

# Sensor Minimization Problems with Static or Dynamic Observers for Fault Diagnosis

Franck Cassez\*    Stavros Tripakis†    Karine Altisen‡

January 30, 2007

## Abstract

We study sensor minimization problems in the context of fault diagnosis. Fault diagnosis consists in synthesizing a diagnoser that observes a given plant and identifies faults in the plant as soon as possible after their occurrence. Existing literature on this problem has considered the case of static observers, where the set of observable events does not change during execution of the system. In this paper, we consider static as well as dynamic observers, where the observer can “switch” sensors on or off, thus dynamically changing the set of events it wishes to observe. It is known that checking diagnosability (whether an observer capable of identifying faults exists) can be solved in polynomial time for static observers, and we show that the same is true for dynamic ones. On the other hand, minimizing the number of (static) observable events required to achieve diagnosability is NP-complete. We show that this is true also in the case of mask-based observation, where some events are observable but not distinguishable. For dynamic observers, we prove that a most permissive observer can be computed in doubly exponential time, using a game-theoretic approach. We further investigate optimization problems for dynamic observers and define a notion of cost of an observer. Finally we show how to compute an optimal observer.

## 1 Introduction

**Monitoring, Testing, Fault Diagnosis and Control.** Many problems concerning the monitoring, testing, fault diagnosis and control of discrete event systems (DES) can be formalized by using finite automata over a set of *observable*

---

\*CNRS/IRCCyN, 1 rue de la Noë, BP 92101, 44321 Nantes Cedex 3, France

†Cadence Berkeley Labs, 1995 University Avenue, Berkeley, CA, 94704, USA, and CNRS, Verimag Laboratory, Centre Equation, 2, avenue de Vignate, 38610 Gières, France. Email: tripakis@cadence.com.

‡INPG and Verimag Laboratory, Centre Equation, 2, avenue de Vignate, 38610 Gières, France.

events  $\Sigma$ , plus a set of *unobservable* events [9, 12]. The invisible actions can often be represented by a single unobservable event  $\varepsilon$ . Given a finite automaton over  $\Sigma \cup \{\varepsilon\}$  which is a model of a *plant* (to be monitored, tested, diagnosed or controlled) and an *objective* (good behaviours, what to test for, faulty behaviours, control objective) we want to check if a monitor/tester/diagnoser/controller exists that achieves the objective, and if possible to synthesize one automatically.

The usual assumption in this setting is that the set of observable events is fixed (and this in turn determines the set of unobservable events as well). Observing an event usually requires some detection mechanism, i.e., a *sensor* of some sort. Which sensors to use, how many of them, and where to place them are some of the design questions that are often difficult to answer, especially without knowing what these sensors are to be used for.

In this paper we study problems of *sensor minimization*. These problems are interesting since observing an event can be costly in terms of time or energy: computation time must be spent to read and process the information provided by the sensor, and power is required to operate the sensor (as well as perform the computations). It is then essential that the sensors used really provide useful information. It is also important for the computer to discard any information given by a sensor that is not really needed. In the case of a fixed set of observable events, it is not the case that all sensors always provide useful information and sometimes energy (sensor operation and computer treatment) is spent for nothing. For example, to diagnose a fault in the system described by the automaton  $\mathcal{B}$ , Figure 4, an observer only has to watch event  $a$ , and *when  $a$  has occurred*, to watch event  $b$ : if the sequence  $a.b$  occurs, for sure a fault has occurred and the observer can raise an alarm. It is then not useful to switch on sensor  $b$  before an  $a$  has occurred.

**Sensor Minimization and Fault Diagnosis.** We focus our attention on sensor minimization, without looking at problems related to sensor placement, choosing between different types of sensors, and so on. We also focus on a particular observation problem, that of *fault diagnosis*. We believe, however, that the results we obtain are applicable to other contexts as well.

Fault diagnosis consists in observing a plant and detecting whether a fault has occurred or not. We follow the discrete-event system (DES) setting of [10] where the behavior of the plant is known and a model of it is available as a finite-state automaton over  $\Sigma \cup \{\varepsilon, f\}$  where  $\Sigma$  is the set of observable events,  $\varepsilon$  represents the unobservable events, and  $f$  is a special unobservable event that corresponds to the faults. Checking *diagnosability* (whether a fault can be detected) for a given plant and a *fixed* set of observable events can be done in polynomial time [10, 13, 6]. (Notice that synthesizing a diagnoser involves determinization in general, thus cannot be done in polynomial time.)

We examine sensor optimization problems with both *static* and *dynamic* observers. A static observer always observes the same set of events, whereas a dynamic observer can modify the set of events it wishes to observe during the course of the plant execution (this could be implemented by switching sensors

on and off in order to save energy, for example).

In the static observer case, we consider both the standard setting of observable/unobservable events as well as the setting where the observer is defined as a *mask* which allows some events to be observable but not *distinguishable* (e.g., see [2]). Our first contribution is to show that the problems of *minimizing* the number of observable events (or distinct observable outcomes in case of the mask) are NP-complete. Membership in NP can be easily derived by reducing these problems to the standard diagnosability problem, once a candidate minimal solution is chosen non-deterministically. NP-hardness can be shown using reductions of well-known NP-hard problems, namely, clique and coloring problems in graphs.

In the dynamic observer case, we assume that an observer can decide after each new observation the set of events it is going to watch. As a second contribution, we provide a definition of the *dynamic observer synthesis problem* and then show that computing a *dynamic observer* for a given plant, can be reduced to a *game problem*. We further investigate optimization problems for dynamic observers and define a notion of *cost* of an observer. Finally we show how to compute an optimal observer.

**Related work.** NP-hardness of finding minimum-cardinality sets of observable events so that diagnosability holds under the standard, projection-based setting has been previously reported in [13]. Our result of section 3 can be viewed as an alternative shorter proof of this result. Masks have not been considered in [13]. As we show in section 4 a reduction from the mask version of the problem to the standard version is not straightforward. Thus the result in section 4 is useful and new.

The complexity of finding “optimal” observation masks, i.e. a set that cannot be reduced, has been considered in [7] where it was shown that the problem is NP-hard for general properties. [7] also shows that finding optimal observation masks is polynomial for “mask-monotonic” properties where increasing the set of observable (or distinguishable) events preserves the property in question. Diagnosability is a mask-monotonic property. Notice that optimal observation masks are not the same as minimum-cardinality masks that we consider in our work.

In [4], the authors investigate the problem of computing a minimal-cost strategy that allows to find a subset of the set of observable events s.t. the system is diagnosable. It is assumed that each such subset has a known associated cost, as well as a known a-priori probability for achieving diagnosability.

To our knowledge, the problems of synthesizing dynamic observers for diagnosability, studied in Section 5, have not been addressed previously in the literature.

**Organisation of the paper.** In Section 2 we fix notation and introduce finite automata with faults to model DES. In Section 3 we show NP-completeness of the sensor minimization problem for the standard projection-based observation

setting. In Section 4 we show NP-completeness of the sensor minimization problem for the mask-based setting. In Section 5 we introduce and study dynamic observers. We define dynamic observers and show that the most permissive dynamic observer can be computed as the strategy in a safety 2-player game.

We also define a notion of cost for dynamic observers and show that the cost of a given observer can be computed using Karp’s algorithm. Finally, we define the optimal-cost observer synthesis problem and show it can be solved using Zwick and Paterson’s result on graph games.

## 2 Preliminaries

### 2.1 Words and Languages

Let  $\Sigma$  be a finite alphabet and  $\Sigma^\varepsilon = \Sigma \cup \{\varepsilon\}$ .  $\Sigma^*$  is the set of finite words over  $\Sigma$  and contains  $\varepsilon$  which is also the empty word. A *language*  $L$  is any subset of  $\Sigma^*$ .  $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$ . Given two words  $\rho, \rho'$  we denote  $\rho.\rho'$  the concatenation of  $\rho$  and  $\rho'$  (which is defined in the usual way).  $|\rho|$  stands for the length of the word  $\rho$  and  $|\rho|_\lambda$  with  $\lambda \in \Sigma$  stands for the number of occurrences of  $\lambda$  in  $\rho$ .

If  $\rho = a_1 \cdots a_n \in \Sigma^*$  and  $0 \leq i \leq |\rho|$ , we let  $\rho(i) = a_1 \cdots a_i$  be the prefix of  $\rho$  up to  $a_i$  ( $\rho(0) = \varepsilon$ ).

Given  $\Sigma_1 \subseteq \Sigma$ , we define the *projection*  $\pi_{/\Sigma_1} : \Sigma^* \rightarrow \Sigma_1^*$  by:  $\pi_{/\Sigma_1}(\varepsilon) = \varepsilon$  and for  $a \in \Sigma, \rho \in \Sigma^*$ ,  $\pi_{/\Sigma_1}(a.\rho) = a.\pi_{/\Sigma_1}(\rho)$  if  $a \in \Sigma_1$  and  $\pi_{/\Sigma_1}(\rho)$  otherwise.

Let  $f \notin \Sigma^\varepsilon$  be a fresh letter that corresponds to the fault action. We use the notations:  $\Sigma^{\varepsilon,f} = \Sigma^\varepsilon \cup \{f\}$ ,  $\Sigma^f = \Sigma \cup \{f\}$ . A word  $\rho = \rho_1.f.\rho_2$  is *k-faulty* if  $|\rho_2| \geq k$ . A word  $\rho$  is *faulty* if it is *k-faulty* for  $k \in \mathbb{N}$ . Otherwise it is *non faulty*.

### 2.2 Finite Automata

An *automaton*  $A$  is a tuple<sup>1</sup>  $(Q, q_0, \Sigma^{\varepsilon,f}, \rightarrow)$  with  $Q$  a set of states,  $q_0 \in Q$  the initial state,  $\rightarrow \subseteq Q \times \Sigma^{\varepsilon,f} \times Q$  the transition relation. We write  $q \xrightarrow{\lambda} q'$  if  $(q, \lambda, q') \in \rightarrow$ . We let  $en(q)$  be the set of *enabled* labels  $a$  in  $\Sigma$  s.t. there is some  $q'$  with  $q \xrightarrow{a} q'$ . If  $Q$  is finite  $A$  is a *finite automaton*. A *run*  $\rho$  of  $A$  from state  $s$  is a sequence of transitions  $s_0 \xrightarrow{\lambda_1} s_1 \xrightarrow{\lambda_2} s_2 \cdots s_{n-1} \xrightarrow{\lambda_n} s_n$  s.t.  $\lambda_i \in \Sigma^{\varepsilon,f}$  and  $s_0 = s$ . The set of runs from  $s$  in  $A$  is denoted  $Runs(s, A)$  and we define  $Runs(A) = Runs(q_0, A)$ .

The *trace* of the run  $\rho$ , denoted  $tr(\rho)$ , is the word obtained by concatenating the symbols  $\lambda_i$  appearing in  $\rho$ , for those  $\lambda_i$  different from  $\varepsilon$ . A word  $w$  is *accepted* by  $A$  if  $w = tr(\rho)$  for some  $\rho \in Runs(A)$ . The *language*  $\mathcal{L}(A)$  of  $A$  is the set of words accepted by  $A$ . A run  $\rho$  of  $A$  is *faulty* if  $tr(\rho)$  is a faulty word. Otherwise it is non faulty. We define  $Faulty(A) \subseteq (\Sigma^f)^*$  to be the set of faulty words accepted by  $A$  and the non faulty words are the complement:  $NonFaulty(A) =$

<sup>1</sup>In this paper we only use finite automata that generate prefix-closed languages, hence we do not need to use a set of final or accepting states.

$\mathcal{L}(A) \setminus \text{Faulty}(A)$ . For  $k \in \mathbb{N}$ ,  $\text{Faulty}_{\geq k}(A)$  is the set of  $k$ -faulty words of  $\mathcal{L}(A)$ . It follows that  $\text{Faulty}_{\geq k+1}(A) \subseteq \text{Faulty}_{\geq k}(A) \subseteq \dots \subseteq \text{Faulty}_{\geq 0}(A) = \text{Faulty}(A)$ .

We assume that each run of  $A$  of length  $n$  can be extended into a run of length  $n+1$ . This is required for technical reasons and can be achieved by adding  $\varepsilon$  loop transitions to each deadlock state of  $A$ . Notice that this transformation does not change the observations produced by the plant, thus, any observer synthesized for the transformed plant also applies to the original one.

### 2.3 Product of Automata

Let  $A_1 = (Q_1, q_0^1, \Sigma_1, \rightarrow_1)$  and  $A_2 = (Q_2, q_0^2, \Sigma_2, \rightarrow_2)$ . The *product* of  $A_1$  and  $A_2$  is the automaton  $A_1 \times A_2 = (Q, q_0, \Sigma, \rightarrow)$  where:

- $Q = Q_1 \times Q_2$ ,
- $q_0 = (q_0^1, q_0^2)$ ,
- $\Sigma = \Sigma_1 \cup \Sigma_2$ ,
- $\rightarrow \subseteq Q \times \Sigma \times Q$  is defined by  $(q_1, q_2) \xrightarrow{\sigma} (q'_1, q'_2)$  if:
  - either  $\sigma \in \Sigma_i \setminus \Sigma_{3-i}$  and  $q_i \xrightarrow{\sigma} q'_i$  and  $q'_{3-i} = q_{3-i}$  or
  - $\sigma \in \Sigma_1 \cap \Sigma_2$  and  $q_k \xrightarrow{\sigma} q'_k$  for  $k = 1, 2$ .

## 3 Sensor Minimization with Static Observers

In this section we address the sensor minimization problem for *static observers*. We point out that the result in this section was already obtained in [13] and we only give here an alternative shorter proof. We are given a finite automaton  $A = (Q, q_0, \Sigma^{\varepsilon, f}, \rightarrow)$ . We want to decide whether there is a subset  $\Sigma_o \subsetneq \Sigma$  such that the faults can be detected by observing only events in  $\Sigma_o$ . Moreover, we would like to find an “optimal” such  $\Sigma_o$ .

A *diagnoser* is a device that observes the plant and raises an “alarm” whenever it detects a fault. We allow the diagnoser to raise an alarm not necessarily immediately after the fault occurs, but possibly some time later, as long as this time is bounded by some  $k \in \mathbb{N}$ , where  $\mathbb{N}$  is the set of non-negative integers. We model time by counting the “moves” the plant makes (including observable and unobservable ones). More If the system generates a word  $\rho$  but only a subset  $\Sigma_o \subseteq \Sigma$  is observable, the diagnoser can only see  $\pi_{/\Sigma_o}(\rho)$ .

**Definition 1 (( $\Sigma_o, k$ )-Diagnoser)** Let  $A$  be a finite automaton over  $\Sigma^{\varepsilon, f}$ ,  $k \in \mathbb{N}$ ,  $\Sigma_o \subseteq \Sigma$ . A mapping  $D : \Sigma_o^* \rightarrow \{0, 1\}$  is a  $(\Sigma_o, k)$ -diagnoser for  $A$  if:

- for each  $\rho \in \text{NonFaulty}(A)$ ,  $D(\pi_{/\Sigma_o}(\rho)) = 0$ ,
- for each  $\rho \in \text{Faulty}_{\geq k}(A)$ ,  $D(\pi_{/\Sigma_o}(\rho)) = 1$ . ■

$A$  is  $(\Sigma_o, k)$ -diagnosable if there is a  $(\Sigma_o, k)$ -diagnoser for  $A$ .  $A$  is  $\Sigma_o$ -diagnosable if there is some  $k \in \mathbb{N}$  s.t.  $A$  is  $(\Sigma_o, k)$ -diagnosable.

**Example 1** Let  $\mathcal{A}$  be the automaton shown on Fig. 1. The run with one action  $f$  is in  $Faulty_{\geq 0}(\mathcal{A})$ , the run  $f.a$  is in  $Faulty_{\geq 1}(\mathcal{A})$  and  $a.\varepsilon^2$  is in  $NonFaulty(\mathcal{A})$ .  $\mathcal{A}$  is neither  $\{a\}$ -diagnosable, nor  $\{b\}$ -diagnosable. This is because, for any  $k$ ,

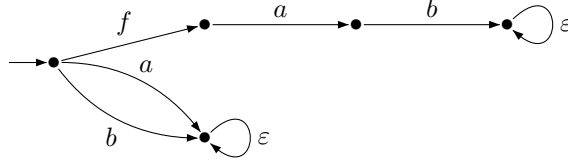


Figure 1: The automaton  $\mathcal{A}$

the faulty run  $f.a.b.\varepsilon^k$  gives the same observation as the non-faulty run  $a.\varepsilon^k$  (in case  $a$  is the observable event) or the non-faulty run  $b.\varepsilon^k$  (in case  $b$  is the observable event). Consequently, the diagnoser cannot distinguish between the two no matter how long it waits. If both  $a$  and  $b$  are observable, however, then we can define:  $D(a.b.\rho) = 1$  for any  $\rho \in \{a, b\}^*$  and  $D(\rho) = 0$  otherwise.  $D$  is a  $(\{a, b\}, 2)$ -diagnoser for  $\mathcal{A}$ .

For given  $A$  and  $\Sigma_o$  it is known how to check diagnosability and build a diagnoser (e.g., see [10]). Checking whether  $A$  is  $\Sigma_o$ -diagnosable can be done in polynomial time in the size of  $A$ , more precisely in  $O(|A|^2)$ . Computing the minimum  $k$  s.t.  $A$  is  $(\Sigma_o, k)$ -diagnosable can be done in  $O(|A|^3)$ . Moreover in case  $A$  is  $\Sigma_o$ -diagnosable, there is a diagnoser  $D$  that can be represented by a finite automaton. Computing this finite automaton is in  $O(2^{|A|})$ . Algorithms for solving these problems are given in appendix A. These algorithms use the fact that  $A$  is  $(\Sigma_o, k)$ -diagnosable iff

$$\pi_{/\Sigma_o}(Faulty_{\geq k}(A)) \cap \pi_{/\Sigma_o}(NonFaulty(A)) = \emptyset \quad (1)$$

or in other words, there is no pair of runs  $(\rho_1, \rho_2)$  with  $\rho_1 \in Faulty_{\geq k}(A)$ ,  $\rho_2 \in NonFaulty(A)$  s.t.  $\rho_1$  and  $\rho_2$  give the same observations on  $\Sigma_o$ .

