Symbolic Testing of Diagnosability

Alban Grastien [*,**]

[*] *NICTA, Canberra Research Lab, Canberra, Australia*
[**] *The Research School of Information Sciences and Engineering (RSISE), The Australian National University, Canberra, Australia*

Abstract: Diagnosability ensures that the occurrence of a failure on the system can always be diagnosed by a diagnosis engine. In this paper, we explore symbolic techniques to test diagnosability of DES. We present several algorithms that can be implemented with symbolic tools, and show how to combine decentralised approach with symbolic approach. Finally, we discuss how to extract the minimal set of sensors that ensures diagnosability.

Keywords: Discrete-Event Systems, Diagnosability, Symbolic Techniques

1. INTRODUCTION

Diagnosis is the AI problem of determining whether a specified system is running as expected thanks to observations on its behaviour; in case of an unexpected behaviour, the diagnosis engine should be able to localise and identify the failure as accurately as possible. Another procedure can then take place to fix the failure or reconfigure the plant. When the observations provide too little information on the current behaviour of the plant, the diagnosis engine is not able to return a precise diagnosis. Therefore an important feature to ensure when designing a system is the *diagnosability* which states that a failure can always be precisely diagnosed.

We are interested in the model-based diagnosis of a dynamic system modeled by a discrete-event system (DES, Cassandras and Lafortune (1999)) – basically a finite-state automaton – and consider only one possible failure as results can be easily generalised. The model is complete, i.e., it represents both the nominal and the failure behaviours. A system is diagnosable iff whenever a fault occurs, the diagnosis engine should be able to diagnose it, possibly after a delay. Equivalently, a system is *not* diagnosable iff there exists a pair of infinite executions on the DES that generate the same observations, one of which exhibits a fault and not the other one (pair of critical paths). Testing diagnosability was proved NLOGSPACE-hard for enumative representations, and PSPACE-hard for succinct (symbolic) representations (Rintanen (2007)). The design of efficient algorithms for diagnosability testing attracted a lot of attention in the past years (Sampath et al. (1995); Console et al. (2000); Jiang et al. (2001); Cimatti et al. (2003); Pencolé (2004); Rintanen and Grastien (2007); Schumann and Pencolé (2007); Cordier et al. (2007)). Unfortunately, because of complexity issues, these techniques are usually not applicable for many large real-world DES.

We propose several algorithms based on symbolic model-checking techniques. The existence of critical trajectories is checked by the computation of fixed-points on pairs of states of the DES. The symbolic techniques allow one to manipulate large sets of states. and have already been proposed for the diagnosis of DES (Schumann et al. (2007); Grastien et al. (2007)) or even to test diagnosability (Rintanen and Grastien (2007); Cimatti et al. (2003)) and are widely used in planning (Kautz and Selman (1996)) and model-checking (McMillan (1993)). We also show how to decentralise these algorithms to improve the runtime. Finally, we present an algorithm to find a minimal set of observable events that makes the system diagnosable.

The paper is structured as follows. Next section gives general definitions on diagnosis, diagnosability, and the twin plant-based algorithm. We then show how the diagnosability testing relates to model-checking and present several algorithms to test it. Extensions are then proposed to allow a decentralised approach. Finally, the computation of minimal set of observable events is presented.

2. DIAGNOSABILITY

This section presents the formal definitions of diagnosability. The discrete-event systems and the diagnosis of DES are first presented. Then, the diagnosability is introduced and the twin-plant algorithm is presented.

2.1 Diagnosis of discrete-event system

A Discrete-Event System (Cassandras and Lafortune (1999)) is a dynamic system that behaves in a discrete manner. It is basically modeled as an automaton with observable and non observable transitions. We restrict ourselves to finite DES.

Definition 1. (Discrete-Event System).
A *discrete-event system* (DES) is the finite-state machine $\langle Q, E, T, I\rangle$ where

- Q is a set of states,
- E is a set of events,
- $T \subseteq Q \times E \times Q$ is a set of transitions, and
- $I \subseteq Q$ is a set of initial states.

Starting from an initial state $q_0 \in I$, a DES takes a sequence of consecutive transitions $t_i = \langle q_{i-1}, e_i, q_i\rangle$ called a path.

A DES is usually represented in a symbolic, compact form.

Definition 2. (Symbolic Model).
The *symbolic model* of a DES is a tuple $\langle \mathcal{A}, E, \mathcal{R}, \mathcal{I}\rangle$ where

- $\mathcal{A}$ is a set of state variables a taking their value in a finite domain $dom(a)$,

- E is a set of events,
- $\mathcal{R}$ is a set of rules that associates each event with a precondition and a set of effects on the valuation of the state variables, and
- $\mathcal{I}$ is a condition on the valuation of the state variables.

A symbolic model can be translated into an enumerative model as follows: a state $q \in Q$ is a total assignment of the variables of $\mathcal{A}$; the transition $\langle q, e, q' \rangle$ exists if i) q satisfies the precondition of $\mathcal{R}(e)$ and ii) q' corresponds to the application of the effects of $\mathcal{R}(e)$ on q. Finally, the set of initial states is defined as the set of states q that satisfy the precondition $\mathcal{I}$.

The set E of events is partitioned in two subsets E_o of observable events and E_u of unobservable events. The set Q of states is partitioned in two subsets Q_N of nominal states and Q_F of faulty states. [1] A path is said *nominal* if it remains within the subset Q_N and *faulty* otherwise. The fault is permanent which means that $\forall \langle q, e, q' \rangle \in T,\ q \in Q_N \vee q' \in Q_F$.

The system runs through a path $\rho = q_0 \xrightarrow{e_1} \dots \xrightarrow{e_n} q_n$, and generates an observation whenever $e_i \in E_o$. A commun hypothesis is that the system satisfies some *fairness property*. Formally, an infinite path must contain an infinite number of specified events. Examples of fairness constraint include but are not limited to: an infinite path generates an infinite number of observations, an infinite path on the system must contain an infinite occurrence of events from each component within the system.

