On-Time Diagnosis of Discrete Event Systems

Aditya Mahajan and Demosthenis Teneketzis

Department of EECS, University of Michigan, Ann Arbor, MI – 48109, USA.
{adityam,teneket}@eecs.umich.edu

Abstract— A formulation and solution methodology for *on-time* fault diagnosis in discrete event systems is presented. This formulation and solution methodology captures the timeliness aspect of fault diagnosis and is therefore different from all other approaches to fault diagnosis in discrete event systems which are asymptotic in nature. A monitor observes a projection of the events that occur in the system. After each observation it can either raise an alarm and shut down the system or allow the system to continue. If the system is stopped when no fault had occurred, a false alarm penalty is incurred; on the other hand if a fault had occurred, a delayed detection penalty is incurred. Both these penalties are trace dependent. The on-time diagnosis problem is formulated as a minimax optimization problem where the objective is to choose a monitoring rule which minimizes the worst case cost along all traces of the language describing the discrete event system. An optimal diagnosis rule is determined using a dynamic programming algorithm. An example is presented which illustrates our methodology and highlights the difference between our formulation of on-time diagnosis with existing results on asymptotic diagnosis of discrete event systems.

I. Introduction

Fault diagnosis is important in many applications such as transportation systems [1], power systems [2], HVAC (heating ventilation and air-conditioning) systems [3, 4] other industrial systems [5]–[8], and communication networks [9]–[14]. Fault diagnosis can be formulated in two different ways: (i) asymptotic fault diagnosis; and (ii) on-time fault diagnosis. In asymptotic fault diagnosis the objective is to determine, as accurately as possible, the occurrence of faults; the delay in fault diagnosis is not critical. On the other hand, on-time fault diagnosis is concerned with situations where the timeliness of fault detection is important. Within the context of discrete event systems, asymptotic fault diagnosis has been investigated in [15]–[24], [25] and references therein. To the best of our knowledge, on-time fault diagnosis has not been addressed so far within the context of DES. In this paper we formulate the on-time diagnosis problem in DES and provide a solution methodology.

The rest of the paper is organized as follows. In Section II we introduce some preliminary notation used in the rest of the paper. In Section III we formulate and solve the on-time diagnosis problem. In Section IV we present an example illustrating our solution methodology. We conclude in Section V.

II. Preliminaries

A. Languages

Let Σ denote a set of events. A *language* defined over Σ is a set of finite-length strings from events in Σ. Σ^* denotes the *Kleene-closure* of Σ, that is, the set of *all* finite strings of elements of Σ, including the empty string ε. The symbol $\cdot$ denotes the *concatenation* operation. If $t \cdot u = s$ with $t, u \in \Sigma^*$, then t is called the *prefix* of s and is denoted by $t \prec s$. Observe that both s and ε are prefixes of s.

The *prefix closure* $\bar{L}$ of L consists of all the prefixes of all the strings in L. A language is called *prefix closed* if any prefix of any string in L is also an element of L.

For any string s belonging to language L, the *post language* L after s, denoted by L/s, is given by

$$L/s := \{t \in \Sigma : s \cdot t \in L\}$$

For any sub-language L' of L, the post-language L after L' is given by

$$L/L' = \bigcup_{s \in L'} L/s$$

A string s belonging to L is called a *terminal* string if the post language L after s is empty (i.e., $L/s = \emptyset$). Any language L can be partitioned into the set L_T of terminal strings and the set L_{NT} of non-terminal strings, i.e.,

$$L = L_T \cup L_{NT},$$

where $L_T := \{s \in L : L/s = \emptyset\}$ and $L_{NT} := L \setminus L_T$. If L is prefix-closed, then

$$L_{NT} = \bar{L}_T \setminus L_T.$$

For example, consider a language L defined over $\Sigma = \{a, b, c\}$ given by

$$L := \{\varepsilon, a, b, ab, ac, aa, ba, bb, bc\}, \tag{1}$$

then

$$L_T = \{ab, ac, aa, ba, bb, bc\} \quad \text{and} \quad L_{NT} = \{\varepsilon, a, b\}$$