In this section we address the problem of *finding* a set of observable events  $\Sigma_o$  that allows faults to be detected. We would like to detect faults using as few observable events as possible.

### Problem 1 (Minimum Number of Observable Events)

INPUT:  $A$ ,  $n \in \mathbb{N}$  s.t.  $n \leq |\Sigma|$ .

PROBLEMS:

(A) Is there any  $\Sigma_o \subseteq \Sigma$  with  $|\Sigma_o| = n$ , such that  $A$  is  $\Sigma_o$ -diagnosable ?

(B) If the answer to (A) is “yes”, find the minimum  $n_0$  such that there exists  $\Sigma_o \subseteq \Sigma$  with  $|\Sigma_o| = n_0$  and  $A$  is  $\Sigma_o$ -diagnosable.

If we know how to solve Problem 1(A) efficiently then we can also solve Problem 1(B) efficiently: we perform a binary search over  $n$  between 0 and  $|\Sigma|$ , and solve Problem 1(A) for each such  $n$ , until we find the minimum  $n_0$  for which Problem 1(A) gives a positive answer.<sup>2</sup> Unfortunately, Problem 1(A) is a combinatorial problem, exponential in  $|\Sigma|$ , as we show next.

**Theorem 1** *Problem 1(A) is NP-complete.*

**Proof:** Membership in NP is proved using the result in appendix A: if we guess a solution  $\Sigma_o$  we can check that  $A$  is  $\Sigma_o$ -diagnosable in time polynomial in  $|A|$ . Here we provide the proof of NP-hardness by giving a reduction of the  $n$ -clique problem. Let  $G = (V, E)$  be a (undirected) graph where  $V$  is set of vertices and  $E \subseteq V \times V$  is a set of edges (we assume that  $(v, v') \in E \iff (v', v) \in E$ ). A *clique* in  $G$  is a subset  $V' \subseteq V$  such that for all  $(v, v') \in V'$ ,  $(v, v') \in E$ . The  $n$ -clique problem asks the following: determine whether  $G$  contains a clique of  $n$  vertices.

The reduction is as follows. Given  $G$ , we build a finite automaton  $A_G$  such that there is a  $n$ -clique in  $G$  iff  $A_G$  is  $\Sigma_o$ -diagnosable, where  $|\Sigma \setminus \Sigma_o| = n$ . Notice that since  $\Sigma_o \subseteq \Sigma$ ,  $|\Sigma \setminus \Sigma_o| = n$  is equivalent to  $|\Sigma_o| = |\Sigma| - n$ . Thus, there is a  $n$ -clique in  $G$  iff  $n$  events from  $\Sigma$  do not need to be observed i.e. iff Problem 1(A) gives a positive answer for  $|\Sigma| - n$ .

Starting from  $G$  we define  $\Sigma = V$ . Then we define  $A_G$  as shown in Fig. 2:  $q_0$  is the initial state and the “branches”  $f.a.b$ ,  $f.a'.b'$ ,  $\dots$  are obtained from the pairs of nodes  $(a, b), (a', b'), \dots$  that are *not* linked with an edge in  $G$ :  $(a, b), (a', b'), \dots \notin E$ .

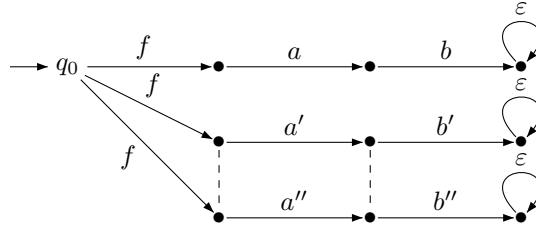


Figure 2: The automaton  $A_G$

**If part.** Assume Problem 1(A) gives a positive answer for  $|\Sigma| - n$ . This means that there exists  $\Sigma_o \subseteq \Sigma$  such that  $|\Sigma_o| = |\Sigma| - n$ , or  $|\Sigma \setminus \Sigma_o| = n$ , and  $A_G$  is  $\Sigma_o$ -diagnosable. Then we claim that  $C = \Sigma \setminus \Sigma_o$  is a clique in  $G$ . Assume  $C$  is not a clique. Then there must be some  $\{a, b\} \subseteq C$  such that  $(a, b) \notin E$ . By the construction of Fig. 2, there is a branch  $f.a.b$  in  $A_G$  and thus both  $\rho = \varepsilon$  and

<sup>2</sup>Notice that knowing  $n_0$  does not imply we know the required set of observable events  $\Sigma_o$ ! We can find (one of the possibly many)  $\Sigma_o$  by searching over all possible subsets  $\Sigma_o$  of  $\Sigma$  of size  $n_0$  (there are  $C(|\Sigma|, n_0)$  such combinations) and check for each such  $\Sigma_o$  whether  $A$  is  $\Sigma_o$ -diagnosable, using the methods described in Appendix A.

$\rho' = f.a.b.\varepsilon^k$  are runs of  $A_G$ , for any  $k$ . As  $\{a, b\} \subseteq C$ ,  $\{a, b\} \cap \Sigma_o = \emptyset$  and the observation of both runs  $\rho$  and  $\rho'$  is  $\varepsilon$ , no diagnoser exists that can distinguish between the two runs, for any  $k$ .

**Only if part.** Assume there exists a  $n$ -clique in  $G$ . Let  $\Sigma_o = \Sigma \setminus C$ . We claim that  $A_G$  is  $(\Sigma_o, 2)$ -diagnosable (thus also  $\Sigma_o$ -diagnosable). Suppose not. Then there exist two runs  $\rho$  and  $\rho'$  such that  $\rho' \in \text{NonFaulty}(A_G)$  and  $\rho = \rho_1.f.\rho_2$ , and  $|\rho_2| \geq 2$ , and  $\pi_{/\Sigma_o}(\rho) = \pi_{/\Sigma_o}(\rho')$ . Then,  $\rho$  must be of the form  $\rho = f.a.b.\varepsilon^k$ , with  $(a, b) \notin E$ . Also, the only way  $\rho'$  can be non faulty is  $\rho' = \varepsilon$ . Then  $\pi_{/\Sigma_o}(\rho') = \varepsilon = \pi_{/\Sigma_o}(\rho)$ , thus, both  $a$  and  $b$  must be non-observable, thus  $\{a, b\} \cap \Sigma_o = \emptyset$  which entails  $\{a, b\} \subseteq C$ . This implies that  $(a, b) \in E$  which is a contradiction. ■

## 4 Sensor Minimization with Masks

So far we have assumed that observable events are also *distinguishable*. However, there are cases where two events  $a$  and  $b$  are observable but not distinguishable, that is, the diagnoser knows that  $a$  or  $b$  occurred, but not which of the two. This is not the same as considering  $a$  and  $b$  to be unobservable, since in that case the diagnoser would not be able to detect occurrence of  $a$  or  $b$ .

Distinguishability of events is captured by the notion of a *mask*.

**Definition 2 (Mask)** A mask  $(M, n)$  over  $\Sigma$  is a total, surjective function  $M : \Sigma \rightarrow \{1, \dots, n\} \cup \{\varepsilon\}$ . ■

$M$  induces a morphism  $M^* : \Sigma^* \rightarrow \{1, \dots, n\}^*$ . For example, if  $\Sigma = \{a, b, c, d\}$ ,  $n = 2$  and  $M(a) = M(b) = 1$ ,  $M(c) = 2$ ,  $M(d) = \varepsilon$ , then we have  $M^*(a.b.c.b.d) = 1.1.2.1 = M^*(a.a.d.c.a)$ .

**Definition 3  $((M, n), k)$ -diagnoser** Let  $(M, n)$  be a mask over  $\Sigma$ . A mapping  $D : \{1, \dots, n\}^* \rightarrow \{0, 1\}$  is a  $((M, n), k)$ -diagnoser for  $A$  if:

- for each  $\rho \in \text{NonFaulty}(A)$ ,  $D(M^*(\rho)) = 0$ ;
- for each  $\rho \in \text{Faulty}_{\geq k}(A)$ ,  $D(M^*(\rho)) = 1$ . ■

$A$  is  $((M, n), k)$ -diagnosable if there is a  $((M, n), k)$ -diagnoser for  $A$ .  $A$  is said to be  $(M, n)$ -diagnosable if there is some  $k$  such that  $A$  is  $((M, n), k)$ -diagnosable.

Given  $A$  and a mask  $(M, n)$ , checking whether  $A$  is  $((M, n)$ -diagnosable can be done in polynomial time. In fact it can be reduced to checking  $\Sigma_o$ -diagnosability of a modified automaton  $A_M$ , with  $\Sigma_o = \{1, \dots, n\}$ .  $A_M$  is obtained from  $A$  by renaming the actions  $a \in \Sigma$  by  $M(a)$ . It can be seen that  $A$  is  $((M, n)$ -diagnosable iff  $A_M$  is  $\{1, \dots, n\}$ -diagnosable. Also notice that  $A$  is  $((M, n), k)$ -diagnosable iff  $M^*(\text{Faulty}_{\geq k}(A)) \cap M^*(\text{NonFaulty}(A)) = \emptyset$ .

As in the previous section, we are mostly interested in minimizing the observability requirements while maintaining diagnosability. In the context of

diagnosis with masks, this means minimizing the number  $n$  of distinct outputs of the mask  $M$ . We thus define the following problem:

**Problem 2 (Minimum Mask)**

INPUT:  $A, n \in \mathbb{N}$  s.t.  $n \leq |\Sigma|$ .

PROBLEM:

- (A) *Is there any mask  $(M, n)$  such that  $A$  is  $(M, n)$ -diagnosable ?*
- (B) *If the answer to (A) is “yes”, find the minimum  $n_0$  such that there is a mask  $(M, n_0)$  such that  $A$  is  $(M, n_0)$ -diagnosable.*

As with Problem 1, if we know how to solve Problem 2(A) efficiently we also know how to solve Problem 2(B) efficiently: again, a binary search on  $n$  suffices.

We will prove that Problem 2 is NP-complete. One might think that this result follows easily from Theorem 1. However, this is not the case. Obviously, a solution to Problem 1 provides a solution to Problem 2: assume there exists  $\Sigma_o$  such that  $A$  is  $(\Sigma_o, k)$ -diagnosable and  $\Sigma_o = \{a_1, \dots, a_n\}$ ; define a mask  $M : \Sigma \rightarrow \{1, \dots, n\}$  such that  $M(a_i) = i$  and for any  $a \in \Sigma \setminus \Sigma_o$ ,  $M(a) = \varepsilon$ . Then,  $A$  is  $((M, n), k)$ -diagnosable. However, a positive answer to Problem 2(A) does not necessarily imply a positive answer to Problem 1(A), as shown by the example that follows.

**Example 2** *Consider again the automaton  $\mathcal{A}$  of Fig. 1. Let  $M(a) = M(b) = 1$ . Then  $\mathcal{A}$  is  $((M, 1), 2)$ -diagnosable because we can build a diagnoser  $D$  defined by:  $D(\varepsilon) = 0, D(1) = 0, D(1^2.\rho) = 1$  for any  $\rho \in 1^*$ . However, as we said before, there is no strict subset of  $\{a, b\}$  that allows  $A$  to be diagnosed.<sup>3</sup>*

Instead of using the previous result, we provide a direct proof of NP-hardness of Problem 2.

**Theorem 2** *Problem 2 is NP-complete.*

**Proof:** Membership in NP is again justified by the fact that checking whether a guessed mask works can be done in polynomial time (it suffices to rename the events of the system according to  $M$  and apply the algorithm of appendix A). We show NP-hardness using a reduction of the  $n$ -coloring problem. The  $n$ -coloring problem asks the following: given an undirected graph  $G = (V, E)$ , is it possible to color the vertices with colors in  $\{1, 2, \dots, n\}$  so that no two adjacent vertices have the same color ?

Let  $G = (V, E)$  be an undirected graph. Let  $E = \{e_1, e_2, \dots, e_j\}$  be the set of edges with  $e_i = (u_i, v_i)$ . We let  $\Sigma = V$  and define the automaton  $A_G$  as pictured in Fig. 3. The initial state of  $A_G$  is  $q_0$ .

We claim that  $G$  is  $n$ -colorizable iff  $A_G$  is  $(M, n)$ -diagnosable.

---

<sup>3</sup> Note that automaton  $\mathcal{B}$  of Fig. 4, although looking very similar to  $\mathcal{A}$ , is not  $(M, 1)$ -diagnosable.

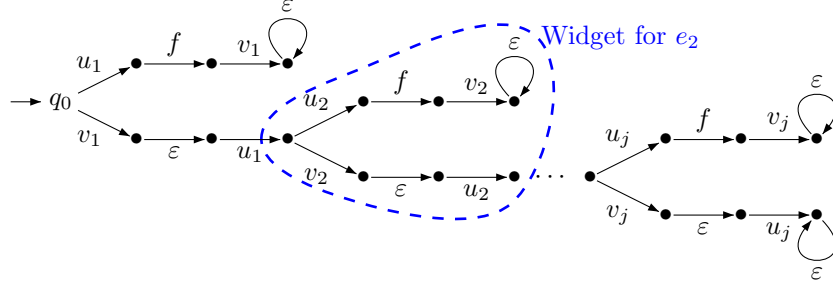


Figure 3: Automaton  $A_G$  for  $n$ -colorizability

**If part.** Assume  $A_G$  is  $(M, n)$ -diagnosable for  $n \geq 0$ . We first show that for all  $i = 1, \dots, j$ ,  $M(u_i) \neq \varepsilon$ ,  $M(v_i) \neq \varepsilon$  and  $M(u_i) \neq M(v_i)$ . For any  $k$ , we can define  $\rho = v_1.\varepsilon.u_1 \cdots u_i.f.v_i.\varepsilon^k$  and  $\rho' = v_1.\varepsilon.u_1 \cdots v_i.\varepsilon.u_i$ . If either  $M(u_i) = \varepsilon$  or  $M(v_i) = \varepsilon$  or  $M(u_i) = M(v_i)$  holds, then  $M^*(\pi_{/\Sigma}(\rho)) = M^*(\pi_{/\Sigma}(\rho'))$ . This way for any  $k$ , there is a faulty run of length with more than  $k$  events after the fault, and a non-faulty run which gives the same observation through the mask. Hence  $A$  cannot be  $((M, n), k)$ -diagnosable for any  $k$  and thus  $A$  is not  $(M, n)$ -diagnosable which contradicts diagnosability of  $A$ .

Note that the above implies in particular that  $n \geq 1$ . We can now prove that  $G$  is  $n$ -colorizable. Let  $C$  be the color mapping defined by  $C(v) = M(v)$ . We need to prove that  $C(u_i) \neq C(v_i)$  for any  $(u_i, v_i) \in E$ . This holds by construction of  $A_G$  and the fact that  $M(u_i) \neq M(v_i)$  as shown above.

**Only if part.** Assume  $G$  is  $n$ -colorizable. There exists a color mapping  $C : V \rightarrow \{1, 2, \dots, n\}$  s.t. if  $(v, v') \in E$  then  $C(v) \neq C(v')$ . Define the mask  $M$  by  $M(a) = C(a)$  for  $a \in V$ . We claim that  $A_G$  is  $((M, n), 1)$ -diagnosable (thus, also  $(M, n)$ -diagnosable). Assume on the contrary that  $A_G$  is not  $((M, n), 1)$ -diagnosable. Then there exist two words  $\rho \in \text{Faulty}_{\geq 1}(A_G)$  and  $\rho' \in \text{NonFaulty}(A_G)$  such that  $M(\pi_{/\Sigma}(\rho)) = M(\pi_{/\Sigma}(\rho'))$ . As  $\rho$  is faulty it must be of the form  $\rho = v_1.\varepsilon.u_1 \cdots u_i.f.v_i.\varepsilon^k$  with  $1 \leq i \leq j$  and  $k \geq 0$ . Notice that  $M(a) \neq \varepsilon$  for all  $a \in V$ . Therefore,  $M(\pi_{/\Sigma}(\rho)) = M(v_1).M(u_1) \cdots M(u_i).M(v_i)$  and  $|M(\pi_{/\Sigma}(\rho))| = 2i$ . Consequently,  $|M(\pi_{/\Sigma}(\rho'))| = 2i$ . The only possible such  $\rho'$  which is also non-faulty is  $\rho' = v_1.\varepsilon.u_1 \cdots v_i.\varepsilon.u_i$ . Now,  $M(\pi_{/\Sigma}(\rho)) = M(\pi_{/\Sigma}(\rho'))$ , which implies  $M(v_i) = M(u_i)$  i.e.  $C(v_i) = C(u_i)$ . But  $(u_i, v_i) \in E$ , and this contradicts the assumption that  $C$  is a valid coloring of  $G$ . ■

## 5 Sensor Minimization with Dynamic Observers

In this section we introduce *dynamic observers*. To illustrate why dynamic observers can be useful consider the following example.

**Example 3 (Dynamic Observation)** Assume we want to detect faults in automaton  $\mathcal{B}$  of Fig. 4. A static diagnoser that observes  $\Sigma = \{a, b\}$  works, however, no proper subset of  $\Sigma$  can be used to detect faults in  $\mathcal{B}$ . Thus the minimum cardinality of the set of observable events for diagnosing  $\mathcal{B}$  is 2 i.e. a static observer

will have to monitor two events during the execution of the DES. If we want to use a mask, the minimum-cardinality for a mask is 2 as well. This means that an observer will have to be receptive to at least two inputs at each point in time to detect a fault in  $\mathcal{B}$ . One can think of being receptive as switching on a device to sense an event. This consumes energy. We can be more efficient using a dynamic observer, that only turns on sensors when needed, thus saving energy. In the case of  $\mathcal{B}$ , this can be done as follows: in the beginning we only switch on the a-sensor; once an a occurs the a-sensor is switched off and the b-sensor is switched on. Compared to the previous diagnosers we use twice as less energy. Compared to the previous diagnosers we use twice as less energy.

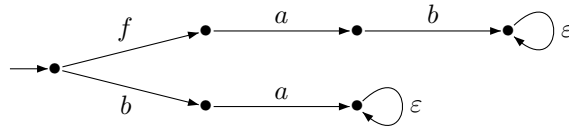


Figure 4: The automaton  $\mathcal{B}$

## 5.1 Dynamic Observers

We formalize the above notion of dynamic observation using *observers*. The choice of the events to observe can depend on the choices the observer has made before and on the observations it has made. Moreover an observer may have *unbounded* memory.