In this article, the diagnosis task is simply a detection problem. From the sequence of observations generated by the system, the *diagnosis* engine must determine what happened in the system, and basically whether the current state q_n is a faulty state.

2.2 Diagnosability

An important feature of a system, known as diagnosability (Sampath et al. (1995)), states that the occurrence of a failure can be identified by a diagnosis engine. The failure is not directly observable, and there is often a delay between its occurrence and the observations that allow a precise diagnosis.

The formal diagnosability definition depends on the fairness constraints on the behaviour, but we give the general informal definition. A system is diagnosable iff for any failure path ρ_F on the system, then the observations of any infinite and fair continuation of ρ_F eventually diverges from the observations of any infinite (possibly unfair, see next paragraph) nominal behaviour. This means that after a certain delay after the occurrence of a failure, the current behaviour will no longer be mistaken for a nominal behaviour.

We restrict ourselves to a unique fault but the results can be easily extended to several fault types.

2.3 Twin plant

We now present the diagnosability algorithm based on the twin plant, as proposed by Jiang et al. (2001).

We consider that the fairness properties are such that for any (possibly not fair) infinite path, for any (finite) prefix of this infinite path, there is a fair infinite path that exhibits the same prefix. This property basically states that it is not possible to reach a deadlock in the DES.

The twin plant is based on the following principle:

Proposition 1. A system is *not* diagnosable iff the DES exhibits a fair faulty infinite path and a nominal infinite path that generates the same observations as the faulty path.

Note that the nominal infinite path does not need to be fair. Indeed, note first that the diagnosability property is only required for faulty paths, since an ambiguous diagnosis is not harmful when no fault occurred. Thus, in the proposition, the faulty path is the one that definitely occurred on the system and it has to be fair. Note that the diagnosis is never performed on the sequence of observations generated by the infinite fair path, because this sequence is unknown; the diagnosis is performed on a prefix (of increasing size) of the infinite sequence. Remember that any prefix of an unfair infinite path is also the prefix of a fair infinite path. Therefore, if an infinite nominal (possibly unfair) path generates the same observations as the infinite fair faulty behaviour, then for any prefix of the infinite faulty behaviour there exists an infinite fair nominal behaviour whose prefix generates the same observations. Basically, the fairness property means that a particular event has to occur eventually, and this event can be postponed as much as we want on the nominal behaviour because the nominal behaviour is not the one that actually happens, but conversely this event cannot be postponed infinitely long on the faulty behaviour.

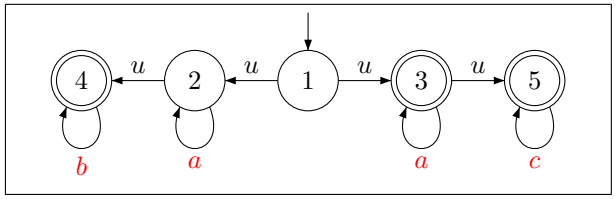

Fig. 1. Importance of the fairness property

This is illustrated Figure 1. In this figure, the fairness is represented by the double circle: an infinite path must infinitely often go through states 3, 4 or 5. The observable events are a, b, and c, and the initial state is 1. Consider the faulty states are 2 and 4. Then a faulty path has to eventually reach state 4, at which stage the system generates bs and the fault can be diagnosed. One the other hand, if the faulty states are 3 and 5, then the faulty path can remain indefinitely in state 3 and generate a sequence of as; this sequence can be mistaken for a non-faulty path remaining in state 2.

This definition of diagnosability allows the twin plant-based algorithm proposed by Jiang et al. (2001). The twin plant (TP) is simply the synchronisation of the DES on a copy of the DES. The transitions are synchronised on the observable events so that any path on the twin plant corresponds to a pair of paths on the DES that generate the same observations. To ensure that a failure occurred only on the copy of the DES, the states $Q_F \times Q$ are pruned from the twin plant. It is then sufficient to find a loop on the twin plant from an ambiguous state $q_a \in Q_N \times Q_F$ reachable from a state of $I \times I$, which loop satisfies the fairness constraint on the copy of the DES.

Definition 3. (Twin Plant).
The *twin plant* of the DES $\langle Q, E, T, I \rangle$ is the finite-state machine $\langle \mathcal{Q}, \mathcal{E}, \mathcal{T}, \mathcal{I} \rangle$ defined by:

- $\mathcal{Q} = Q_N \times Q$,

[1] The literature usually defines the fault on the events, but for permanent faults, it can be equivalently defined on the states.

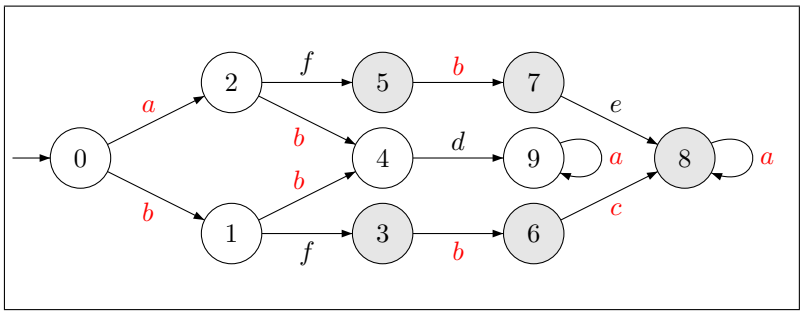

Fig. 2. Example of a DES (a, b, and c observable; 3, 5, 6, 7, and 8 faulty)