B. Prefix-preserving projections

Consider a prefix-closed langauge L, and a set valued function $O : L_{NT} \to 2^\Sigma$. For any string $s \in L$, $O(s)$ denotes the observable set of observable events immediately following s. This function O defines a projection P as follows: for any $\sigma \in \Sigma$, and any $s \cdot \sigma \in L$,

$$P(\varepsilon) := \varepsilon,$$

$$P(\sigma) := \begin{cases} \sigma, & \text{if } \sigma \in O(\varepsilon); \\ \varepsilon, & \text{otherwise,} \end{cases}$$

$$P(s \cdot \sigma) := \begin{cases} P(s) \cdot \sigma, & \text{if } \sigma \in O(s); \\ P(s), & \text{otherwise.} \end{cases}$$

The *projected language* $P(L)$ is the set of projections of all strings of L, i.e.,

$$P(L) := \{P(s) : s \in L\}.$$

By construction, the projected language is prefix-closed; hence, we call such projections *prefix-preserving* projections. The inverse projection P^{-1} is a map from $P(L)$ to 2^L defined as follows

$$P^{-1}(t) := \{s \in L : P(s) = t\}.$$

These projections are a generalization of the natural projections. In this paper we do not restrict ourselves to natural projections, because, in the future, we want to investigate on-time diagnosis of decentralized systems with communication. In such systems, communication rules can be trace dependent; so, it is necessary to understand diagnosis in centralized systems with such projections.

C. Some sub-languages and inverse mappings

We define some sub-languages of L and some inverse mappings from $P(L)$ to 2^L that will be useful in future analysis.

Definition 1: Define sub-languages L_{NT}^C, L^C of L as follows:

$$L_{NT}^C := \{s \cdot \sigma \in L_{NT} : \sigma \in O(s)\},$$
$$L^C := L_{NT}^C \cup L_T$$

Definition 2: Define inverse maps Q_T, Q from $P(L)$ to 2^L as follows:

$$Q_T(t) := \{s \in L_T : t = P(s)\},$$
$$Q(t) := \{s \cdot \sigma \in L : \sigma \in O(s) \text{ and } t = P(s \cdot \sigma)\}.$$

Notice that $Q_T(t)$ denotes the set of terminal strings of the language that give projection t; $Q(t)$ denotes the set of strings of the language that give projection t and whose last event is observable.

We illustrate the concepts introduced in Sections B and C by means of the following example.

D. An example

Consider the language L defined in (1). Let the observable events O be given by:

$$O(\varepsilon) = \{a\}, \quad O(a) = \emptyset, \quad \text{and} \quad O(b) = \{a, b\}. \tag{2}$$

This observation function is illustrated in Figure 1. Notice that the set of observable events is trace-dependent. After ε the event b is not observable, but it is observable after b. Further, the projections are given by
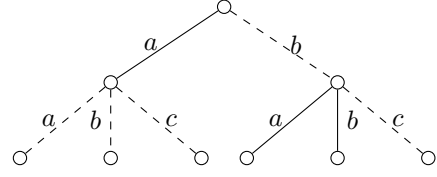

Fig 1. A graphical representation of the language L of (1) and the observable events given by (2). An observable event is denoted by a solid line and an unobservable event by a dashed line.

$$P(\varepsilon) = \varepsilon, \quad P(a) = a, \quad P(b) = \varepsilon,$$
$$P(aa) = P(ab) = P(ac) = a,$$
$$P(ba) = a, \quad P(bb) = b, \text{ and } P(bc) = \varepsilon. \tag{3}$$

The projected language $P(L) = \{\varepsilon, a, b\}$, and the inverse projection P^{-1} is given by

$$P^{-1}(\varepsilon) = \{\varepsilon, b, bc\}, \quad P^{-1}(a) = \{a, ab, aa, ab, ac, ba\},$$
$$\text{and} \quad P^{-1}(b) = \{bb\}.$$

