Sensor Minimization Problems with Static or Dynamic Observers for Fault Diagnosis

Franck Cassez* Stavros Tripakis† Karine Altisen‡

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.

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* events Σ, plus a set of *unobservable* events [7, 10]. The invisible actions can often be represented by a single unobservable event ε. 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 [8] 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 Σ is the set of observable events, ε 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 [8, 11, 5]. (Notice that synthesizing a diagnoser involves determinization in general, thus cannot be done in polynomial time.)

We examine sensor optimization problems with both

*CNRS/IRCCyN, 1 rue de la Noë, BP 92101, 44321 Nantes Cedex 3, France. Email: franck.cassez@cnrs.irccyn.fr

†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.

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*.

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 [11]. Our result of section 3 can be viewed as an alternative shorter proof of this result. Masks have not been considered in [11]. 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 [6] where it was shown that the problem is NP-hard for general properties. [6] 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 [3], 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.

2 Preliminaries

2.1 Words and Languages

Let Σ be a finite alphabet and $\Sigma^\varepsilon = \Sigma \cup \{\varepsilon\}$. Σ^* is the set of finite words over Σ and contains ε which is also the empty word. A *language* L is any subset of Σ^*. $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$. Given two words ρ, ρ' we denote $\rho.\rho'$ the concatenation of ρ and ρ' (which is defined in the usual way). $|\rho|$ stands for the length of the word ρ and $|\rho|_\lambda$ with $\lambda \in \Sigma$ stands for the number of occurrences of λ in ρ. Given $\Sigma_1 \subseteq \Sigma$, we define the *projection* $\boldsymbol{\pi}_{/\Sigma_1} : \Sigma^* \to \Sigma_1^*$ by: $\boldsymbol{\pi}_{/\Sigma_1}(\varepsilon) = \varepsilon$ and for $a \in \Sigma, \rho \in \Sigma^*$, $\boldsymbol{\pi}_{/\Sigma_1}(a.\rho) = a.\boldsymbol{\pi}_{/\Sigma_1}(\rho)$ if $a \in \Sigma_1$ and $\boldsymbol{\pi}_{/\Sigma_1}(\rho)$ otherwise.

2.2 Finite Automata

Let $f \notin \Sigma^\varepsilon$ be a fresh letter that corresponds to the fault action. A *finite automaton* A is a tuple[1] $(Q, q_0, \Sigma^{\varepsilon,f}, \to)$ with Q a finite set of states, $q_0 \in Q$ is the initial state, $\to \subseteq Q \times \Sigma^{\varepsilon,f} \times Q$ is the transition relation. We write $q \xrightarrow{\lambda} q'$ if $(q, \lambda, q') \in \to$. For $q \in Q$, $en(q)$ is the set of actions enabled at q. A *run* ρ 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 $s_0 = s$. We let $tgt(\rho) = s_n$. 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 ρ, denoted $tr(\rho)$, is the word obtained by concatenating the symbols λ_i appearing in ρ, for those λ_i different from ε. Given a set $R \subseteq Runs(A)$, $Tr(R) = \{tr(\rho) \text{ for } \rho \in R\}$ is the set of traces of the runs in R. A run ρ is *k-faulty* if there is some $1 \leq i \leq n$ s.t. $\lambda_i = f$ and $n - i \geq k$. $Faulty_{\geq k}(A)$ is the set of k-faulty runs of A. A run is *faulty* if it is k-faulty for some $k \in \mathbb{N}$ and $Faulty(A)$ denotes the set of faulty runs. It follows that $Faulty_{\geq k+1}(A) \subseteq Faulty_{\geq k}(A) \subseteq \cdots \subseteq Faulty_{\geq 0}(A) = Faulty(A)$. Finally

[1] 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.

NonFaulty(A) = *Runs*$(A) \setminus$ *Faulty*(A) is the set on *non-faulty* runs of A. We let *Faulty*$^{tr}_{\geq k}(A)$ = *Tr*(*Faulty*$_{\geq k}(A)$) and *NonFaulty*$^{tr}(A)$ = *Tr*(*NonFaulty*(A)).

A word w is *accepted* by A if w = *tr*(ρ) for some $\rho \in$ *Runs*(A). The *language* $\mathcal{L}(A)$ of A is the set of words accepted by A.

We assume that each faulty run of A of length n can be extended into a run of length $n+1$. This is required for technical reasons (in order to guarantee that the set of faulty runs where sufficient time has elapsed after the fault is well-defined) and can be achieved by adding ε 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.

