Random generator of k-diagnosable discrete event systems

Yannick Pencolé[1 2]
[1]CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France
[2]Univ de Toulouse, LAAS, F-31400 Toulouse, France
e-mail: yannick.pencole@laas.fr

Abstract

This paper presents a random generator of discrete event systems that are by construction k-diagnosable. The aim of this generator is to provide an almost infinite set of diagnosable systems for creating benchmarks. The goal of such benchmarks is to provide a solid set of examples to test and compare algorithms that solve many problems around diagnosable discrete event systems.

1 Introduction

For many years, the problem of fault diagnosis in discrete event systems has been actively addressed by different scientific communities such as DX (AI-based diagnosis) [1; 2], FDI (Fault Detection and Isolation), DES (Discrete Event Systems) [3]. Depending on the community, many differents aspects of the same problem have been addressed such as the design of efficient diagnosers, the checking of diagnosability properties, the effective modelling of real systems. When dealing with performance, most of the contributions present experimental results on specific examples of their own usually inspired or based on real world systems. The main problem about these contributions is that they are not really comparable as they are not applied on the same benchmarks. Moreover, used benchmarks may not be always completly defined in a paper due, most of the time, to confidential data that cannot be published so other academic contributors cannot use them for comparison purposes. In order to analyse and boost the effective performance of algorithms addressing the fault diagnosis problem in DES, common and fully available benchmarks become a necessity.

This paper addresses the random generation of k-diagnosable systems. We here propose the possibility to generate (and store on a web page) k-diagnosable systems that have been generated without any kind of bias that would come from a specific diagnosis/diagnosability method. By doing so, we propose to design a random category for benchmarks as the SAT community proposed for SAT problems and to get the same advantages by comparing different diagnosis/diagnosability approaches on the same but random systems. The choice of generating k-diagnosable systems is motivated by the fact that they can be used as examples for:

1. diagnosis algorithms: given a fault f, we know by construction that the most precise algorithm will determine its occurrence with certainty within the next k observations after the occurrence of f;
2. diagnosability algorithms: the fact that a fault f is k-diagnosable is usually the worst case for this type of algorithms (as they all look for the existence of an ambiguous scenario to conclude the system is not diagnosable).

The paper is organised as follows. After formally recalling the problem that motivates the generation of benchmarks, we describe the fundamental property which is being used for the effective generation of systems where a given fault f is k-diagnosable. Then the description of the algorithm of the generator is provided as well as some details about its effective implementation.

2 Background

This paper addresses the random generation of benchmarks for the problem of the fault diagnosis of discrete event system. This problem is briefly recalled in this section. We assume that the reader is familiar with the notations of the language theory (notion of Kleene closure, prefixes,...).

2.1 Modelling

We suppose that the system under monitoring behaves as an event generator that can be modelled as an automaton.

Definition 1 (System description)**.** *The model (system description)* SD *of a discrete event system S is a finite state automaton* $\mathrm{SD} = (Q, \Sigma, T, q_0)$ *where:*

- *Q is a finite set of states;*
- *Σ is a finite set of events;*
- *$T \subseteq Q \times T \times Q$ is a finite set of transitions;*
- *q_0 is the initial state of the system.*

Σ is the set of events that the system can produce. Among Σ we distinguish events that are observable $\Sigma_o \subseteq \Sigma$ and events that are not observable. When the system operates, its effective behaviour is represented by a trace of the automaton (also called a run).

Definition 2 (Trace)**.** *A trace $\tau \in \Sigma^*$ of the system is a finite sequence of events associated with a transition path from the initial state q_0 to a state q in the model of the system.*

The set of traces of the system is the language generated by its model and is denoted $\mathcal{L}(S)$ (so the automaton SD generates the language $\mathcal{L}(S)$). Let $P_{\Sigma'}(\tau)$ be the classical projection of a sequence τ of Σ^* on the alphabet Σ' recursively defined as follows:

1. $P_{\Sigma'}(\varepsilon) = \varepsilon$;

2. $P_{\Sigma'}(\tau.e) = P_{\Sigma'}(\tau)$ if $e \notin \Sigma'$;
3. $P_{\Sigma'}(\tau.e) = P_{\Sigma'}(\tau).e$ if $e \in \Sigma'$.