The sub-languages L_{NT}^C and L^C are given by:

$$L_{NT}^C = \{a\} \quad \text{and} \quad L^C = \{a, aa, ab, ac, ba, bb, bc\}.$$

The mappings Q_T and Q are given by:

$$Q_T(\varepsilon) = \emptyset, \quad Q_T(a) = \{aa, ab, ac, ba\}, \quad Q_T(b) = \{bb\},$$

and

$$Q(\varepsilon) = \{\varepsilon\}, \quad Q(a) = \{a, ba\}, \quad Q(b) = \{bb\}.$$

III. The on-time diagnosis problem

A. The model

We consider a dynamic system represented by a prefix-closed language L over an event set Σ. In this paper we restrict attention to finite and bounded languages (i.e., the language has finite number of strings, and the length of each string in the language is bounded.) Suppose $\Sigma_f \subset \Sigma$ is the set of fault event. An observer, which we call the *monitor*, observes a prefix-preserving projection P with an observation function O. The monitor has to ascertain whether a fault has occurred in the system or not. After taking each observation it can raise an alarm, in which case the system is shut down, or it can decide to do nothing, in which case the system continues to operate. The rule used to decide when to raise an alarm is called a *monitoring* rule and is a function from $P(L)$ to $\{0, 1\}$, where 0 means monitor does not raise an alarm and 1 means that the monitor raises an alarm.

When an alarm is raised, the system is shut down immediately and cannot execute any other event. This statement can best be explained by an example. Consider the language shown in Figure 2a. Suppose on observing ba the monitor decides to raise an alarm. Note that $P(L) = \{\varepsilon, b, ba, bac\}$, $P^{-1}(ba) = \{ba, baa, bab, bac, bba, bbaa, bbab\}$, and $Q(ba) = \{ba, bba\}$. All strings in $P^{-1}(ba)$ lead to the observation a.
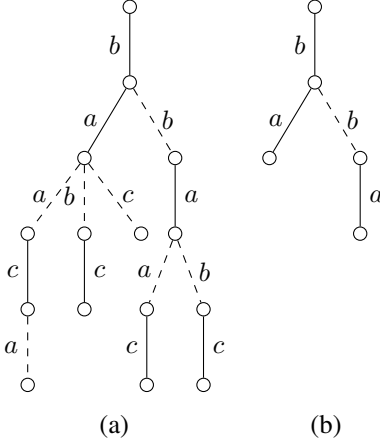
b

a b a b

a b c a a

c c a b

a c c

(a)   (b)

Fig 2. (a) A language L,
(b) monitored sub-language $L|_g$.

However, the moment the monitor observes a after b, the system has to be in strings in $Q(ba)$. The monitor raises an alarm and the system is shut down immediately. So, when the system stops, it would be in one of the strings in $Q(ba)$. The strings belonging to $P^{-1}(ba) \setminus Q(ba)$ are never reached. For any monitoring rule, the system will always stop in a string belonging to L^C.

A monitoring policy $g : P(L) \to \{0,1\}$ restricts the language L to a sub-language. We call this restricted language *the monitored sub-language* and denote it by $L|_g$. For the language shown in Figure 2a, the monitoring rule g given by $g(ba) = 1$, $g(\varepsilon) = g(b) = g(bac) = 0$ gives the monitored sub-language shown in Figure 2b.

We now want to define a metric to measure the quality of a monitoring rule. We want this metric to capture the notion of "timeliness of fault diagnosis". In order to define such a metric we first introduce a cost function $C(\cdot)$ on all strings in L^C. For strings belonging to L^C_{NT}, if the string contains a fault, the cost corresponds to the damage done by allowing the system to run in a faulty state; if the string does not contain a fault, the cost corresponds to the false alarm penalty. For strings belonging to L_T, if the string contains a fault, the cost corresponds to the damage done by allowing the system to complete all its tasks in a faulty state; if the (terminal) string does not contain a fault, the cost is zero.