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 [11] and we only give here an alternative shorter proof. We are given a finite automaton $A = (Q, q_0, \Sigma^{\varepsilon,f}, \rightarrow)$. The maximal set of observable events is Σ (ε is not observable). 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 Σ_o. Moreover, we would like to find an "optimal" such Σ_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). If the system generates a word ρ but only a subset $\Sigma_o \subseteq \Sigma$ is observable, the diagnoser can only see $\boldsymbol{\pi}_{/\Sigma_o}(\rho)$.

Definition 1 ((Σ_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* (Σ_o, k)-diagnoser for A *if (i) for each $\rho \in$ NonFaulty(A), $D(\boldsymbol{\pi}_{/\Sigma_o}(tr(\rho))) = 0$, and (ii) for each $\rho \in$ Faulty$_{\geq k}(A)$, $D(\boldsymbol{\pi}_{/\Sigma_o}(tr(\rho))) = 1$.* ■

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

Example 1 *Let $\mathcal{A}$ be the automaton shown on Fig. 1. The run 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, 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 $\mathcal{A}$.

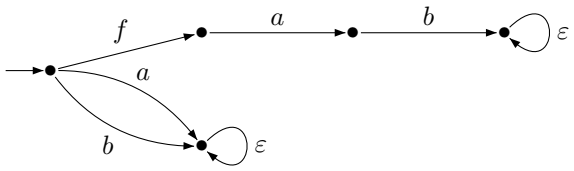

Figure 1. The automaton $\mathcal{A}$

For given A and Σ_o it is known how to check diagnosability and build a diagnoser (e.g., see [8]). Checking whether A is Σ_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 (Σ_o, k)-diagnosable can be done in $O(|A|^3)$. Moreover in case A is Σ_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 in [1]. These algorithms use the fact that A is (Σ_o, k)-diagnosable iff

$$\boldsymbol{\pi}_{/\Sigma_o}(\textit{Faulty}^{tr}_{\geq k}(A)) \cap \boldsymbol{\pi}_{/\Sigma_o}(\textit{NonFaulty}^{tr}(A)) = \emptyset \quad (1)$$

or in other words, there is no pair of runs (ρ_1, ρ_2) with $\rho_1 \in$ *Faulty*$_{\geq k}(A)$, $\rho_2 \in$ *NonFaulty*(A) s.t. ρ_1 and ρ_2 give the same observations on Σ_o. In this section we address the problem of *finding* a set of observable events Σ_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.* $n \leq |\Sigma|$.
PROBLEM:

(A) Is there any $\Sigma_o \subseteq \Sigma$ with $|\Sigma_o| = n$, such that A is Σ_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 Σ_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.[2] Unfortu-

[2]Notice that knowing n_0 does not imply we know the required set of observable events Σ_o! We can find (one of the possibly many) Σ_o by searching over all possible subsets Σ_o of Σ of size n_0 (there are $C(|\Sigma|, n_0)$ such combinations) and check for each such Σ_o whether A is Σ_o-diagnosable, using the methods described in the appendix A of [1].

nately, 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 in [1]: if we guess a solution Σ_o we can check that A is Σ_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 Σ_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 Σ do not need to be observed i.e. iff Problem 1(A) gives a positive answer for $|\Sigma| - n$. Let $\Sigma = V$. Then we define A_G as shown in Fig. 2: q_0 is the initial state and the "branches" $f.a.b$, $f.a'.b'$, $\cdots$ are obtained from the pairs of nodes $(a, b), (a', b'), \cdots$ that are *not* linked with an edge in G: $(a, b), (a', b'), \cdots \notin E$.

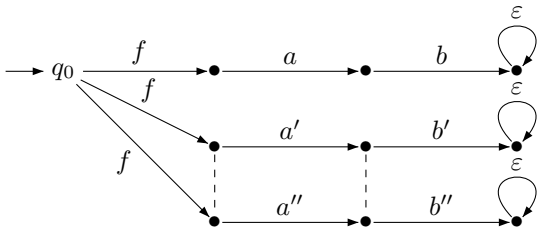

Figure 2. The automaton A_G

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 Σ_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 $\rho' = f.a.b.\varepsilon^K$ 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 ρ and ρ' is ε, 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 Σ_o-diagnosable). Suppose not. Then there exist two runs ρ and ρ' such that $\rho' \in \mathit{NonFaulty}(A_G)$ and $\rho = \rho_1.f.\rho_2$, and $|\rho_2| \geq 2$, and $\boldsymbol{\pi}_{/\Sigma_o}(\rho) = \boldsymbol{\pi}_{/\Sigma_o}(\rho')$. Then, ρ must be of the form $\rho = f.a.b.\varepsilon^k$, with $(a, b) \notin E$. Also, the only way ρ' can be non faulty is $\rho' = \varepsilon$. Then $\boldsymbol{\pi}_{/\Sigma_o}(\rho') = \varepsilon = \boldsymbol{\pi}_{/\Sigma_o}(\rho)$, 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* Σ *is a total, surjective function* $M : \Sigma \to \{1, \cdots, n\} \cup \{\varepsilon\}$. ■