Based on this notion of projection, we can associate with any trace of the system its observable part.

Definition 3 (Observable trace)**.** *Let τ be a trace of the system, the observable trace σ_τ is the projection of τ over the set of observable events Σ_o:*

$$\sigma_\tau = P_{\Sigma_o}(\tau).$$

2.2 Diagnosis problem and solution

Now we are ready to define the classical Fault diagnosis problem on DES.

Definition 4 (Fault)**.** *A* fault *is a non-observable event $f \in \Sigma$.*

A fault is represented as a special type of non-observable event that can occur on the underlying system. Once the event has occurred, we say that the fault is *active* in the system, otherwise it is *inactive*. We consider here the problem of permanent faults as initially introduced in [4].

Definition 5 (Diagnosis problem)**.** *A diagnosis problem is a triple* $(\mathrm{SD}, \mathrm{OBS}, \mathrm{FAULTS})$ *where* SD *is the model of a system,* OBS *is the sequence of observations of $\Sigma_o^\star$ and* FAULTS *is the set of fault events defined over* SD*.*

Informally speaking, $(\mathrm{SD}, \mathrm{OBS}, \mathrm{FAULTS})$ represents the problem of finding the set of active faults from FAULTS that have occurred relying on the model SD and the sequence of observations OBS.

Definition 6 (Diagnosis Candidate)**.** *A diagnosis candidate is a couple (q, F) where q is a state of* SD *($q \in Q$) and F is a set of faults.*

A diagnosis candidate represents the fact that the underlying system is in state q and the set F of faults has occurred before reaching state q.

Definition 7 (Solution Diagnosis)**.** *The* solution Δ *of the problem* $(\mathrm{SD}, \mathrm{OBS}, \mathrm{FAULTS})$ *is the set of diagnosis candidates (q, F) such that there exists for each of them at least one trace τ of* SD *such that:*

1. *the observable trace of τ is exactly the sequence* $\mathrm{OBS} = o_1 \dots o_m$ *and the last event of τ is o_m;*
2. *the set of fault events that has occurred in τ is exactly F;*
3. *the final state of τ is q.*

Informally, candidate (q, F) is part of the solution if it is possible to find out in SD a behaviour of the system satisfying OBS which leads to the state q after the last observation of OBS and in which the faults F have occurred.

2.3 Diagnosability

Diagnosability is a property of the system that asserts whether a fault f of a system S can be always diagnosed with certainty after the observation of a finite set of observations [4]. In other words, once the fault f has occurred in S, it is sufficient to wait a certain amount of observations to ensure that any candidate (q, F) of the solution contains f ($f \in F$).

Definition 8 (Diagnosability)**.** *The fault f is diagnosable in a system S if:*

$$\exists n \in \mathbb{N}^+, \mathit{Diagnosable}(n)$$

where $\mathit{Diagnosable}(n)$ stands for:

$$\forall \tau_1.f \in \mathcal{L}(S), \forall \tau_2 : \tau_1.f.\tau_2 \in \mathcal{L}(S)$$
$$|P_{\Sigma_o}(\tau_2)| \geq n \Rightarrow$$
$$(\forall \tau \in \mathcal{L}(S), (P_{\Sigma_o}(\tau) = P_{\Sigma_o}(\tau_1.f.\tau_2) \Rightarrow f \in \tau)).$$

Definition 9 (k-Diagnosability)**.** *The fault f is k-diagnosable, $k \in \mathbb{N}^+$, in a system S if:*

$$\mathit{Diagnosable}(k) \wedge \neg \mathit{Diagnosable}(k-1).$$

Diagnosability is a property that relies on the liveness of the observability of the system which means that, to be (k)-diagnosable, a system must not generate unbounded sequences of unobservable events (no cycle of unobservable events in SD). Throughout this paper, we consider that the observability of the system is live.

3 Random Generator

The aim of this section is to present the algorithm that is being used to randomly generate a discrete event systems where a fault f is k-dignosable and that has been implemented inside the Diades software. We focus on the generation of a system with one fault only. (see th conclusion for the generation for $n, n > 1$ faults).