**Definition 4 (Observer)** An observer *Obs* over  $\Sigma$  is a deterministic labeled automaton  $\text{Obs} = (S, s_0, \Sigma, \delta, L)$ , where  $S$  is a (possibly infinite) set of states,  $s_0 \in S$  is the initial state,  $\Sigma$  is the set of observable events,  $\delta : S \times \Sigma \rightarrow S$  is the transition function (a total function), and  $L : S \rightarrow 2^\Sigma$  is a labeling function that specifies the set of events that the observer wishes to observe when it is at state  $s$ . We require for any state  $s$  and any  $a \in \Sigma$ , if  $a \notin L(s)$  then  $\delta(s, a) = s$ : this means the observer does not change its state when an event it chose not to observe occurs. We use the notation  $\delta(s_0, w)$  to denote the state  $s$  reached after reading the word  $w$  and  $L(\delta(s_0, w))$  is the set of events *obs* observes after  $w$ . ■

An observer implicitly defines a *transducer* that consumes an input event  $a \in \Sigma$  and, depending on the current state  $s$ , either outputs  $a$  (when  $a \in L(s)$ ) and moves to a new state  $\delta(s, a)$ , or outputs nothing or  $\varepsilon$ , (when  $a \notin L(s)$ ) and remains in the same state waiting for a new event. Thus, an observer defines a mapping *Obs* from  $\Sigma^*$  to  $\Sigma^*$  (we use the same name “*Obs*” for the automaton and the mapping). Given a run  $\rho$ ,  $\text{Obs}(\rho)$  is the output of the transducer when it reads  $\rho$ . It is called the *observation* of  $\rho$  by *Obs*. We next provide an example of a particular case of observer which can be represented by a finite-state machine.

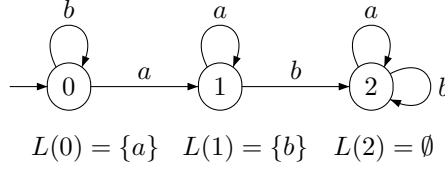


Figure 5: A Finite-State Observer Obs

**Example 4** Let  $Obs$  be the observer of Fig. 5.  $Obs$  maps the following inputs as follows:  $Obs(baab) = ab$ ,  $Obs(bababbaab) = ab$ ,  $Obs(bbbba) = a$  and  $Obs(bbaaa) = a$ . If  $Obs$  operates on the DES  $\mathcal{B}$  of Fig 4 and  $\mathcal{B}$  generates  $f.a.b$ ,  $Obs$  will have as input  $\pi_{/\Sigma}(f.a.b) = a.b$  with  $\Sigma = \{a, b\}$ . Consequently the observation of  $Obs$  is  $Obs(\pi_{/\Sigma}(f.a.b)) = a$ .

## 5.2 Fault Diagnosis with Dynamic Diagnosers

**Definition 5 ((Obs,  $k$ )-diagnoser)** Let  $Obs$  be an observer over  $\Sigma$ . A mapping  $D : \Sigma^* \rightarrow \{0, 1\}$  is a (Obs,  $k$ )-diagnoser for  $A$  if (i)  $\forall \rho \in NonFaulty(\mathcal{A})$ ,  $D(Obs(\rho)) = 0$  and (ii)  $\forall \rho \in Faulty_{\geq k}(\mathcal{A})$ ,  $D(Obs(\rho)) = 1$ . ■

$A$  is (Obs,  $k$ )-diagnosable if there is an (Obs,  $k$ )-diagnoser for  $A$ .  $A$  is Obs-diagnosable if there is some  $k$  such that  $A$  is (Obs,  $k$ )-diagnosable.

If a diagnoser always selects  $\Sigma$  as the set of observable events, it is a static observer and (Obs,  $k$ )-diagnosability amounts to the standard  $(\Sigma, k)$ -diagnosis problem [10]. In this case  $A$  is  $(\Sigma, k)$ -diagnosable iff

$$Faulty_{\geq k}(\mathcal{A}) \cap NonFaulty(\mathcal{A}) = \emptyset.$$

As for  $\Sigma$ -diagnosability, we have the following equivalence for dynamic observers:  $A$  is (Obs,  $k$ )-diagnosable iff  $Obs(Faulty_{\geq k}(\mathcal{A})) \cap Obs(NonFaulty(\mathcal{A})) = \emptyset$ . This follows directly from definition 5.

### Problem 3 (Finite-State Obs-Diagnosability)

INPUT:  $A$ ,  $Obs$  a finite-state observer.

PROBLEM:

(A) Is  $A$  Obs-diagnosable ?

(B) If the answer to (A) is “yes”, compute the minimum  $k$  such that  $A$  is (Obs,  $k$ )-diagnosable.

**Theorem 3** Problem 3 is in P.

**Proof:** We build a product automaton<sup>4</sup>  $A \otimes Obs$  such that:  $A$  is (Obs,  $k$ )-diagnosable  $\iff A \otimes Obs$  is  $(\Sigma, k)$ -diagnosable.

<sup>4</sup>We use  $\otimes$  to clearly distinguish this product from the synchronous product  $\times$ .

**Definition 6** ( $A \otimes \text{Obs}$ ) Let  $A = (Q, q_0, \Sigma^{\varepsilon, f}, \rightarrow)$  be a finite automaton and  $\text{Obs} = (S, s_0, \Sigma, \delta, L)$  be a finite-state observer. We define the automaton  $A \otimes \text{Obs} = (Q \times S, (q_0, s_0), \Sigma^{\varepsilon, f}, \rightarrow)$  as follows:

1.  $(q, s) \xrightarrow{\beta} (q', s')$  iff  $\exists \lambda \in \Sigma$  s.t.  $q \xrightarrow{\lambda} q'$ ,  $s' = \delta(s, \lambda)$  and  $\beta = \lambda$  if  $\lambda \in L(s)$ ,  $\beta = \varepsilon$  otherwise;
2.  $(q, s) \xrightarrow{\lambda} (q', s)$  iff  $\exists \lambda \in \{\varepsilon, f\}$  s.t.  $q \xrightarrow{\lambda} q'$ .

To prove Theorem 3 we lemma 1 and 2:

**Lemma 1** Let  $\rho \in \text{Faulty}_{\geq k}(A)$ . There is a word  $\nu \in \text{Faulty}_{\geq k}(A \otimes \text{Obs})$  s.t.  $\text{Obs}(\rho) = \pi_{/\Sigma}(\nu)$ . Let  $\rho' \in \text{NonFaulty}(A)$ . There is a word  $\nu' \in \text{NonFaulty}(A \otimes \text{Obs})$  s.t.  $\text{Obs}(\rho') = \pi_{/\Sigma}(\nu')$ .

**Proof:** Let  $r = q_0 \xrightarrow{a_1} q_1 \cdots q_{n-1} \xrightarrow{a_n} q_n$  be in  $\text{Runs}(A)$ . By definition of  $A \otimes \text{Obs}$ , the run  $\tilde{r} = (q_0, s_0) \xrightarrow{b_1} (q_1, s_1) \cdots (q_{n-1}, s_{n-1}) \xrightarrow{b_n} (q_n, s_n)$  with for  $1 \leq i \leq n$ :

- if  $a_i \in \{\varepsilon, f\}$ ,  $b_i = a_i$  and  $s_i = s_{i-1}$ ;
- if  $a_i \in L(s_{i-1})$ ,  $b_i = a_i$  and  $s_i = \delta(s_{i-1}, a_i)$ ;
- if  $a_i \notin L(s_{i-1})$ ,  $b_i = \varepsilon$  and  $s_i = s_{i-1}$ ;

is in  $\text{Runs}(A \otimes \text{Obs})$ . Moreover (1)  $a_1 a_2 \cdots a_n \in \text{Faulty}_{\geq k}(A) \iff b_1 b_2 \cdots b_n \in \text{Faulty}_{\geq k}(A \otimes \text{Obs})$  and (2)  $\text{Obs}(a_1 a_2 \cdots a_n) = \pi_{/\Sigma}(b_1 b_2 \cdots b_n)$  which means that  $\text{Obs}(tr(r)) = \pi_{/\Sigma}(tr(\tilde{r}))$ .

Let  $\rho \in \mathcal{L}(A)$ . There is a run  $r \in \text{Runs}(A)$  s.t.  $\rho = tr(r)$ . By (1), if  $\rho \in \text{Faulty}_{\geq k}(A)$ ,  $tr(\tilde{r}) \in \text{Faulty}_{\geq k}(A \otimes \text{Obs})$  and if  $\rho \in \text{NonFaulty}(A)$ ,  $tr(\tilde{r}) \in \text{NonFaulty}(A \otimes \text{Obs})$  and in any case  $\text{Obs}(\rho) = \pi_{/\Sigma}(tr(\tilde{r}))$  by (2). ■

**Lemma 2** Let  $\nu \in \text{Faulty}_{\geq k}(A \otimes \text{Obs})$ . There is a word  $\rho \in \text{Faulty}_{\geq k}(A)$  s.t.  $\text{Obs}(\rho) = \pi_{/\Sigma}(\nu)$ . Let  $\nu' \in \text{NonFaulty}(A \otimes \text{Obs})$ . There is a word  $\rho' \in \text{NonFaulty}(A)$  s.t.  $\text{Obs}(\rho') = \pi_{/\Sigma}(\nu')$ .

**Proof:** This proof is the counterpart of the proof of Lemma 1. Let  $r = (q_0, s_0) \xrightarrow{b_1} (q_1, s_1) \cdots (q_{n-1}, s_{n-1}) \xrightarrow{b_n} (q_n, s_n)$  be in  $\text{Runs}(A \otimes \text{Obs})$ . Define a run  $\tilde{r} = q_0 \xrightarrow{a_1} q_1 \cdots q_{n-1} \xrightarrow{a_n} q_n$  with  $1 \leq i \leq n$ :

- if  $b_i = f$ , then  $a_i = f$ ,
- if  $b_i \in \Sigma$ , then  $a_i = b_i$ ,
- if  $b_i = \varepsilon$ , choose some  $a_i \in \{\varepsilon\} \cup \Sigma \setminus L(s_{i-1})$  s.t.  $q_{i-1} \xrightarrow{a_i} q_i$ .

$\tilde{r}$  is a run in  $\text{Runs}(A)$  and  $\text{Obs}(tr(\tilde{r})) = \pi_{/\Sigma}(tr(r))$ . It is the easy to see that Lemma 2 holds by. ■

Now assume  $A$  is  $(\text{Obs}, k)$ -diagnosable and  $A \otimes \text{Obs}$  is not  $(\Sigma, k)$ -diagnosable. There are two words  $\nu \in \text{Faulty}_{\geq k}(A \otimes \text{Obs})$  and  $\nu' \in \text{NonFaulty}(A \otimes \text{Obs})$  s.t.

$\pi_{/\Sigma}(\nu) = \pi_{/\Sigma}(\nu')$ . By Lemma 2, there are two words  $\rho \in \text{Faulty}_{\geq k}(A)$  and  $\rho' \in \text{NonFaulty}(A)$  s.t.  $\text{Obs}(\rho) = \pi_{/\Sigma}(\nu) = \pi_{/\Sigma}(\nu') = \text{Obs}(\rho')$  and thus  $A$  is not  $(\text{Obs}, k)$ -diagnosable.

Assume  $A \otimes \text{Obs}$  is  $(\Sigma, k)$ -diagnosable and  $A$  is not  $(\text{Obs}, k)$ -diagnosable. There are two words  $\rho \in \text{Faulty}_{\geq k}(A)$  and  $\rho' \in \text{NonFaulty}(A)$  with  $\text{Obs}(\rho) = \text{Obs}(\rho')$ . By Lemma 1, there are two words  $\nu \in \text{Faulty}_{\geq k}(A \otimes \text{Obs})$  and  $\nu' \in \text{NonFaulty}(A \otimes \text{Obs})$  s.t.  $\text{Obs}(\rho) = \pi_{/\Sigma}(\nu)$  and  $\pi_{/\Sigma}(\nu') = \text{Obs}(\rho')$  and thus  $\pi_{/\Sigma}(\nu)$  and  $\pi_{/\Sigma}(\nu')$ . This would imply that  $A \otimes \text{Obs}$  is not  $(\Sigma, k)$ -diagnosable.

The number of states of  $A \otimes \text{Obs}$  is at most  $|Q| \cdot |S|$  and the number of transitions is bounded by the number of transitions of  $A$ . Hence the size of the product is polynomial in the size of the input  $|A| + |\text{Obs}|$ . Checking that  $A \otimes \text{Obs}$  is diagnosable can be done in polynomial time (Appendix A) thus Problem 3 is in P. This completes the proof. ■

**Example 5** Let  $A$  be the DES given in Fig. 1 and  $\text{Obs}$  the observer of Fig. 5. The product  $A \otimes \text{Obs}$  is given in Fig. 6. Using an algorithm for checking  $\Sigma$ -diagnosability of  $A \otimes \text{Obs}$  we obtain that it is  $(\Sigma, 2)$ -diagnosable (and 2 is the minimum value). Hence  $A$  is  $(\text{Obs}, 2)$ -diagnosable with 2 the minimum value.

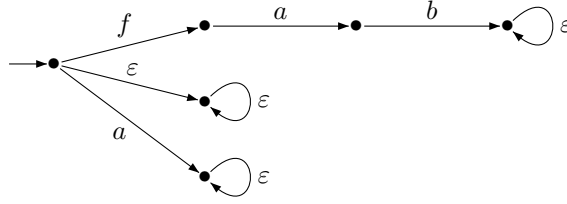


Figure 6: The automaton  $A \otimes \text{Obs}$

For Problem 3, we have assumed that an observer was given. It would be even better if we could *synthesize* an observer  $\text{Obs}$  such that the plant is diagnosable with  $\text{Obs}$ . Before attempting to synthesize such an observer, we should first check that the plant is  $\Sigma$ -diagnosable: if it is not, then obviously no such observer exists; if the plant is  $\Sigma$ -diagnosable, then the trivial observer that observes all events in  $\Sigma$  at all times works<sup>5</sup>. Therefore, we need a way to compute the other diagnosers. Hence we define the problem of computing the set of *all* valid observers.

#### Problem 4 (Dynamic-Diagnosability)

INPUT:  $A$ .

PROBLEM: Compute the set  $\mathcal{O}$  of all observers such that  $A$  is  $\text{Obs}$ -diagnosable iff  $\text{Obs} \in \mathcal{O}$ .

We do not have a solution to the above problem: it can be reduced to finding a trace-based winning strategy for Büchi game with partial observation.

<sup>5</sup> Notice that this also shows that existence of an observer implies existence of a finite-state observer, since the trivial observer is finite-state.

We know how to do this for safety games (Appendix B) but we do not have a solution to solve Büchi games of this type. Instead, we introduce a restricted variant:

**Problem 5 (Dynamic- $k$ -Diagnosability)**

INPUT:  $A$  and  $k \in \mathbb{N}$ .

PROBLEM: Compute the set  $\mathcal{O}$  of observers such that  $A$  is  $(\text{Obs}, k)$ -diagnosable iff  $\text{Obs} \in \mathcal{O}$ .

We now give an algorithm to solve Problem 5.

**5.3 Problem 5 as a Game Problem**

To solve Problem 5 we reduce it to a *safety* 2-player game. The definitions and results for such games are given in appendix B. We also provide an intuitive explanation of such games in this section, as we construct our reduction proof. In short, the reduction we propose is the following:

- Player 1 chooses the set of events it wishes to observe, then it hands over to Player 2 ;
- Player 2 chooses an event and tries to produce a run which is the observation of a  $k$ -faulty run and a non-faulty run.

Player 2 wins if it can produce such a run. Otherwise Player 1 wins. Player 2 has complete information of Player 1’s moves (i.e., it can observe the sets that Player 1 chooses to observe). Player 1, on the other hand, only has partial information of Player 2’s moves because not all events are observable (details follow). Let  $A = (Q, q_0, \Sigma^{\varepsilon, f}, \rightarrow)$  be a finite automaton. To define the game, we use two copies of automaton  $A$ :  $A_1^k$  and  $A_2$ . The accepting states of  $A_1^k$  are those corresponding to runs of  $A$  which are faulty and more than  $k$  steps occurred after the fault.  $A_2$  is a copy of  $A$  where the  $f$ -transitions have been removed. The game we are going to play is the following (see Fig. 7, Player 1 states are depicted with square boxes and Player 2’ states with round shapes):

1. the game starts in an state  $(q_1, q_2)$  corresponding to the initial state of the product of  $A_1^k$  and  $A_2$ . Initially, it is Player 1’s turn to play. Player 1 chooses a set of events it is going to observe i.e. a subset  $X$  of  $\Sigma$  and hands it over to Player 2;
2. assume the automata  $A_1^k$  and  $A_2$  are in states  $(q_1, q_2)$ . Player 2 can change the state of  $A_1^k$  and  $A_2$  by:
  - (a) firing an action which is not in  $X$  in either  $A_1^k$  or  $A_2$  (no synchronization). In this case a new state  $(q, q')$  is reached and Player 2 can play again from this state;

- (b) firing an action  $\lambda$  in  $X$ : to do this both  $A_1^k$  and  $A_2$  must be in a state where  $\lambda$  is possible (synchronization); after the action is fired a new state  $(q'_1, q'_2)$  is reached: now it is Player 1's turn to play, and the game continues as in step 1 above (from the new state  $(q'_1, q'_2)$ ).