M induces a morphism $M^* : \Sigma^* \to \{1, \cdots, n\}^*$. 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* Σ. *A mapping* $D : \{1, \cdots, n\}^* \to \{0, 1\}$ *is a* $((M, n), k)$-diagnoser *for* A *if (i) for each* $\rho \in \mathit{NonFaulty}(A)$, $D(M^*(\rho)) = 0$ *and (ii) for each* $\rho \in \mathit{Faulty}_{\geq k}(A)$, $D(M^*(\rho)) = 1$. ■

A is $((M, n), k)$-diagnosable if there is a $((M, n), k)$-diagnoser for A. A is (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 Σ_o-diagnosability of a modified automaton A_M, with $\Sigma_o = \{1, ..., n\}$. 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 $\{1, \cdots, n\}$-diagnosable. Notice that A is $((M, n), k)$-diagnosable iff $M^*(\mathit{Faulty}_{\geq k}(A)) \cap M^*(\mathit{NonFaulty}(A)) = \emptyset$.

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 Σ_o such that A is (Σ_o, k)-diagnosable and $\Sigma_o = \{a_1, ..., a_n\}$; define a mask $M : \Sigma \to \{1, \cdots, n\}$ 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.*

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 in [1]). 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 $\{1, 2, \cdots, n\}$ so that no two adjacent vertices have the same color ? Let $G = (V, E)$ be an undirected graph. Let $E = \{e_1, e_2, \cdots, e_j\}$ 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.

If part. Assume A_G is (M, n)-diagnosable for $n \geq 0$. We first show that for all $i = 1, ..., j$, $M(u_i) \neq \varepsilon$, $M(v_i) \neq \varepsilon$ 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 $M^*(\boldsymbol{\pi}_{/\Sigma}(\rho)) = M^*(\boldsymbol{\pi}_{/\Sigma}(\rho'))$. 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 $C : V \to \{1, 2, \cdots, n\}$ 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 $\rho \in \mathit{Faulty}_{\geq 1}(A_G)$ and $\rho' \in \mathit{NonFaulty}(A_G)$ such that $M(\boldsymbol{\pi}_{/\Sigma}(\rho)) = M(\boldsymbol{\pi}_{/\Sigma}(\rho'))$. As ρ 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, $M(\boldsymbol{\pi}_{/\Sigma}(\rho)) = M(v_1).M(u_1) \cdots M(u_i).M(v_i)$, and $|M(\boldsymbol{\pi}_{/\Sigma}(\rho))| = 2i$. Consequently, $|M(\boldsymbol{\pi}_{/\Sigma}(\rho'))| = 2i$. The only possible such ρ' which is also non-faulty is $\rho' = v_1.\varepsilon.u_1 \cdots v_i.\varepsilon.u_i$. Now, $M(\boldsymbol{\pi}_{/\Sigma}(\rho)) = M(\boldsymbol{\pi}_{/\Sigma}(\rho'))$, 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 Dynamic Observers

In this section we introduce *dynamic observers*. To illustrate why dynamic observers can be helpful 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 Σ can be used to detect faults in $\mathcal{B}$. Thus the minimum value for Problem 1 is 2. If we want to use a mask, the minimum value for Problem 2 is 2 as well. This means that a diagnoser 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.*