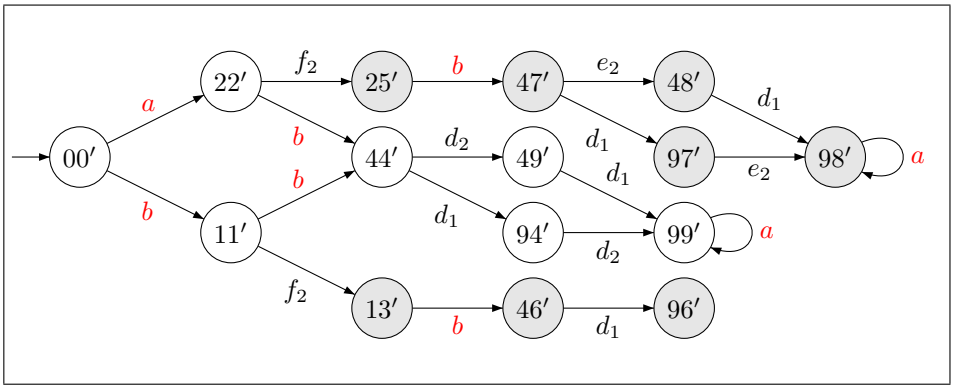

Fig. 3. Twin Plant for the DES of Figure 2

- $\mathcal{E} = E_o \cup ((E \setminus E_o) \times \{1, 2\})$
- $\mathcal{T} = \{\langle q, e, q' \rangle \mid q = \langle q_1, q_2 \rangle \wedge q' = \langle q'_1, q'_2 \rangle \wedge \big((e \in E_o \wedge \langle q_1, e, q'_1 \rangle \in T \wedge \langle q_2, e, q'_2 \rangle \in T) \vee (e \in (E \times \{i\}) \wedge \langle q_i, e, q'_i \rangle \in T \wedge q_j = q'_j \wedge j \neq i)\big)$,
- $\mathcal{I} = I \times I$.

The part of the twin plant that cannot be accessed from the initial state are pruned.

The twin plant algorithm is illustrated with the DES Figure 2 inspired from Briones et al. (2008). The observable events are a, b, and c. The faulty states are 3, 5, 6, 7, and 8. The twin plant is presented Figure 3 where the copy of the DES is represented with a prime. The state $98'$ is ambiguous, and the loop on the state generates an observation (which satisfies the fairness contraint). Therefore, the system is not diagnosable.

3. SYMBOLIC TESTING OF DIAGNOSABILITY

In this section, we propose a new algorithm for the diagnosability of discrete-event system. We discuss the implementation with symbolic tools at the end of the section.

3.1 Diagnosability as a Model-Checking Problem

Model-checking (Clarke et al. (2000)) is an automatic technique for verifying dynamic plants modeled as DES such as communication protocols. Specifications are expressed in temporal logic. Efficient procedures exist to determine whether or not the DES satisfies the property. The use of model-checking to test diagnosability was already proposed (Cimatti et al. (2003)) but with a simplified definition of diagnosability. In particular Cimatti and coauthors give a different definition of ambiguous behaviours, and the delay to disambiguate these behaviours is restricted to one.

The definition of non-diagnosability given in the beginning of the subsection 2.3 can be expressed as a model-checking problem.

$$TP \models E(fair \wedge F(Q_N \times Q_F)) \tag{1}$$

where E is the traditional logic operator that stands for *there exists an infinite path*, $fair$ is the formula that expresses that the path on the copy of the DES is fair, F is the traditional logic operator *eventually*, and $Q_N \times Q_F$ means that the diagnosis state is ambiguous. This definition states that a system is not diagnosable if there is a fair infinite path on the twin plant that reaches an ambiguous state (and thus stays in this ambiguous state). To our best knowledge, it is the first time diagnosability of DES is expressed with (1).

Diagnosability can thus be expressed in LTL (*linear-time logic*). The choice of LTL logic allows to express more general fairness properties than what is usually considered. The traditional fairness property states that an infinite trajectory should generate an infinite number of observations which can be expressed by $fair = GF\,O$. This formula means *at any point in the path, it is always true that O will eventually be reached*, where O models the set of states in which an observation just occurred (a Boolean state variable can represent this property). It is also possible to represent more elaborate properties as *any request is eventually fulfilled* formally expressed by $fair = G\,(ask \rightarrow F\ answer)$.

Any LTL model-checker can solve a diagnosability problem, but we propose to present the general algorithm to study domain-dependant features (as decentralised computation).

3.2 Forward Search

We are looking for an infinite path on the twin plant that reaches an ambiguous state. The path will have the pattern

$$\langle q_0, q'_0\rangle \rightarrow \ldots \rightarrow \langle q_k, q'_k\rangle \rightarrow \langle q_{k+1}, q'_{k+1}\rangle \rightarrow \ldots$$

where $q_i \in Q_N$ for any i and $q'_i \in Q_N$ if $i \in \{0, \ldots, k\}$ (k possibly equals 0), and $q'_i \in Q_F$ if $i > k$. The idea is then to compute all the pairs of states $\langle q_k, q'_k\rangle \subseteq Q_N \times Q_N$ reachable from $I \times I$ for increasing values of k, next compute all the $\langle q_{k+j}, q'_{k+j}\rangle \subseteq Q_N \times Q_F$ reachable in j transitions or more [2] for increasing values of j until a fix point is found.

This procedure is illustrated Algorithm 1 where

- $\mathcal{T}$ represents the set of transitions on the twin plant,
- $Next_{\mathcal{T}}(X)$ is the set of states reached from X by one transition of $\mathcal{T}$ (formally $\{q' \mid \exists q \in X : \langle q, q'\rangle \in \mathcal{T}\}$),
- $\mathcal{W}_k$ represents the set of pairs $\langle q, q'\rangle$ ($q' \in Q_N$) reachable in k or less transitions,
- $\mathcal{R}_i$ represents the set of pairs $\langle q, q'\rangle$ ($q' \in Q_F$) reachable in $(i+1)$ or less transitions from a pair of states of $\mathcal{W}_\infty$, and
- $\mathcal{S}_j$ represents the set of pairs reachable from $(j+1)$ or more transitions from a pair of states of $\mathcal{W}_\infty$.