The choice of the cost function C depends on the application. For example, consider a system where we want to detect a fault as soon as possible after its occurrence (i.e., after as few as possible events have occurred after the fault) and the false alarm penalty is fixed. In this situation, suppose the system is stopped after a non-terminal trace s has occurred: if s does not contain a fault, the cost is equal to the false alarm penalty; if s contains a fault, the cost is proportional to the number of events that have occurred after the fault. If the system is not-stopped and executes a terminal trace s, then if s does not contain a fault the cost is zero; if s contains a fault the cost is proportional to the number of events that have occurred after the fault plus a fixed penalty

accounting for not stopping a faulty system. Thus the cost function C is: for $s \in L^C_{NT}$

$$C(s) = \begin{cases} c \cdot (n - \tau(s)), & \text{if } s \text{ contains a fault} \\ H_{NT}, & \text{otherwise} \end{cases}$$

for $s \, \text{in} \, L^C \setminus L^C_{NT}$

$$C(s) = \begin{cases} c \cdot (n - \tau(s)) + H_T, & \text{if } s \text{ contains a fault} \\ 0, & \text{otherwise} \end{cases}$$

where $\tau(s)$ denotes the first time a fault occurs in string s, c is a constant, H_{NT} is the false alarm penalty, and H_T is the terminal penalty.

Now, the performance of a monitoring rule can be quantified by the worst case cost when the system stops (either shut down due to an alarm, or finishes all its tasks) and is given by

$$\mathcal{J}(g) := \max_{s \in (L|_g)_T} C(s). \tag{4}$$

B. Problem formulation and solution

We are interested in the following optimization problem.

Problem 1 (The on-time diagnosis problem): Given a prefix-closed, finite and bounded language L, a prefix-preserving projection P with an observation function O, and a cost function C defined on L^C, choose a monitoring rule g^* belonging to the family $\mathscr{G}$ of all functions from $P(L)$ to $\{0,1\}$, which minimizes the worst case cost given by (4), i.e.

$$\mathcal{J}(g^*) = \mathcal{J}^* := \min_{g \in \mathscr{G}} \max_{s \in (L|_g)_T} C(s). \tag{5}$$

Since there are only a finite number of monitoring rules, Problem 1 is well defined and an optimal rule always exists. Problem 1 is a centralized minimax optimization problem and can be solved by dynamic programming. An optimal solution can be obtained as follows.

Theorem 1: An optimal monitoring rule for Problem 1 can be obtained by solving the following set of recursive equations.

$$\forall t \in (P(L))_T$$
$$V(t) = \min \left\{ \max_{s \in Q(t)} C(s), \max_{s \in Q_T(t)} C(s) \right\} \tag{6}$$

$$\forall t \in (P(L))_{NT},$$
$$V(t) = \min \Big\{ \max_{s \in Q(t)} C(s),$$
$$\max \big\{ \max_{s \in Q_T(t)} C(s), \max_{e \in O_C(t)} V(t \cdot e) \big\} \Big\}, \tag{7}$$

where $O_C(t) := \{e \in \Sigma : t \cdot e \in P(L)\}$. Furthermore, an optimal monitoring rule is described as follows: $\forall t \in (P(L))_T$,

$$g^*(t) = \begin{cases} 1, & \text{if } \max_{s \in Q(t)} C(s) < \max_{s \in Q_T(t)} C(s), \\ 0, & \text{otherwise,} \end{cases}$$

$$\tag{8}$$

and $\forall t \in (P(L))_{NT}$,

$$g^*(t) = \begin{cases} 1, & \text{if } A < B \\ 0, & \text{otherwise.} \end{cases} \qquad (9)$$

where

$$A = \max_{s \in Q(t)} C(s)$$

$$B = \max\left\{ \max_{s \in Q_T(t)} C(s), \max_{s \in e \in O_C(t)} V(t \cdot e) \right\}.$$

The recursive equations (6) and (7) have the following interpretation:

Eq (6): For terminal strings t, calculate the worst cost of stopping (i.e., $\max_{s \in Q(t)} C(s)$) and the worst cost of continuing (i.e., $\max_{s \in Q_T(t)} C(s)$) and choose the action which incurs the smaller cost.