5.1 Diagnosers and 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.

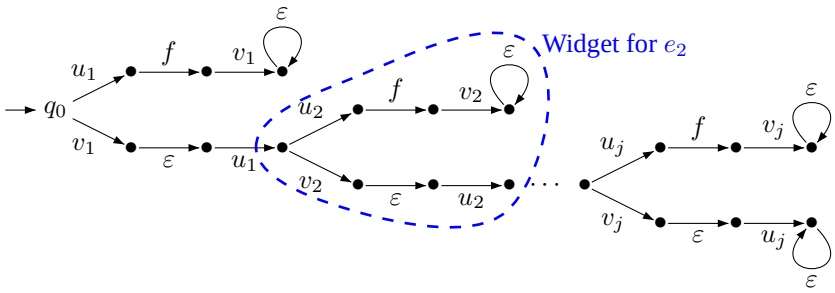

Figure 3. Automaton A_G for n-colorizability

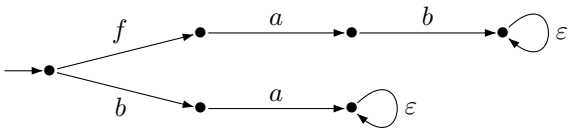

Figure 4. The automaton $\mathcal{B}$

Definition 4 (Observer) *An* observer Obs *over* Σ *is a deterministic labeled automaton* Obs $= (S, s_0, \Sigma, \delta, L)$*, where* S *is a (possibly infinite) set of states,* $s_0 \in S$ *is the initial state,* Σ *is the set of observable events,* $\delta : S \times \Sigma \to S$ *is the transition function (a total function), and* $L : S \to 2^{\Sigma}$ *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 ε, (when $a \notin L(s)$) and remains in the same state waiting for a new event. Thus, an observer defines a mapping Obs from Σ^* to Σ^* (we use the same name "Obs" for the automaton and the mapping). Given a run ρ, Obs(ρ) is the output of the transducer when it reads ρ. It is called the *observation* of ρ by Obs. We next provide an example of a particular case of observer which can be represented by a finite-state machine.

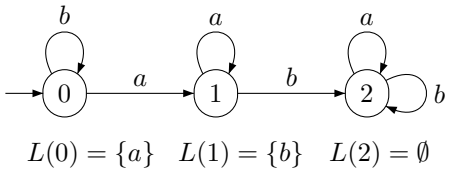

Figure 5. A Finite-State Observer Obs

Example 4 *Let Obs be the observer of Fig. 5. Obs maps the following inputs as follows:* Obs$(baab) = ab$*,* Obs$(bababbaab) = ab$*,* Obs$(bbbbba) = a$ *and* Obs$(bbaaa) = a$*.*

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

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. As for Σ-diagnosability, we have the following equivalence: A is (Obs, k)-diagnosable iff Obs(*Faulty*$_{\geq k}(\mathcal{A})) \cap$ Obs(*NonFaulty*$(\mathcal{A})) = \emptyset$.

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: The proof runs as follows: we build a *product* automaton[3] $A \otimes$ Obs such that: A is (Obs, k)-diagnosable $\iff$ $A \otimes$ Obs is (Σ, k)-diagnosable.

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

- $(q, s) \xrightarrow{\beta} (q', s')$ 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;
- $(q, s) \xrightarrow{\lambda} (q', s)$ iff $\exists \lambda \in \{\varepsilon, f\}$ s.t. $q \xrightarrow{\lambda} q'$.

To prove Theorem 3 we use the following lemmas:

[3] We use $\otimes$ to clearly distinguish this product from the synchronous product $\times$.

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

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

The proofs of these lemmas are given in [1]. Now assume A is (Obs, k)-diagnosable and $A \otimes \text{Obs}$ is not (Σ, k)-diagnosable. There are two words $\nu \in$ *Faulty*$_{\geq k}(A \otimes \text{Obs})$ and $\nu' \in$ *NonFaulty*$(A \otimes \text{Obs})$ s.t. $\boldsymbol{\pi}_{/\Sigma}(\nu) = \boldsymbol{\pi}_{/\Sigma}(\nu')$. By Lemma 2, there are two words $\rho \in$ *Faulty*$_{\geq k}(A)$ and $\rho' \in$ *NonFaulty*(A) s.t. $\text{Obs}(\rho) = \boldsymbol{\pi}_{/\Sigma}(\nu) = \boldsymbol{\pi}_{/\Sigma}(\nu') = \text{Obs}(\rho')$ and thus A is not (Obs, k)-diagnosable.

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

The number of states of $A \otimes \text{Obs}$ is at most $|Q|.|S|$ 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 in [1]) thus Problem 3 is in P. This completes the proof. ■

For Problem 3, we have assumed that an observer was given. It would be even better if we could *synthesize* an observer Obs such that the plant is diagnosable with Obs. Before attempting to synthesize such an observer, we should first check that the plant is Σ-diagnosable: if it is not, then obviously no such observer exists; if the plant is Σ-diagnosable, then the trivial observer that observes all events in Σ at all times works[4]. Therefore, we need a way to eliminate such trivial 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* 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. We know how to do this for safety games (Appendix A) but we do not have a solution to solve Büchi games of this type. Instead, we introduce a restricted variant:

[4]Notice that this also shows that existence of an observer implies existence of a finite-state observer, since the trivial observer is finite-state.

Problem 5 (Dynamic-k-Diagnosability)
INPUT: A *and* $k \in \mathbb{N}$.
PROBLEM: *Compute the set* $\mathcal{O}$ *of all observers such that* A *is* (Obs, k)*-diagnosable iff* $\text{Obs} \in \mathcal{O}$.

5.2 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 A. 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. 6, 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 Σ 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 λ in X: to do this both A_1^k and A_2 must be in a state where λ 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')).