The algorithm uses classical fix point techniques.

Algorithm 1 Forward algorithm to test diagnosability

```
 1: k := 0
 2: W_k := I × I
 3: repeat
 4:     k++
 5:     W_k := W_{k-1} ∪ ((Q_N × Q_N) ∩ Next_T(W_{k-1}))
 6: until W_k = W_{k-1}
 7: i := 0
 8: R_i := (Q_N × Q_F) ∩ Next_T(W_k)
 9: repeat
10:     i++
11:     R_i := R_{i-1} ∪ Next_T(R_{i-1})
12: until R_i = R_{i-1}
13: j := 0
14: S_j := R_i
15: repeat
16:     j++
17:     S_j := Next_T(S_{j-1})
18: until S_j = S_{j-1}
19: if S_j = ∅ then
20:     return diagnosable
21: else
22:     return not diagnosable
```

We illustrate the algorithm on the example of Figure 2. The evolution of the variables in the algorithm are presented Table 1.

This is the first algorithm that allows to check diagnosability in a symbolic manner rather than using an enumerate approach. For instance, the sets of states Table 1 are enumerated, but they could be represented in a symbolic manner as done to some extend Table 2. Symbolic techniques allow one to manipulate larger sets of states than the enumerate ones (on this topic, see discussion in the subsection 3.4). It is easy to retrieve a critical path from the sets $\mathcal{R}_i$, $\mathcal{S}_j$ and $\mathcal{W}_k$ by picking a state of S_j and building the trajectory backwardly.

This algorithm did not consider fairness properties, but can be easily extended for this case. We illustrate this feature for the backward search.

[2] If one does not consider j transitions *or more*, the set of pairs may not reach a fix point but oscillate (for instance oscillate between the set of states $\{\langle q_1, q'_1\rangle\}$ and $\{\langle q_2, q'_2\rangle\}$).

$\mathcal{W}_0$	$\{00'\}$
$\mathcal{W}_1$	$\{00', 11', 22'\}$
$\mathcal{W}_2$	$\{00', 11', 22', 44'\}$
$\mathcal{W}_3$	$\{00', 11', 22', 44', 49', 94'\}$
$\mathcal{W}_4$	$\{00', 11', 22', 44', 49', 94', 99'\}$
$\mathcal{W}_5$	$\{00', 11', 22', 44', 49', 94', 99'\}$
$\mathcal{R}_0$	$\{25', 13'\}$
$\mathcal{R}_1$	$\{25', 13', 47', 46'\}$
$\mathcal{R}_2$	$\{25', 13', 47', 46', 48', 97', 96'\}$
$\mathcal{R}_3$	$\{25', 13', 47', 46', 48', 97', 96', 98'\}$
$\mathcal{R}_4$	$\{25', 13', 47', 46', 48', 97', 96', 98'\}$
$\mathcal{S}_0$	$\{25', 13', 47', 46', 48', 97', 96', 98'\}$
$\mathcal{S}_1$	$\{47', 46', 48', 97', 96', 98'\}$
$\mathcal{S}_2$	$\{48', 97', 96', 98'\}$
$\mathcal{S}_3$	$\{98'\}$
$\mathcal{S}_4$	$\{98'\}$

Table 1. Unfolding Algorithm 1 (forward search)

3.3 Backward Search

In the backward search, the existence of the critical paths is tested starting from the end. This approach is based on the following idea that motivated this work: in some systems, the behaviour after the occurrence of a failure is much different from the nominal behaviour. First looking at the end of the paths may quickly show that nominal and faulty behaviours can be easily distinguished from each other, and therefore save a lot of computation. For instance, if the system generates alarms if and only if a failure occurs, it should be easy to determine that a failure behaviour cannot be mistaken for a nominal behaviour. In comparison, a forward search on the twin plant requires to explore at least all the nominal behaviour space: $\mathcal{W}_\infty$ as defined in Algorithm 1 contains at least $\langle q, q\rangle$ for any $q \in Q_N$.

The backward search is illustrated Algorithm 2 for fairness property $fair = (GF\, X_1) \wedge \cdots \wedge (GF\, X_x)$ where X_i represents a set of pairs. This fairness property simply indicates that the infinite path should go infinitely often through the set of states X_i for all i. In this algorithm, $Prev_T(S)$ refers to the set of states q such that $\exists q' \in S : \langle q, q'\rangle \in T$. The algorithm first determines the set $\mathcal{W}_p$ of ambiguous pairs from which there is an infinite path that satisfies the fairness properties X_1 to X_p. To do so, starting from the set of pairs $\mathcal{W}_{p-1}$, we repeatedly (lines 5 to 14) force the set into X_p and then allow an unlimited (but non null) number of transitions (lines 8 to 11) until a fix point is reached. Each fairness property can be tested separately but the algorithm works only if $x \neq 0$. [3] Finally, the procedure goes backward (lines 16 to 19) until an initial pair is found (the existence of a critical path is proved: non diagnosable) or a fix point is reached (diagnosable).

The application of this algorithm is presented Table 2. Recall that the fairness constraint indicates that an infinite trace should generate an infinite sequence of observations, which means that $x = 1$ and that S_0^i is computed by taking an observable transition from R_{i-1}. For instance, $\mathcal{W}_0$ is defined as the Cartesian product of Q_N and Q_F. $\mathcal{R}_0 = \mathcal{W}_0$. $\mathcal{S}_0^0$ is defined as the set of predecessors of $\mathcal{R}_0$ through an observable event. There are three observable events: the synchronisation of a leads to $(\{0, 9\} \times \{8'\})$, the synchronisation of b leads to $(\{0, 1, 2\} \times \{3', 5'\})$, while c leads to $\emptyset$. The next step computes the union of the previous step and the set of predecessors of these states. Finally, $\mathcal{S}$ reaches a fix-point leading to $\mathcal{R}_1$. When $\mathcal{R}$ reaches

[3] It is possible to force $X_1 = Q \times Q$.

