - \sharp^N_\Delta(\Pi(v)) \leq 1$.

and that if $\Delta(\Pi(v)) = F$ then $\sharp^F_A(s) = \sharp^F_\Delta(\Pi(v))$ and if $\Delta(\Pi(v)) = N$ then $\sharp^N_A(s) = \sharp^N_\Delta(\Pi(v))$.

Consider now $v' = v v^N_{k+1} v^F_{k+1} \in L(A)$ and let $s_k \in S$ be the unique state reached by triggering v in A. By definition $s_k \in F$ and the state reached by triggering the first event in v^N_{k+1} belongs to N. As A has no vanishing-repair, $|\Pi(v^N_{k+1})| \geq 1$. Let u^N_{k+1} be the minimal prefix of v^N_{k+1} that ends with an observable. We then have two different cases

- either $\Delta(\Pi(v u^N_{k+1})) = N$ and we have

  $$\sharp^N_A(v u^N_{k+1}) = \sharp^N_\Delta(\Pi(v u^N_{k+1})) = k+1$$

  as $\sharp^N_A(v) = \sharp^N_\Delta(\Pi(v)) = k$, and as A is T-Diagnosable w.r.t F, $\Delta(\cdot)$ emitted the verdict F after $\Pi(v)$ while observing a sub-sequence of v^F_k. Thus $\sharp^N_\Delta(\cdot)$ is incremented by 1 as well as $\sharp^N_A(\cdot)$ as the sequence leads to a repair state.
- or $\Delta(\Pi(v u^N_{k+1}))) = U$, meaning that the diagnoser is still not able to say that the system is repaired and in that case $\sharp^N_A(v u^N_{k+1}) = \sharp^N_\Delta(\Pi(v u^N_{k+1})) + 1$. However, as A is T-Diagnosable w.r.t. N, there exists a prefix $u^N \in \Sigma^* \Sigma_o$ such that $u^N_{k+1} \leq u^N \leq v^N_{k+1}$, and $\Delta(\Pi(v u^N)) = N$. At this point $\sharp^N_A(v u^N) = \sharp^N_\Delta(\Pi(v u^N)) = k+1$. For the remaining sub-sequence up to v^N_{k+1} either the diagnoser can only emit the verdict N or the verdict U as the corresponding sequence is not faulty and thus the function $\sharp^N_\Delta()$ is not incremented.

The proof showing that the function $\sharp^F_\Delta()$ is also incremented by 1 after $v.v^N_{k+1}$ while reading v^F_{k+1} is symmetric to the previous case by replacing N by F and vice-versa.

Now, consider the case where $v = v^N_1 v^F_1 \cdots v^F_{k-1}.v^N_k \in L(A)$ ends in a normal state. We have $\Delta(v) \in \{N, U\}$, and $s_k \in N$.

Consider the extension of v with a sequence of faulty states v^F_{k+1}. As s_k is in N, as v^F_{k+1} starts with a faulty state and as we have no vanishing repair and A is T-diag w.r.t N, we have $\sharp^N_A(v) = \sharp^N_\Delta(v)$. As v^F_{k+1} is a sequence of faulty states, $\sharp^N_\Delta(v) = \sharp^N_\Delta(v.v^F_{k+1})$ We hence have $\sharp^N_A(v.v^F_{k+1}) = \sharp^N_\Delta(v) = k$, and the property is still satisfied. □

Intuitively, this proposition states that we can build from the diagnoser a function that counts the number of times the system becomes faulty (resp. is repaired) with a difference of at most 1. Furthermore, the difference is null as soon as the fault (resp repair) is diagnosed by the diagnoser. Note that the condition for counting in proposition 7 is sufficient, but not necessary as shown by the automaton of Figure 16.

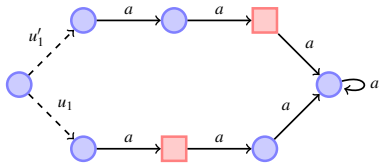

Fig. 16 A T-Diagnosable automaton wr.t. N but not w.r.t. F

In this example, the automaton is T-diagnosable w.r.t. N but not w.r.t. F, moreover the sequence of verdicts emitted by Δ is $NUUN$. However, after reading aa we know a single fault happened for sure.

5 Related work

The diagnosis of such transient faults has been considered in [9], which proposed four notions of diagnosability. One of them ("O-diagnosability") consists in detecting the occurrence of a transient fault, even after it has been repaired, which amounts to saturating $L_F(A)$ (see Section 2.2). Symmetrically, the "I-diagnosability" aims at detecting the occurrence of a repair, even if fault(s) followed, which amounts to inverting the roles of S_F and S_N, or to saturating the safe language $L_N(A)$. Both notions thus match the standard (or historical) notion of diagnosability for a slightly modified version of A.