3.1 Signatures and fault ambiguity

The algorithm that generates a k-diagnosable system relies on the notion of signatures. Let f be a faulty event, the signature of f is the set of observable traces resulting from the projection of system traces that contain at least one occurrence of an event f before the last observation of the trace.

Definition 10 (Signature)**.** *The signature of an event f into a system S is the language $Sig(f) \subseteq \Sigma_o^\star$ such that*

$$Sig(f) = \{\sigma_\tau | \tau = \tau_1.o.\tau_2 \in \mathcal{L}(S), f \in \tau_1, o \in \Sigma_o, \tau_2 \in \Sigma^\star, \sigma_\tau = P_{\Sigma_o}(\tau)\}.$$

In the following, we will also denote by $Sig(\neg f)$ the set of observable traces associated with the traces of the system that do not contain any fault f before the last observation. Intuitively speaking, as long as the current observable trace is in $Sig(\neg f) \cap Sig(f)$, we know that the system may have produced a faulty trace or a non-faulty trace before the last observation. k-diagnosability ensures that the ambiguity can last at most for k observations. The principle of the generator relies on the following result that formalizes this intuition. Let $\textsc{LargestPrefixes}(\tau, n) = \{\tau_i' : \tau = \tau_i'\tau_i, |\tau_i| = i, i \in \{0, \dots, n-1\}\}$ be the set of the n largest prefixes of τ (τ being a prefix of itself).

Theorem 1. *In the system S, the event f is k-diagnosable if and only if:*

1. *For any observable trace σ in $Sig(\neg f) \cap Sig(f)$, there exists $n < k$ such that* $\textsc{LargestPrefixes}(\sigma, n) \subseteq Sig(\neg f) \cap Sig(f)$ *and* $\textsc{LargestPrefixes}(\sigma, n+1) \not\subseteq Sig(\neg f) \cap Sig(f)$.
2. *There exists at least one observable trace σ in $Sig(\neg f) \cap Sig(f)$ such that* $\textsc{LargestPrefixes}(\sigma, k-1) \subseteq Sig(\neg f) \cap Sig(f)$ *and an observable o such that $\sigma o \in Sig(f)$.*

Proof: ($\Rightarrow$)Let $\tau_1.f$ be a trace of the system S. As S is k-diagnosable, there exists $m \leq k$ such that $\forall \tau_2 : \tau_1.f.\tau_2 \in \mathcal{L}(S), |P_{\Sigma_o}(\tau_2)| \geq m \Rightarrow (\forall \tau \in \mathcal{L}(S), (P_{\Sigma_o}(\tau) = P_{\Sigma_o}(\tau_1.f.\tau_2) \Rightarrow f \in \tau))$. Consider one of these trace $\tau_1.f.\tau_2$ such that τ_2 contains exactly m observations ($P_{\Sigma_o}(\tau_2) = o_1 \dots o_m$). k-diagnosability implies that there exists a minimal integer $n \in \{1, \dots, m-1\}$ such that $P_{\Sigma_o}(\tau_1.f).o_1 \dots o_{n+1} \Rightarrow f \in \tau$ as soon as $\tau \in \mathcal{L}(S)$ and $P_{\Sigma_o}(\tau) = P_{\Sigma_o}(\tau_1.f).o_1 \dots o_{n+1}$, therefore $P_{\Sigma_o}(\tau_1.f).o_1 \dots o_{n+1} \in Sig(f) \setminus Sig(\neg f)$ and $\forall i \in \{1, \dots, n\}, P_{\Sigma_o}(\tau_1.f).o_1 \dots o_i \in Sig(f) \cap Sig(\neg f)$. So $\text{LargestPrefixes}(P_{\Sigma_o}(\tau_1.f).o_1 \dots o_n, n) \subseteq Sig(\neg f) \cap Sig(f)$. So for any $\tau_1.f$ there exists $n < k$ such that $\text{LargestPrefixes}(P_{\Sigma_o}(\tau_1.f).o_1 \dots o_n, n) \subseteq Sig(\neg f) \cap Sig(f)$. Now, remark that for any observable sequence σ that belongs to $Sig(\neg f) \cap Sig(f)$, there must exist a trace $\tau_1.f.\tau_2$ of the system, with τ_2 containing at least one observable event, such that $\sigma = P_{\Sigma_o}(\tau_1.f.\tau_2)$, so there must exist $n < k$ such that $\sigma \in \text{LargestPrefixes}(P_{\Sigma_o}(\tau_1.f).o_1 \dots o_n, n) \subseteq Sig(\neg f) \cap Sig(f)$ so, for any σ that belongs to $Sig(\neg f) \cap Sig(f)$, there is no set $\text{LargestPrefixes}(\sigma, n+1)$ that only contains ambiguous signatures.