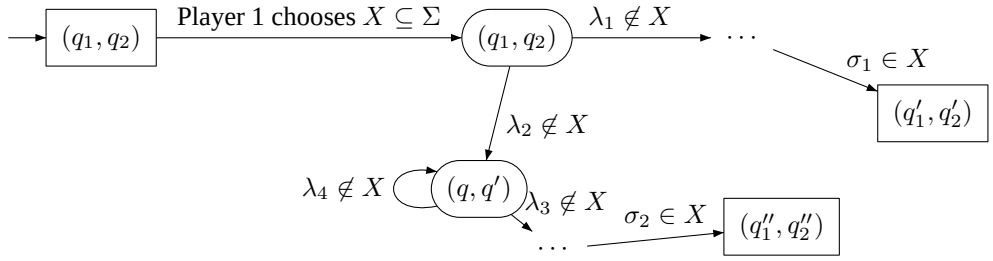

Figure 6. Game Reduction for Problem 5

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

- $S_1 = (Q \times \{-1, \cdots, k\}) \times Q$ 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 $j = k$.
- $S_2 = (Q \times \{-1, \cdots, k\}) \times Q \times 2^\Sigma$ 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 ε).
- 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 (ε) 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. Consequently, a *silent* move $((q_1, i), q_2, X), \varepsilon, (q'_1, j), q'_2, X))$ is in δ 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 : *Runs*$(G_A) \rightarrow \Sigma_1$ that associates a move $f(\rho)$ in Σ_1 to each run ρ of G_A that ends in a S_1-state. A strategy f is trace-based (see appendix A for details), if given two runs ρ, ρ', if *tr*$(\rho) =$ *tr*(ρ') 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: $f(O)(\rho) = L(\delta(s_0, \boldsymbol{\pi}_{/\Sigma}(tr(\rho))))$. The intuition is that we take the run ρ in G_A, take the trace of ρ (choices of Player 1 and moves of Player 2) and remove the choices of Player 1. This gives a run in Σ^*. The strategy for Player 1 for ρ is the set of events the observer O chooses to observe after reading $\boldsymbol{\pi}_{/\Sigma}(tr(\rho))$ i.e. $L(\delta(s_0, \boldsymbol{\pi}_{/\Sigma}(tr(\rho))))$.

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 $Out(G, f)$ 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 A for a formal definition of $Out(G_A, f)$). Assume on the contrary $f(O)$ is not winning. This implies that there is a run ρ in $Out(G_A, f(O))$ as defined by equations (2–5). Each part of the run given by equations (2–5) consists of a choice of Player 1 (X_i move) followed by a number of

$$\rho = (q_0^1,0),q_0^2 \xrightarrow{X_0} (q_0^1,0),q_0^2,X_0 \xrightarrow{\lambda_0^1} (q_0^1(1),k_0(1)),q_0^2(1),X_0\cdots(q_0^1(j),k_0(j)),q_0^2(j),X_0\cdots \xrightarrow{\lambda_0^{n_0}} \quad (2)$$

$$(q_1^1,k_1),q_1^2 \xrightarrow{X_1} (q_1^1,k_1),q_1^2,X_1 \xrightarrow{\lambda_1^1} (q_1^1(1),k_1(1)),q_1^2(1),X_1\cdots(q_1^1(j),k_1(j)),q_1^2(j),X_1\cdots \xrightarrow{\lambda_1^{n_1}} \quad (3)$$

$$(q_2^1,k_2),q_2^2 \qquad \cdots \quad (4)$$

$$(q_n^1,k_n),q_n^2 \xrightarrow{X_n} (q_n^1,k_n),q_n^2,X_n\cdots(q_n^1(j),k_n(j)),q_n^2(j),X_n\cdots \xrightarrow{\lambda_0^\alpha} (q_n^1(\alpha),k_n(\alpha)),q_n^2(\alpha),X_n \quad (5)$$

$$\nu = q_0^1 \xrightarrow{\lambda_0^1} q_0^1(1) \xrightarrow{\lambda_0^2} \cdots \xrightarrow{\lambda_0^{n_0}} q_1^1 \xrightarrow{\lambda_1^1} \cdots \xrightarrow{\lambda_1^{n_1}} q_2^1\cdots q_n^1 \xrightarrow{\lambda_n^1} \cdots \xrightarrow{\lambda_n^\alpha} q_n^1(\alpha) \quad (6)$$

$$\nu' = q_0^2 \xrightarrow{\lambda_0^1} q_0^2(1) \xrightarrow{\lambda_0^2} \cdots \xrightarrow{\lambda_0^{n_0}} q_1^2 \xrightarrow{\lambda_1^1} \cdots \xrightarrow{\lambda_1^{n_1}} q_2^2\cdots q_n^2 \xrightarrow{\lambda_n^1} \cdots \xrightarrow{\lambda_n^\alpha} q_n^2(\alpha) \quad (7)$$

