Active Diagnosis with Observable Quiescence

Stanislav Böhm, Stefan Haar, Serge Haddad, Piotr Hofman, and Stefan Schwoon

October 2015

Research report LSV-15-01

Laboratoire Spécification & Vérification

École Normale Supérieure de Cachan
61, avenue du Président Wilson
94235 Cachan Cedex France

Active Diagnosis with Observable Quiescence

Stanislav Böhm[1], Stefan Haar[2,3], Serge Haddad[2,3],
Piotr Hofman[2], and Stefan Schwoon[2,3]

Address [1] IT4 Innovations, National Supercomputing Center, Ostrava, Czech Republic
[2] LSV (CNRS & ENS Cachan) & INRIA, Cachan, France
[3] Inria, France

Abstract Active diagnosis of a discrete-event system consists in controlling the system such that faults can be detected. Here we extend the framework of active diagnosis by introducing modalities for actions and states and a new capability for the controller, namely observing that the system is quiescent. We design a game-based construction for both the decision and the synthesis problems that is computationally optimal. Furthermore we prove that the size and the delay provided by the active diagnoser (when it exists) are almost optimal.

1 Introduction

Diagnosis and control are important tasks in managing discrete-event systems (DES). In this paper, we contribute to the study of *active diagnosis*, which combines the two aspects: In a system whose events are partially observable, a controller observing an ongoing execution is charged with diagnosing whether a certain event, usually called *fault* and not directly observable, has happened or not. To this end, the controller may intervene and restrict the behaviour of the system in precisely defined ways. The active-diagnosis problem is to determine whether it is possible to control the system in such a way that diagnosis is always possible and, if so, synthesize a corresponding controller.

The active-diagnosis problem for DES was first studied in [8]. More recently, [6] proposed a new construction for active diagnosers based on automata and game theory that is provably optimal w.r.t. the size of the computed controller and the computational complexity for building it. Moreover, a number of variations have been studied. They range from the selection of minimal sets of observable labels that make the system diagnosable [3], to online aspects that either turn on and off sensors [9, 3] or modify an action plan [4] in order to reduce the amount of ambiguity, and to the case of probabilistic systems [1]. See also [2] for an extensive survey of DES problems.

In this work, we extend the frameworks of [8, 6]. Most importantly, we consider the case where the controller is able to observe that the system is quiescent and to exploit such observations for active diagnosis. While we believe that such an extension is natural and worth studying, it has a surprisingly large effect on the formal framework for active diagnosis. Notably, one of the tasks of the controller is, given the stream of observations σ, to decide whether σ indicates that a fault has happened. In [8, 6], this decision depends on σ alone, whereas in our framework it also depends on the control exercised during the execution that produced σ.

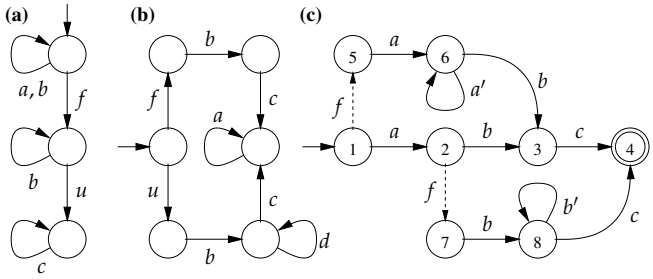

Figure 1: Examples of LTS. In (c), lazy events are indicated by dashed arrows and idle states by a double boundary.

We will present details of the model together with motivating examples in Section 2. Section 3 presents our framework for active diagnosis, while in Section 4 we show how to solve the active-diagnosis problem with the aforementioned additions, and Section 5 establishes worst-case lower bounds for certain aspects.

2 The Model

Before giving a formal definition of our framework, we discuss some motivating examples for active diagnosis and our extensions. As in [8, 6], we consider systems whose events can be *observable* or *unobservable*; a controller observing an ongoing execution of the system will only see its observable events. Consider Figure 1 (a). As in all examples that follow and unless mentioned otherwise, all events except f and u are observable, where f represents the fault. In a purely passive diagnosis setting, that system is considered *undiagnosable*: an observer seeing only a stream of bs cannot decide whether f has happened or not. However, suppose that event b is *controllable*, meaning that the observer can enable or disable it. Then, after seeing a number of bs, the observer may temporarily disable b, and the next observable action of the system must be a or c, revealing whether the fault has happened or not.

