Incremental Diagnosis of DES with a Non-Exhaustive Diagnosis Engine

Alban Grastien * **Anbulagan** *

* *NICTA and The Australian National University (ANU), Canberra, Australia*
{*alban.grastien*|*anbulagan*}*@nicta.com.au*

Abstract: The incremental diagnosis of discrete-event systems is the problem of updating the diagnosis when new observations are available. For solving this problem, most approaches use diagnosis engines that return *all* the diagnoses, and concatenate the diagnoses of the new observations to the previous diagnoses. In this paper, we define the notion of *non-exhaustive* diagnosis engine, which returns only the *preferred* diagnosis, and we show how such an engine can be used for the incremental diagnosis. This is done by the mean of the so-called *prediction windows* that specify a delay that is sufficient to return a correct and accurate diagnosis.

Keywords: Diagnosis, Discrete-Event Systems, Incremental Diagnosis

1. INTRODUCTION

Many man-made systems require proper supervision to detect the occurrence of failures, in order to take appropriate actions and to recover to an acceptable state. This task, known as *diagnosis*, can be performed based on *expert knowledge*, but the most robust techniques use *model-based reasoning*. In this context, discrete-event systems often offer a satisfying level of abstraction, *i.e.* a good trade-off between complexity and accuracy. Model-based diagnosis of discrete-event systems attracted a lot of research effort over the last two decades (Sampath et al. [1995], Lamperti and Zanella [2003], Fabre et al. [2005] to cite a few).

The model of the system contains information that is relevant for diagnosis but that does not directly interest the agent in charge of the supervision. For instance, the diagnosis engine, also called *diagnoser* in this paper, may have to monitor the value of a particular state variable to determine the occurrence of a failure, while the agent is interested only by all possible occurrences of failures and definitely not by the evolution of this specific variable assignment over the time. A diagnoser must be able to disregard these time- and memory-consuming details and only consider the relevant high-level information.

Furthermore in some contexts, several diagnoses are consistent with the observations, but not all these diagnoses are required. A preference criterion exists on these diagnoses, based e.g. on the probability of the diagnosis or the level of gravity of the diagnosis. In this context, any decision based on the diagnosis will only consider the preferred diagnosis hypothesis, and an efficient procedure only needs to return this hypothesis. This paper makes two contributions based on these considerations.

The first contribution is the characterisation of *non-exhaustive* diagnosis engine. Such a diagnoser exhibits the following features: the engine is able to determine whether a specified class of behaviours occurred and can exhibit one such behaviour consistent with the observations if existing. A *downgrade* to a non-exhaustive engine can significantly improve the efficiency of a diagnosis algorithm since not all the diagnoses are required.

On-line diagnosis is the problem of diagnosing the system while it is running, by updating the diagnosis regularly. Though required by most real-world systems, on-line diagnosis is not well-studied. Notable exceptions include the work of Pencolé and Cordier [2005], Grastien et al. [2005], Lamperti and Zanella [2007], Torta and Torasso [2007], and Schumann et al. [2007]. A well-known characteristic that makes on-line diagnosis difficult is the temporal uncertainties on the observations: when a delay is required to transmit the observations, it gets hard to determine the list of observations that were emitted at a given date. In this paper, we identify an orthogonal characteristic that makes the on-line diagnosis non trivial for some diagnosers.

Thus the second contribution is the investigation of the on-line diagnosis using a non-exhaustive diagnosis engine. As opposed to exhaustive diagnosers, such a diagnoser needs in general to restart from scratch every time new observations are provided. As the number of observations grows, this procedure will eventually fail to return the diagnosis in reasonable time. We want to bound the complexity of updating the diagnosis in order to allow a theoretically never-ending supervision.

The idea behind this contribution is to assume that the behaviour computed until some date in the past is established, i.e. that it is known for sure. We run the diagnosis starting from the state reached at the end of this established behaviour, and concatenate this new behaviour to the established one. The issue here is to make sure that establishing the behaviour is not harmful to completeness.

A system may indeed generate a behaviour that cannot be precisely and immediately diagnosed by an external observer. As the behaviour goes on, the system generates more observations that help disambiguate the original

behaviour (although additional ambiguities can arise regarding the more recent parts of the behaviour). There is usually a delay after which new observations are no longer useful either because the ambiguity has already been cleared or because it can no longer be cleared. We propose to establish the behaviour that occurred before this delay. Since we consider that no more contradictory information can be received, this approach should be correct, provided that the delay is correctly computed.

The paper is structured as follows. The next section presents the diagnosis of discrete-event systems and non-exhaustive diagnosers. Then, we present how to perform incremental diagnosis and how to determine the prediction window. Experiments are presented to validate this approach. Finally, we discuss related works and possible extensions.

2. NON-EXHAUSTIVE DIAGNOSIS

This section describes the discrete-event system (DES), the diagnosis of DES, and non-exhaustive diagnosis engines.

2.1 Diagnosis of Discrete-Event System

A discrete-event system is basically a finite-state machine with partially observable transitions.