In [15], two definitions involving multiple occurrences of faults are given. A system is $K-$diagnosable if the execution of any state-trace containing at least K failures can be deduced within a finite delay from the observed behavior. K-diagnosability is not monotonic, and the authors also introduce $[1\cdots K]$-diagnosability, that is met by systems that are J-diagnosable for every $1 \leq J \leq K$. Compared with $[1\cdots K]$-diagnosability or simply K diagnosability, we introduced a sufficient condition under which it is possible to count exactly the number of faults that occurred in the system. Furthermore, similarly to [9], the definitions of diagnosability introduced in [15], do not request the detection of the fault before its repair.

In the same manner, [9] introduced the notions of "P-diagnosability" and "R-diagnosability". These two notions are dual: P-diagnosability states that after the

occurrence of a fault, it is always possible to detect the fact that the system is currently faulty, based on the observation (even though the fault has been repaired in the past). Conversely, $R-$diagnosability states that after a fault is repaired, it is possible to detect in finite time whether the system is currently in a safe state. As we are mainly detecting fault occurrences, our work should only be compared to the notion of "P-diagnosability". Our notion of T-Diagnosability is then stronger than P-diagnosability, as we require that detection of faults occur *before they are repaired*. It is then easy to show that whenever a system is T-diagnosable then it is also P-diagnosable. We now express the notion of "P-diagnosability" in our context (i.e. using a state-based approach (Note that a similar definition has been recently proposed in [5])) and show that deciding whether an automaton A is P-diagnosable is a PSPACE complete problem. Formally, we have:

Definition 1 An automaton A is *P-diagnosable* w.r.t. Σ_o, F iff

$$\begin{array}{c} \exists n \in \mathbb{N},\ \forall v_1 \in L_F^{min}(A),\ \forall v_1 v_2 \in L(A), \\ [\ |v_2|_o \geq n\ \ \Rightarrow\ \ \exists v_2' \leq v_2\ :\ \Pi^{-1} \circ \Pi(v_1 v_2') \subseteq L_F(A)\ ] \end{array} \tag{12}$$

The authors of [9] are interested in the dynamic behaviour of discrete event systems where failure and reset events occur continuously along any path of the system's evolution. To represent this, they introduce the notions of Σ_f-recurrence and Σ_r-recurrence.

An automaton A is *Σ_f-recurrent* iff

$$\begin{array}{c} \exists n \in \mathbb{N},\ \forall v_1 \in L_F^{min}(A),\ \forall v_1 v_2 \in L(A), \\ [\ |v_2|_o \geq n\ \ \Rightarrow\ \ \exists v_2' \leq v_2\ :\ v_1 v_2' \subseteq L_N(A)\ ] \end{array} \tag{13}$$

We denote by $L_R^{min} = \{v\alpha \in L_N(A) \mid v \in L_F(A) \wedge \alpha \in \Sigma\}$ the set of minimal repaired words of A, *i.e.* words that correspond to a run ending with a transition from a faulty state to a normal one in A. An automaton A is *Σ_r-recurrent* iff

$$\begin{array}{c} \exists n \in \mathbb{N},\ \forall v_1 \in L_R^{min}(A),\ \forall v_1 v_2 \in L(A), \\ [\ |v_2|_o \geq n\ \ \Rightarrow\ \ \exists v_2' \leq v_2\ :\ v_1 v_2' \subseteq L_F(A)\ ] \end{array} \tag{14}$$

Under those restrictions, they obtained the following decidability result.

Proposition 8 ([9]) *Given a Σ_f-recurrent and Σ_r-recurrent automaton A, and assuming there is no vanishing fault, P-diagnosability of A can be decided in PSPACE.*

Note that, in [5], the authors propose another algorithm to decide the P-diagnosability of a system A based on the classical notion of twin-plant. However the tests that are necessary to check the P-diagnosability requires to check all the faulty runs that are equivalents in this machine, leading to a PSPACE algorithm for the test.

We complete these results by proving the PSPACE-hardness of $P-$Diagnosability. The proof reuses ingredient from the proof of hardness of T-diagnosability (Thm. 2) with some subtle differences.