Algorithm 2 Backward algorithm to test diagnosability (under fairness properties)

1: $\mathcal{W}_0 = Q_N \times Q_F$
2: **for** ($p := 1$; $p \leq x$; p++) **do**
3: $i := 0$
4: $R_i := \mathcal{W}_{p-1}$
5: **repeat**
6: $j := 0$
7: $S_j := Prev_{\mathcal{T}}(R_i \cap X_p)$
8: **repeat**
9: j++
10: $S_j := S_{j-1} \cup Prev_{\mathcal{T}}(S_{j-1})$
11: **until** $S_j = S_{j-1}$
12: i++
13: $R_i := S_j$
14: **until** $R_i = R_{i-1}$
15: $\mathcal{W}_p := R_i$
16: **if** $\mathcal{W}_x \cap (I \times I) \neq \emptyset$ **then**
17: **return non diagnosable**
18: **else**
19: **return diagnosable**

a fix-point, it intersects $I \times I$ which means the system is not diagnosable.

$\mathcal{W}_0$	$\{0,1,2,4,9\} \times \{3',5',6',7',8'\}$
$\mathcal{R}_0$	$\{0,1,2,4,9\} \times \{3',5',6',7',8'\}$
$\mathcal{S}_0^0$	$(\{0,9\} \times \{8'\}) \cup (\{0,1,2\} \times \{3',5'\})$
$\mathcal{S}_1^0$	$(\{0,4,9\} \times \{8'\}) \cup (\{0,9\} \times \{7'\}) \cup$ $(\{0,1,2\} \times \{1',2',3',5'\})$
$\mathcal{S}_2^0$	$(\{0,4,9\} \times \{7',8'\}) \cup (\{0,1,2\} \times \{1',2',3',5'\}) \cup \{00'\}$
$\mathcal{S}_3^0$	$(\{0,4,9\} \times \{7',8'\}) \cup (\{0,1,2\} \times \{1',2',3',5'\}) \cup \{00'\}$
$\mathcal{R}_1$	$(\{0,4,9\} \times \{7',8'\}) \cup (\{0,1,2\} \times \{1',2',3',5'\}) \cup \{00'\}$
$\mathcal{S}_0^1$	$\{98',00',15',25'\}$
$\mathcal{S}_1^1$	$\{98',00',15',25',48',97',12',22'\}$
$\mathcal{S}_2^1$	$\{98',00',15',25',48',97',12',22',47'\}$
$\mathcal{S}_3^1$	$\{98',00',15',25',48',97',12',22',47'\}$
$\mathcal{R}_2$	$\{98',00',15',25',48',97',12',22',47'\}$
$\mathcal{S}_0^2$	$\{98',00',15',25'\}$
$\mathcal{S}_1^2$	$\{98',00',15',25',48',97',12',22'\}$
$\mathcal{S}_2^2$	$\{98',00',15',25',48',97',12',22',47'\}$
$\mathcal{S}_3^2$	$\{98',00',15',25',48',97',12',22',47'\}$
$\mathcal{R}_3$	$\{98',00',15',25',48',97',12',22',47'\}$
$\mathcal{W}_1$	$\{98',00',15',25',48',97',12',22',47'\}$

Table 2. Unfolding Algorithm 2 (backward search)

This is the first backward algorithm for testing diagnosability. When diagnosable, the forward algorithm explores at least all the nominal search space. In case the behaviour of the system is much different after a failure, we expect the backward algorithm to explore a smaller search space.

Note that the double **repeat-until** loop can be avoided in the algorithm by *compiling* the model. Basically, a new transition system is defined on $Q_N \times Q_F$ such that a transition links $s_1 = \langle q_1, q_1' \rangle$ to $s_2 = \langle q_2, q_2' \rangle$ iff $s_2 \in X_p$ and there is a non empty path on $\mathcal{T}$ between s_1 and s_2. This new set of transitions denoted $\mathcal{T}_p$ is computed by the fix-point Algorithm 3. Now R_i can be simply computed by $R_i = Prev_{\mathcal{T}_p}{}^{\infty}(R_{i-1})$.

3.4 Complexity

The algorithms presented in this section do not explicitly refer to a symbolic implementation. However, the set of pairs in these algorithms should be implemented with symbolic tools. For instance, enumerating all the states of $\mathcal{W}_0 = Q_N \times Q_F$ in Algorithm 2 is already quadratic in the number of states in the DES; this was avoided in Table 2 where the set of states was represented by Cartesian product. Note that an incomplete algorithm using symbolic techniques (SAT, Rintanen and Grastien (2007)) was proposed that proves non diagnosability, but cannot prove diagnosability.

Algorithm 3 Compiling $\mathcal{T}_p$

1: $j := 0$
2: $\mathcal{T}_p^j := \mathcal{T} \cap (Q_N \times (Q_F \cap X_p))$
3: **repeat**
4: j++
5: $\mathcal{T}_p^j := \mathcal{T}_p^{j-1} \cup \{\langle s_1, s_2' \rangle \mid \exists s' : \langle s_1, s' \rangle \in \mathcal{T} \wedge \langle s', s_2' \rangle \in \mathcal{T}_p^{j-1}\}$
6: **until** $\mathcal{T}_p^j = \mathcal{T}_p^{j-1}$
7: **return** $\mathcal{T}_p^j$

As explained before, the symbolic approach usually represents sets of states and transitions by propositional formulas, [4] and operations as union, intersection, emptiness-checking, set equality, projection can be expressed as operations or transformations on propositional formulas, such as disjunction, conjunction, satisfiability, equivalence, forgetting. With this respect, any language completely representing propositional formula can be used as long as these operations are efficient and the representation succinct. A lot of work is currently being done to identify these languages. The most popular language is OBDD$_<$ (Bryant (1986)), but a lot of attention is captured by other sub-families of DNNF, as d-DNNF (Huang and Darwiche (2007)), (d-)SDNNF (Pipatsrisawat and Darwiche (2008)).