Quiescence

The first of our extensions w.r.t. [8, 6] concerns the ability of the diagnoser to deal with temporary quiescence of the system. The concept of quiescence is well-established and used in *conformance testing*, see e.g. Tretmans [10, 11, 12], where it is used to observe that the system under test will not produce any output unless provided with additional input.

We shall construct controllers with the capacity to observe that the system is quiescent and react by re-enabling some events. Quiescence can be exploited in active diagnosis, as the example in Figure 1 (b) shows, where only c is controllable. That system is considered non-actively-diagnosable in the frameworks of [8, 6], which are language-based: the upper half of the system can only do (a prefix of) fbc, which is undistinguishable from a sequence possible in the lower half; yet, a controller cannot prevent the system from entering the upper half with uncontrollable actions fb. Slightly more formally, a finite execution prefix is called *ambiguous* in [8, 6] if it contains a fault (resp. does not contain one) and is observationally

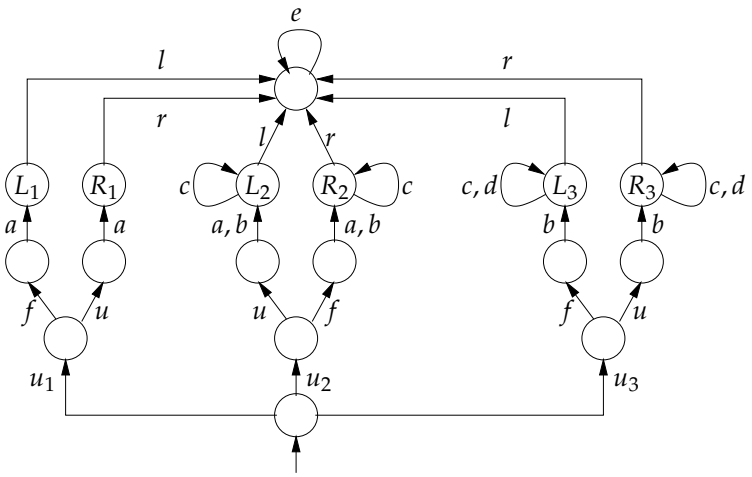

Figure 2: Example that is undiagnosable for previous frameworks even with arbitrary additional δ-loops.

equivalent to a prefix that does not contain a fault (resp. contains one). Then, a system is called actively diagnosable if it can be controlled so as to remain in a non-ambiguous sublanguage of the system. The reader can easily see that the only unambiguous execution prefixes of Figure 1 (b) are of the form ubd^mca^n, $m \geq 1$, $n \geq 0$; however, the uncontrollable action f in the beginning makes it impossible to remain in this set.

Suppose now another approach in which a controller blocks c after seeing b and then has the capacity to observe that either d happens (i.e. the system is in q') or that the system is quiescent (i.e. the current state is q). The controller can then issue the corresponding diagnosis verdict and, e.g., re-enable c to let the system continue.

Semantics of Quiescence

We remark that quiescence is semantically different from extending the system with some event δ symbolising "passage of time". While, e.g., Figure 1 (b) can be made actively diagnosable in the sense of [8, 6] by adding a δ-loop to q, our method has at least two advantages: First, it does not require the designer of a system to add artefacts that are merely required by deficiencies of some analysis method. Secondly, such a transformation is not always possible. Consider the system shown in Figure 2, where a, b, c, d, e, l, r are observable and c, d, l, r controllable. We shall see that this system is (i) diagnosable in our framework, (ii) undiagnosable in the frameworks of [8, 6], and (iii) no addition of δ-loops can render the system diagnosable for [8, 6].