Player 2 wins if he can reach a state  $(q_1, q_2)$  in  $A_1^k \times A_2$  where  $q_1$  is an accepting state of  $A_1^k$  (this means that Player 1 wins if it can avoid ad infinitum this set of states). In this sense this is a safety game for Player 1 (and a reachability game for Player 2). Formally, the game  $G_A = (S_1 \uplus S_2, s_0, \Sigma_1 \uplus \Sigma_2, \delta)$  is defined

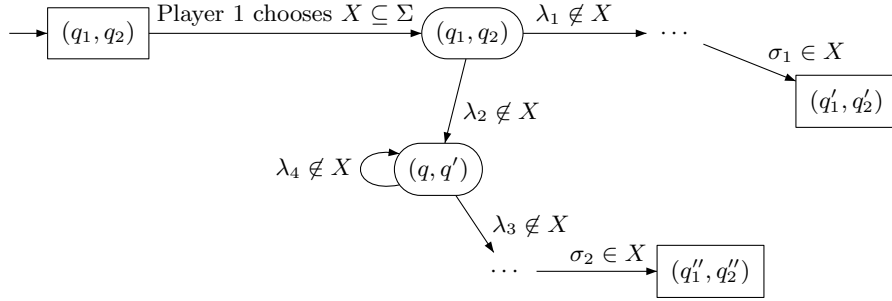


Figure 7: Game Reduction for Problem 5

as follows ( $\uplus$  denotes union of disjoint sets):

- $S_1 = (Q \times \{-1, \dots, k\}) \times Q$  is the set of Player 1 states; a state  $(q_1, j)$ ,  $q_2$  indicates that  $A_1^k$  is in state  $q_1$ ,  $j$  steps have occurred after a fault, and  $q_2$  is the current state of  $A_2$ . If no fault has occurred,  $j = -1$  and if more than  $k$  steps occurred after the fault, we use  $j = k$ .
- $S_2 = (Q \times \{-1, \dots, k\}) \times Q \times 2^\Sigma$  is the set of Player 2 states. For a state  $((q_1, j), q_2, X)$ , the component  $(q_1, j)$ ,  $q_2$  has the same meaning as for  $S_1$ , and  $X$  is the set of moves Player 1 has chosen to observe on its last move.
- $s_0 = ((q_0, -1), q_0)$  is the initial state of the game belonging to Player 1;
- $\Sigma_1 = 2^\Sigma$  is the set of moves of Player 1;  $\Sigma_2 = \Sigma^\varepsilon$  is the set of moves of Player 2 (as we encode the fault into the state, we do not need to distinguish  $f$  from  $\varepsilon$ ).
- the transition relation  $\delta \subseteq (S_1 \times \Sigma_1 \times S_2) \cup (S_2 \times \{\varepsilon\} \times S_2) \cup (S_2 \times \Sigma \times S_1)$  is defined by:

- Player 1 moves: let  $\sigma \in \Sigma_1$  and  $s_1 \in S_1$ . Then  $(s_1, \sigma, (s_1, \sigma)) \in \delta$ .
- Player 2 moves: a move of Player 2 is either a silent move ( $\varepsilon$ ) i.e. a move of  $A_1^k$  or  $A_2$  or a joint move of  $A_1^k$  and  $A_2$  with an observable action in  $X$ . A *silent* move  $((q_1, i), q_2, X), \varepsilon, (q'_1, j), q'_2, X)$  is in  $\delta$  if one of the following conditions holds:

1. either  $q'_2 = q_2$ ,  $q_1 \xrightarrow{\ell} q'_1$  is a step of  $A_1^k$ ,  $\ell \notin X$ , and if  $i \geq 0$  then  $j = \max(i+1, k)$ ; if  $i = -1$  then if  $\ell = f$   $j = 0$  otherwise  $j = i$ .
2. either  $q'_1 = q_1$ ,  $q_2 \xrightarrow{\ell} q'_2$  is a step of  $A_2$ ,  $\ell \notin X$  (and  $\ell \neq f$ ), and if  $i \geq 0$   $j = \max(i+1, k)$ , otherwise  $j = i$ .

A *visible* move can be taken by Player 2 if both  $A_1^k$  and  $A_2$  agree on doing such a move. In this case the game proceeds to a Player 1 state:  $((q_1, i), q_2, X), \ell, ((q'_1, j), q'_2) \in \delta$  if  $\ell \in X$ ,  $q_1 \xrightarrow{\ell} q'_1$  is a step of  $A_1^k$ ,  $q_2 \xrightarrow{\ell} q'_2$  is a step of  $A_2$ , and if  $i \geq 0$   $j = \max(i+1, k)$ , otherwise  $j = i$ .

We can show that for any observer  $O$  s.t.  $A$  is  $(O, k)$ -diagnosable, there is a strategy  $f(O)$  for Player 1 in  $G_A$  s.t.  $f(O)$  is *trace-based* and winning. A *strategy* for Player 1 is a mapping  $f : \text{Runs}(G_A) \rightarrow \Sigma_1$  that associates a move  $f(\rho)$  in  $\Sigma_1$  to each run  $\rho$  of  $G_A$  that ends in a  $S_1$ -state. A strategy  $f$  is trace-based (see appendix B for details), if given two runs  $\rho, \rho'$ , if  $\text{tr}(\rho) = \text{tr}(\rho')$  then  $f(\rho) = f(\rho')$ . Conversely, for any trace-based winning strategy  $f$  (for Player 1), we can build an observer  $O(f)$  s.t.  $A$  is  $(O(f), k)$ -diagnosable.

Let  $O = (S, s_0, \Sigma, \delta, L)$  be an observer for  $A$ . We define the strategy  $f(O)$  on finite runs of  $G_A$  ending in a Player 1 state by:  $f(O)(\rho) = L(\delta(s_0, \pi_{/\Sigma}(\text{tr}(\rho))))$ . The intuition is that we take the run  $\rho$  in  $G_A$ , take the trace of  $\rho$  (choices of Player 1 and moves of Player 2) and remove the choices of Player 1. This gives a run in  $\Sigma^*$ . The strategy for Player 1 for  $\rho$  is the set of events the observer  $O$  chooses to observe after reading  $\pi_{/\Sigma}(\text{tr}(\rho))$  i.e.  $L(\delta(s_0, \pi_{/\Sigma}(\text{tr}(\rho))))$ .

**Theorem 4** *Let  $O$  be an observer s.t.  $A$  is  $(O, k)$ -diagnosable. Then  $f(O)$  is a trace-based winning strategy in  $G_A$ .*

**Proof:** First  $f(O)$  is trace-based by definition. We have to prove that  $f(O)$  is winning. We denote  $\text{Out}(G, f)$  the set of outcomes i.e. the set of possible runs of a game  $G$  when the strategy  $f$  is played by Player 1 (see appendix B for a formal definition of  $\text{Out}(G_A, f)$ ). Assume on the contrary  $f(O)$  is not winning. This implies that there is a run  $\rho$  in  $\text{Out}(G_A, f(O))$  as defined by equations (2-7).

$$\rho = (q_0^1, 0), q_0^2 \xrightarrow{X_0} (q_0^1, 0), q_0^2, X_0 \xrightarrow{\lambda_0^1} (q_0^1(1), k_0(1)), q_0^2(1), X_0 \cdots \quad (2)$$

$$\cdots (q_0^1(j), k_0(j)), q_0^2(j), X_0 \cdots \xrightarrow{\lambda_0^{n_0}} \quad (3)$$

$$(q_1^1, k_1), q_1^2 \xrightarrow{X_1} (q_1^1, k_1), q_1^2, X_1 \xrightarrow{\lambda_1^1} (q_1^1(1), k_1(1)), q_1^2(1), X_1 \cdots \quad (4)$$

$$\cdots (q_1^1(j), k_1(j)), q_1^2(j), X_1 \cdots \xrightarrow{\lambda_1^{n_1}} \quad (5)$$

$$(q_2^1, k_2), q_2^2 \quad \cdots \quad (6)$$

$$(q_n^1, k_n), q_n^2 \xrightarrow{X_n} (q_n^1, k_n), q_n^2, X_n \cdots (q_n^1(j), k_n(j)), q_n^2(j), X_n \cdots \quad (7)$$

$$\cdots \xrightarrow{\lambda_n^\alpha} (q_n^1(\alpha), k_n(\alpha)), q_n^2(\alpha), X_n \quad (8)$$

Each part of the run given by equations (2–7) consists of a choice of Player 1 ( $X_i$  move) followed by a number of moves by Player 1 ( $\lambda_i^j$  actions). The last state encountered in  $\rho$ ,  $(q_n^1(\alpha), k_n(\alpha), q_n^2(\alpha), X_n)$  is a losing state for Player 1, which means that  $k_n(\alpha) \geq k$ , by definition of losing states in  $G_A$ . From the run  $\rho$ , we can build two runs  $\nu$  and  $\nu'$  defined by:

$$\begin{aligned}\nu &= q_0^1 \xrightarrow{\lambda_0^1} q_0^1(1) \xrightarrow{\lambda_0^2} \dots \xrightarrow{\lambda_0^{n_0}} q_1^1 \xrightarrow{\lambda_1^1} \dots \xrightarrow{\lambda_1^{n_1}} q_2^1 \dots q_n^1 \xrightarrow{\lambda_n^1} \dots \xrightarrow{\lambda_n^\alpha} q_n^1(\alpha) \\ \nu' &= q_0^2 \xrightarrow{\lambda_0^1} q_0^2(1) \xrightarrow{\lambda_0^2} \dots \xrightarrow{\lambda_0^{n_0}} q_1^2 \xrightarrow{\lambda_1^1} \dots \xrightarrow{\lambda_1^{n_1}} q_2^2 \dots q_n^2 \xrightarrow{\lambda_n^1} \dots \xrightarrow{\lambda_n^\alpha} q_n^2(\alpha)\end{aligned}$$

By definition of  $G_A$ , each  $\lambda_i^j$  is either a common visible action of  $A_1^k$  and  $A_2$  and it is in  $\Sigma$ , or a silent action ( $\varepsilon$ ) i.e. it comes from an action of  $A_1^k$  or  $A_2$  that is not in the current set of visible actions  $X_i$ . We can remove from  $\nu$  (resp.  $\nu'$ ) the actions  $\varepsilon$  that are obtained from an action of  $A_2$  (resp.  $A_1^k$ ) leaving the state of  $A_1^k$  (resp.  $A_2$ ) unchanged. Let  $\tilde{\nu}$  and  $\tilde{\nu}'$  be the runs obtained this way. By definition of  $G_A$ ,  $tr(\tilde{\nu}) \in \text{Faulty}_{\geq k}(A)$  and  $tr(\tilde{\nu}') \in \text{NonFaulty}(A)$ . We claim that  $O(tr(\tilde{\nu})) = O(tr(\tilde{\nu}'))$ . Indeed, each part of the runs from  $q_i^1 \dots q_{i+1}^1$  and  $q_i^2 \dots q_{i+1}^2$  yields the same observation by  $O$ : it is the sequence of events  $\lambda_{j_1} \dots \lambda_{j_{n_i}}$  s.t. each  $\lambda_{j_l}$  is a letter of both  $A_1^k$  and  $A_2$  and is  $X_i$ . As there are two words  $tr(\tilde{\nu}) \in \text{Faulty}_{\geq k}(A)$  and  $tr(\tilde{\nu}') \in \text{NonFaulty}(A)$  with the same observation,  $A$  is not  $(O, k)$ -diagnosable which contradicts the assumption. Hence  $f(O)$  must be winning.  $\blacksquare$

Conversely, with each trace-based strategy  $f$  of the game  $G_A$  we can associate a transition system  $O(f) = (S, s_0, \Sigma, \delta, L)$  defined by:

- $S = \{\pi_{/\Sigma}(tr(\rho)) \mid \rho \in \text{Out}(G_A, f) \text{ and } tgt(\rho) \in S_1\}$ ;
- $s_0 = \varepsilon$ ;
- $\delta(v, \ell) = v'$  if  $v \in S$ ,  $v' = v.\ell$  and there is a run  $\rho \in \text{Out}(G_A, f)$  with  $\rho = q_0 \xrightarrow{X_0} q_0^1 \xrightarrow{\varepsilon^*} q_0^{n_0} \xrightarrow{\lambda_1} q_1 \xrightarrow{X_1} q_1^1 \xrightarrow{\varepsilon^*} q_1^{n_1} \xrightarrow{\lambda_2} q_2 \dots q_{k_1} \xrightarrow{\varepsilon^*} q_{k_1}^{n_{k-1}} \xrightarrow{\lambda_k} q_k$  with each  $q_i \in S_1$ ,  $q_i^j \in S_2$ ,  $v = \pi_{/\Sigma}(tr(\rho))$ , and  $\rho' = \rho \xrightarrow{X_k} q_k^1 \xrightarrow{\varepsilon^*} q_k^{n_k} \xrightarrow{\ell} q_{k+1}$  with  $q_{k+1} \in S_1$ ,  $\ell \in X_k$ .  
 $\delta(v, l) = v$  if  $v \in S$  and  $\ell \notin f(\rho)$ ;
- $L(v) = f(\rho)$  if  $v = \pi_{/\Sigma}(tr(\rho))$ .

**Lemma 3**  $O(f)$  is an observer.

**Proof:** We first have to prove that  $O(f)$  (more precisely  $L$ ) is well defined. Assume  $v = \pi_{/\Sigma}(tr(\rho))$  and  $v = \pi_{/\Sigma}(tr(\rho'))$ . As  $f$  is trace-based,  $f(\rho) = f(\rho')$  and there is unique value for  $L(v)$ .

We also have to prove that the last requirement of Definition 4 is satisfied i.e. if  $a \notin L(s)$  then  $\delta(s, a) = s$ . If  $\ell \notin L(v)$ , then  $\ell \notin f(\pi_{/\Sigma}(tr(\rho)))$  for any  $\rho$  s.t.  $v = \pi_{/\Sigma}(tr(\rho))$  because  $f$  is trace-based. Thus  $\delta(v, \ell) = v$ .  $\blacksquare$

**Theorem 5** Let  $f$  be a trace-based winning strategy in  $G_A$ . Then  $A$  is  $(O(f), k)$ -diagnosable.

**Proof:** Assume  $A$  is not  $(O(f), k)$ -diagnosable. There are two words  $\nu \in \text{Faulty}_{\geq k}(A)$  and  $\nu' \in \text{NonFaulty}(A)$  s.t.  $O(\nu) = O(\nu')$ . Assume  $\nu = w_{-1}\lambda_0w_0\lambda_1w_1 \cdots \lambda_nw_n$  and  $\nu' = w'_{-1}\lambda_0w'_0\lambda_1w'_1 \cdots \lambda_nw'_n$  with  $w_i, w'_i \notin O(f)(\lambda_0\lambda_1 \cdots \lambda_i)$  for  $i \geq 0$  and  $w_{-1}, w'_{-1} \notin O(f)(\varepsilon)$ , and  $\lambda_{i+1} \in O(f)(\lambda_0\lambda_1 \cdots \lambda_i)$ . We build a run in  $\text{Out}(G_A, f)$  as follows:

1. Player 1 chooses the set  $X_0 = O(f)(\varepsilon)$  which is by definition equal to  $f((q_0^1, 0), q_0^2)$  where  $(q_0^1, 0), q_0^2$  is the initial state of the game.
2. Player 2 chooses actions in  $w_1 \cup w'_1$ . The game moves through  $S_2$  states because each action is an invisible move. Finally Player 2 chooses  $\lambda_0$ . The game reaches a new  $S_1$ -state  $(q_1^1, k_1), q_1^2$ .
3. from  $(q_1^1, k_1), q_1^2$ , the strategy  $f$  is to play  $X_1$  which by definition is  $O(\lambda_0)$ . Thus Player 2 can play moves in  $w_2 \cup w'_2$  and finally  $\lambda_1$

We can iteratively build a run in  $\text{Out}(G_A, f)$  that reaches a state  $(q_n^1, k_n), q_n^2$  with  $k_n \geq k$  and thus  $\text{Out}(G_A, f)$  contains a losing run. Hence  $f$  is not winning which contradicts the assumption. This way we conclude that  $A$  is  $(O(f), k)$ -diagnosable.  $\blacksquare$

The result on  $G_A$  (Appendix B) is that, if there is a winning trace-based strategy for Player 1, then there is a most permissive strategy  $\mathcal{F}_A$  which has finite memory. It can be represented by a finite automaton  $S_{\mathcal{F}_A} = (W_1 \uplus W_2, s_0, \Sigma \cup 2^\Sigma, \Delta_A)$  s.t.  $\Delta_A \subseteq (W_1 \times 2^\Sigma \times W_2) \cup (W_2 \times \Sigma \times W_1)$  which has size exponential in the size of  $G_A$ . For a given run  $\rho \in (\Sigma \cup 2^\Sigma)^*$  ending in a  $W_1$ -state, we have  $\mathcal{F}_A(w) = \text{en}(\Delta_A(s_0, w))$ .

## 5.4 Most Permissive Observer

We now define the notion of a most *permissive* observer and show the existence of a most permissive observer for a system in case  $A$  is diagnosable.  $\mathcal{F}_A$  is the mapping defined at the end of the previous section.

For an observer  $O = (S, s_0, \Sigma, \delta, L)$  and  $w \in \Sigma^*$  we let  $L(w)$  be the set  $L(\delta(s_0, w))$ : this is the set of events  $O$  chooses to observe on input  $w$ . Given a word  $\rho \in \pi_{/\Sigma}(\mathcal{L}(A))$ , we recall that  $O(\rho)$  is the observation of  $\rho$  by  $O$ . Assume  $O(\rho) = a_0 \cdots a_k$ . Let  $\bar{\rho} = L(\varepsilon).\varepsilon.L(a_0).a_0 \cdots L(O(\rho)(k)).a_k$  i.e.  $\bar{\rho}$  contains the history of what  $O$  has chosen to observe at each step and the events that occurred after each choice.

Let  $\mathcal{O} : (2^\Sigma \times \Sigma^\varepsilon)^+ \rightarrow 2^{2^\Sigma}$ .  $\mathcal{O}$  is the most permissive observer for  $(A, k)$  if the following holds:

$$\begin{aligned} O = (S, s_0, \Sigma, \delta, L) \text{ and is an observer} \\ \text{and } A \text{ is } (O, k)\text{-diagnosable} \end{aligned} \iff \forall w \in \Sigma^* L(\delta(s_0, w)) \in \mathcal{O}(\bar{w})$$

The definition of the most permissive observer states that:

- any good observer Obs (one such that  $A$  is  $(\text{Obs}, k)$ -diagnosable) must choose a set of observable events in  $\mathcal{O}(\bar{w})$  on input  $w$ ;

- if an observer chooses its set of observable events in  $\mathcal{O}(\bar{w})$  on input  $w$ , then it is a good observer.

Assume  $A$  is  $(\Sigma, k)$ -diagnosable. Then there is an observer  $O$  s.t.  $A$  is  $(O, k)$ -diagnosable because the constant observer that observes  $\Sigma$  is a solution. By Theorem 4, there is a trace-based winning strategy for Player 1 in  $G_A$ . As said at the end of the previous section, in this case there is a most permissive trace-based winning strategy which is  $\mathcal{F}_A$ .