Definition 1. (Discrete-Event System).
A *discrete-event system* is a tuple $\mathcal{A} = \langle Q, E_u, E_o, T, I\rangle$ where Q is a finite set of *states*, E_u is a finite set of *unobservable events*, E_o is a finite set of *observable events*, $T \subseteq Q \times (E_u \cup E_o) \times Q$ is a finite set of *transitions*, and $I \subseteq Q$ is a finite set of *initial states*.

A DES represents a set of behaviours modeled as paths on the DES. A path on the DES $\mathcal{A}$ is a sequence of consecutive transitions $\rho = q_0 \xrightarrow{e_1} q_1 \xrightarrow{e_2} \dots \xrightarrow{e_n} q_n$ on the system (*i.e.* $\langle q_{i-1}, e_i, q_i\rangle \in T$ for $i \in \{1, \dots, n\}$). The system takes a path ρ starting from an initial state $q_0 \in I$ and generates a sequence of j observations $obs(\rho) = [o_1, \dots, o_j]$ that corresponds to the elimination of the $n - j$ unobservable events in the sequence $[e_1, \dots, e_n]$. This sequence is observed by the diagnosis engine.[1] The goal of the diagnosis engine is to recover what happened in the system according to these observations.

An agent on a DES is usually concerned with tracking unacceptable faulty behaviours. The definition of such behaviours can be elaborate (Jéron et al. [2006]), but for sake of simplicity we consider that a subset $E_f \subseteq E_u$ of unobservable events are defined as *faulty*. A path is faulty if it contains at least one faulty event. The goal of the diagnosis is then to determine the sequence $[f_1, \dots, f_k]$ of k faulty events that corresponds to the elimination of the non faulty events $(E_o \cup E_u \setminus E_f)$ of the sequence of events $[e_1, \dots, e_n]$. The path that supports a diagnosis is called an *explanation* of the diagnosis. Note that this work is not restricted to this definition of diagnosis.

Example 1. Figure 1 presents a running example for this paper. The states are represented by circles, and the transitions by arrows. The unique initial state is 1. The observable events are a, b, and c, and the unobservable events are u and the faulty event f.

Given the sequence of observations $obs = [b, c, b, a, b]$, the only behaviour consistent with obs is $1 \xrightarrow{b} 2 \xrightarrow{c} 4 \xrightarrow{u} 5 \xrightarrow{b} 6 \xrightarrow{a} 1 \xrightarrow{b} 2$; the diagnosis is the empty sequence of faults ([]). However, if the sequence of observations is $[c, b, c]$, three behaviours are possible: $1 \xrightarrow{c} 3 \xrightarrow{b} 4 \xrightarrow{u} 5 \xrightarrow{c} 7$, $1 \xrightarrow{f} 2 \xrightarrow{c} 4 \xrightarrow{u} 5 \xrightarrow{b} 6 \xrightarrow{c} 8$, or $1 \xrightarrow{f} 2 \xrightarrow{c} 4 \xrightarrow{u} 5 \xrightarrow{b} 6 \xrightarrow{c} 8 \xrightarrow{u} 5$; the two corresponding diagnosis candidates (or diagnoses) are: [] or $[f]$. □

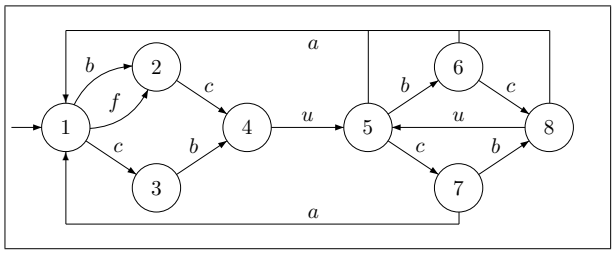

Fig. 1. Example of a discrete-event system.

2.2 Non-Exhaustive Diagnosers

As seen in Example 1, the observations of the behaviour of a system can be *ambiguous*, i.e. it is not possible to retrieve precisely what behaviour generated these observations. In particular, many systems are not *diagnosable* (Sampath et al. [1995]) which means that the sequence of faults that actually occurred on the system cannot always be precisely deduced from the sequence of observations. In this case, several diagnoses can be returned to explain the observations.

What the diagnoser should do when ambiguous diagnoses arise depends on the application: the agent may expect all diagnoses, all minimal diagnoses, the most probable diagnosis, the worst-case diagnosis, etc. For this purpose, we consider a notion of *preference*, such that the expected output of the diagnoser is the preferred diagnosis.

Definition 2. (Diagnosis Preference).
The (diagnosis) *preference* is an ordering relation $\preceq$ on the set of diagnoses such that the diagnosis Δ is preferred to the diagnosis Δ' if $\Delta \preceq \Delta'$.

We extend the notion of preference to paths, in the sense that if the path ρ is preferred to the path ρ' (denoted $\rho \preceq \rho'$), then the diagnosis of ρ is preferred to the diagnosis of ρ'. The preferability notion may include e.g. the probability of a diagnosis.