Finally, as S is k-diagnosable, we know that there exists at least one trace $\tau.f.\tau_1.o_1$, such that τ is a trace of the system that does not contain f, τ_1 is a finite continuation of $\tau.f$ that is unobservable and o_1 is observable and there is a finite continuation $\tau_2 o_2 \tau_3 o_3 \dots \tau_k o_k$ with $P_{\Sigma_o}(\tau_i) = \varepsilon$ such that for any $i \in \{1, \dots, k-1\}$, $P_{\Sigma_o}(\tau.f.\tau_1.o_1 \dots \tau_i o_i) \in Sig(\neg f) \cap Sig(f)$ $P_{\Sigma_o}(\tau.f.\tau_1.o_1 \dots \tau_k o_k) \in Sig(f) \setminus Sig(\neg f)$ which implies the condition 2 with $\sigma = P_{\Sigma_o}(\tau.f.\tau_1.o_1 \dots \tau_{k-1} o_{k-1})$.

($\Leftarrow$) Suppose now that conditions 1 and 2 hold. Consider an observable trace σ that is ambiguous ($\sigma \in Sig(f) \cap Sig(\neg f)$). Condition 1 states that there exists $n < k$ such that $\text{LargestPrefixes}(\sigma, n) \subseteq Sig(\neg f) \cap Sig(f)$ and $\text{LargestPrefixes}(\sigma, n+1) \not\subseteq Sig(\neg f) \cap Sig(f)$. Consider now any largest observable trace σ' such that $|\sigma'| - |\sigma| = m$ and $\sigma \in \text{LargestPrefixes}(\sigma', m) \subseteq Sig(\neg f) \cap Sig(f)$, it follows that $\text{LargestPrefixes}(\sigma', m+n) \subseteq Sig(\neg f) \cap Sig(f)$ and $\text{LargestPrefixes}(\sigma', m+n+1) \not\subseteq Sig(\neg f) \cap Sig(f)$. As σ' is one of the largest observable trace holding this condition, any observable trace $\sigma' o, o \in \Sigma_o$, is either in $Sig(f)$ or in $Sig(\neg f)$ but not in both of them. Condition 1 states that $m + n < k$, so k observations at least are required to solve the ambiguity. Condition 2 states that there exists at least such an observable trace σ' with $m + n = k - 1$ and an observation o so that $\sigma' o$ is definitively in $Sig(f)$ so f can be diagnosed with certainty in this case with exactly k observations. Hence the result. □

3.2 Algorithm

The principle of the random generator is depicted in Algorithm 1. Given a parameter k and a fault event f, the algorithm randomly generates a system S where the event f is k-diagnosable by construction. We also provide another parameter deg which is the maximal number of output transitions that is allowed per state during the generation of the system. Parameter deg is important for the creation of benchmarks as the output degree has a strong influence on the diagnosis/diagnosability computations.

Algorithm 1 General algorithm for the random generation of k-diagnosable systems.

Input: $k \in \mathbb{N}, k \geq 1$
Input: f an event
Input: deg maximal output degree
$(\Sigma_o, \Sigma) \leftarrow$ GenerateEvents()
$S \leftarrow \varnothing$
$AmbSig(f) \leftarrow$ GenerateAmbSignature(k,Σ_o, deg)