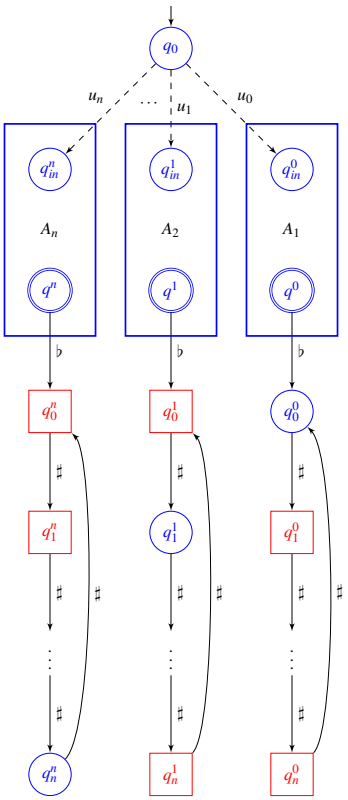

Fig. 17 PSPACE-hardness of P-diagnosability. Red states are faulty, and blue states normal states.

Proposition 9 *Given a Σ_f-recurrent and Σ_r-recurrent automaton A with no vanishing fault, P-diagnosability of A is PSPACE-hard.*

Proof The proof shows hardness of the problem by reduction from a language inclusion problem, which is known to be PSPACE-complete. The language inclusion problem can be formulated as follows: given $A_0,...,A_n$ some deterministic finite automata, does $\bigcap_{i\in 0..n} L(A_i) = \emptyset$?

Let $n \in \mathbb{N}$ and for $0 \leq i \leq n$, $A_i = (S_i, \Gamma, T_i, q^i_{in}, F_i)$ be some deterministic finite automaton on alphabet Γ. We build the finite automaton $A = (S, \Sigma, T, q_0)$ (see Figure 17) where:

- $\Sigma = \Gamma \cup \{u_0, \dots u_n\} \cup \{f, \sharp, \flat\}$
- $S = \{q_0\} \cup \{q^j_i \mid i, j \in^{0..n}\} \cup \{q^j_{in} \mid j \in 0..n\} \cup \bigcup_{0\leq i\leq n} S_i$
- $T = \{(q_0, u_j, q^j_{in}) \mid j \in 0..n\} \cup \{(q^j_n, \sharp, q^j_0 \mid j \in 0..n\} \cup \{(q, \flat, q^j_0) \mid q \in F_j, j \in 0..n\} \cup \{(q^j_i, \sharp, q^j_{i+1} \mid i = 0..n, j = 0 \dots n-1\} \cup \bigcup_{0\leq j\leq n} T_j$

The set of safe states is $S_F = \{q^i_i \mid i = 0 \dots n\}$. The set of faulty states is $S \setminus S_F$. We set $\Sigma_o = \Gamma \cup \{\sharp\}$.

We claim that A is P-diagnosable if and only if $\bigcap_{i\in 0..n} L(A_i) = \emptyset$.

First, remark that after observing $w\flat\sharp^m$ for $m \in \mathbb{N}$, the current run leads the automaton in some state q_k^j with $j \leq n$, $k \leq n$ and $k = m \mod (n+1)$ such that A_j accepts w.

Suppose that A is P-diagnosable. Let $w \in L(A_0)$, v_1 be the unique run of A such that $\sigma(v_1) = u_0 w\flat\sharp$. This run is unique as u_0 leads to the initial state of the deterministic automaton A_0. As v_1 is a minimal faulty run, there exists $m \in \mathbb{N}$ such that the run v_1v_2 with $\sigma(v_1v_2) = u_0w\flat\sharp^m$ verifies $\Pi^{-1} \circ \Pi(v_1v_2) \subseteq L_F(A)$. From our initial remark, as q_k^k with $k \leq n$ and $k = m \mod (n+1)$ is safe, it means that A_k does not accept w. As this is true for every $w \in L(A_0)$, $\bigcap_{i\in 0..n} L(A_i) = \emptyset$.

Conversely suppose that $\bigcap_{i\in 0..n} L(A_i) = \emptyset$. Let v_1 be a minimal faulty run. v_1 is of the form $\sigma(v_1) = u_i w\flat\sharp^m$ with $i \leq n$ and $w \in L(A_i)$. As $\bigcap_{i\in 0..n} L(A_i) = \emptyset$, there exists $j \leq n$ such that $w \notin L(A_j)$. Let $m_j \in \mathbb{N}$ be such that $j = m + m_j \mod n$ and consider the run v_1v_2 with $\sigma(v_1v_2) = u_i w\flat\sharp^{m+m_j}$. Thanks to our initial remark, as q_j^r with $r \neq j$ is a faulty state and A_j does not accept w, $\Pi^{-1} \circ \Pi(v_1v_2) \in L_F(A)$. Thus the fault can be claimed. As this is true for every minimal faulty run v_1, A is P-diagnosable. □