Consider that the agent in charge of the system only requires the preferred diagnosis, e.g. because the number of possible diagnoses is prohibitive and no decision can be made that satisfies all possible diagnosis. We call *non-exhaustive diagnosis engine* (NEDE) a diagnoser that returns only the preferred diagnosis. To allow the incremental diagnosis, we also require that this engine returns an explanation of the diagnosis.

Definition 3. (Non-Exhaustive Diagnosis Engine).
A *non-exhaustive diagnosis engine* is a procedure that

[1] This sequence may be uncertain (partial order, noise, etc.) – see (Grastien et al. [2005], Lamperti and Zanella [2007]) for instance –, but this issue is rather orthogonal to this article.

returns the preferred diagnosis, along with an explanation of this diagnosis.

Example 2. Through the examples of this paper, the preferred diagnosis is the most probable diagnosis, defined as the minimum cardinality diagnosis, *i.e.* the diagnosis that exhibits the smallest number of faulty events.

In Example 1, for the sequence of observations $[c, b, c]$, the NEDE will only return the empty diagnosis [] with the associated explanation $1 \xrightarrow{c} 3 \xrightarrow{b} 4 \xrightarrow{u} 5 \xrightarrow{c} 7$. □

In many circumstances, a non-exhaustive diagnosis will be sufficient for the agent. Indeed, the agent will only consider the preferred diagnoses. It is possible to turn an exhaustive diagnoser that returns the explanations into a NEDE simply by determining the preferred diagnosis and extracting one of its explanations. However, the restriction proposed in Definition 3 allows more efficient computation as the search space can be pruned as soon as a first diagnosis is computed. The only NEDE we are aware of in the domain of DES is the SAT-based diagnoser we proposed recently (Grastien et al. [2007]). [2] Other existing approaches prefer to ensure completeness. The present article is not restricted to any NEDE in particular.

3. INCREMENTAL DIAGNOSIS

3.1 Definition of Incremental Diagnosis

On-line diagnosis is the problem of supervising a system that is running. It usually includes managing the flow of observations (Pencolé and Cordier [2005], Grastien et al. [2005], Lamperti and Zanella [2007]), which is independent from our problem. Another problem associated with on-line diagnosis is that the number of observations increases regularly, and the diagnosis cannot be computed from scratch every time it is updated. This is the problem of *incremental diagnosis*.

Definition 4. (Incremental Diagnosis).
Given a discrete-event system $\mathcal{A}$, given a sequence of observation $obs_j = [o_1, \ldots, o_j]$, given a continuation $obs_{j'} = [o_1, \ldots, o_{j'}]$ of obs_j (*i.e.* such that $j' \geq j$), the *incremental diagnosis* is the computation of the diagnosis of $obs_{j'}$ using the diagnosis of obs_j.

An important concept in incremental diagnosis is the notion of *consistency* with preference. Basically, if a path ρ_1 is preferable to another path ρ_2, then the concatenation $\rho_1\rho'$ of the two paths ρ_1 and ρ' is usually preferable to the concatenation $\rho_2\rho'$. This strongly suggests that the computations performed for obs_j should be partially reusable for $obs_{j'}$.

The incremental diagnosis must satisfy a complexity requirement. This requirement states that the complexity of updating a diagnosis only depends on the number of additional observations, and not on the total number of observations. Formally, the complexity of the incremental diagnosis must be bounded by some complexity function $f(j' - j)$ and not by $f(j')$.

[2] Although we are aware that planning-based approaches are being developed.

The incremental diagnosis can be defined in a computationally-hard yet simple-to-define manner for the exhaustive diagnosis engines. For any possible state $q \in Q$ of the system reached in the previous diagnosis after observation o_j, new paths starting from q and consistent with the sequence of observations $[o_{j+1}, \ldots, o_{j'}]$ are computed. The sequence of faults of these paths are then concatenated to the sequences of faults already associated with reaching state q. In case the diagnosis engine is a NEDE, the problem is not that trivial since only one possible state q at the end of the previous diagnosis is known. This was already pointed out in (Kurien and Nayak [2000]), but we illustrate it here with Example 3.

Example 3. Recall Example 1 with the observations $[c, b, c]$. The diagnoses were [] with current state 7, or $[f]$ with current state 5 or 8. Consider the additional observations $[a, c, b]$. The path ending in state 7 can be extended by $7 \xrightarrow{a} 1 \xrightarrow{c} 3 \xrightarrow{b} 4$, $7 \xrightarrow{a} 1 \xrightarrow{c} 3 \xrightarrow{b} 4 \xrightarrow{u} 5$, or $7 \xrightarrow{a} 1 \xrightarrow{f} 2 \xrightarrow{c} 4 \xrightarrow{u} 5 \xrightarrow{b} 6$, leading to incremented diagnoses [] or $[f]$. The paths from states 5 and 8 can also be extended leading to diagnoses $[f]$ or $[f, f]$.