There is no guarantee that this approach will be more efficient than an enumerative approach. In particular, the length of a critical path may be quadratic in the number of states in the DES (Rintanen (2007)), which means that the fix-point represented by the **until**s in the algorithms may require a quadratic number of loops. Furthermore, the compactness of a symbolic representation of the set of states does not hold in general: some propositional formulas require a representation whose size is exponential in the number of state variables (Darwiche and Marquis (2002)) (but admittedly, the number of states represented is then exponential as well).

Finally, note that this approach is convenient to extract one critical path but not to generate all critical paths.

4. DECENTRALISED COMPUTATIONS

The decentralised approach for diagnosability was proposed to test diagnosability on large systems. It can be implemented to some extend in our proposed approach by using relaxation of constraints. We first present the general idea of the decentralised approach. Then we show how relaxations can be implemented. Finally, we show the implementation of the decentralised approach.

4.1 Background

Decentralised testing of diagnosability was proposed in (Pencolé (2004)). The idea is that if a failure occurs on a component, the observations that allow one to identify the failure are probably those emitted by this component or its close neighbours. It is sufficient to prove diagnosability locally to ensure global

[4] Although this is not the only approach.

diagnosability; moreover, the cost of the checking is very little compared to the probability to be successful. Similar techniques were applied in model-checking (Balarin and Sangiovanni-Vincentelli (1993)). Furthermore, if the subsystem is not locally diagnosable, it is still possible to prune the search space locally before searching for a larger subsystem.

The principle of this approach is the following. First, the diagnosability is checked locally on the component where the fault can occur. The twin plant is built from the current subsystem model. An analysis is performed on this structure to find critical paths. If such a critical path is found, the diagnosability cannot be proved; an additionnal component is added to the original one and the twin plant is efficiently updated (rather than computed from the subsystem model) from the previous twin plant. Because of fairness properties, it is possible to remove from the twin plant the parts that are diagnosable, thus efficiently pruning the search.

Note that this is different from the modular diagnosability proposed e.g. in (Debouk et al. (2002); Contant et al. (2006)). In the latter case, the modular diagnosability is used to determine whether a system is diagnosable under a certain architecture. In the former case, the goal is to speed up the diagnosability process. Furthermore, the cited article do not rely on the twin plant.

4.2 How to Relax the Constraints

The non-diagnosability testing consists in finding a particular path on the space $Q_N \times Q$. This particular path has to satisfy some constraints that are listed next. Removing some constraint can make the problem simpler while still returning the good result.

The constraints on the critical path are the following: the path has to

(1) satisfy all the fairness constraints,
(2) be consistent with the model,
(3) generate identical observations on both the system and its copy.

If some of these constraints are relaxed and still no critical path is found, then no path will be found if taking all constraints into account, and the system is therefore diagnosable.

Removing some fairness constraints can be very efficient as it removes some loops in Algorithm 2. Removing some constraint on the model allow one to remove some state variables, and therefore reduce the size of the symbolic representations of states [5] . In this section, we concentrate on these two relaxations.

In case the diagnosability cannot be proved, it should still be possible to reuse the result to simplify the computation at the next iteration. The aim is thus to find a restriction on the property that i) is easier to solve than the original property, and ii) allows to reuse what was computed for an incremental approach.

4.3 Implementation in a Symbolic Framework

The set of state variables $\mathcal{A}$ is partitioned into subsets $\mathcal{A}_1, \ldots, \mathcal{A}_n$. The set $\mathcal{A}_i$ implicitly defines the component $\mathcal{C}_i$ with set of states $Q_i = \Pi_{v \in \mathcal{A}_i} dom(v)$. To simplify, we consider $Q_F = Q_{F1} \times Q_2 \times \cdots \times Q_n$ where $Q_{F1} \subseteq Q_1$; which basically means the fault is defined on component $\mathcal{C}_1$. The set E_i of events of component $\mathcal{C}_i$ is the set of events whose precondition depends on or whose effect affects some variable $a \in \mathcal{A}_i$.

Given a subset of j components $Sub = \{\mathcal{C}_1, \ldots, \mathcal{C}_j\}$, it is possible to run Algorithm 2 on the model defined by the projection of the global model on this set of components, with the fairness constraints projected on this model. The algorithm computes a set of states $\mathcal{W}_x$ that corresponds to the set of states from where there is a critical path on the subsystem model. If this set of states does not intersect the set of initial states of the twin plant, then the subsystem is diagnosable, and the system is therefore diagnosable. Otherwise, nothing can be concluded.

The set $\mathcal{W}_x$ represents a set of states on the subsystem Sub, but it also represents the set of states $\mathcal{W}_x \times Q_{j+1}$ on $Sub \cup \{\mathcal{C}_{j+1}\}$. Since the restrictions satisfied when testing diagnosability for Sub must also be satisfied when testing diagnosability for $Sub \cup \{\mathcal{C}_{j+1}\}$, it is possible to start Algorithm 2 with $\mathcal{W}_x \times Q_{j+1}$ rather than $\mathcal{W}_0$ which reduces the number of iteration before reaching the fix-point.

Note that, as for Pencolé (2004), the order in which the components are added to the subset will have a big impact on the efficiency of the algorithm.