moves by Player 1 (λ_i^j actions). The last state encountered in ρ, $(q_n^1(\alpha),k_n(\alpha)),q_n^2(\alpha),X_n$ 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 ρ, we can build two runs ν and ν' defined by equations (6) and (7). By definition of G_A, each λ_i^j is either a common visible action of A_1^k and A_2 and it is in Σ, or a silent action (ε) 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 ν (resp. ν') the actions ε 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, $tr(\tilde{\nu})\in$ *Faulty*$_{\geq k}(A)$ and $tr(\tilde{\nu}')\in$ *NonFaulty*(A). We claim that $O(tr(\tilde{\nu}))=O(tr(\tilde{\nu}'))$. Indeed, each part of the runs from $q_i^1\cdots q_{i+1}^1$ and $q_i^2\cdots q_{i+1}^2$ yields the same observation by O: it is the sequence of events $\lambda_{j_1}\cdots\lambda_{j_{n_i}}$ s.t. each λ_{j_l} is a letter of both A_1^k and A_2 and is X_i. As there are two words $tr(\tilde{\nu})\in$ *Faulty*$_{\geq k}(A)$ and $tr(\tilde{\nu}')\in$ *NonFaulty*(A) with the same observation, A is not (O,k)-diagnosable which contradicts the assumption. Hence $f(O)$ must be winning. ■

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:

- $S=\{\boldsymbol{\pi}_{/\Sigma}(tr(\rho))\,|\,\rho\in Out(G_A,f)$ *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 $\rho\in Out(G_A,f)$ with $\rho=q_0\xrightarrow{X_0}q_0^1\xrightarrow{\varepsilon^*}q_0^{n_0}\xrightarrow{\lambda_1}q_1\xrightarrow{X_1}q_1^1\xrightarrow{\varepsilon^*}q_1^{n_1}\xrightarrow{\lambda_2}q_2\cdots q_{k_1}\xrightarrow{\varepsilon^*}q_{k-1}^{n_{k-1}}\xrightarrow{\lambda_k}q_k$ with each $q_i\in S_1$, $q_i^j\in S_2$, $v=\boldsymbol{\pi}_{/\Sigma}(tr(\rho))$, 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 $v=\boldsymbol{\pi}_{/\Sigma}(tr(\rho))$.

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

Proof: We first have to prove that $O(f)$ (more precisely L) is well defined. Assume $v=\boldsymbol{\pi}_{/\Sigma}(tr(\rho))$ and $v=\boldsymbol{\pi}_{/\Sigma}(tr(\rho'))$. 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 $\ell\notin f(\boldsymbol{\pi}_{/\Sigma}(tr(\rho)))$ for any ρ s.t. $v=\boldsymbol{\pi}_{/\Sigma}(tr(\rho))$ because f is trace-based. Thus $\delta(v,\ell)=v$. ■

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 $\nu\in$ *Faulty*$_{\geq k}(A)$ and $\nu'\in$ *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 $Out(G_A,f)$ 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 λ_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 λ_1

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

The result on G_A (Appendix A) 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 $\mathcal{F}_A(w) = en(\Delta_A(s_0, w))$.

5.3 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 $O = (S, s_0, \Sigma, \delta, L)$ we let $L(\rho)$ be the set $L(\delta(s_0, \rho))$. Given a run $\rho \in \mathit{Runs}(A)$, we recall that $O(\rho)$ is the observation of ρ by O. Assume $O(\rho) = a_0 \cdots a_k$. Let $\overline{\rho} = L(\varepsilon).\varepsilon.L(a_0).a_0.\cdots L(O(\rho)(k)).a_k$ i.e. $\overline{\rho}$ contains the history of what O has chosen to observe at each step and the events that occurred.

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

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

Assume A is (Σ, k)-diagnosable. Then there is an observer O s.t. A is (O, k)-diagnosable because the constant observer that observes Σ 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 $L(\delta(s_0, w)) \in \mathcal{F}_A(\overline{w})$ 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 ν of G_A. By definition of $\overline{w}$, $\boldsymbol{\pi}_{/\Sigma}(\overline{w}) = w$. By definition of $f(O)$, $f(O)(\overline{w}) = L(\delta(s_0, \boldsymbol{\pi}_{/\Sigma}(\overline{w}))) = L(\delta(s_0, w))$ and thus $L(\delta(s_0, w)) \in \mathcal{F}_A(\overline{w})$.