**Theorem 6**  $\mathcal{F}_A$  is the most permissive observer.

**Proof:** Let  $O = (S, s_0, \Sigma, \delta, L)$  be an observer such that  $A$  is  $(O, k)$ -diagnosable. We have to prove that  $L(\delta(s_0, w)) \in \mathcal{F}_A(\bar{w})$  for any  $w \in \Sigma^*$ . By Theorem 4, the strategy  $f(O)$  is a winning state-based strategy and this implies that  $f(O)(\nu) \in \mathcal{F}_A(\nu)$  for any run  $\nu$  of  $G_A$ . By definition of  $\bar{w}$ ,  $\pi_{/\Sigma}(\bar{w}) = w$ . By definition of  $f(O)$ ,  $f(O)(\bar{w}) = L(\delta(s_0, \pi_{/\Sigma}(\bar{w}))) = L(\delta(s_0, w))$  and thus  $L(\delta(s_0, w)) \in \mathcal{F}_A(\bar{w})$ .

Conversely, assume  $O$  is such that  $\forall w \in \Sigma^*, L(s_0, w) \in \mathcal{F}_A(\bar{w})$ . We have to prove that  $A$  is  $(O, k)$ -diagnosable. Again, we build  $f(O)$ . As before,  $f(O)$  is a winning trace-based strategy in  $G_A$  and thus  $O(f(O))$  is such that  $A$  is  $(O(f(O)), k)$ -diagnosable by Theorem 5. Assume  $O(f(O)) = (S', s'_0, \Sigma, \delta', L')$ . By construction of  $O(f(O))$ ,  $L'(\delta'(s'_0, w)) = f(O)(\rho)$  if  $w = \pi_{/\Sigma}(tr(\rho))$ . Hence  $O(f(O)) = O$  and  $A$  is  $(O, k)$ -diagnosable. ■

This enables us to solve Problem 5 and compute a finite representation of the set  $\mathcal{O}$  of all observers such that  $A$  is  $(O, k)$ -diagnosable iff  $O \in \mathcal{O}$ .

Computing  $\mathcal{F}_A$  can be done in  $O(2^{|G_A|})$  (Appendix B). The size of  $G_A$  is linear in  $|A|$  and exponential in the size of  $\Sigma$  and  $k$  i.e.  $|G_A| = O(|A| \cdot 2^{|\Sigma|} \cdot 2^k)$ . This means that computing  $\mathcal{F}_A$  can be done in exponential time in the size of  $A$  and doubly exponential time in the size of  $\Sigma$  and  $k$ . Assume  $A$  has  $n$  states and  $m$  transitions. The number of states of  $G_A$  is in  $O(n^2 \cdot k \cdot 2^{|\Sigma|})$ . The number of transitions of  $G_A$  is in  $O(n^2 \cdot k \cdot 2^{|\Sigma|} \cdot m)$ . Hence  $\mathcal{F}_A$  has at most  $O(2^{n^2 \cdot k \cdot 2^{|\Sigma|}})$  states and  $O(2^{n^2 \cdot k \cdot 2^{|\Sigma|}} \cdot n^2 \cdot k \cdot m)$  transitions.

Theorem 6 establishes that there is a most permissive observer  $\mathcal{F}_A$  in case  $A$  is  $(\Sigma, k)$ -diagnosable and it can be computed in exponential time in the size of  $A$ , doubly exponential time in  $|\Sigma|$  and  $k$ , and has size exponential in  $A$ , and doubly exponential in  $|\Sigma|$  and  $k$ . Moreover the most permissive observer  $\mathcal{F}_A$  can be represented by a finite state machine  $S_{\mathcal{F}_A} = (\{0, 2 \dots, k\} \cup \{1, 3, \dots, 2k' + 1\} \times 2^\Sigma, 0, \Sigma \cup 2^\Sigma, \delta)$  which has the following properties:

- even states are states where the observer chooses a set of events to observe;
- odd states  $(2k+1, X)$  are states where the observer waits for an observable event in  $X$  to occur;
- if  $\delta(2k, X) = (2k' + 1, X)$  with  $X \in 2^\Sigma$ , it means that from an even state  $2k$ , the automaton  $S_{\mathcal{F}_A}$  can select a set  $X$  of events to observe. The successor state is an odd state together with the set  $X$  of events that are being observed;

- if  $\delta((2k+1, X), a) = 2k'$  with  $a \in X$ , it means that from  $(2k+1, X)$ ,  $S_{\mathcal{F}_A}$  is waiting for an observable event to occur. When some occurs it switches to an even state.

By definition of  $\mathcal{F}_A$ , any observer  $O$  s.t.  $A$  is  $(O, k)$ -diagnosable must select a set of observable events in  $\mathcal{F}_A(\bar{w})$  after having observed  $w \in \pi_{/\Sigma}(\mathcal{L}(A))$ .

**Example 6** For the automaton  $\mathcal{A}$  of Fig. 1, we obtain the most permissive observer  $\mathcal{F}_A$  of Fig. 8. For odd states we have not mentioned the component  $X$  that has last been picked up by the observer.  $X$  is the label of the unique incoming transition. In the even states, the observer chooses what to observe and in the odd states it moves according to what it observes. When the system starts, it can choose either  $\{a, b\}$  or  $\{a\}$ . Once an  $a$  has been observed it can choose any subset containing  $b$ . When a  $b$  has been observed the observer can choose to observe the empty set.

We point out that from an odd state  $(2k+1, X)$ , outgoing transitions are labeled by elements of  $X$ . This does not mean that the DES under observation cannot do other actions from  $2k+1$ : it might be able to do so but there are unobservable for the observer.

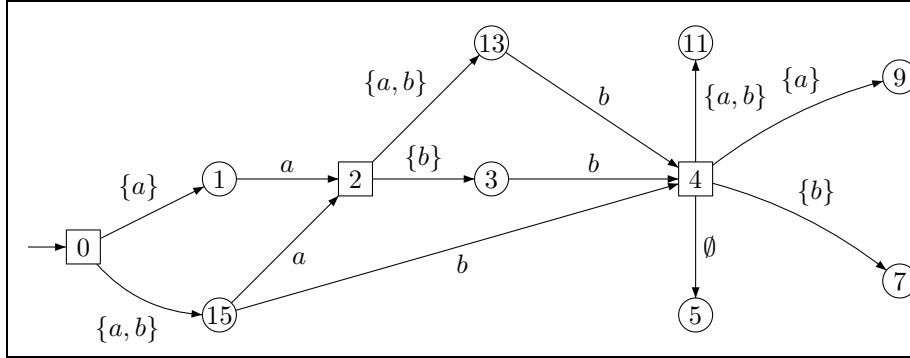


Figure 8: Most Permissive Observer for the Automaton  $\mathcal{A}$  of Fig. 1

The next step we are interested in is to define a notion of *cost* for observers.

## 5.5 Optimal Dynamic Observer

Let  $\text{Obs} = (S, s_0, \Sigma, \delta, L)$  be an observer and  $A = (Q, q_0, \Sigma^{\varepsilon, f}, \rightarrow)$ . We would like to define a notion of *cost* for observers in order to select an optimal one among all of those which are valid, i.e. s.t.  $A$  is  $(\text{Obs}, k)$ -diagnosable. Intuitively this notion of cost should capture the fact that the more events we observe at each time, the more expensive it is.

We are going to define a notion of cost for observers. This notion is inspired by *weighted automata*.

**Weighted Automata.** The notion of cost for automata has already been defined and algorithms to compute some optimal values related to this model are described in many papers. We recall here the results of [8] which will be used later.

**Definition 7 (Weighted Automaton)** A weighted automaton  $(A, w)$  is a pair s.t.  $A = (Q, q_0, \Sigma, \delta)$  is a finite automaton and  $w : Q \rightarrow \mathbb{N}$  associates a weight with each state. ■

**Definition 8 (Mean Cost)** Let  $\rho = q_0 \xrightarrow{a_2} q_1 \xrightarrow{a_1} \dots \xrightarrow{a_n} q_n$  be a run of  $A$ . The mean cost of  $\rho$  is

$$\mu(\rho) = \frac{1}{n+1} \cdot \sum_{i=0}^n w(q_i) \quad .$$

■

We remind that the length of  $\rho = q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \dots \xrightarrow{a_n} q_n$  is  $|\rho| = n$ . We assume that  $A$  is complete w.r.t.  $\Sigma$  (and  $\Sigma \neq \emptyset$ ) and thus contains at least one run for any arbitrary length  $n$ . Let  $Runs^n(A)$  be the set of runs of length  $n$  in  $Runs(A)$ . The *maximum mean-weight* of runs of length  $n$  for  $A$  is  $\nu(A, n) = \max\{\mu(\rho) \text{ for } \rho \in Runs^n(A)\}$ . The *maximum mean weight* of  $A$  is  $\nu(A) = \limsup_{n \rightarrow \infty} \nu(A, n)$ . Actually the value  $\nu(A)$  can be computed using Karp's maximum mean-weight cycle algorithm [8] on weighted graphs. If  $c = s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} \dots \xrightarrow{a_n} s_n$  is a cycle of  $A$  i.e.  $s_0 = s_n$ , the *mean weight* of  $c$  is  $\mu(c) = \frac{1}{n+1} \cdot \sum_{i=0}^n w(s_i)$ . The *maximum mean-weight cycle* of  $A$  is the value  $\nu^*(A) = \max\{\mu(c) \text{ for } c \text{ a cycle of } A\}$ . As stated in [14], for weighted automata,  $\nu(A) = \limsup_{n \rightarrow \infty} \nu(A, n) = \lim_{n \rightarrow \infty} \nu(A, n) = \nu^*(A)$ . Karp's maximum mean-weight cycle algorithm [8] on weighted graphs is explained in Appendix C.

**Definition of Cost.** There is not one way of defining a notion of cost for observers and we first discuss two different notions:

- the first one is to define the cost of a word  $w$  generated by the DES w.r.t. to  $\text{Obs}(w)$ :

$$Cost_1(w) = \frac{\sum_{i=0}^{i=n} |L(\delta(s_0, \text{Obs}(w)(i)))|}{n+1}$$

with  $n = |\text{Obs}(w)|$ . Using the observer of Fig. 9,  $Cost_1(b^n.a) = \frac{1+0}{2} = \frac{1}{2}$ . And this regardless of the value of  $n$ .

- the second one is to define the cost of  $w$  w.r.t. to  $w$  itself:

$$Cost_2(w) = \frac{\sum_{i=0}^{i=n} |L(\delta(s_0, w(i)))|}{n+1}$$

with  $n = |w|$ . Using the observer of Fig. 9, we obtain  $Cost_2(b^n.a) = \frac{n+1+0}{n+2} = \frac{n+1}{n+2}$ . And by simple arithmetic, it is true that  $Cost_2(b^n.a) < Cost_2(b^{n+1}.a)$ .

The example of Fig. 9 shows that the two notions are different. In the sequel we will use the second one  $Cost_2$  because  $Cost_2$  also captures the notion of the *time* we have been observing a set of events. Indeed, if the word  $b^{n+1}$  occurs, we have been observing the set  $L(0)$   $n + 1$  times in a logical time. It is natural that this is more expensive than observing  $L(0)$   $n$  times. Thus  $Cost_2$  is more satisfying than abstracting away the length of the input word as in  $Cost_1$ .

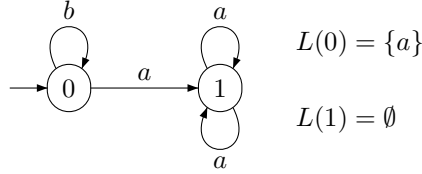


Figure 9: The Finite-State Observer Obs

**Cost of an Observer.** We now show how to define and compute the cost of an observer Obs operating on a DES  $A$ .

Given a word  $\rho \in Runs(A)$ , the observer only processes  $\pi_{/\Sigma}(tr(\rho))$  ( $\varepsilon$  and  $f$ -transitions are not processed). To have a consistent notion of costs that takes into account the logical time elapsed from the beginning, we need to take into account one way or another the number of *steps* of  $\rho$  even if some of them are non observable. A simple way to do this is to consider that  $\varepsilon$  and  $f$  are now observable events, let's say  $u$ , but that the observer never chooses to observe them. Indeed we assume we have already checked that  $A$  is (Obs,  $k$ )-diagnosable, and the problem is now to compute the cost of the observer we have used.

**Definition 9 (Cost of a Run)** Given a run  $\rho = q_0 \xrightarrow{a_1} q_1 \cdots q_{n-1} \xrightarrow{a_n} q_n \in Runs(A)$ , let  $w_i = Obs(tr(\rho(i)))$ ,  $0 \leq i \leq n$ . The cost of  $\rho \in Runs(A)$  is defined by:

$$Cost_2(\rho, A, Obs) = \frac{1}{n+1} \cdot \sum_{i=0}^n |L(\delta(s_0, w_i))|.$$

■

We recall that  $Runs^n(A)$  is the set of runs of length  $n$  in  $Runs(A)$ . The cost of the runs of length  $n$  of  $A$  is

$$Cost_2(n, A, Obs) = \max\{Cost_2(\rho, A, Obs) \text{ for } \rho \in Runs^n(A)\}methpunct.$$

The cost of the pair (Obs,  $A$ ) is

$$Cost_2(A, Obs) = \limsup_{n \rightarrow \infty} Cost_2(n, A, \rho).$$

Notice that  $Cost_2(n, A, Obs)$  is defined for each  $n$  because we have assumed  $A$  generates runs of arbitrary large length.

To compute  $Cost_2(n, A, \text{Obs})$  we consider that  $\varepsilon$  and  $f$  are now observable events, let's say  $u$ , but that the observer never chooses to observe them.

Let  $\text{Obs}^+ = (S, s_0, \Sigma^u, \delta', L)$  where  $\delta'$  is  $\delta$  augmented with  $u$ -transitions that loop on each state  $s \in S$ . Let  $A^+$  be  $A$  where  $\varepsilon$  and  $f$  transitions are renamed  $u$ . Let  $A^+ \times \text{Obs}^+$  be the synchronized product of  $A^+$  and  $\text{Obs}^+$ .  $A^+ \times \text{Obs}^+ = (Z, z_0, \Sigma^u, \Delta)$  is complete w.r.t.  $\Sigma^u$  and we let  $w(q, s) = |L(s)|$  so that  $(A^+ \times \text{Obs}^+, w)$  is a weighted automaton.

**Theorem 7**  $Cost_2(A, \text{Obs}) = \nu^*(A^+ \times \text{Obs}^+)$ .

**Proof:** The proof follows easily from the definitions. Let  $\rho$  be a run of  $A$ . There exists a run  $\tilde{\rho}$  in  $A^+ \times \text{Obs}^+$  s.t.  $Cost_2(\rho, A, \text{Obs}) = \mu(\tilde{\rho})$ .  $\tilde{\rho}$  is obtained from  $\rho$  by replacing  $\varepsilon$  and  $f$  transitions by some  $u$  transitions. Conversely for any run  $\tilde{\rho}$  in  $A^+ \times \text{Obs}^+$  there is a run  $\rho$  in  $A$  s.t.  $\mu(\tilde{\rho}) = Cost_2(\rho, A, \text{Obs})$ . ■

We can compute the cost of a given pair  $(A, \text{Obs})$ : this can be done using Karp's maximum mean weight cycle algorithm [8] on weighted graphs. This algorithm is polynomial in the size of the weighted graph and thus we have:

**Theorem 8** *Computing the cost of  $(A, \text{Obs})$  is in  $P$ .*

**Proof:** The size of  $A^+ \times \text{Obs}^+$  is polynomial in the size of  $A$  and  $\text{Obs}$ . ■

Notice that instead of the values  $|L(s)|$  we could use any mapping from states of  $\text{Obs}$  to  $\mathbb{Z}$  and consider that the cost of observing  $\{a, b\}$  is less than observing  $a$ .

**Example 7** *We give the results for the computation of the cost of two observers for the DES  $A$  given in Fig. 1. Let  $O_1$  be the most powerful observer that observes  $\{a, b\}$  at each step, and  $O_2$  be the observer given in Fig. 5.*

*The automata  $A^+ \times O_1^+$  and  $A^+ \times O_2^+$  are given in Fig. 10 and Fig. 11. The weight function is pictured above each state. Notice that to compute  $\nu^*(A^+ \times O_i^+)$  we do not need the labels of the transitions as we are dealing with weighted graphs: if two transitions  $(s, a, s')$  and  $(s, b, s')$  are in  $A^+ \times O_i^+$  we only need one of them. For instance in Fig. 1 one of the transitions  $(0, a, 4)$  and  $(0, b, 4)$  is redundant. We apply the algorithm of Appendix C. The values  $D_k(v)$  and  $\min(v)$  for each state  $v$  of  $A^+ \times O_i^+$  are given in Table 1 and Table 2. The maximum mean-weight value  $\nu^*$  is the maximum value  $\max_v \min(v)$  for  $v$  ranging over the set of states of  $A^+ \times O_i^+$ . We obtain  $Cost_2(A, O_1) = 2$  and  $Cost_2(A, O_2) = 1$ .*

**Computing an Optimal Dynamic Diagnoser** In section 5.5, we assumed we had an observer. In section 5.4, we proved that a most permissive observer could be computed, given  $A$  and  $k$ . In this section, we focus on the problem of computing a best observer in the sense that diagnosing the DES with it has minimal cost. We address the following problem:

**Problem 6 (Bounded Cost Observer)**

INPUT:  $A, k \in \mathbb{N}$  and  $c \in \mathbb{N}$ .