5. OPTIMAL OBSERVABILITY

Diagnosability is an important feature to check when designing a system that needs supervision. This property can usually be enforced by adding sensors on the system. Adding new sensors has a significant cost, from a design point of view, from a production point of view and also from a maintenance point of view. Thus, an important design capability is to minimize the number of sensors as to minimize the total cost. We propose a conflict-based algorithm using a symbolic approach to solve this problem. The generation of minimal sets of sensors has been studied by Briones et al. (2008). Their approach is based on *signatures*, *i.e.* sequences of observations generated by a fault. The authors propose two incremental algorithms: their first algorithm removes sensors until it is no longer possible to remove any sensor; their second algorithm adds sensors until the system is diagnosable and then applies the first algorithm.

Let $S^\star = \langle \sigma_1, \ldots, \sigma_k \rangle$ be a set of *sensors*. The sensor σ_i, if added to the system, makes the events $E_{\sigma_i} \subseteq E$ observable. The *set of potentially observable events* is denoted $E_o^\star$ and defined by $E_o^\star = \bigcup_{\sigma \in S^\star} E_\sigma$. Note that different sensors can monitor the same observable events ($\sigma \neq \sigma' \not\Rightarrow E_\sigma \cap E_{\sigma'} = \emptyset$); in other words, the sensors of $S^\star$ cover $E_o^\star$ but do not form a partition of $E_o^\star$. The set of observable events monitored by a set of sensors $S \subseteq S^\star$ is denoted E_S and defined by $\bigcup_{\sigma \in S} E_\sigma$. To simplify, with each sensor σ is attached a *cost* denoted $cost(\sigma) > 0$; the cost of a set of sensors $S \subseteq S^\star$ is the sum of the cost of each sensor: $cost(S) = \Sigma_{\sigma \in S} cost(\sigma)$.

The problem is to compute a set $S \subseteq S^\star$ that ensures $\langle Q, E, T, I \rangle$ is diagnosable if $E_o = E_S$ and minimises the cost $cost(S)$. We consider that the fairness properties attached with the definition of diagnosability do not rely on E_S (but potentially on $E_o^\star$).

A general approach based on the classical *conflict-hitting set* method is provided Algorithm 4. $\mathcal{S}$ is a set of candidate sets

[5] Note that it is not always the case in practice.

Algorithm 4 Computing optimal set of sensors

1: $\mathcal{S} := \{\emptyset\}$
2: **while** No solution found and $\mathcal{S} \neq \emptyset$ **do**
3: $S := \text{argmin}_{S \in \mathcal{S}}(cost(S))$
4: **if** DES is diagnosable under E_S **then**
5: **return** S
6: **pick** a conflict $E^{con} \subseteq E_o^\star \setminus E_S$
7: $S^{con} := \{\sigma \in S^\star \mid E_\sigma \cap E^{con} \neq \emptyset\}$
8: $\mathcal{S} := \{S'' \mid \exists \sigma \in S^{con} : \exists S' \in \mathcal{S} : S'' = S' \cup \{\sigma\}\}$
9: $\mathcal{S} := \mathcal{S} \setminus \{S'' \in \mathcal{S} \mid \exists S' \in \mathcal{S} : S' \subset S''\}$
10: **return** non diagnosable

of sensors. At any time in the search, $\mathcal{S}$ contains candidate sets of sensors; if a set S of sensors makes the system diagnosable, then by construction $\exists S' \subseteq S : s' \in \mathcal{S}$. The diagnosability is tested with the element S of $\mathcal{S}$ with smallest $cost(S)$, as it is the best current candidate. If S does not make the system diagnosable, then a critical path ρ can be extracted from the diagnosability checking. The conflict E^{con} generated from this critical path ρ is the set of events of $E_o^\star \setminus E_S$ that occur from this critical path; it is possible to tighten this conflict by incrementally removing any $e \in E^{con}$ such that making e observable does not make the critical path diagnosable. The conflict set E^{con} means that the pair of ambiguous paths ρ can be distinguished only if at least one event of E^{con} is observable. Thus, we update $\mathcal{S}$ by ensuring that each candidate set of $\mathcal{S}$ contains a sensor monitoring an element of E^{con} (line 8). Any strict superset S'' of a $S' \in \mathcal{S}$ can be removed from $\mathcal{S}$ to limit its size. Finally, if a set of sensors $S \subseteq S^\star$ is found with an empty conflict, it means that a critical path was found on the system, which path satisfies the fairness properties and cannot be distinguished even when all sensors of $S^\star$ are set on the system. After an empty conflict is found, $\mathcal{S}$ is automatically set to $\emptyset$ line 8 and the system is proved non diagnosable. Note that this algorithm can be trivially extended in order to enumerate all minimal sets.

The general method presented in Algorithm 4 can be used with a symbolic approach. To improve the efficiency, specific improvements can be implemented.

The first improvement concerns reusing previously computed results. The previous section showed that a way to relax the property expected on all paths when testing diagnosability is not to enforce the simultaneous occurrence of some observable event or, in other words, to consider that some observable event is not observable. We have shown that it is possible to initialise $\mathcal{W}_0$ to the conjunction of the $\mathcal{W}_x$s computed for each restriction of the current restriction. It is possible to apply it here as well. For instance, consider that the system is not diagnosable if the system contains the sensors $\{\sigma_1, \sigma_2\}$ or $\{\sigma_1, \sigma_3\}$. We now want to test whether the system will be diagnosable with the set of sensors $\{\sigma_1, \sigma_2, \sigma_3\}$. Then, it is possible to initialize $\mathcal{W}_0$ to $\mathcal{W}' \cap \mathcal{W}''$ where $\mathcal{W}'$ is the $\mathcal{W}_x$ computed for $\{\sigma_1, \sigma_2\}$, and $\mathcal{W}''$ for $\{\sigma_1, \sigma_3\}$.