Eq (7): For non-terminal strings t, calculate the worst cost of stopping (i.e., $\max_{s \in Q(t)} C(s)$) and the worst cost of continuing which is computed as follows: if the monitor allows the system to continue, the trace that is being executed may (a) terminate before the monitor obtains another observation, or (b) its continuation may lead to another observation. The worst cost for (a) is $\max_{s \in Q_T(t)} C(s)$, the worst cost for (b) is $\max_{e \in O_C(t)} V(t \cdot e)$. The maximum of these two costs gives the worst cost of continuing. The monitor takes the action (stop or continue) that incurs the smaller cost.

Proof of Theorem 1. We first establish the following three lemmas.

Lemma 1: Let $J(t; g)$ denote the worst case continuation cost after the monitor has observed t and is using the monitoring rule g. Then $\forall t \in ((P(L))_T$,

$$J(t; g) = \begin{cases} \max_{s \in Q(t)} C(s), & \text{if } g(t) = 1, \\ \max_{s \in Q_T(t)} C(s), & \text{if } g(t) = 0, \end{cases} \qquad (10)$$

and $\forall t \in ((P(L))_{NT}$, if $g(t) = 1$

$$J(t; g) = \max_{s \in Q(t)} C(s),$$

while $\forall t \in ((P(L))_{NT}$, if $g(t) = 0$

$$J(t; g) = \max\left\{ \max_{s \in Q_T(t)} C(s), \max_{e \in O_C(t)} J(t \cdot e; g) \right\}$$

Lemma 2: For any monitoring rule g and any $t \in P(L|_g)$,

$$J(t; g) \geq V(t) \qquad (11)$$

Lemma 3: The monitoring rule g^* defined by (8) satisfies

$$J(t; g^*) = V(t) \qquad (12)$$

The above three lemmas are proved in Appendix A. Observe that for any monitoring rule g, $\varepsilon \in P(L|_g)$ and $\mathcal{J}(g) = J(\varepsilon; g)$. Therefore, Lemmas 2 and 3 imply that for any monitoring policy g,

$$\mathcal{J}(g^*) = J(\varepsilon; g^*) \leq J(\varepsilon; g) = \mathcal{J}(g). \qquad (13)$$

Thus, g^* is an optimal monitoring rule. $\qquad \square$

IV. An Example

In this section we one language with three instances of cost functions. These instances capture the scenarios with high penalty for false alarm, and high penalty for delayed detection, and the scenario where a catastrophic event can occur. For each of these three instances we find an optimal monitoring rule. The details of the computations are available at [26].
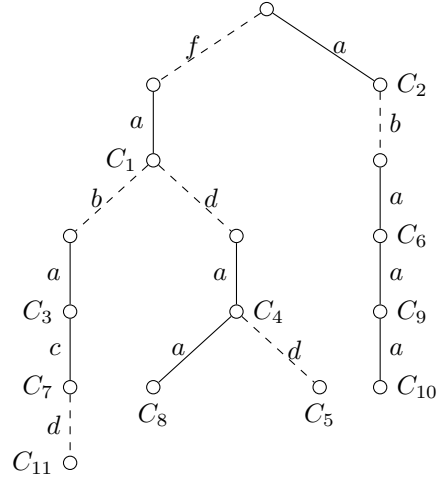

Fig 3. An example of on-time diagnosis problem. The event f denotes the fault event. The projection in this case is a natural projection where events a and c are observable, while events b, d, and f are unobservable.

Consider the language shown in Figure 3. The projection is a natural projection where events a and c are observable while events b, d, and f are unobservable. The event f is a fault event. Thus, the observed language $P(L)$ is $\{\varepsilon, a, aa, aaa, aac, aaaa\}$. There are 11 strings that belong to L^C; we denote the cost of stopping at these strings by C_i as shown in Figure 3. We consider the following instances for the cost functions.

Instance 1 (High false alarm penalty): Suppose the costs for the language of Figure 3 are $C_1 = 1$, $C_2 = 10$, $C_3 = C_4 = 3$, $C_5 = 4$, $C_6 = 10$, $C_7 = C_8 = 4$, $C_9 = C_{10} = 10$, and $C_{11} = 5$. The cost of stopping at non-faulty states (a, aba, $abaa$, and $abaaa$) is much higher than the cost of stopping at other states, thus, these costs corresponds to a situation where the false alarm penalty is high. The optimal monitoring rule in this case is

$$g(\varepsilon) = g(a) = g(aa) = g(aaa) = g(aaaa) = 0$$

$$g(aac) = 1$$

Due to the high false alarm penalty, the monitor does not stop the system until it is sure that a fault has occurred. In general, if the false alarm penalty is infinite, optimal monitoring rule would be the same as the asymptotic diagnoser of [15].