Consider now that the additional observations are not $[a, c, b]$ but $[c, b]$. The path ending in state 7 *cannot* be extended, while the paths ending in state 5 or state 8 can be extended, leading to the diagnosis $[f]$. In case the diagnoser is non-exhaustive, it only returned the path ending in state 7. The procedure presented above fails to compute a diagnosis. □

As shown in the Example 3, the procedure may return no diagnosis if the diagnosis is performed by a NEDE. This is the worst case scenario: in a slightly less harmful scenario (which cannot be reproduced in the particular example of this article), the procedure returns a diagnosis that is *not* the preferred one. Basically, the incremented diagnosis ρ is preferred to ρ', but ρ' is returned because the prefix of ρ' was preferred to the prefix of ρ.

It is easy to determine when no diagnosis is found, and a backtracking technique can then be applied. However, we want to limit the backtracking so as to not violate the complexity requirement. Furthermore, the less harmful scenario is usually very hard to identify without a total backtracking, which is definitely impossible.

Kurien and Nayak [2000] proposed to abstract the problem for allowing the incremental procedure. We present a different approach for this problem.

3.2 Incremental Diagnosis with NEDE

We first illustrate the principle of our approach through Example 4.

Example 4. Back to Example 3 with the incremental observations $[c, b]$, the main ambiguity here is the following: when the system is in state 1 (at the beginning or after the occurrence of a) and it generates the observable event c, did the system go through state 3 (without a fault) or state 2 (with a fault)? This ambiguity remains as long as the system generates a sequence of $(c, b)^n$. If the system generates two bs or two cs in a row, the ambiguity disappears. If it generates a, then the state is reset to 1 and the ambiguity can no longer be solved. Consider now that,

independently from the path leading to the state 5, the probability of triggering b in this state is the same as that of triggering c or a. Then, the probability of generating $(c,b)^m$ decreases exponentially with m ($< 3^{-m}$).[3] It is reasonable to consider that for a value of m large enough (say $m = 50$), this probability gets negligible. Thus, if there is an ambiguity about the occurrence of a failure at a given time, after $n = 2 \times m$ observations, this ambiguity will either

(1) be impossible to ever solve because of the occurrence of a, and the best is then to consider the preferred possibility, *i.e.* no failure occurred, or
(2) be dispelled by the occurrence of a pair of bs or cs.

Therefore, a property of the system is that the behaviour before the last n observations can be safely established, and that no backtrack should be required further than this state. □

This property is not a feature of this particular example, but it is a feature of any system though the value n may be difficult to determine. When a system generates an ambiguous behaviour, the next observations may help to solve or determine the most probable path (according to the preferred diagnosis). The observations immediately after the date of this ambiguity are likely to provide the more information. After a longer delay, the information provided by the observations about this particular ambiguity weakens for two reasons:

(1) because the ambiguous behaviours can no longer be distinguished, or
(2) because the ambiguity was already cleared by the previous observations.

We propose a diagnosis technique based on this property. When the behaviour of the system is computed at a given time, the property makes reliable the first part of this behaviour. Thus, we consider that this part is the prefix of what really happened on the system. The last part is not established yet at this stage, but will be when new observations are available. Thanks to this technique, the first part of the behaviour does not need to be computed again when new observations are gathered. On the contrary, the last part of the diagnosis is computed again, and a different explanation (and a different diagnosis) may be generated for this part.

This technique is illustrated in Algorithm 1, which requires the following three inputs:

(1) Prediction window μ, which indicates how many observations are required before a diagnosis is established.
(2) Non-exhaustive diagnosis engine Ω, which takes as input a set of initial states[4] and a sequence of observations, and returns a preferred explanation.
(3) An observation flow $\mathcal{OF}$, which provides the ordered sequence of observations on the system. This flow does not return all the observations at once.

[3] This upper-bound considers only the transitions on state 5; the other possible transitions on states 6, 7, and 8 should be considered for a more accurate value.

[4] A set of states is required because of *diagnosis resets* as explained later.

The established explanation of the observations is stored in ρ and its final state is stored in S_0. The path ρ_u represents the current explanation of all the observations; it may be inconsistent with the final diagnosis; it can be returned any time if the procedure is on-line (see section 3.3). *obs* contains the list of observations that are not yet explained in ρ; its size will generally be bounded by μ.

Whenever new observations *newobs* are available, the following procedure takes place:

- the new observations are added to the observations not explained and a path ρ' is computed from a state of S_0;
- the **if** statement is explained later;
- both paths ρ_u and ρ are updated, and the last μ observations are kept;
- finally the set of states S_0 is updated.

The algorithm does not return the diagnosis but a path from which the diagnosis can be trivially computed.