A second improvement regards the extraction of the conflict. Smaller conflicts make the algorithm converge faster, as they reduce the number of elements in $\mathcal{S}$. The extraction of the critical path simply runs the sets of states computed backward and pick one transition at each stage. To increase the probability of a small conflict, when the transition is chosen we suggest to avoid if possible any event from $E_o^\star \setminus E_S$ that was not already chosen in the prefix of the path.

6. CONCLUSION

A system is diagnosable if it contains enough sensors for an observer to diagnose a failure. In this paper, we presented a symbolic-based approach to efficiently test whether a discrete-event system is diagnosable. Several variations are proposed with different advantages. Using symbolic techniques allows one to consider large discrete-event systems that existing enumerative approaches fail to test. The search can be performed in a classical forward manner, or in a backward manner which potentially avoids exploring all the search space of the discrete-event system. This approach can also be mixed with a decentralised computation, which allows early detection of diagnosability and reduction of the search space in general. The main drawback of this approach is that it does not return all the critical non-diagnosable paths. We also studied the problem of finding a minimal set of sensors that makes the system diagnosable The general procedure only requires a diagnosability engine at its core, and we proposed to use our engine. We also improved our algorithm for this purpose.

As explained at the end of section 3, some questions remain unanswered. In particular, it is not clear which symbolic tool should be used to represent the set of states (BDD, DNNF, etc.). This question is the topic of intensive search in all path-finding communities (planning, model-checking). We also want to check to which extend decomposition (Schumann and Pencolé (2007)) can be merged with this approach.

7. ACKNOWLEDGEMENT

This research was supported by NICTA. NICTA is funded through the Australian Government's Backing Australia's Ability initiative, in part through the Australian National Research Council.

The author wants to thank Perrine and Zacharie for their useful comments.

REFERENCES

F. Balarin and A. Sangiovanni-Vincentelli. An iterative approach to language containment. In *Fifth International Conference on Computer-Aided Verification*, 1993.

L. Briones, A. Lazovik, and Ph. Dague. Optimal observability for diagnosability. In *Nineteenth International Workshop on Principles of Diagnosis (DX-08)*, 2008.

R. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, 35(8):677–691, 1986.

C. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Kluwer Academic Publishers, 1999.

A. Cimatti, Ch. Pecheur, and R. Cavada. Formal verification of diagnosability with via symbolic model-checking. In Georg Gottlob, editor, *Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03)*, pages 363–369. Morgan Kaufmann Publishers, 2003.

E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, Cambridge, MA, 2000.

L. Console, Cl. Picardi, and M. Ribaudo. Diagnosis and diagnosability analysis using PEPA. In *Fourteenth European Conference on Artificial Intelligence (ECAI-00)*, pages 131–135, 2000.

O. Contant, St. Lafortune, and D. Teneketzis. Diagnosability of discrete event systems with modular structure. *Discrete Event Dynamic Systems*, 16:9–37, 2006.

M.-O. Cordier, Y. Pencolé, L. Travé-Massuyès, and Th. Vidal. Self-healablity = diagnosability + repairability. In *Eighteenth International Workshop on Principles of Diagnosis (DX-07)*, pages 251–258, 2007.

A. Darwiche and P. Marquis. A knowledge compilation map. *Artificial Intelligence (AIJ)*, 17:229–264, 2002.

R. Debouk, R. Malik, and B. Brandin. A modular architecture for diagnosis of discrete event systems. In *41st IEEE Conference on Decision and Control*, volume 1, pages 417–422, 2002.

A. Grastien, Anbulagan, J. Rintanen, and E. Kelareva. Diagnosis of discrete-event systems using satisfiability algorithms. In R. Holte, editor, *Twenty-Second AAAI Conference on Artificial Intelligence*, pages 305–310. AAAI Press, 2007.

J. Huang and A. Darwiche. The language of search. *Journal of Artificial Intelligence Research (JAIR)*, 29:191–219, 2007.

Sh Jiang, Zh. Huang, V. Chandra, and R. Kumar. A polynomial algorithm for diagnosability of discrete-event systems. *IEEE Transactions on Automatic Control*, 46:1318–1321, 2001.

H. Kautz and B. Selman. Pushing the envelope: planning, propositional logic, and stochastic search. In *13th National Conference on Artificial Intelligence*, pages 1194–1201. AAAI Press, 1996.

K. McMillan. *Symbolic Model Checking*. Kluwer Academic Publishers, 1993.

Y. Pencolé. Diagnosability analysis of distributed discrete event systems. In *European Conference on Artificial Intelligence (ECAI'04)*, pages 173–178, 2004.

K. Pipatsrisawat and A. Darwiche. New compilation languages based on structured decomposability. In *Twenty-Third AAAI Conference on Artificial Intelligence*, pages 517–522, 2008.

J. Rintanen. Diagnosers and diagnosability of succinct transition systems. In *Proc. 20th International Joint Conference on Artificial Intelligence (IJCAI-07)*, pages 538–544, 2007.

J. Rintanen and A. Grastien. Diagnosability testing with satisfiability algorithms. In M. Veloso, editor, *Twentieth International Joint Conference on Artificial Intelligence (IJCAI-07)*, pages 532–537. AAAI Press, 2007.

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

A. Schumann and Y. Pencolé. Scalable diagnosability checking of event-driven systems. In M. Veloso, editor, *Twentieth International Joint Conference on Artificial Intelligence (IJCAI-07)*, pages 575–580. AAAI Press, 2007.

A. Schumann, Y. Pencolé, and Sylvie Thiébaux. A spectrum of symbolic on-line diagnosis approaches. In R. Holte, editor, *Twenty-Second AAAI Conference on Artificial Intelligence (AAAI-07)*. AAAI Press, 2007.