Instance 2 (High penalty of delayed detection): Suppose

the costs for the language of Figure 3 are $C_1 = 10$, $C_2 = 1$, $C_3 = C_4 = 11$, $C_5 = 12$, $C_6 = 1$, $C_7 = C_8 = 12$, $C_9 = C_{10} = 1$, $C_{11} = 13$. These cost correspond to a situation where the delayed detection penalties are much higher than false alarm penalties. The optimal monitoring rule in this case is

$$g(\varepsilon) = g(aaa) = g(aaaa) = 0$$
$$g(a) = g(aa) = g(aac) = 1$$

Due to the high delayed detection penalty, the monitor raises an alarm and stops the system as soon as there is a *possibility* that a fault could have occurred.

Instance 3 (Catastrophic event): Suppose the costs for the language of Figure 3 are $C_1 = 1$, $C_2 = 10$, $C_3 = 2$, $C_4 = 4$, $C_5 = 100$, $C_6 = 10$, $C_7 = 3$, $C_8 = 12$, $C_9 = 10$, $C_{10} = 15$, $C_{11} = 12$. These costs correspond to a situation when the occurrence of two d events after a fault are catastrophic. The optimal monitoring rule in this case is

$$g(\varepsilon) = g(a) = g(aaaa) = 0$$
$$g(aa) = g(aaa) = g(aac) = 1$$

Due to the catastrophic nature of $fadad$, the monitor raises an alarm when there is a possibility that continuing the system could lead to the catastrophic string.

V. Conclusion

We have formulated and solved a fault diagnosis problem within the context of discrete event models. Our formulation takes into account the timeliness of fault diagnosis. The key idea in the formulation is to penalize false alarm as well as the string-dependent amount of delay in fault detection. With these penalties the on-time diagnosis problem can be formulated as a minimax optimization problem. We provided an algorithm for the solution of this problem and illustrated via examples the nature of the solution and its differences from the solution of the "asymptotic fault diagnosis problem" that has appeared in the literature.

Appendix A. Proof of the three lemmas

1. Proof of Lemma 1

If $g(t) = 1$, then an alarm is raised and the system is shutdown. The system can have executed any of the strings in $Q(t)$ when the last event in t is observed. So, the worst case continuation cost is the maximum cost incurred amongst all strings in $Q(t)$. Thus,

$$J(t; g) = \max_{s \in Q(t)} C(s), \quad \text{when } g(t) = 1. \quad (14)$$

If $g(t) = 0$ then the monitor allows the system to continue. If $t \in ((P(L))_T$, then the monitor will not see any other event

in the future. The system will ultimately stop in one of the strings in $Q_T(t)$ and will incur a corresponding cost. Thus, the worst case continuation cost in this case is

$$J(t; g) = \max_{s \in Q_T(t)} C(s), \quad \text{when } g(t) = 0 \text{ and } t \in (P(T))_T. \quad (15)$$

If $t \in (P(L))_{NT}$, the system may either end up in a terminal string in $Q_T(t)$, in which case the monitor will not see anything in the future, or the system will not end in $Q_T(t)$, in which case the monitor will observe some event in $O_C(t)$ in the future. So, the worst case continuation cost in this case is the higher of these costs, hence

$$J(t; g) = \max \Big\{ \max_{s \in Q_T(t)} C(s),$$
$$\max \Big\{ \max_{s \in Q_T(t)} C(s), \max_{e \in O_C(t)} V(t \cdot e) \Big\} \Big\}$$
$$\text{when } g(t) = 0 \text{ and } t \in (P(L))_{NT}. \quad (16)$$

Equations (14), (15), and (16) complete the proof of the lemma.