PROBLEM:

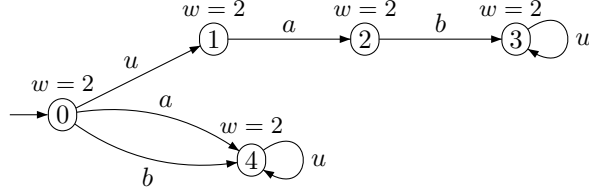


Figure 10:  $\mathcal{A}^+ \times O_1^+$

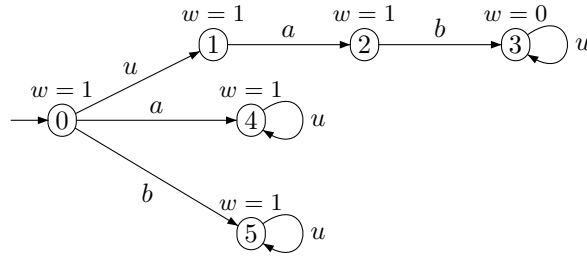


Figure 11:  $\mathcal{A}^+ \times O_2^+$

	0	1	2	3	4
$D_0$	1	$-\infty$	$-\infty$	$-\infty$	$-\infty$
$D_1$	$-\infty$	4	$-\infty$	$-\infty$	4
$D_2$	$-\infty$	$-\infty$	6	$-\infty$	6
$D_3$	$-\infty$	$-\infty$	$-\infty$	8	8
$D_4$	$-\infty$	$-\infty$	$-\infty$	10	10
min	$-\infty$	$-\infty$	$-\infty$	<b>2</b>	<b>2</b>

Table 1: Iterations for  $\mathcal{A}^+ \otimes O_1^+$

	0	1	2	3	4	5
$D_0$	1	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
$D_1$	$-\infty$	2	$-\infty$	$-\infty$	2	2
$D_2$	$-\infty$	$-\infty$	3	$-\infty$	3	3
$D_3$	$-\infty$	$-\infty$	$-\infty$	4	4	4
$D_4$	$-\infty$	$-\infty$	$-\infty$	4	5	5
$D_5$	$-\infty$	$-\infty$	$-\infty$	4	6	6
min	$-\infty$	$-\infty$	$-\infty$	0	<b>1</b>	<b>1</b>

Table 2: Iterations for  $\mathcal{A}^+ \otimes O_2^+$

- (A). Is there an observer  $\text{Obs}$  s.t.  $A$  is  $(\text{Obs}, k)$ -diagnosable and  $\text{Cost}_2(\text{Obs}) \leq c$ ?
- (B). If the answer to (A) is “yes”, compute a witness optimal observer  $\text{Obs}$  with  $\text{Cost}_2(\text{Obs}) \leq c$ .

To compute an optimal observer, we use a result by Zwick and Paterson [14] on *weighted graph games* (see Appendix D for Zwick and Paterson algorithm). These are graphs  $(V, E)$  with the set of nodes partitioned into two sets:  $V_1$  for Player 1 and  $V_2$  for Player 2. In a  $V_i$  state it is Player  $i$ 's turn to play. There is a *weight* function that associates with each edge  $e$  the weight  $w(e)$ . The players build paths  $e_1 \cdots e_n$  by choosing an edge when it is their turn to play. The goal of the game is for Player 1 to maximize the value  $\liminf_{n \rightarrow \infty} \frac{1}{n} \sum_{i=1}^n w(e_i)$  and for Player 2 to minimize  $\limsup_{n \rightarrow \infty} \frac{1}{n} \sum_{i=1}^n w(e_i)$ . One of the results by Zwick and Paterson [14] is that:

- there is a value  $\nu \in \mathbb{Q}$ , called the *value of the game* s.t. Player 1 has a strategy to ensure that  $\liminf_{n \rightarrow \infty} \frac{1}{n} \sum_{i=1}^n w(e_i) \geq \nu$  and Player 2 has a strategy to ensure that  $\limsup_{n \rightarrow \infty} \frac{1}{n} \sum_{i=1}^n w(e_i) \leq \nu$ ; this value can be computed in  $O(|V|^3 \cdot |E| \cdot W)$  where  $W$  is the range of the weight function (assuming the weights are in the interval  $[-W..W]$ ). Note that deciding whether this value satisfies  $\nu \bowtie k$  for  $\bowtie \in \{=, <, >\}$  for  $k \in \mathbb{Q}$  can be done in  $O(|V|^2 \cdot |E| \cdot W)$ .
- there are optimal memoryless strategies for both players that can be computed in  $O(|V|^4 \cdot |E| \cdot \log(|E|/|V|) \cdot W)$ .

To solve the Problem 6, we use the most permissive observer we computed in section 5.4. Given  $A$  and  $\mathcal{F}_A$ , we build a weighted graph game  $G(A, \mathcal{F}_A)$  s.t. the value of the game is the optimal cost for the set of all observers. Moreover an optimal observer can be obtained by taking an optimal memoryless strategy in  $G(A, \mathcal{F}_A)$ .

To build  $G(A, \mathcal{F}_A)$  we use the same idea as in section 5.5: we replace  $\varepsilon$  and  $f$  transitions in  $A$  by  $u$  obtaining  $A^+$ . We also modify  $\mathcal{F}_A$  to obtain a weighted graph game  $(\mathcal{F}_A^+, w)$  by adding transitions so that each state  $2k+1$  is complete w.r.t.  $\Sigma^u$ . This is done as follows:

- from each  $(2k+1, X)$  state, create a new even state i.e. pick some  $2k'$  that has not already been used. Add transitions  $((2k+1, X), \sigma, 2k')$  for each  $\sigma \in \Sigma^u \setminus \text{en}(2k+1, X)$ . Add also a transition  $(2k', X, (2k+1, X))$ . This step means that if a  $A$  produces an event and it is not observable,  $\mathcal{F}_A^+$  just reads the event and makes the same choice again.
- the weight of a transition  $(2k, X, (2k'+1, X))$  is  $|X|$ .

The automaton  $\mathcal{F}_A^+$  obtained from  $\mathcal{F}_A$  is depicted on Fig. 12. The game  $G(A, \mathcal{F}_A)$  is then  $A^+ \times \mathcal{F}_A^+$ . This way we can obtain a weighted graph game  $WG(A, \mathcal{F}_A)$  by abstracting away the labels of the transitions. Notice that it still enables us to convert any strategy in  $WG(A, \mathcal{F}_A)$  to a strategy in  $\mathcal{F}_A$ . A



A consequence of Theorem 7 is that the cost of the optimal observer is a rational number (see Appendix D).

**Example 8** For the example  $\mathcal{A}$  of Fig. 6 and  $\mathcal{F}_A^+$  of Fig. 12, using Zwick and Paterson’s algorithm (Appendix D) we obtain that the optimal cost is 1 and the optimal strategy is to use the observer Obs of Fig. 5.

## 6 Conclusions

In this paper we have addressed sensor minimization problems in the context of fault diagnosis, using both static and dynamic observers. We showed that computing the smallest number of observable events necessary to achieve diagnosis with a static observer is NP-complete: this result also holds in the mask-based setting which allows to consider events that are observable but not distinguishable. We then focused on dynamic observers and proved that, for a given such observer, diagnosability can be checked in polynomial time (as in the case of static observers). We also solved the synthesis problem of dynamic observers and showed that a most-permissive dynamic observer can be computed in doubly-exponential time. Finally we have shown how to define a notion of cost for dynamic observers and how to compute the optimal cost of the set of all observers that can  $k$ -diagnose  $A$ .

There are several directions we are currently investigating:

- Problem 4 has not been solved so far. The major impediment to solve it is that the reduction we propose in section 5 yields a Büchi game. The algorithm we give in appendix B does not work for Büchi games and cannot be extended trivially.
- Problem 5 is solved in doubly exponential time. To reduce the number of states of the most permissive observer, we point out that only *minimal* sets of events we need to observe are needed. Indeed, if we can diagnose a system by observing only  $A$  from some point on, we surely can diagnose it using any superset  $A' \supseteq A$ . So far we keep all the sets that can be used to diagnose the system. We could possibly take advantage of the previous property using techniques described in [5].

## References

- [1] Franck Cassez, Stavros Tripakis, and Karine Altisen. Sensor minimization problems with static or dynamic observers for fault diagnosis. Technical Report RI-2007-1, IRCCyN/CNRS, 1 rue de la Noë, BP 92101, 44321 Nantes Cedex, France, January 2007.
- [2] R. Cieslak, C. Desclaux, A. Fawaz, and P. Varaiya. Supervisory control of discrete-event processes with partial observations. *IEEE Transactions on Automatic Control*, 33:249–260, 1988.

- [3] Ali Dasdan, Sandy S. Irani, and Rajesh K. Gupta. Efficient algorithms for optimum cycle mean and optimum cost to time ratio problems. In *Annual ACM IEEE Design Automation Conference*, pages 37–42, New Orleans, Louisiana, United States, 1999. ACM Press New York, NY, USA. ISBN:1-58133-109-7.
- [4] R. Debouk, S. Lafortune, and D. Teneketzis. On an optimization problem in sensor selection. *Discrete Event Dynamic Systems*, 4(12), November 2004.
- [5] L. Doyen, K. Chatterjee, T.A. Henzinger, and J.-F. Raskin. Algorithms for omega-regular games with imperfect information. In *CSL: Computer Science Logic*, Lecture Notes in Computer Science 4207, pages 287–302. Springer, 2006.
- [6] S. Jiang, Z. Huang, V. Chandra, and R. Kumar. A polynomial algorithm for testing diagnosability of discrete event systems. *IEEE Transactions on Automatic Control*, 46(8), August 2001.
- [7] S. Jiang, R. Kumar, and H. E. Garcia. Optimal sensor selection for discrete event systems with partial observation. *IEEE Transactions on Automatic Control*, 48(3):369–381, March 2003.
- [8] Richard M. Karp. A characterization of the minimum mean cycle in a digraph. *Discrete Mathematics*, 23:309–311, 1978.
- [9] P. Ramadge and W. Wonham. Supervisory control of a class of discrete event processes. *SIAM J. Control Optim.*, 25(1), January 1987.
- [10] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Diagnosability of discrete event systems. *IEEE Transactions on Automatic Control*, 40(9), September 1995.
- [11] Wolfgang Thomas. On the synthesis of strategies in infinite games. In *Proc. 12th Annual Symposium on Theoretical Aspects of Computer Science (STACS'95)*, volume 900, pages 1–13. Springer, 1995. Invited talk.
- [12] J.N. Tsitsiklis. On the control of discrete event dynamical systems. *Mathematics of Control, Signals and Systems*, 2(2), 1989.
- [13] T. Yoo and S. Lafortune. On the computational complexity of some problems arising in partially-observed discrete event systems. In *American Control Conference (ACC'01)*, 2001. Arlington, VA.
- [14] U. Zwick and M. Paterson. The complexity of mean payoff games on graphs. *Theoretical Computer Science*, 158(1–2):343–359, 1996.

## A Checking Diagnosability

Here is an algorithm to check diagnosability in polynomial time.

Let  $A = (Q, q_0, \Sigma^{\varepsilon, f}, \rightarrow)$  be a finite automaton over  $\Sigma^{\varepsilon, f}$ . By definition of *diagnosability*,  $A$  is diagnosable iff  $\exists k \in \mathbb{N}^*$  s.t.  $A$  is  $(\Sigma_o, k)$ -diagnosable. This is equivalent to

$$A \text{ is not diagnosable} \iff \forall k \in \mathbb{N}^*, A \text{ is not } (\Sigma, k)\text{-diagnosable} \quad (9)$$

$$\iff \forall k \in \mathbb{N}^*, \begin{cases} \exists \rho \in \text{NonFaulty}(A) \\ \exists \rho' \in \text{Faulty}_{\geq k}(A) \\ \pi_{/\Sigma}(\rho) = \pi_{/\Sigma}(\rho') \end{cases} \quad (10)$$

Define  $A_1 = (Q \times \{0, 1\}, (q_0, 0), \Sigma^{\varepsilon}, \rightarrow_1)$  s.t.

- $(q, k) \xrightarrow{l}_1 (q', k')$  iff  $q \xrightarrow{l} q'$  and  $l \in \Sigma$  and  $k = k'$ ;
- $(q, k) \xrightarrow{\varepsilon}_1 (q', 1)$  iff  $q \xrightarrow{f} q'$ , ( $k$  is set to 1 after a fault occurs and will remain 1 once it has been set to 1);
- $(q, k) \xrightarrow{\varepsilon}_1 (q', k)$  iff  $q \xrightarrow{\varepsilon} q'$ .

Define  $A_2 = (Q, q_0, \Sigma^{\varepsilon}, \rightarrow_2)$  with

- $q \xrightarrow{l}_2 q'$  if  $q \xrightarrow{l} q'$  and  $l \in \Sigma$ ;
- $q \xrightarrow{\varepsilon}_2 q'$  if  $q \xrightarrow{\varepsilon} q'$ .

Let  $A_1 \times A_2$  be the synchronized product of  $A_1$  and  $A_2$  (synchronization on common actions, i.e. actions in  $\Sigma$  and otherwise interleaving).  $\rightarrow_{1,2}$  stands for the transition relation of  $A_1 \times A_2$ .

Let  $\text{Runs}_{\geq k}(A_1 \times A_2)$  be the set of runs in  $A_1 \times A_2$  s.t. a (faulty) state  $(q_1, 1), q_2$  is followed by at least  $k$   $A_1$ -actions. Then we have the following lemma:

**Lemma 4**

$$\exists \rho \in \text{Runs}_{\geq k}(A_1 \times A_2) \iff \begin{cases} \exists \rho_1 \in \text{NonFaulty}(A) \\ \exists \rho_2 \in \text{Faulty}_{\geq k}(A) \\ \pi_{/\Sigma_o}(\rho_1) = \pi_{/\Sigma_o}(\rho_2) \end{cases}$$

**Proof:** It suffices to note that a compound run in  $\text{Runs}_{\geq k}(A_1 \times A_2)$  can be split into a run of  $A_1$  and a run of  $A_2$  having the same projection on  $\Sigma$ . ■

Assume we have Lemma 4 for all  $k \in \mathbb{N}^*$ . Then there must be a state  $(q_1, 1), q_2$  in  $A_1 \times A_2$  which is the source of an infinite number of runs  $\rho_k$  having more than  $k$   $A_1$ -actions. As  $A_1 \times A_2$  is of finite branching, by König's Lemma, there is a infinite run in  $A_1 \times A_2$  containing infinitely many  $A_1$  actions. The converse holds as well: if there is an infinite faulty run in  $A_1 \times A_2$  with infinitely

many  $A_1$  actions, then for all  $k \in \mathbb{N}^*$  there is a run  $\rho_1 \in \text{NonFaulty}(A)$ ,  $\rho_2 \in \text{Faulty}_{\geq k}(A)$  s.t.  $\pi_{/\Sigma}(\rho_1) = \pi_{/\sigma}(\rho_2)$ .

We can then define an extended version of  $A_1 \times A_2$ : we add a boolean variable  $z$  that is set to 0 in the initial state. An extended state is a pair  $(s, z)$  with  $s$  a state of  $A_1 \times A_2$ . Whenever there is an  $A_1$ -action,  $z$  is set to 1, and when there is an  $A_2$  action,  $z$  is set 0. We denote  $\Longrightarrow_{1,2}$  the new transition relation for extended states. Define the set of states  $F = \{(((q, 1), q'), 1) \mid ((q, 1), q') \in A_1 \times A_2\}$ . Define the Büchi automaton  $\mathcal{B} = ((Q \times \{0, 1\} \times Q) \times \{0, 1\}, ((q_0, 0), q_0, 0), \Sigma^\varepsilon, \Longrightarrow_{1,2}, F)$  by: The following theorem holds:

**Theorem 11**  $\mathcal{L}^\omega(\mathcal{B}) \neq \emptyset \iff A$  is not  $\Sigma$ -diagnosable.

This gives a procedure to check whether an automaton  $A$  is diagnosable or not. The product (extended) system built from  $A_1 \times A_2$  has size  $O(|A|^2)$ .

As language emptiness of Büchi automata can be checked in polynomial time w.r.t. number of states and transitions, checking diagnosability of a finite (non-deterministic) automaton is also polynomial.

To compute the smallest  $k$  s.t.  $A$  is  $(\Sigma, k)$ -diagnosable we proceed as follows: in case  $A$  is  $\Sigma$ -diagnosable, there is no infinite faulty run with infinitely many  $A_1$ -actions in  $A_1 \times A_2$ ; this implies that each run beginning in a faulty state of  $A_1 \times A_2$  is followed by a finite number of  $A_1$ -actions and this number is bounded by a constant  $k$ . Indeed, otherwise, we could construct an infinite faulty run with a infinite number of  $A_1$ -actions and  $A$  would not be diagnosable. Assume  $k$  is the maximum number of  $A_1$ -actions that can follow a faulty state. Then  $A$  is  $(\Sigma, k + 1)$ -diagnosable: otherwise they would be a faulty run with  $k + 1$   $A_1$ -actions after a faulty state in  $A_1 \times A_2$ . Also  $A$  is not  $(\Sigma, k)$ -diagnosable because there is faulty run with  $k$   $A_1$ -actions after a faulty state in  $A_1 \times A_2$ , which also means there is another non faulty run with the same observable actions. Thus  $k + 1$  is the minimum value for which  $A$  is  $\Sigma$ -diagnosable. Computing  $k + 1$  amounts to computing the maximum number of  $A_1$ -actions that can follow a faulty state in  $A_1 \times A_2$ .

The complexity of this computation is  $O(|Q|^3)$  if  $A$  as  $|Q|$  states. To compute the minimum  $k$ , we define the following operators: let  $W$  be a subset of the faulty states of  $A_1 \times A_2$  and  $k \in \mathbb{N}^*$ ; define  $\text{SUCC}_{A_1}(W, k) = (\text{SUCC}_{A_1}(W), k + 1)$  and  $\text{SUCC}_{A_2 \setminus A_1}(W, k) = (\text{SUCC}_{A_2 \setminus A_1}(W), k)$  where  $\text{SUCC}_B(S)$  is the set of states reachable in one step from a state of  $S$  doing a  $B$ -action in  $A_1 \times A_2$ . Define also  $\text{SUCC}(W, k) = \text{SUCC}_{A_1}(W, k) \cup \text{SUCC}_{A_2 \setminus A_1}(W, k)$ . We then use the following inductive definition:  $Y_0 = \{\{f \mid f \text{ is a faulty state of } A_1 \times A_2\}, 0\}$ , and  $Y_{i+1} = \cup_{(x_i, k_i) \in Y_i} \text{SUCC}(x_i, k_i)$ . Because the number of  $A_1$ -actions following a faulty state is bounded by  $k$ , there are at most  $k \cdot |Q|^2$  different sets  $Y_i$ . Also  $k$  must be less than  $|Q|^2$  because otherwise they would be a cycle in  $A_1 \times A_2$  with  $k$  actions and thus an infinite faulty run with infinitely many  $A_1$ -actions. Hence after at most  $|Q|^3$  steps  $Y_i = Y_{i+1}$  and the largest value of  $k_i$  in  $Y_i$  is  $k$ . This way we only need  $O(|Q|^3)$  steps to compute  $k$ .