Algorithm 1 Incremental Diagnosis with a NEDE.

```
1:  input prediction window μ
2:  input NEDE Ω
3:  input observation flow OF
4:  S0 := I
5:  obs := []
6:  ρ := ∅
7:  ρu := ∅
8:  while OF generates newobs do
9:     obs := obs ⊕ newobs
10:    ρ' := Ω(obs, S0)
11:    if ρ' not found then
12:       ρ' := Ω(obs, Q)                    // Diagnosis reset
13:    end if
14:    ρu := ρ ⊕ ρ'
15:    ρ := ρ ⊕ remove-last(ρ', μ)
16:    obs := keep-last(obs, μ)
17:    S0 := {last-state(ρ)}
18: end while
19: return ρu
```

Such a procedure can be seen as a backtracking technique. The diagnoser is allowed to backtrack until the μ previous observations, and the backtracking is limited to this position to avoid complexity explosion. The algorithm may return an incorrect diagnosis. Obviously, if the value μ is set poorly, the algorithm may return a diagnosis that is not the preferred one, in particular if $\mu = 0$. The system presented in Figure 1 has no minimal value for μ that makes the algorithm correct. We discuss how to determine μ in Section 4.

Since this procedure may be incorrect, it may fail to compute a diagnosis (line 10). What shall we do in this case? As specified precedently, the backtracking is limited until a bounded number of observations. The state of the system being lost, the procedure can be simply stopped, and return a failure message.

Alternatively, we can consider that the set of observations *obs* will be sufficient to recover the state of the system. Since the current state is unknown, we run the algorithm from any state of Q (line 12), which is called a *diagnosis reset*. This means that the behaviour computed is not

globally consistent: the state at the beginning of the prediction window abruptly jumps for no sensible reason. However, we expect the procedure to bring the supervision system back on track. If the supervision system is able to recover the system state, the diagnosis will be incorrect around the diagnosis reset but should be correct before and after. In case the system generates many failures, in particular in case of intermittent failures, this procedure should be acceptable and have a good ratio of correct fault identifications. Obviously, the incremental algorithm should warn the human agent of this incorrectness.

3.3 Extension to On-line Diagnosis

This algorithm was presented in the context of incremental diagnosis, but not of *on-line* diagnosis. In on-line diagnosis, the goal is to determine what is currently happening, as opposed to what happened some time ago. It is very difficult to determine what is currently happening because only a very little information is available. The current behaviour is usually ambiguous. However, although exhaustive algorithms return all possible scenarios, the agent generally considers only the preferred one. Exhaustive methods ensure monotonicity (Lamperti and Zanella [2007]), which basically states that any diagnosis at time t is consistent with at least one diagnosis at any time $t' < t$; the diagnosis at time t possibly contains more faults than the diagnosis at time t' since faults possibly occurred between times t' and t.

It is possible to return the explanation ρ_u at any time. This explanation indicates what the current preferred diagnosis is. Note the suffix of this diagnosis is not established, which means that a latter explanation after receiving new observations may be inconsistent with the current one. The monotonicity is thus not ensured. However, we consider that this is not harmful in most cases: since the control decisions on the system are usually performed by the human operator on the preferred diagnoses, the monotonicity is not a strong requirement. Besides, when the behaviour is ambiguous, decisions must still be taken.

4. DETERMINING THE PREDICTION WINDOW

In this section, we discuss how to determine the value of the prediction window μ. Some systems have the property, where Algorithm 1 will always return the preferred diagnosis for value μ greater than a particular value n. We first discuss this value. Because most systems (in particular the decentralised ones) actually do not exhibit such a feature, we propose to use a *large* value for μ. Experiments in the next section show how the selected values impact on the quality of the diagnosis and the efficiency of the computation.

4.1 Correct Prediction Window

Some systems have the following property: any ambiguity on their behaviour raised by an observer can be solved after a bounded number of additional observations is provided. We call these systems *finitely trackable*, and define the property in Definition 5. In this definition, we consider ρ_k and $\rho_k\rho'_k$ are paths on the system model. Furthermore, $obs(\rho'_i) = [o_1, \ldots, o_n, \ldots]$ means that the observations generated by ρ'_j start with $[o_1, \ldots, o_n]$.

Definition 5. (Finite trackability).
A discrete-event system $\mathcal{A}$ is *finitely trackable* iff:

$$\exists n \in \mathbf{N} : \forall \rho_1, \rho_2,\ obs(\rho_1) = obs(\rho_2),$$
$$\forall [o_1, \ldots, o_n] \in (E_o)^n,\ \exists i, j : \{i, j\} = \{1, 2\} \wedge$$
$$\forall \rho'_j,\ obs(\rho'_j) = [o_1, \ldots, o_n, \ldots] \Rightarrow$$
$$\exists \rho'_i : obs(\rho'_i) = obs(\rho'_j) \wedge \rho_i\rho'_i \preceq \rho_j\rho'_j.$$

This definition means that there is a finite delay n such that for any ambiguity (expressed by $obs(\rho_1) = obs(\rho_2)$), and for any sequence of n observations generated after this ambiguity, it is possible to keep only one path ρ_i. Indeed for any path $\rho_j\rho'_j$ consistent with the observations (including the next unknown observations), a preferable path $\rho_i\rho'_i \preceq \rho_j\rho'_j$ also consistent with these observations can be found. Given two paths ρ_1 and ρ_2, the preferred path ρ_i can be different depending on the next observations $[o_1, \ldots, o_n]$.