2. Proof of Lemma 2

Partition $P(L|_g)$ into disjoint sets M_0, M_1, ... such that

$$M_0 = \Big\{ t \in (P(L|_g))_T \Big\}$$
$$M_1 = \Big\{ t \in (P(L|_g))_{NT} : \exists \sigma_1 \in \Sigma$$
$$\text{such that } t \cdot \sigma_1 \in (P(L|_g))_T \Big\}$$
$$\cdots = \cdots$$
$$M_i = \Big\{ t \in (P(L|_g))_{NT} : \exists \sigma_1, \cdots, \sigma_i \in \Sigma$$
$$\text{such that } t \cdot \sigma_1 \cdots \sigma_i \in (P(L|_g))_T \Big\}$$

Note that by construction, for all $t \in M_i$, $i \neq 0$, $g(t) = 0$. We want to show that for all $t \in P(L|_g)$

$$J(t; g) \geq V(t) \quad (17)$$

We will show this by induction on the sets M_i. For $t \in M_0$,

$$J(t; g) = \max_{s \in Q(t)} C(s) \text{ or } \max_{s \in Q_T(t)} C(s)$$
$$\geq \min \Big\{ \max_{s \in Q(t)} C(s), \max_{s \in Q_T(t)} C(s) \Big\}$$
$$=: V(t) \quad (18)$$

This is the basis for induction. Now assume that for all $t \in M_0 \cup \cdots \cup M_i$, (17) is true. We will show that (17) is also true for all $t \in M_{i+1}$. If $M_{i+1} = \emptyset$, the claim is a vacuous truth. Otherwise, consider $t \in M_{i+1}$. As noted above $g(t) = 0$. Thus,

$$J(t; g) = \max \Big\{ \max_{s \in Q_T(t)} C(s), \max_{e \in O_C(t)} J(t \cdot e; g) \Big\}$$

Observe that for all $e \in O_C(t)$, $t \cdot e$ belongs to $M_0 \cup \cdots \cup M_i$. By the induction hypothesis $J(t \cdot e; g) \geq V(t \cdot e)$. Thus,

$$J(t; g) = \max \left\{ \max_{s \in Q_T(t)} C(s), \max_{e \in O_C(t)} J(t \cdot e; g) \right\}$$

$$\geq \max \left\{ \max_{s \in Q_T(t)} C(s), \max_{e \in O_C(t)} V(t \cdot e) \right\}$$

$$=: V(t) \qquad (19)$$

Thus, by the principle of induction, the lemma is true.

3. Proof of Lemma 3

This lemma can be proved along the same lines as the proof of Lemma 2. By the definition of $g^*(t)$, the relation of (18) holds with equality. As a consequence of this, the relation in (19) also holds with equality. Therefore, by the principle of induction, Lemma 3 is true.

REFERENCES

[1] R. Sengupta, "Discrete event diagnosis of automated vehicles and highways," in *Proc. of 2001 American Control Conf.*, June 2001.

[2] T.-S. Yoo and H. Garcia, "Event diagnosis of discrete event systems with uniformly and non-uniformly bounded diagnosis delays," in *Proc of 2004 American Control Conf*, 2004, pp. 5102-5107.

[3] K. Sinnamohideen, "Discrete-event diagnosis of heating, ventilation, and air-conditioning systems," in *Proc. of 2001 American Control Conf.*, June 2001.

[4] M. Sampath (1995). Discrete event systems based diagnostics for a variable air volume terminal box application. Techincal Report Advanced Development Team, Johnsons Control Inc..

[5] M. Sampath, A. Godambe, E. Jackson, and E. Mallow, "Combining qualitative and quantitative reasoning — a hybrid approach to failure diagnosis of industrial systems," in *IFAC Saveprocesses*, 2000, pp. 494–501.

[6] M. Sampath (1999). Embedded print engine diagnostics: The DC265 project and beyond. Technical Report X9900094, Xerox Corporation.

[7] E. Garcia, F. Morant, R. Blasco-Giménez, and E. Quiles, "Centralized modular diagnosis and the phenomenon of coupling," in *Proc of the 6th International Workshop on Discrete Event Systems (WODES'02)*, 2002, pp. 161–168.