Conversely, assume O is such that $\forall w \in \Sigma^*, L(s_0, w) \in \mathcal{F}_A(\overline{w})$. 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 $w = \boldsymbol{\pi}_{/\Sigma}(tr(\rho))$. 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 A). The size of G_A is linear in $|A|$ and exponential in the size of Σ and k i.e. $|G_A| = O(|A|.2^{|\Sigma|}.2^k)$. 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 Σ and k.

Example 5 *For the automaton $\mathcal{A}$ of Fig. 1, we obtain the most permissive observer $\mathcal{F}_\mathcal{A}$ of Fig. 7. In the square states, the observer chooses what to observe and in the round states it moves according to what it observers. 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.*

6 Conclusion and Future Work

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.

We are currently investigating the following directions:

- 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 A 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 need. 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 [4].
- Another line of work is to define a notion of *cost* for dynamic observers. This can be done and an optimal observer can be computed as it is reported in [1].

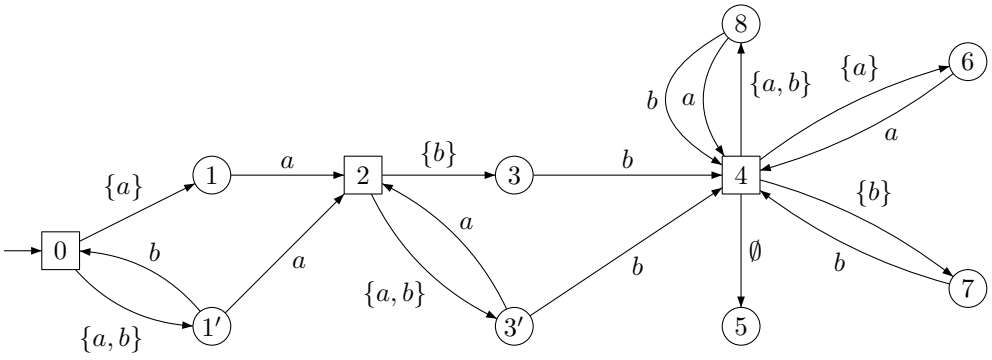

Figure 7. Most Permissive Observer for the Automaton $\mathcal{A}$ of Fig. 1

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 Cedes, 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] R. Debouk, S. Lafortune, and D. Teneketzis. On an optimization problem in sensor selection. *Discrete Event Dynamic Systems*, 4(12), November 2004.

[4] 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.

[5] 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.

[6] 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.

[7] P. Ramadge and W. Wonham. Supervisory control of a class of discrete event processes. *SIAM J. Control Optim.*, 25(1), January 1987.

[8] 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.

[9] 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.

[10] J.N. Tsitsiklis. On the control of discrete event dynamical systems. *Mathematics of Control, Signals and Systems*, 2(2), 1989.

[11] 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.

A 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 Σ_1) and hands it over to player 2. Player 2 can play two types of moves: invisible moves (ε is the invisible action) and visible moves (Σ_2-actions). If Player 2 plays an invisible move ε, 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:

Definition 6 (Two-Player ε-Games) *A* two-player ε-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;*
- *Σ_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 $q\, a\, q'$. 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 ε transitions, G is an alternating fully-observable two-player game (we use the term two-player game in this case).

Definition 7 (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 $q_0 \xrightarrow{\ell_1 \ell_2 \cdots \ell_n} q_n$ if $q_0 \ell_1 q_1 \cdots \ell_n q_n$ 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 ρ, denoted tr(ρ) is the sequence*[5] $\ell_0 \ell_1 \cdots \ell_n \cdots$.

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

[5] As ε also stands for the empty word, a trace does not contains ε.

Definition 8 (Player 1 Strategy) *A strategy for player* 1 *is a mapping f : Runs$_1^*(G) \to \Sigma_1$. A strategy f is* trace-based *if for all $\rho, \rho' \in$ Runs$_1^*(G)$, tr$(\rho) =$ tr(ρ') implies $f(\rho) = f(\rho')$. A strategy f is* memoryless *if tgt$(\rho) =$ tgt(ρ') implies $f(\rho) = f(\rho')$.*

Definition 9 (outcome) *Given a strategy f for player* 1*, the outcome $Out(G, f)$ of the game G under f is the set of plays*

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

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

Definition 10 (Objective) *An* objective *ϕ for Player 1 is a subset of $(Q_1 \cup Q_2)^\omega \cup (Q_1 \cup Q_2)^*$.*

Given a pair (G, ϕ), 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 ε-games asks the following:

Problem 6 (Control Problem)
INPUT: *A two-player ε-game G, an objective ϕ.*
PROBLEM: *Is there a winning strategy for (G, ϕ) ?*