There are some similarities between this property and the diagnosability property (Sampath et al. [1995]). The latter one indicates that a complete diagnoser can always diagnose the occurrence of a fault after a bounded number of observations after the fault occurred. This is very similar to our finite trackability property especially regarding the finite bound. However, there is no logical relation between diagnosability and finite trackability:

- A finitely trackable system can be non diagnosable. Indeed, a trackable diagnoser solves the ambiguity by *choosing the preferred diagnosis* while a diagnosable diagnoser finds the exact one. E.g., remove the transition $5 \xrightarrow{c} 7$ in the system presented in Figure 1; then this system is finitely trackable with $n = 3$ but is not diagnosable: did a fault occur if we observe $[(c, b, a)^\infty]$?
- A diagnosable system may be not finitely trackable. The reason is that diagnosability is tested on an exhaustive diagnoser: it may be required to keep track of several possible paths on an unbounded delay before the occurrence of the failure. Consider the system presented in Figure 2. Clearly, the system is diagnosable because a failure is always followed by the observable event b. However, it is not finitely trackable because the ambiguity in the first transition ($3 \xrightarrow{u} 2$ or $3 \xrightarrow{u} 4$) cannot be solved in a bounded delay.

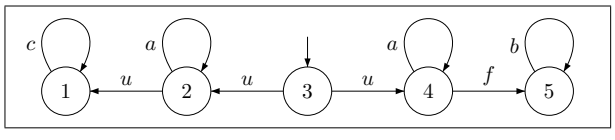

Fig. 2. Example of a diagnosable yet not finitely trackable discrete-event system.

Distributed systems are generally not finitely trackable for the following reason. Whenever an ambiguous behaviour occurs on some part of the system and given an upper bound n, another part of the system can generate $(n + 1)$ observations unrelated to the ambiguity; this independent behaviour does not help clearing the original ambiguity, thus contradicting the existence of n.

4.2 Incorrect Prediction Window

If the system is not finitely trackable, it is still possible to choose a large enough value for μ to ensure a correct diagnosis in most cases.

A larger prediction window increases the probability of a correct diagnosis. But it also has drawbacks. First the computation is almost certainly more expensive with a larger prediction window, since each call to the NEDE is performed on a larger window. Moreover, a behaviour is established — and considered sure — only after a long delay μ, which can be unsatisfactory for on-line diagnosis.

There is a second parameter that can influence the choice of μ. This parameter, denoted λ and called *diagnosis window*, is the number of new observations at each new step, *i.e.* the size of *newobs* in Algorithm 1. If the parameter λ is small, the diagnosis is updated more often. However, a large value of λ may make the diagnosis simpler to solve because the NEDE is called less often.

We conducted a series of experiments on this topic.

5. EXPERIMENTS

For this study, we reuse the benchmark we proposed in (Grastien et al. [2007]). The system contains 20 components with 6 states each. This makes the number of states reach 6^{20}. In many diagnosis problems, some states are no longer reachable after some time, which makes the incremental diagnosis simpler as the size of the search space is diminished. This is not the case in this example since it is always possible to return to the initial state.

The problem is to find one minimal cardinality diagnosis on a sequence of $1,000$ totally-ordered observations. [5] The NEDE is the implementation of the procedure presented in (Grastien et al. [2007]) using one of the state-of-the-art SAT solvers, MINISAT v2.0 (Eén and Sörensson [2003]). [6] The experiments were conducted on a 1.73 GHz Pentium 4 computer.

The diagnosis problem is translated into the following problem: *Finding a path on the system consistent with the observations* with the optimality criterion that the number of faults in the path must be minimal. The path-finding problem is in turn translated into a SAT problem as in the classical SAT planning (Kautz and Selman [1996]), and the search is performed for an incremental number of faults starting from 0. To test the existence of a path (line 11 in Algorithm 1), the path-finding is first performed without constraint on the number of faults.

We studied the performances of the diagnoser with two parameters: μ (size of the prediction window) and λ (number of observations added at each iteration). The incremental SAT feature (Hooker [1993]) was used to speed-up the solving of similar SAT problems within the same pair of parameters but is out of the scope of this article.

[5] Althought several different diagnosis with the same cardinality may be generated, we consider only one diagnosis.

[6] This SAT solver is available from http://www.satcompetition.org/ or http://minisat.se/.

Table 1. Runtime in seconds.

$\lambda\downarrow$ $\mu\rightarrow$	0	10	20	30	40	50
2	7	43	81	180	673	1 429
5	10	28	54	127	288	758
10	8	21	43	86	214	485
20	11	25	45	121	263	453
40	36	58	101	285	321	685

The runtime to incrementally solve the diagnosis problem is given in Table 1. As one could expect, a larger prediction window increases the runtime. A small diagnosis window also has a negative impact on the time, as the number of incremental diagnosis problems solved increases (this number is roughly the total number of observations divided by λ). Note however that a large diagnosis window also increases the runtime especially for small prediction windows because the incremental diagnosis is then performed on larger windows. We tried to solve the diagnosis problem in a non-incremental way with the algorithm used in this experiment; however, no solution was found within two hours ($7,200$ seconds).