To compute the diagnoser we do the following: take  $A_1$  and determinize it. The deterministic automaton obtained this way has  $2^{O(|Q|)}$  states. If a state  $S$

of the deterministic version contains only faulty states, i.e., every  $(q, k) \in S$  is such that  $k = 1$  then declare  $S$  as faulty. Define the diagnoser  $D$  as follows: for a run  $\rho \in \Sigma$ ,  $D(\rho) = 1$  if  $S_0 \xrightarrow{\rho} S'$  in the deterministic version of  $A_1$  and  $S'$  is faulty; otherwise  $D(\rho) = 0$ .  $D$  is a diagnoser for  $A$  and issues a 1 after at most  $k + 1$  steps if  $k$  is the minimum value for which  $A$  is  $(\Sigma, k)$ -diagnosable. Computing the diagnoser is thus exponential in the size of  $A$ , i.e. can be done in  $2^{O(|A|)}$ .

## B Results For Safety 2-Player Games

In this section we give results on games with partial observation where two players are playing, but Player 1 cannot observe all Player 2 moves. The proofs of lemmas and theorems in this section are given in Appendix B of [1]. The aim of Player 1 is to win but with a *trace-based* strategy i.e. a strategy that is based on the set of moves that have been observed. In each state of the game, it is either up to Player 1 to play or to Player 2. The game starts in a Player 1 state. Player 1 only plays one move (in  $\Sigma_1$ ) and hands it over to player 2. Player 2 can play two types of moves: invisible moves ( $\varepsilon$  is the invisible action) and visible moves ( $\Sigma_2$ -actions). If Player 2 plays an invisible move  $\varepsilon$ , it is again his turn to play. Otherwise if he plays a visible move, the turn switches to Player 1. This setting is formally defined by:

### B.1 Finite Game Automata

**Definition 10 (Two-Player  $\varepsilon$ -Games)** A two-player  $\varepsilon$ -game  $G$  is a tuple  $(Q_1 \uplus Q_2, q_0, \Sigma_1 \uplus \Sigma_2^\varepsilon, \delta)$  with:

- $Q_i$  is a finite set of states for player  $i, i = 1, 2$ ;
- $q_0 \in Q_1$  is the initial state of the game;
- $\Sigma_i$  is a finite set of actions for player  $i, i = 1, 2$ ;
- $\delta \subseteq (Q_1 \times \Sigma_1 \times Q_2) \cup (Q_2 \times \{\varepsilon\} \times Q_2) \cup (Q_2 \times \Sigma_2 \times Q_1)$  is the transition relation.

We assume that the game is deterministic w.r.t Player 1 moves, i.e. for all  $q_1 \in Q_1, \sigma_1 \in \Sigma_1$  there is at most one state  $q_2 \in Q_2$  such that  $(q_1, \sigma_1, q_2) \in \delta$ . For  $(q, a, q') \in \delta$  we use the notation  $q \xrightarrow{a} q'$  or  $qaq'$ . We let  $en(q) = \{\sigma \mid \exists q' \mid \delta(q, \sigma) = q'\}$  i.e. the set of moves enabled in a state  $q$ . If  $G$  contains no  $\varepsilon$  transitions,  $G$  is an alternating full-observation two-player game (we use the term two-player game in this case).

**Definition 11 (Play, Trace)** A play in  $G$  is a finite or infinite sequence

$$\rho = q_0 \ell_1 q_1 \cdots q_n \ell_{n+1} q_{n+1} \cdots$$

such that for each  $i$ ,  $q_i \xrightarrow{\ell_{i+1}} q_{i+1}$ . We write  $q_0 \xrightarrow{\ell_1 \ell_2 \dots \ell_n} q_n$  if  $q_0 \ell_1 q_1 \dots \ell_n q_n$  is a finite play in  $G$ . We let  $Runs(G)$  be the set of plays in  $G$  and  $Runs^*(G)$  be the set of finite plays.  $Runs_i^*(G)$ ,  $i = 1, 2$  are the sets of finite runs ending in a  $Q_i$ -state. The trace of  $\rho$ , denoted  $tr(\rho)$  is the sequence<sup>6</sup>  $\ell_0 \ell_1 \dots \ell_n \dots$ . The state sequence of  $\rho$ , denoted  $States(\rho)$  is the sequence  $q_0 q_1 \dots q_n \dots$ .

We let  $\rho(i)$  be the prefix of  $\rho$  up to state  $q_i$ , so  $\rho(0) = q_0$ ,  $\rho(n) = q_0 \ell_1 q_1 \dots q_n$ , and so on.

**Definition 12 (Player 1 Strategy)** A strategy for player 1 is a mapping  $f : Runs_1^*(G) \rightarrow \Sigma_1$ . A strategy  $f$  is trace-based if for all  $\rho, \rho' \in Runs_1^*(G)$ ,  $tr(\rho) = tr(\rho')$  implies  $f(\rho) = f(\rho')$ . A strategy  $f$  is memoryless if  $tgt(\rho) = tgt(\rho')$  implies  $f(\rho) = f(\rho')$ .

**Definition 13 (outcome)** Given a strategy  $f$  for player 1, the outcome of the game  $G$  under  $f$  is the set  $Out(G, f)$  of states sequences<sup>7</sup>  $States(\rho)$  for plays

$$\rho = q_0 \ell_1 q_1 \dots q_n \ell_{n+1} q_{n+1} \dots$$

such that for each  $q_i \in Q_1$ ,  $\ell_{i+1} = f(\rho(i))$ .

**Definition 14 (Objective)** An objective  $\phi$  for Player 1 is a subset of  $(Q_1 \cup Q_2)^\omega \cup (Q_1 \cup Q_2)^*$ .

Given a pair  $(G, \phi)$ , a strategy  $f$  is winning if  $Out(G, f) \subseteq \phi$ . A state  $q$  of  $G$  is winning if there is a winning strategy from  $q$ .

The usual control problem on two-player  $\varepsilon$ -games asks the following:

**Problem 8 (Control Problem)**

INPUT: A two-player  $\varepsilon$ -game  $G$ , an objective  $\phi$ .

PROBLEM: Is there a winning strategy for  $(G, \phi)$  ?

## B.2 Safety Two-Player Games

$\phi$  is a safety objective if there is a set  $F \subseteq Q_1 \cup Q_2$  such that  $\phi = F^\omega \cup F^*$ . If  $G$  is a  $\varepsilon$ -game and  $\phi$  a safety objective we say that the pair  $(G, \phi)$  is a safety game.  $F$  is the set of safe states. To solve safety two-player  $\varepsilon$ -games we define the *Cpre* operator [11]:

$$Cpre(S) = \{s \in Q_1 \mid \exists \sigma_1 \in \Sigma_1 \mid \delta(s, \sigma_1) \in S\} \quad (11)$$

$$\cup \{s \in Q_2 \mid \forall \sigma_2 \in \Sigma_2^\varepsilon \mid \delta(s, \sigma_2) \subseteq S\} \quad (12)$$

It is well-known [11] that iterating *Cpre* and computing the fixpoint  $Cpre^*(F)$  gives the set of winning states<sup>8</sup> of the game. In case  $G$  is finite this computation

<sup>6</sup>As  $\varepsilon$  also stands for the empty word, a trace does not contains  $\varepsilon$ .

<sup>7</sup>In the sequel we assume that the game objectives are states sequences and do not refer to the labels of the game.

<sup>8</sup>Notice that by definition of *Cpre*, Player 1 cannot win by refusing to play.

terminates and can be done in linear time for safety games [11]. If the initial state of the game  $q_0 \in Cpre^*(F)$ , there is a strategy for Player 1 to win. Moreover for this type of games *memoryless* strategy are sufficient to win. Indeed, as we can see the state of the game,  $\varepsilon$  transitions are not really unobservable (or invisible) and thus knowing the current state gives some useful information. Moreover we can define a *most permissive strategy*<sup>9</sup>  $\mathcal{F} : Q_1 \cap Cpre^*(F) \rightarrow 2^{\Sigma_1} \setminus \emptyset$  by:  $\mathcal{F}(q) = \{\sigma \mid \delta(q, \sigma) \in Cpre^*(F)\}$ . This is the most permissive strategy in the following sense:  $f$  is a winning strategy for  $(G, \phi)$  iff for any  $\rho \in Runs_1^*(G)$ ,  $f(\rho) \in \mathcal{F}(tgt(\rho))$ , i.e. every move defined by  $f$  is a move of the most permissive strategy.

### B.3 Safety Two-Player $\varepsilon$ -Games

The problem we want to address is the following:

**Problem 9 (Trace-Based Control)**

INPUT: a game  $G$ , an objective  $\phi$ .

PROBLEM: Is there a trace-based winning strategy for  $(G, \phi)$  ?

This problem is more demanding than the usual Control Problem 8 that asks only for a winning strategy for Player 1, i.e. a strategy in which full observation of the state is assumed. To solve Problem 9, we reduce it to a full-observation two-player game.

We define the following operator for  $\sigma \in \Sigma_2^\varepsilon$ ,  $s \in Q_2$ ,

$$Next_2(s, \sigma) = \{s' \mid s \xrightarrow{\varepsilon^*} s_1 \xrightarrow{\sigma} s'\} \quad (13)$$

It follows that if  $\sigma \in \Sigma_2$ ,  $Next_2(s, \sigma) \subseteq Q_1$  and if  $\sigma = \varepsilon$ ,  $Next_2(s, \sigma) \subseteq Q_2$ . Let  $\sigma \in \Sigma_1$  and  $Q \subseteq Q_1$ . We define

$$Next_1(Q, \sigma) = \{s' \in Q_2 \mid s' = \delta(s, \sigma) \text{ with } s \in Q\} \quad (14)$$

We address Problem 9 where  $\phi$  is safety objective i.e.  $\phi = F^\omega \cup F^*$ . To solve  $(G, \phi)$ , we build a full-observation two player game  $(G_H, F_H)$  (no  $\varepsilon$  transitions) such that:

**Theorem 12** *There is a (standard) winning strategy in  $(G_H, F_H)$  iff there is a trace-based winning strategy in  $(G, F)$ .*

**Definition 15** *Assume  $G = (Q_1 \uplus Q_2, q_0, \Sigma_1 \uplus (\Sigma_2 \cup \{\varepsilon\}), \delta)$ . The game  $G_H = (S, s_0, \Sigma', \Delta)$  is defined by:*

- $W = W_1 \uplus W_2$  with  $W_1 = (2^{Q_1} \cup \perp_1)$  are the Player 1 states,  $W_2 = (2^{Q_2} \cup \perp_2)$  are Player 2 states and
- $s_0 = \{q_0\}$ ,

---

<sup>9</sup>According to Definition 12, it is not a strategy as it prescribes a set of moves for a given state instead of one move.

- $\Sigma' = \Sigma_1 \cup \Sigma_2^u$  where  $u$  is a fresh name, and  $\Sigma_1$  are Player 1 moves and  $\Sigma_2^u$  Player 2 moves;
- $\Delta \subseteq (W_1 \times \Sigma_1 \times W_2) \cup (W_2 \times \Sigma_2^u \times W_1)$  is defined by:  $(S, \sigma, S') \in \Delta$  iff one of the three conditions holds:
  - $C_1$ :  $S \subseteq Q_1, \sigma \in \Sigma_1$  and  $S' = \text{Next}_1(S, \sigma)$  if for all  $s \in S, \sigma \in \text{en}(s)$  and otherwise  $S' = \perp_2$ ;
  - $C_2$ :  $S \subseteq Q_2, \sigma \in \Sigma_2, S' = \text{Next}_2(S, \sigma)$  and  $S' \neq \emptyset$ ;
  - $C_3$ :  $S \subseteq Q_2, \sigma = u, \text{Next}_2(S, \varepsilon) \cap \overline{F} \neq \emptyset$  and  $S' = \perp_1$ .

We let  $F_H = \{Q \in S \mid Q \subseteq F\}$  i.e.  $F_H$  be the set of safe states for  $G_H$ .  $\perp_1$  and  $\perp_2$  are not safe states.  $(G_H, F_H)$  is a safety game as well. Notice also that  $G_H$  is a turn-based two player game in which the moves of the two players alternate. The following fact holds as well:

**Fact 1** *By definition  $G_H$  is deterministic. Hence for any word  $w \in (\Sigma_1 \cup \Sigma_2^u)^*$ , there is a unique run  $\beta(w) = s_0 \xrightarrow{w} s'$  in  $G_H$  with  $\text{tr}(\beta(w)) = w$  and a unique last state  $\Delta(s_0, w) = s'$ .*

We can now prove Theorem 12:

**Proof:**

**Only If Part** Let  $f_H$  be a winning strategy in  $G_H$  (as the safety objective is fixed we omit it). Let  $\rho \in \text{Runs}_1^*(G)$  and let  $f(\rho) = f_H(\beta(\text{tr}(\rho)))$ .  $f$  is trace-based by its definition and Fact 1.

$f$  is also winning. To prove this we use the following Lemma: Given  $\rho$  a run of  $\text{Out}(G, f)$ , we can build a run  $\rho_H = S_0 \sigma_0 \cdots S_{n_\rho}$  in  $G_H$ , such that  $\rho_H$  is the longest run that can match  $\rho$ . It is defined as follows:

- $S_0 = \{q_0^0\}$ ,
- for  $1 \leq 2i < n_\rho, S_{2i} = \text{Next}_2(S_{2i-1}, \lambda_{2i-1})$
- for  $1 \leq 2i + 1 < n_\rho, S_{2i+1} = \text{Next}_1(S_{2i}, \sigma_{2i})$
- either  $S_{n_\rho} \in \{\perp_1, \perp_2\}$  or
  - if  $n_\rho = 2k, S_{n_\rho} = \text{Next}_2(S_{2k-1}, \lambda_{2k-1})$ ,
  - if  $n_\rho = 2k + 1, S_{n_\rho} = \text{Next}_1(S_{2k}, \sigma_{2k})$ .

**Lemma 5** *This run  $\rho_H$  is in  $\text{Out}(G_H, f_H)$  and has the properties:*

- $P_1$ : for  $1 \leq 2i < n_\rho, q_{2i}^0 \in S_{2i}$  and for  $1 \leq 2i + 1 < n_\rho, q_{2i+1}^1 \in S_{2i+1}$ ;
- $P_2$ : either (a)  $\text{tgt}(\rho_H) = S_{n_\rho} \in \{\perp_1, \perp_2\}$  or (b) if  $n_\rho = 2k, q_{2k}^0 \in S_{n_\rho}$ , and if  $n_\rho = 2k + 1, q_{2k+1}^1 \in S_{n_\rho}$ .

**Proof:** We prove properties Lemma 5 by induction on the number of Player 1 moves in  $\rho$ . Assume  $k = 0$ . Clearly the run  $\rho_H = \{q_0^0\}$  is in  $Out(G_H, f_H)$  and Lemma 5 holds.

Assume the property holds for  $k$  Player 1 moves. Let  $\rho$  be a run with  $k + 1$  Player 1 moves  $\sigma_0, \sigma_2, \dots, \sigma_{2k}$ . We write

$$\begin{aligned} \rho &= q_0^0 \xrightarrow{\sigma_0} q_1^1 \xrightarrow{\varepsilon} \dots \xrightarrow{\varepsilon} q_1^{n_1} \xrightarrow{\lambda_1} \\ & q_2^0 \xrightarrow{\sigma_2} q_3^1 \xrightarrow{\varepsilon} \dots \xrightarrow{\varepsilon} q_3^{n_3} \xrightarrow{\lambda_1} \\ & \vdots \\ & q_{2k}^0 \xrightarrow{\sigma_{2k}} q_{2k+1}^1 \xrightarrow{\varepsilon} \dots \xrightarrow{\varepsilon} q_{2k+1}^{n_{2k+1}} \xrightarrow{\lambda_{2k+1}} \\ & q_{2(k+1)}^0 \xrightarrow{\sigma_{2(k+1)}} q_{2(k+1)+1}^1 \xrightarrow{\varepsilon} \dots \xrightarrow{\varepsilon} q_{2(k+1)+1}^{n_{2(k+1)+1}} \end{aligned}$$

Let  $\rho' = q_0^0 \dots q_{2k+1}^{n_{2k+1}}$ .  $\rho'$  has  $k - 1$  Player 1 moves and by induction hypothesis on  $\rho'$ , we obtain:

- $\rho'_H \in Out(G_H, f_H)$  and  $P_1$  and  $P_2$  hold;
- as  $\rho'_H$  satisfies  $P_2$ :
  - either ia satisfies  $P_2$ .(a) and  $tgt(\rho'_H) \in \{\perp_1, \perp_2\}$ ; in this case  $\rho'_H$  cannot be extended in  $G_H$  and thus  $\rho_H = \rho'_H$  and  $\rho_H$  satisfies  $P_1$  and  $P_2$ .(a);
  - or  $P_2$ .(b) holds and  $\rho'_H$  does not contain any state in  $\{\perp_1, \perp_2\}$ ; Then  $\rho'_H = S_0 \dots S_{2k} \sigma_{2k} S_{2k+1}$  satisfies  $P_1$  and  $P_2$ .(b) and we have in particular  $q_{2k}^0 \in S_{2k}$  and  $q_{2k+1}^1 \in S_{2k+1}$ . Because  $q_{2k+1}^1 \xrightarrow{\varepsilon} \dots q_{2k+1}^{n_{2k+1}} \xrightarrow{\lambda_{2k+1}} q_{2(k+1)}^0$ , the set  $Next_2(S_{2k+1}, \lambda_{2k+1})$  is not empty and thus  $= S_0 \dots S_{2k} \sigma_{2k} S_{2k+1} \lambda_{2k+1} S_{2(k+1)}$  is in  $Out(G_H, f_H)$  and  $S_{2(k+1)} \neq \perp_2$ .  $f_H(S_0 \dots \sigma_{2k} S_{2k+1} \lambda_{2k+1} S_{2(k+1)}) = f(q_0^0 \dots q_{2(k+1)}^0)$  by definition of  $f$  and  $S_0 \sigma_0 \dots \sigma_{2k} S_{2(k+1)} = \beta(tr(q_0^0 \sigma_0 \dots q_{2(k+1)}^0))$ . Two cases arise:
    1. either  $S_{2(k+1)} \xrightarrow{\sigma_{2(k+1)}} \perp_2$ :  
in this case  $\rho_H = S_0 \dots S_{2k} \dots \lambda_{2k+1} S_{2(k+1)} \sigma_{2(k+1)} \perp_2$  and  $\rho_H \in Out(G_H, f_H)$  and satisfies  $P_1$  and  $P_2$ .(a);
    2. or  $S_{2(k+1)} \xrightarrow{\sigma_{2(k+1)}} S_{2(k+1)+1}$ .  
This means that  $S_{2(k+1)+1} \neq \perp_2$ . In this case  $q_{2(k+1)+1}^1 \in S_{2(k+1)+1}$ . Again two cases arise:
      - (a) either there is some  $2 \leq j \leq n_{2(k+1)+1}$  s.t.  $q_{2(k+1)+1}^j \notin F$  and in this case  $S_{2(k+1)+1} \xrightarrow{u} \perp_1$  is in  $\Delta$  and  $\rho_H = S_0 \dots S_{2(k+1)} \sigma_{2(k+1)} S_{2(k+1)+1} u \perp_1$  is in  $Out(G_H, f_H)$ .  $\rho_H$  satisfies  $P_1$  and  $P_2$ .(a);
      - (b) or  $q_{2(k+1)+1}^j \in F$  for  $1 \leq j \leq n_{2(k+1)+1}$  and in this case  $\rho_H = S_0 \dots S_{2(k+1)} \sigma_{2(k+1)} S_{2(k+1)+1}$  is in  $Out(G_H, f_H)$  and satisfies  $P_1$  and  $P_2$ .(b).