/* $AmbSig(f) = (Q, \Sigma_o, T, q_0, A)$ a deterministic automaton */
MF[q_0] $\leftarrow$ GenerateStates()
MNF[q_0] $\leftarrow$ GenerateStates()
for all $q \in Q$ in Breadth-First Order from q_0 **do**
  $(\Sigma_o^f, \Sigma_o^{\neg f}) \leftarrow$ RandomSplit(Σ_o, q)
  $S \leftarrow S \cup$ GenFaultExts*(MF[q],Σ_o^f,deg)
  $S \leftarrow S \cup$ GenNomExts*(MNF[q],$\Sigma_o^{\neg f}$,deg)
  for all $q \xrightarrow{o} q' \in T$ **do**
    if MF[q'] $= \varnothing$ **then**
      MF[q'] $\leftarrow$ GenerateStates()
      MNF[q'] $\leftarrow$ GenerateStates()
    end if
    $S \leftarrow S \cup$
    GenNomExts(MNF[q],MNF[q'],o,deg)
    if $q' \notin A$ **then**
      $S \leftarrow S \cup$ GenNomExts(MF[q],MF[q'],o,deg)
    else
      if $q \in A$ **then**
        $S \leftarrow S \cup$ GenExts(MF[q],MF[q'],o,deg)
      else
        $S \leftarrow S \cup$
        GenFaultExts(MF[q],MF[q'],o,deg)
      end if
    end if
  end for
end for
Output: S where f is k-diagnosable.

The generation is composed of two steps. The first one is the generation of the ambiguous signature with GenerateAmbSignature. The result of this function is a deterministic automaton $AmbSig(f) = (Q, \Sigma_o, T, q_0, A)$ that actually generates the language $Sig(f) \cap Sig(\neg f)$ (any transition path from state q_0 to an accepting state of A represents a sequence of $Sig(f) \cap Sig(\neg f)$). The automaton $AmbSig(f)$ is generated with respect to the conditions 1 and 2 that are defined in Theorem 1 to ensure the k-diagnosability of the resulting system S. The second step of the generation is the effective generation of S based on the ambiguous signature $AmbSig(f)$. The idea is to map every state q of $AmbSig(f)$ with two sets of states in S denoted $MF[q]$ and $MNF[q]$. Given any path σ of $AmbSig(f)$ that leads to state q with, as a last observation, the event o, any state of $MF[q]$ (resp. $MNF[q]$) will be reached by at least one transition path τ of S starting from a state of $MF[q_0]$ (resp. $MNF[q_0]$) that ends with a transition labelled with o and the observable projection of τ is exactly σ. The difference between MF and MNF is that any underlying path of S leading to a state of $MF[q]$ (resp. $MNF[q]$) has an observable projection which is a prefix of $Sig(f)$ (resp. a prefix of $Sig(\neg f)$). To generate S we explore $AmbSig(f)$ from its initial state in a breadth-first search manner. For a given state q, we have to consider three types of transition

generations going out of any state of $MF[q], MNF[q]$. The first ones are the transition paths that will lead to an observation o that belongs to $AmbSig(f)$, the second one is the set of transition paths that do not lead to an observation o that belongs to $AmbSig(f)$ but lead to an observation o' that belongs to $Sig(f)$ only and the third one is the set of transition paths that do not lead to an observation o that belongs to $AmbSig(f)$ but lead to an observation o'' that belongs to $Sig(\neg f)$ only.

The second and third cases are handled by randomly splitting Σ_o into two subsets $(\Sigma_o^f, \Sigma_o^{\neg f})$ each of them only containing observable events that are not output event of q in $AmbSig(f)$ (RANDOMSPLIT(Σ_o, q)). Then given Σ_o^f, we randomly generate faulty extensions for a subset of Σ_o^f (the selection of the subset is also random and might even be empty if q has no output events in $AmbSig(f)$, indeed if q has no output events, it must be extended to ensure that the observability of the system is live). An extension is a set of acyclic and unobservable transition paths that lead to a transition labeled with an observable event from Σ_o^f. A faulty extension ensures that an event f has at least occurred on any generated transition path before the observable transition (GENFAULTEXTS). Given $\Sigma_o^{\neg f}$, we proceed the same way to generate non-faulty extensions (GENNOMEXTS). As, in these two cases, the traces generated by these extensions are not associated with observable traces involved in $AmbSig(f)$ any more, it is sufficient to generate further extensions on these traces and guarantee that the observable language associated with these further extensions is live (this procedure is denoted by the * in GENNOMEXTS* and GENFAULTEXTS*).