Table 2. Number of diagnosis resets.

$\lambda\downarrow$ $\mu\rightarrow$	0	10	20	30	40	50
2	374	130	78	31	5	0
5	182	105	47	0	0	0
10	98	74	17	7	0	0
20	47	28	0	4	0	0
40	21	10	1	0	0	0

The number of diagnosis resets during the computation is given in Table 2. With a prediction window of more than 40, the number of resets is null. On this particular system, a good value for μ is probably 50 or more. Another interesting result is that small diagnosis windows increase the number of resets because they increase the number of incremental problems.

Table 3. Number of false negatives (out of 108).

$\lambda\downarrow$ $\mu\rightarrow$	0	10	20	30	40	50
2	100	66	47	25	7	0
5	97	69	37	0	0	0
10	88	56	15	5	0	0
20	68	39	1	3	0	0
40	36	14	3	0	0	0

We now propose to evaluate the quality of the diagnosis. We study whether the diagnosis algorithm was able to identify each fault and the time they occurred. The scenario used for this evaluation contains 108 faults; the occurrence date within the flow of observations is known for each fault. Using only the observations, it is impossible to determine exactly at what time a fault occurred: there is a varying delay between the occurrence of the fault and the reception of the relevant observations. We considered that the diagnoser identified correctly a fault if it suggested the occurrence of this fault at a date within the flow of observations that differs by no more than 25 observations. The number of missed faults (false negative) is given in Table 3. The faults are well identified when the prediction is large enough. Interestingly, with small prediction windows, the algorithm performs better for the large diagnosis windows than for the small ones. The reason is that a large diagnosis windows acts as a prediction window for the events that

occurred in the beginning of the diagnosis window. The false positives follow a similar tendency.

These experiments clearly show that a large prediction window can lead to a correct incremental diagnosis. This technique can be used to perform diagnosis in an incremental manner or in an on-line context. The experiments also show that the two parameters together with the NEDE have an impact on the computation time which must be considered.

6. RELATED WORK

Diagnosis is often concerned with the on-line monitoring of system, because on-line detection of faults enables an agent to circumvent unfortunate consequences. Yet on-line diagnosis and in particular the issue of incremental diagnosis, are often omitted in the literature.

The Sampath Diagnoser (Sampath et al. [1995]) allows incremental diagnosis with great efficiency. The system model is compiled into a deterministic finite state machine whose transitions are labeled by observable events. Given a sequence of observations, it suffices to follow the single path on the diagnoser and to read the diagnosis labeling the state reached thereby. The limitations of this method are well-known. First although being computed off-line, this approach does not scale up at all, since the size of the diagnoser is exponential in the size of the system model, which is in turn exponential in the number of components. To some extend, this issue can be prevented by the use of specialised and local diagnosers (Pencolé et al. [2006]) but with the loss of global view and generally the loss of accuracy. The second issue is that this method was defined for totally-ordered sequences of observations, and the extension to uncertain observations was not studied and probably adds another complexity. Finally the diagnoser works as a black-box and returns no explanation; an explanation is sometimes expected by the (human) agent in charge of the system.

More recently, some frameworks were developed for the incremental diagnosis (Pencolé and Cordier [2005], Grastien et al. [2005], Lamperti and Zanella [2007]). The diagnosis is reduced to finding all the paths on the model consistent with the observations. Since all the paths are computed, the issue of incremental diagnosis is not that some paths may be missed but how to slice the flow of observations in case of delays. In (Pencolé and Cordier [2005]), the efficiency is ensured by the use of a decentralised approach together with partial-order reduction techniques. In (Cordier and Grastien [2006]), the momentary independencies are used to avoid global and complex computations.

Symbolic methods have also been proposed for the diagnosis of discrete-event systems. In (Schumann et al. [2007]), the unfolding of the model consistently with the observations is performed with BDDs. This approach is complete and the use of symbolic tools greatly simplifies the procedure.

What does the diagnoser need to keep from previous and current computations in incremental diagnosis? The Fault Detection and Identification (FDI) community proposed the use of Kalman filters (Kalman [1960]), which only requires the estimated state from the previous time step.

7. CONCLUSION AND FUTURE WORK

We introduced the notion of non-exhaustive algorithm for the diagnosis of discrete-event systems. The characteristic of such algorithms is that they do not compute all the diagnoses but only the preferred diagnosis. This restriction simplifies the problem as the number of possible diagnoses and their explanations is usually exponential in the number of components. Yet, this restriction is acceptable as the (human) agent is usually only interested by the preferred diagnosis.

Incremental diagnosis is the problem of updating the diagnosis when new observations are received. Because the currently preferred diagnosis may contradict latter observations, a correct incremental algorithm usually needs to compute all the diagnoses, or start from scratch everytime new observations are available. Our proposal relies on the following remark: when a diagnosis is computed, the first part is more reliable than the last part because it is supported by more future observations. When computing a diagnosis, we thus propose to establish the first part, and let the last part, called *prediction window*, free; this latter part is established only when more observations are available. We have shown that our algorithm does not ensure completeness or even consistency in general. However, experiments have demonstrated that for some parameters, the diagnosis returned by this procedure is correct.