This completes the proof of Lemma 5. ■

Now assume  $f$  is not winning, we can build a run  $\rho \in \text{Out}(G, f)$  with  $\text{tgt}(\rho) \notin F$ . Applying Lemma 5 we obtain: there is a run  $\rho_H \in \text{Out}(G_H, f_H)$  that satisfies  $P_1$  and  $P_2$ :

- either  $\text{tgt}(\rho_H) \in \{\perp_1, \perp_2\}$  and in this case  $\rho_H$  is a losing run for Player 1 and thus  $f_H$  is not winning which contradicts the assumption;
- or  $\rho_H$  contains no state in  $\{\perp_1, \perp_2\}$ . Two cases arise depending on the parity of the length of  $\rho_H$ :
  - either  $\rho_H = S_0 \cdots S_{2k}$ . In this case  $\text{tgt}(\rho) \in S_{2k}$  ( $P_2$ .(b)) and thus  $\rho_H$  ends in a losing state and  $f_H$  is not winning.
  - or  $\rho_H = S_0 \cdots S_{2k+1}$  and again  $\text{tgt}(\rho) \in S_{2k+1}$  and  $\rho_H$  is losing and  $f$  is not winning.

This contradicts the assumption that  $f_H$  is winning.

This concludes the proof of the *Only If* part.

**If Part.**

Let  $w$  be a run of  $G_H$  ending in  $W_1$ -state. If  $w$  does not contain any  $u$  action, there is a run  $\rho \in \text{Runs}_1^*(G)$  s.t.  $\text{tr}(\rho) = \text{tr}(w)$  by construction of  $G_H$  (Definition 15). We let  $f_H(w) = f(\rho)$ . If  $w = S_0 \cdots S_k u \perp_1$ , we let  $f_H(w) = f_H(S_0 \cdots S_k)$ .

$f_H$  is well defined as if  $\rho, \rho'$  are such that  $\text{tr}(\rho) = \text{tr}(\rho')$  then  $f(\rho) = f(\rho')$  because  $f$  is trace-based.

$f_H$  is winning. To prove this we use the following lemma:

**Lemma 6** *Let  $w = S_0 \sigma_0 S_k$  be a run in  $\text{Out}(G_H, f_H)$ . Then one of two following conditions hold:*

- $D_1$ :  $S_k \in \{\perp_1, \perp_2\}$  and there is a run  $q_0^0 \cdots q$  in  $\text{Out}(G, f)$  such that either (i) with  $q \notin F$  or (ii)  $f(q_0^0 \cdots q) = \sigma$  and  $\sigma \notin \text{en}(q)$ ;
- $D_2$ :  $S_i \notin \{\perp_1, \perp_2\}$  for  $0 \leq i \leq k$  and for any  $q \in S_k$ , there is a run  $\rho = q_0^0 \cdots q$  in  $\text{Out}(G, f)$  s.t.  $\text{tr}(\rho) = \text{tr}(w)$ .

**Proof:** We prove Lemma 6 by induction of the number of Player 1 moves in  $w$ . If  $w$  contains 0 moves clearly it holds.

Assume  $w$  is a run of  $\text{Out}(G_H, f_H)$  that contains  $k+1$  Player 1 moves. There are two possible forms for  $w$ :

- $w$  ends in a  $W_2$ -state.  $w = S_0 \sigma_0 \cdots S_{2k} \sigma_{2k} S_{2k+1}$ . All the states  $S_i$ ,  $0 \leq i \leq 2k$  must be different from  $\{\perp_1, \perp_2\}$ . Then either  $S_{2k+1} = \perp_2$  or not.
  - if  $S_{2k+1} = \perp_2$ , there is some state  $q \in S_{2k}$  s.t.  $\sigma_{2k} \notin \text{en}(q)$  by definition of  $G_H$ . We can apply the induction assumption on the run  $w' = S_0 \sigma_0 \cdots S_{2k}$  that contains  $k$  Player 1 moves. This leads: there

is a run  $\rho' = q_0^0 \cdots q$  in  $Out(G, f)$  s.t.  $tr(\rho') = tr(w')$  (because  $w'$  does not contain any  $\perp_{1,2}$  state). Moreover  $f(\rho') = f(w') = \sigma_{2k}$ ,  $\sigma_{2k} \notin en(q)$  and thus condition  $D_1.(ii)$  holds for  $w$ .

– otherwise  $S_{2k+1} \neq \perp_2$ . By definition of  $G_H$ , this implies that  $S_{2k+1} = Next_1(S_{2k}, \sigma_{2k})$ . For any  $q \in S_{2k+1}$  there is some  $q' \in S_{2k}$  s.t.  $q' \xrightarrow{\sigma_{2k}} q$  in  $G_H$ . We can again apply the induction assumption on the run  $w' = S_0 \sigma_0 \cdots S_{2k}$  that contains  $k$  Player 1 moves. This leads: there is a run  $\rho' = q_0^0 \cdots q'$  in  $Out(G, f)$  s.t.  $tr(\rho') = tr(w')$  and  $f(\rho') = f(w') = \sigma_{2k}$ . Hence  $\rho = \rho' \xrightarrow{\sigma_{2k}} q$  is a run of  $Out(G, f)$  and  $tr(\rho) = tr(w)$ .  $w$  satisfies  $D_2$ .

- $w$  ends in a  $W_1$ -state. Let  $w = S_0 \sigma_0 \cdots S_{2k} S_{2k+1} \lambda_{2k+1} S_{2(k+1)}$ . It must be the case that  $S_{2k+1} \neq \perp_2$ . Two cases arise:

1.  $\lambda_{2k+1} \neq u$ . By definition of  $G_H$ ,  $S_{2(k+1)} = Next_2(S_{2k+1}, \lambda_{2k+1})$ . Let  $q \in S_{2(k+1)}$ . By definition of  $Next_2$ , there is a state  $q' \in S_{2k+1}$  s.t.  $q' \xrightarrow{\varepsilon} \cdots \xrightarrow{\lambda_{2k+1}} q$ . As in the previous case for any  $q' \in S_{2k+1}$  we can build a run  $\rho' = q_0^0 \cdots q'$  in  $Out(G, f)$  with  $tr(\rho') = tr(S_0 \cdots S_{2k+1})$ . Hence  $\rho = \rho' \xrightarrow{\lambda_{2k+1}} q$  is a run of  $Out(G, f)$  and satisfies  $D_2$ .
2.  $\lambda_{2k+1} = u$ . In this case  $S_{2(k+1)} = \perp_1$ . This mean that there is a state  $q' \in S_{2k+1}$  s.t.  $q' \xrightarrow{\varepsilon} \cdots \xrightarrow{\varepsilon} q$  and  $q \notin F$ . Again we can build a run  $\rho' = q_0^0 \cdots q'$  in  $Out(G, f)$  with  $tr(\rho') = tr(S_0 \cdots S_{2k+1})$  and clearly  $\rho = \rho' \xrightarrow{\varepsilon} \cdots \xrightarrow{\varepsilon} q$  is run of  $Out(G, f)$  and thus  $w$  satisfies  $D_1.(i)$ .

This completes the proof of Lemma 6. ■

Now assume  $f_H$  is not winning. There is a run  $S_0 \cdots S_k$  in  $Out(G_H, f_H)$  such that either (i)  $S_k \in \{\perp_1, \perp_2\}$  or (ii) there is some  $q \in S_k$  such that  $q \notin F$ . Applying Lemma 6 we obtain:

- if (i) holds,  $D_1$  holds and there is a run  $\rho \in Out(G, f)$  s.t. either (a)  $tgt(\rho) \notin F$  or (b)  $f(\rho) \notin en(tgt(\rho))$ . If (a) holds  $f$  is not a winning strategy. If (b) holds  $f$  is not a strategy. In any case this contradicts the fact that  $f$  is a winning strategy.
- if (ii) holds, as  $S_k$  contains a state  $q \notin F$ , there is a run  $q_0^0 \cdots q$  in  $Out(G, f)$  and  $q \notin F$  which again contradicts the fact that  $f$  is winning.

This completes the proof of Theorem 12. ■

From this we obtain an algorithm for Problem 9 and as the size of  $G_H$  is exponential in the size  $G$ :

**Theorem 13** *Problem 9 is in EXPTIME.*

## B.4 Most Permissive Strategies for $\varepsilon$ -Games

Given  $G_H$  and  $F_H$  we can compute the most permissive strategy  $\mathcal{F}_H$ . Given  $\mathcal{F}_H$  we define the following mapping on  $Runs_1^*(G)$ :  $\mathcal{F}(\rho) = \mathcal{F}_H(tgt(\beta(tr(\rho))))$ .

**Theorem 14**  $\mathcal{F}$  is the most permissive trace-based strategy for  $G$ .

**Proof:** First  $\mathcal{F}$  is trace-based<sup>10</sup>. Let  $f$  be a trace-based winning strategy. We define  $f_H$  as in the *If* part proof of Theorem 12.  $f_H$  is winning and thus for any run  $w$   $f_H(w) \in \mathcal{F}_H(\text{tgt}(w))$ . By definition of  $f_H$ ,  $f(\rho) = f_H(\beta(\text{tr}(\rho)))$  and  $f(\rho) \in \mathcal{F}_H(\text{tgt}(\beta(\text{tr}(\rho)))) = \mathcal{F}(\rho)$ .

Now assume  $f$  is a trace-based strategy such that for each run  $\rho \in \text{Runs}_1^*(G)$ ,  $f(\rho) \in \mathcal{F}(\rho)$ . Build  $f_H$  as before.  $f_H$  is winning. Indeed,  $f_H(w) = f(\rho)$  for any  $\rho$  s.t.  $\text{tr}(\rho) = \text{tr}(w)$ . Hence,  $f_H(w) \in \mathcal{F}_H(\text{tgt}(w))$  and thus  $f_H$  is winning. Now from  $f_H$  build a strategy  $\tilde{f}$  as in the *Only If* part of the proof of Theorem 12. It turns out that  $\tilde{f} = f$  and as  $\tilde{f}$  is winning,  $f$  is winning. ■

**Corollary 1** The most permissive trace-based strategy  $\mathcal{F}$  for  $(G, \phi)$  can be represented by an automaton which has at most an exponential number of states.

This follows from the fact that  $G_H$  is exponential in the size of  $G$ . The most permissive trace-based strategy is obtained from  $G_H$  by removing from each state  $q$  the transitions that are not in  $\mathcal{F}_H(q)$ .

## C Karp's Maximum Mean Cycle Algorithm

In this section, we recall Karp's maximum mean cycle algorithm [8]. The original algorithm works for weighted graph where the weights are on the edges. We give a version where weights are on vertices.

**Definition 16 (Weighted Graph)** A weighted directed graph is a pair  $(G, w)$  s.t.  $G = (V, E)$  is a directed graph and  $w : V \rightarrow \mathbb{R}$ . We assume that each vertex  $v \in V$  is reachable from a unique source vertex  $s_0$ . ■

**Definition 17 (Mean Weight of a Cycle)** Let  $c = (v_1, v_2, \dots, v_k)$  be a sequence of vertices s.t. each  $(v_i, v_{i+1}) \in E$ ,  $1 \leq i \leq k-1$  which is a cycle i.e.  $v_1 = v_k$ . The mean weight of  $c$  is  $\mu(c) = \frac{1}{k} \cdot \sum_{i=1}^k w(v_i)$ . ■

Let  $\nu^* = \max_c \mu(c)$  where  $c$  ranges over all directed cycles in  $G$ . A cycle  $c$  with  $\mu(c) = \nu^*$  is a *maximum mean-weight cycle*.

Let  $D(v)$  be the weight of a most expensive path from  $s_0$  to  $v$  and  $D_k(v)$  be the weight of a most expensive path which has exactly  $k$  edges (if there is no such path  $D_k(v) = -\infty$ ).

Assume  $|V| = n$ . Karp's algorithm is based on the fact that

$$\nu^* = \max_{v \in V} \min_{0 \leq k \leq n-1} \frac{D_n(v) - D_k(v)}{n - k}$$

<sup>10</sup>Even if  $\mathcal{F}$  is not strictly speaking a strategy, the trace-based property extends to sets of moves with set equality.

The values  $D_k(v)$  can be computed iteratively:

$$D_0(s_0) = w(s_0) \quad (15)$$

$$D_0(v) = -\infty \quad \text{for } v \neq s_0 \quad (16)$$

$$D_{k+1}(v) = \max_{(u,v) \in E} \{D_k(u) + w(v)\} \quad (17)$$

Thus for each vertex we can compute  $\min(v) = \min_{0 \leq k \leq n-1} \frac{D_n(v) - D_k(v)}{n-k}$  and then compute the value  $\max_{v \in V} \min(v)$  to obtain  $\nu^*$ . This algorithm runs in  $O(n.m)$  where  $|V| = n$  and  $|E| = m$ . Improvements [3] can be made to this algorithm still the worst case run-time is  $O(n.m)$ .

## D Zwick and Paterson's Algorithm

In this section we give an overview of some results of [14]. Assume  $G = (V, E)$  is a weighted graph as in definition 16 except that the weight function is defined on edges:  $w : E \rightarrow \{-W, \dots, 0, \dots, W\}$  assigns an integral weight to each edge of  $G$  with  $W \in \mathbb{N}$ . We assume each vertex has at least one outgoing transition.

**Definition 18 (Weighted Graph Game)** *A weighted graph game is a bipartite weighted graph  $G = (V, E)$  with  $V = V_1 \cup V_2$  and  $E = E_1 \cup E_2$ ,  $E_1 \subseteq V_1 \times V_2$  and  $E_2 \subseteq E_2 \times E_1$ . We assume the initial vertex  $v_0$  of  $G$  belongs to  $V_1$ . ■*

Vertices  $V_i$  are Player  $i$ 's vertex. A weighted graph game is a turn based game in which the turn alternates between Player 1 and Player 2. The game starts at a vertex  $v_0 \in V_1$ . Player 1 chooses an edge  $e_1 = (v_0, v_1)$  and then Player 2 chooses an edge  $e_2 = (v_1, v_2)$  and so on and they build an infinite sequence of edges. Player 1 wants to maximise  $\liminf_{n \rightarrow \infty} \frac{1}{n} \cdot \sum_{i=1}^n w(e_i)$  and Player 2 wants to minimize  $\limsup_{n \rightarrow \infty} \frac{1}{n} \cdot \sum_{i=1}^n w(e_i)$ .

One of the result of [14] is that there is a rational value  $\nu \in \mathbb{Q}$  s.t. Player 1 has a strategy to ensure  $\liminf_{n \rightarrow \infty} \frac{1}{n} \cdot \sum_{i=1}^n w(e_i) \geq \nu$  and Player 2 has a strategy to ensure that  $\limsup_{n \rightarrow \infty} \frac{1}{n} \cdot \sum_{i=1}^n w(e_i) \leq \nu$ .  $\nu$  is called the value of the game.

Let  $n = |V|$ . To compute  $\nu$ , proceed as follows ([14]):

1. Let  $\nu_0(v) = 0$  for  $v \in V$ . For  $v \in V$  and  $k \geq 1$ ,  $\nu_k(v)$  is defined by:

$$\nu_k(v) = \begin{cases} \max_{(v,w) \in E} \{w(v, w) + \nu_{k-1}(w)\} & \text{if } v \in V_1 \\ \min_{(v,w) \in E} \{w(v, w) + \nu_{k-1}(w)\} & \text{if } v \in V_2 \end{cases}$$

This is the equivalent of the  $D_k(v)$  values for Karp's algorithm using a min max strategy depending on which player is playing;

2. for each  $v \in V$ , compute  $\nu'(v) = \nu_k(v)/k$  for  $k = 4 \cdot n^3 \cdot W$ .

3. for each vertex, the value of the game from  $v$  is the only rational number with a denominator at most  $n$  that lies in the interval  $] \nu'(v) - \alpha, \nu'(v) + \alpha [$  with  $\alpha = \frac{1}{2n(n-1)}$ .

The value of the game is  $\nu = \nu(v_0)$  where  $v_0$  is the initial vertex.

To compute an optimal strategy for Player 1, proceed as follows:

1. compute the values  $\nu(v)$  for each  $v \in V$ ;
2. if all the vertices of  $V_1$  have outgoing degree 1, there is a unique strategy and it is positional and optimal;
3. otherwise, take a vertex  $v \in V_1$  with outgoing degree  $d \geq 2$ . Remove  $\lceil \frac{d}{2} \rceil$  edges from  $v$  leaving at least one. Recompute the value  $m_v$  for each  $v$ . If  $m_v = \nu(v)$ , there is an optimal positional strategy which uses the remaining edges from  $v$ . Otherwise there is a positional strategy that uses one of the removed edges.

We can iterate the previous scheme to find an optimal strategy for Player 1.