The last case to handle now is the case where the observable event o is an output event of q in $AmbSig(f)$, which means that there exists one and only one transition $q \xrightarrow{o} q'$ in $AmbSig(f)$. If q' has never been visited, the set of states $MF[q']$ and $MNF[q']$ are generated first. A nominal extension is generated from $MNF[q]$ to $MNF[q']$. Depending on the status of q', the extension between $MF[q]$ and $MF[q']$ is different. If $q' \notin A$, it means that any prefix generated by $AmbSig(f)$ with paths from q_0 to q' are prefixes of sequences in $Sig(f) \cap Sig(\neg f)$ but they are not in $Sig(f) \cap Sig(\neg f)$, they can therefore be only in $Sig(\neg f)$: extensions between $MF[q]$ and $MF[q']$ are then nominal extensions. Now, if $q' \in A$, there are two cases. If $q \notin A$, it means the system must become faulty between the states of $MF[q]$ and $MF[q']$ so that paths of the system that reach any state of $MF[q']$ is associated with an observable trace that belongs to $Sig(f)$ (GENFAULTEXTS). If $q \in A$, any path that reaches a state of $MF[q]$ is already faulty (its observable trace is already in $Sig(f)$), any type of extension from $MF[q]$ to $MF[q']$ is therefore possible (faulty or not), hence the use of GENEXTS.

3.3 Implementation

Algorithm 1 is implemented with the help of the Diades library package [5]. Diades is a set of C++ libraries that implement discrete event systems in a component-based way, different diagnosis algorithms as defined in the spectrum of [6] (from component-based algorithms to diagnoser-based algorithms). DIADES also implements a diagnosability checker as well as an accuracy checker. The generator results in a Linux terminal command `dd-diagnosable-des-generate` with a set of parameters like the number of (un)observable events the output degree of transitions, the parameter k, the minimal number of observable events involved in the ambiguous signature, the number of states (still experimental). One particular parameter is the seed parameter that allows the generation of the same system (the seed ensures the same generation of random numbers). By construction, the algorithm is linear in the number of states. A set of pre-computed benchmarks as well as the implemented generator are available at the following url:
`http://homepages.laas.fr/ypencole/benchmarks`

4 Conclusions

To test and compare diagnosis and/or diagnosability algorithms, fully detailed and available benchmarks are a necessity. In order to test how generic is an algorithm, we propose here an algorithm that randomly generates systems where a fault f is k-diagnosable. We also propose an implementation within the DIADES framework. Extension to generate systems with n k-diagnosable faults is easy, it requires to repeat the generation of the ambiguous signatures for the n faults and explore them in parallel to generate the k-diagnosable system. Our short-term perspective is to improve the generator to allow a better control of the number of generated states. A fixed number of generated states requires to add new constraints in the generation that propagate during the generation process. Without any control about this propagation, the generation may just fail as it could become an over-constrained problem. Our perspective is to also go one step further by generating diagnosable systems that are component-based in order to scale up the size of the generated system. The DIADES framework already has a tool to generate component-based systems [5] which ensures that any component is globally consistent, but adding the constraint of diagnosability makes the generation far more complex to implement.

References

[1] Gianfranco Lamperti and Marina Zanella. Diagnosis of discrete-event systems from uncertain temporal observations. *Artificial Intelligence*, 137:91–163, 2002.

[2] Yannick Pencolé and Marie-Odile Cordier. A formal framework for the decentralised diagnosis of large scale discrete event systems and its application to telecommunication networks. *Artificial Intelligence*, 164(2):121–170, 2005.

[3] Janan Zaytoon and Stéphane Lafortune. Overview of fault diagnosis methods for discrete event systems. *Annual Reviews in Control*, 37:308–320, 2013.

[4] Meera Sampath, Raja Sengupta, Stéphane Lafortune, Kasim Sinnamohideen, and Demosthenis Teneketzis. Diagnosability of discrete-event systems. *Transactions on Automatic Control*, 40(9):1555–1575, 9 1995.

[5] Yannick Pencolé. Fault diagnosis in discrete-event systems: How to analyse algorithm performance? In *Diagnostic reasoning: Model Analysis and Performance*, pages 19–25, Montpellier, France, 2012.

[6] Anika Schumann, Yannick Pencolé, and Sylvie Thiébaux. A spectrum of symbolic on-line diagnosis approaches. In *17th International Workshop on Principles of Diagnosis*, pages 194–201, Nashville, TN USA, 2007.