We see several extensions to the presented work.

We considered the NEDE returns only the preferred diagnosis. Extensions include returning the k preferred diagnoses or even all the diagnoses (but not necessarily all the paths). It is also possible to increase the number of paths returned by the NEDE. For instance, the intrackable problem in Figure 2 becomes trackable if *two* paths are monitored since the ambiguity of the transitions $3 \xrightarrow{u} 2$ and $3 \xrightarrow{u} 4$ no longer needs to be solved. The definition of finite trackability indicates that after a number of observations, it suffices to establish a single path; it can be extended for k paths. These problems include formally defining when two paths are considered different. Typically, it is not useful to store a path, plus a copy of this path followed by unobservable events, or an equivalent version with respect to concurrency.

Our work can be used for any non-exhaustive algorithm. Whichever algorithm is used, one concern should be the reuse of past computations: the suffix of the path computed at time t is not established, and this suffix will be computed again when new observations are available. It should be possible to recycle several conclusions that arose from the previous computations. This should be investigated more closely.

Finally, we used a temporal decomposition to simplify a diagnosis problem. We want to investigate the spatial decomposition as well. For instance, it may be possible to diagnose a subsystem, establish the diagnosis of a part (the *core*) of this subsystem, and incrementally consider a bigger subsystem. After the core behaviour is set, the diagnosis of the latter subsystem is easier.

ACKNOWLEDGEMENT

We want to thank the reviewers for the very useful comments they provided during the reviewing process. This research was supported by NICTA. NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

REFERENCES

M.-O. Cordier and A. Grastien. Exploiting independence in a decentralised and incremental approach of diagnosis. In *Proc. of Seventeenth International Workshop on Principles of Diagnosis (DX'06)*, 2006.

N. Eén and N. Sörensson. An extensible SAT-solver. In *Proc. of Sixth International Conference on Theory and Applications of Satisfiability Testing (SAT-03)*, 2003.

É. Fabre, Al. Benveniste, St. Haar, and Cl. Jard. Distributed monitoring of concurrent and asynchronous systems. *Journal of Discrete Event Systems*, pages 33–84, 2005. Special issue.

A. Grastien, M.-O. Cordier, and Ch. Largouët. Incremental diagnosis of discrete-event systems. In *Proc. of Sixteenth International Workshop on Principles of Diagnosis (DX-05)*, pages 119–124, 2005.

A. Grastien, Anbulagan, J. Rintanen, and E. Kelareva. Diagnosis of discrete-event systems using satisfiability algorithms. In *Proc. of Twenty-Second AAAI Conference on Artificial Intelligence (AAAI-07)*, pages 305–310. AAAI Press, 2007.

J. Hooker. Solving the incremental satisfiability problem. *Journal of Logic Programming*, 15(1-2):177–186, 1993.

Th. Jéron, H. Marchand, S. Pinchinat, and M.-O. Cordier. Supervision patterns in discrete-event systems diagnosis. In *Proc. of Seventeenth International Workshop on Principles of Diagnosis (DX-06)*, pages 117–124, 2006.

R. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME Journal of Basic Engineering*, 82(Series D):35–45, 1960.

H. Kautz and B. Selman. Pushing the envelope: planning, propositional logic, and stochastic search. In *Proc. of Thirteenth National Conference on Artificial Intelligence (AAAI-96) and the Eighth Innovative Applications of Artificial Intelligence Conference (IAAI-96)*, pages 1194–1201. AAAI Press., 1996.

J. Kurien and P. Nayak. Back to the future for consistency-based trajectory tracking. In *Proc. of the Seventeenth National Conference on Artificial Intelligence (AAAI-00) and Twelfth Conference on Innovative Applications of Artificial Intelligence (IAAI-00)*, pages 370–377. AAAI Press, 2000.

G. Lamperti and M. Zanella. *Diagnosis of Active Systems*. Kluwer Academic Publishers, 2003.

G. Lamperti and M. Zanella. On monotonic monitoring of discrete-event systems. In *Proc. of Eighteenth International Workshop on Principles of Diagnosis (DX-07)*, 2007.

Y. Pencolé and M.-O. Cordier. A formal framework for the decentralised diagnosis of large scale discrete event systems and its application to telecommunication networks. *Artificial Intelligence (AIJ)*, 164:121–170, 2005.

Y. Pencolé, D. Kamenetsky, and A. Schumann. Towards low-cost diagnosis of component-based systems. In *Proc of the Sixth IFAC Symposium on Fault Detection, Supervision and Safety of Technical Process (SAFEPROCESS)*, 2006.

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

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

G. Torta and P. Torasso. An on-line approach to the computation and presentation of preferred diagnoses for dynamic systems. *AI Communications*, 20(2):93–116, 2007.