ϕ 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 ε-game and ϕ a safety objective we say that the pair (G, ϕ) is a safety game. F is the set of *safe* states. To solve safety two-player ε-games we define the the *Cpre* operator [9]:

$$\begin{aligned} \mathit{Cpre}(S) &= \{s \in Q_1 \mid \exists \sigma_1 \in \Sigma_1 \mid \delta(s, \sigma_1) \in S\} \quad (8) \\ &\cup \{s \in Q_2 \mid \forall \sigma_2 \in \Sigma_2^{\varepsilon} \mid \delta(s, \sigma_2) \subseteq S\} \quad (9) \end{aligned}$$

It is well-known [9] that iterating *Cpre* and computing the fixpoint *Cpre*$^*(F)$ gives the set of winning states[6] of the game. In case G is finite this computation terminates and can be done in linear time for safety games [9]. 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, ε 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*[7] $\mathcal{F} : Q_1 \cap$ *Cpre*$^*(F) \to 2^{\Sigma_1} \setminus \emptyset$ 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, ϕ) 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.

The problem we want to solve is the following:

[6] Notice that by definition of *Cpre*, Player 1 cannot win by refusing to play.

[7] According to Definition 8, it is not a strategy as it prescribes a set of moves for a given state instead of one move.

Problem 7 (Trace-Based Control)
INPUT: *a game G, an objective ϕ.*
PROBLEM: *Is there a trace-based winning strategy for (G, ϕ) ?*

This problem is more demanding than the usual Control Problem 6 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 7, we reduce it to a problem of a fully-observable two player game.

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

$$\mathit{Next}_2(s, \sigma) = \{s' \mid s \xrightarrow{\varepsilon^*} s_1 \xrightarrow{\sigma} s'\} \tag{10}$$

It follows that if $\sigma \in \Sigma_2$, $\mathit{Next}_2(s, \sigma) \subseteq Q_1$ and if $\sigma = \varepsilon$, $\mathit{Next}_2(s, \sigma) \subseteq Q_2$. Let $\sigma \in \Sigma_1$ and $Q \subseteq Q_1$. We define

$$\mathit{Next}_1(Q, \sigma) \quad = \quad \{s' \mid s' = \delta(s, \sigma) \textit{ with } s \in Q\} \tag{11}$$

We address Problem 7 where ϕ is safety objective i.e. $\phi = F^\omega \cup F^*$. To solve (G, ϕ), we build a fully-observable two player game (G_H, F_H) (no ε transitions) such that:

Theorem 7 *There is a (standard) winning strategy in (G_H, F_H) iff there is a trace-based winning strategy in (G, F).*

Definition 11 *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 $W_1 = (2^{Q_1} \cup \bot_1)$ are the Player 1 states, $W_2 = (2^{Q_2} \cup \bot_2)$ are Player 2 states;*
- *$s_0 = \{q_0\}$,*
- *$\Sigma' = \Sigma_1 \cup \Sigma_2^u$ where u is a fresh name, and Σ_1 is the set Player 1 moves and Σ_2^u the set of 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 $S' = \mathit{Next}_1(S, \sigma)$ if for all $s \in S, \sigma \in \mathit{en}(s)$ and otherwise $S' = \bot_2$;*
  - *C_2: $S \subseteq Q_2, \sigma \in \Sigma_2$, $S' = \mathit{Next}_2(S, \sigma)$ and $S' \neq \emptyset$;*
  - *C_3: $S \subseteq Q_2, \sigma = u$, $\mathit{Next}_2(S, \varepsilon) \cap F \neq \emptyset$ and $S' = \bot_1$.*

We let $F_H = \{Q \in S \mid Q \subseteq F\}$ i.e. F_H be the set of safe states for G_H. $\bot_1$ and $\bot_2$ 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 $\mathit{tr}(\beta(w)) = w$ and a unique last state $\Delta(s_0, w) = s'$.*

From the proof of Theorem 7 (see Appendix B of [1]) we obtain an algorithm for Problem 7 and as the size of G_H is exponential in the size G:

Theorem 8 *Problem 7 is in EXPTIME.*

Given G_H and F_H we can compute the most permissive strategy $\mathcal{F}_H$. Given $\mathcal{F}_H$ we define the mapping $\mathcal{F}$ on $\mathit{Runs}_1^*(G)$: $\mathcal{F}(\rho) = \mathcal{F}_H(\mathit{tgt}(\beta(\mathit{tr}(\rho))))$.

Theorem 9 *$\mathcal{F}$ is the most permissive trace-based strategy for G.*

Corollary 1 *The most permissive trace-based strategy $\mathcal{F}$ for (G, ϕ) 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 $\mathcal{F}_H(q)$.