6 Conclusion

We have proposed a notion of "timely-diagnosability" that requires the detection (in bounded time) of transient faults after they occur, and before they are repaired. T-diagnosability for a deterministic partially observed automaton is decidable (in PSPACE). T-diagnosability is stronger than the P-diagnosability of [9] in the sense that the latter does not require that a transient fault be detected before it is repaired. Nevertheless, P-Diagnosability remains PSPACE complete, as shown in this paper. For deterministic systems that do not contain vanishing faults nor repairs, checking T-Diagnosability amounts to detecting ambiguous cycles and occurrences of repairs before any possible diagnosis. The question of whether a system contains vanishing faults (resp. repairs) is decidable with reasonable complexity. Requiring a system to be free from vanishing faults and repairs is quite a sensible restriction, and when a system does not satisfy this property, it can be corrected to avoid considering vanishing faults as real faults. Now, while determinism allows one to express diagnosability properties in terms of faulty and safe languages, it leads to quite complicated criteria for T-diagnosability, as in Theorem 1. Cleary, the T-diagnosability setting extends to systems with several types of faults by considering a set $F = \{F_1, \ldots F_k\}$ of sets of faulty states, each one corresponding to a particular fault. T-diagnosability of F is deduced from the T-diagnosability of each type of fault.

As future line of research, it could be interesting to define T-diagnosability for non-deterministic automata, and to explore whether criteria simplify. For example, it is likely that in the absence of vanishing faults and of vanishing repairs, T-diagnosability is preserved by Σ_o-closure. Also, while the T-diagnosability of faults relies on a complicated criterion, it is likely that systems which are both T-diagnosable for faults and for repairs are much easily characterized. This subclass is quite interesting, as it corresponds to systems where all changes of the status of the system from safe to faulty and conversely are detected in bounded time, and in any case before they change again. So ambiguity, when it appears, can not last forever.

Theorem 2 shows that checking T-diagnosability is a *PSPACE*-complete problem. *PSPACE* complexity yields an exponential time complexity *in worst cases*. It should be noted that the worst cases appear due to the fact that a diagnoser has to maintain a set of possible states reached after an observation, and that maintaining state estimates yields an exponential blowup. Again, this is a worst case, and in practice, $Diag(A)$ may not have an exponential number of states. We believe that some properties of automata, such as the *observer property* (see for instance [20]) can help mastering complexity. Indeed, in automata that meet this property, all observationally equivalent runs ending on different states are extended with equivalent runs. If an automaton satisfies the observer property, then all branching in the automaton remain visible in its projection on observable actions. In a diagnosis setting, this allows to keep the same order of magnitude for the sizes of A and $Diag(A)$.

Besides these immediate perspectives, the future of this work is definitely in the direction of quantitative analysis. Being able to characterize exactly, after a bounded delay, in which state class lies system A is a very strong property. A more relevant question would be to determine how likely it is that A is in S_N or S_F given partial observations, and whether this relative certainty passes some threshold in a bounded time after that status of a system A has changed.

Acknowledgements

The authors would like to thank Francois Godi, Xavier Montillet and Chen Qian, master students at ENS Rennes, for interesting discussions that led to this work.

References

1. Eric Badouel, Marek Bednarczyk, Andrzej Borzyszkowski, Benoît Caillaud, and Philippe Darondeau. Concurrent secrets. *Discrete Event Dynamic Systems*, 17(4):425–446, 2007.
2. A. Benveniste, E. Fabre, S. Haar, and C. Jard. Diagnosis of asynchronous discrete event systems: A net unfolding approach. *IEEE Trans. Automat. Contr.*, 48(5):714–727, 2003.
3. Nathalie Bertrand, Serge Haddad, and Engel Lefaucheux. Foundation of diagnosis and predictability in probabilistic systems. In Venkatesh Raman and S. P. Suresh, editors, *34th International Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2014, December 15-17, 2014, New Delhi, India*, volume 29 of *LIPIcs*, pages 417–429. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2014.
4. Nathalie Bertrand, Serge Haddad, and Engel Lefaucheux. Diagnosis in infinite-state probabilistic systems. In Josée Desharnais and Radha Jagadeesan, editors, *27th International Conference on Concurrency Theory, CONCUR 2016, August 23-26, 2016, Québec City, Canada*, volume 59 of *LIPIcs*, pages 37:1–37:15. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.
5. Ab. Boussif, B. Liu, and M. Ghazel. A twin-plant based approach for diagnosability analysis of intermittent failures. In *13th International Workshop on Discrete Event Systems*, pages 237–244, Xi'an, China, 2016.
6. Jeremy Bryans, Maciej Koutny, Laurent Mazaré, and Peter Y. A. Ryan. Opacity generalised to transition systems. *International Journal of Information Security*, 7(6):421–435, 2008.
7. Maria Paola Cabasino, Alessandro Giua, Stéphane Lafortune, and Carla Seatzu. Diagnosability analysis of unbounded petri nets. In *Proceedings of the 48th IEEE Conference on Decision and Control, CDC 2009, combined withe the 28th Chinese Control Conference, December 16-18, 2009, Shanghai, China*, pages 1267–1272. IEEE, 2009.
8. F. Cassez and S. Tripakis. Fault diagnosis with static and dynamic diagnosers. *Fundamenta Informaticae*, 88(4):497–540, November 2008.