[8] Y.-L. Chen and G. Provan, "Modelling and diagnosis of timed discrete event systems — a factory automation example," in *Proc. of 1997 American Control Conf.*, 1997, pp. 31–36.

[9] Y. Pencolé, "Diagnostic décentralisé de systèmes à évènements discrets: application aux résaux de télécommunications," Ph.D. Thesis, Université de Rennes, France, 2002.

[10] ——, "Decentralized diagnoser approach: Application to telecommunication networks," in *Proc. DX'00: Eleventh International Workshop on Principles of Diagnosis*, 2000, pp. 185–192.

[11] Y. Pencolé, M.-P. Cordier, and L. Rozé, "A decentralized model-based diagnostic tool for complex systems," in *Proc of 13th IEEE Int. Conf. on Tools with Arif. Intel. (IC-TAI'01)*, 2001, pp. 95–102.

[12] L. Rozé, "Supervision de réseaux de télécommunications: Une approche à base de modèles," Ph.D. Thesis, Université de Rennes I, France, 1997.

[13] L. Rozé and M.-O. Cordier, "Diagnosing discrete-event systems: An experiment in telecommunication netowrks," in *Proc of the 1998 International Worksho[on Discrete Event Systems (WODES'98)*, 1998, pp. 130–137.

[14] ——, "Diagnosing discrete-event systems: Extending the diagnoser approach to deal with telecommunication netowrks," *Discrete Event Dynamic Systems: Thoery and Applications*, vol. 12, pp. 43–81, 2002.

[15] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis, "Diagnosability of discrete event systems,", vol. 40, no. 9, pp. 1555–1575, Sep. 1995.

[16] ——, "Failure diagnosis using discrete event models,", vol. 4, no. 2, pp. 105-124, Mar. 1996.

[17] M. Sampath, S. Lafortune, and D. Teneketzis, "Active diagnosis of discrete event systems,", vol. 43, no. 7, pp. 908–929, Jul. 1998.

[18] M. Larson, "Diagnosis and analysis of diagnosis properties using discrete event systems," in *Proc of 37th IEEE Conference on Decision and Control*, 1998, pp. 3775–3780.

[19] T. Chun (1996). Diagnostic supervisory control: A discrete event system approach. Master's thesis, University of Toronto, Dept. of Elec. Eng..

[20] S. Jiang, R. Kumar, and H. Garcia, "Diagnosis of repeated/intermittent failures in discrete event systems," *Transportation Research*, vol. 19, no. 2, pp. 310–323, Apr. 2003.

[21] J. Ashley and L. E. Holloway, "Qualitative diagnosis of condition systems," *Discrete Event Dynamic Systems: Theory and Applications*, vol. 14, no. 4, pp. 395–412, Oct. 2004.

[22] P. Baroni, G. Lamperti, P. Pogliano, and M. Zanella, "Diagnosis of a class of distributed discrete-event systems," *Systems, Man and Cybernetics, Part A, IEEE Transactions on*, vol. 30, no. 6, pp. 731–752, Nov. 2000.

[23] S. Bavishi and E. K. P. Chong, "Automated fault diagnosis using a discrete event systems framework," in *Intelligent Control, 1994., Proceedings of the 1994 IEEE International Symposium on*, 1994, Columbus, OH, pp. 213–218.

[24] O. Contant, S. Lafortune, and D. Teneketzis, "Failure diagnosis of discrete event systems: The case of intermittent failures," in *Proc of 41st IEEE Conference on Decision and Control*, 2002.

[25] S. Lafortune, D. Teneketzis, M. Sampath, R. Sengupta, and K. Sinnamohideen, "Failure diagnosis of dynamic systems: an approach based on discrete event systems," in *American Control Conference, 2001. Proceedings of the 2001*, vol. 3, 2001, Arlington, VA, pp. 2058–2071.

[26] A. Mahajan and D. Teneketzis, *On-time diagnosis of discrete event systems—some examples*, 2008. Available at http://www.eecs.umich.edu/˜adityam/publications/conferences/wodes2008/appendix.pdf.