9. Olivier Contant, Stéphane Lafortune, and Demosthenis Teneketzis. Diagnosis of intermittent faults. *Discrete Event Dynamic Systems*, 14(2):171–202, 2004.
10. E. Fabre, L. HÍouet, E. Lefaucheux, and H. Marchand. Diagnosability of repairable faults. In *13th International Workshop on Discrete Event Systems*, pages 256–262, Xi'an, China, 2016.
11. S. Genc and S. Lafortune. Distributed diagnosis of discrete-event systems using petri nets. In *Applications and Theory of Petri Nets (ICATPN) 2003*, volume 2679 of *LNCS*, pages 316–336, 2003.
12. Stefan Haar, Serge Haddad, Tarek Melliti, and Stefan Schwoon. Optimal constructions for active diagnosis. *J. Comput. Syst. Sci.*, 83(1):101–120, 2017.
13. T. Jéron, H. Marchand, S. Pinchinat, and M-O. Cordier. Supervision patterns in discrete event systems diagnosis. In *Workshop on Discrete Event Systems, WODES'06*, pages 262–268, Ann-Arbor (MI, USA), July 2006.
14. S. Jiang, Z. Huang, V. Chandra, and R. Kumar. A polynomial time algorithm for diagnosability of discrete event systems. *IEEE Transactions on Automatic Control*, 46(8):1318–1321, 2001.
15. 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, April 2003.
16. Dexter Kozen. Lower bounds for natural proof systems. In *18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October - 1 November 1977*, pages 254–266. IEEE Computer Society, 1977.
17. B. Lampson. A note on the confinement problem. *Communication of the ACM*, 16(10):613–615, 1973.
18. Klaus-Jörn Lange and Peter Rossmanith. The emptiness problem for intersections of regular languages. In *Mathematical Foundations of Computer Science 1992, 17th International Symposium, MFCS'92, Prague, Czechoslovakia, August 24-28, 1992, Proceedings*, pages 346–354, 1992.
19. NSA/NCSC. A guide to understanding covert channel analysis of trusted systems. Technical report, NSA/NCSC, 1993.
20. Patrícia N. Pena, Hugo J. Bravo, Antonio Eduardo Carrilho da Cunha, Robi Malik, Stéphane Lafortune, and José E. R. Cury. Verification of the observer property in discrete event systems. *IEEE Trans. Automat. Contr.*, 59(8):2176–2181, 2014.
21. M. Sampath, R. Sengupta, S. Lafortune, K. Sinaamohideen, and D. Teneketzis. Diagnosability of discrete event systems. *IEEE Transactions on Automatic Control*, 40(9):1555–1575, 1995.
22. Meera Sampath, Raja Sengupta, Stéphane Lafortune, Kasim Sinnamohideen, and Demosthenis Teneketzis. Failure diagnosis using discrete-event models. *IEEE Trans. Contr. Sys. Techn.*, 4(2):105–124, 1996.
23. Walter J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *J. Comput. Syst. Sci.*, 4(2):177–192, 1970.
24. A. Prasad Sistla and Edmund M. Clarke. The complexity of propositional linear temporal logics. *J. ACM*, 32(3):733–749, 1985.
25. D. Thorsley and D. Teneketzis. Diagnosability of stochastic discrete-event systems. *IEEE Trans. Automat. Contr.*, 50(4):476–492, 2005.
26. James C. Tiernan. An efficient search algorithm to find the elementary circuits of a graph. *Commun. ACM*, 13(12):722–726, 1970.
27. J. Zaytoon and S. Lafortune. Overview of fault diagnosis methods for discrete event systems. *Annual Reviews in Control*, 37(2):308–320, 2013.