As for (i), the first observation will be either a or b, without any possibility of intervention for the controller. Also, note that if the first two observations are one of al, ar, bl, or br, then the controller will be forever unable to decide whether a fault has happened. Let us first suppose that the first observation is a, then the system is known to be in one of $\{L_1, R_1, L_2, R_2\}$. The controller must now block both l and r. If the next observation is c, then the set of possible states shrinks to $\{L_2, R_2\}$; if the next observation is quiescence, then to

$\{L_1, R_1\}$. (Note that here we implicitly assume that c necessarily happens when it is the only available event. Later we will introduce the notion of *lazy* and *eager* events to let the designer specify this aspect of the system's behaviour.) Having observed either c or quiescence, the controller can then disable c and re-allow l and r, and after the next observation, knows precisely which path the system has taken. Likewise, if the first observation is b; then the system is known to be in $\{L_2, R_2, L_3, R_3\}$. The controller then blocks c, which is possible in all these states, as well as l and r. Now the next observation will be either d or quiescence; the controller then proceeds by blocking c, d and has complete information about the path taken by the system after observing either l or r.

Concerning (ii), the only unambiguous sequences possible in the system are $u_2 x a c^m y e^n$ and $u_3 x b d^m y e^n$, for any choice of $x \in \{f, u\}$, $y \in \{l, r\}$ and $m \geq 1$, $n \geq 0$. However, the controller cannot prevent the system from beginning with, e.g., $u_1 x a$ or $u_2 x b$, so the system is undiagnosable in the sense of [8, 6].

Concerning (iii), we shall show the following: Let S be a subset of the states in Figure 2, and $\mathcal{A}_S$ the system where every state in S has an additional loop with an observable event δ. Then, for any choice of S, $\mathcal{A}_S$ remains non-actively diagnosable for [8, 6], independently of whether δ is controllable or not. We prove this claim by contradiction: suppose that for some set S, $\mathcal{A}_S$ *is* actively diagnosable. To begin with, we can easily see that the choice for S can be limited to a subset of $\{L_1, R_1, L_2, R_2, L_3, R_3\}$; all other states contain at least one uncontrollable action and can therefore choose to never use the δ-loop. Now suppose that S does not contain state L_1. Then every execution starting with the uncontrollable prefix $u_1 f a$ is ambiguous, notably $u_1 f a l$ is observationally equivalent to $u_2 u a l$. The cases for states R_1, L_2, and R_2 are analogous, for the uncontrollable prefixes $u_1 u a$, $u_2 u b$, and $u_2 f b$, respectively. Thus, $\mathcal{A}_S$ must have δ-loops in all four states. But then $\mathcal{A}_S$ is again non-actively diagnosable: if the first observation is a, then δ is available in any possible state, and after l or r the system remains forever undiagnosable.

Put simply, Figure 2 exhibits a fundamental semantic difference between quiescence (which only happens when no other action is available), and a loop signifying "passage of time", which is otherwise treated like any other action. Treating quiescence directly thus genuinely extends the power of active diagnosers.

Lazy actions and idle states

Moreover, quiescence allows us to meaningfully distinguish two types of events that we call *eager* and *lazy*. An eager event enabled in some state eventually happens unless another event pre-empts it; typically, eager events are those happening during the normal course of the system. By contrast, a *lazy* event represents an entirely non-deterministic action, e.g. a fault or input from the environment. Thus, quiescence can be observed in a state even when one or more lazy actions are enabled.

Figure 1 (c) represents a three-stage production process with actions a, b, c, all controllable. The fault f is lazy, indicated by dashed arcs. If all goes well, the system proceeds from state 1 to state 4 with abc. However, steps a and b can be erroneous, which must be tested for before proceeding with b resp. c. E.g., after seeing a, the system could be in states 2 or 6, and the controller will temporarily disable b to see whether it observes a' or quiescence. However, if

the system is in state 2, this temporary disabling does not necessarily provoke the lazy event f. In our framework, the system can reach state 4 without fault while enabling a controlling observer to verify that the execution was fault-free. By contrast, [8, 6] consider all events to be eager; for them, the system is actively diagnosable but at the price of eliminating all non-faulty executions.

Finally, notice that state 4 in Figure 1 (c) has a double boundary. We use this to indicate so-called *idle* states, and we forbid controllers to observe quiescence forever unless the system is in such an idle state. In the example, this allows the designer of the system to formulate the requirement that an active diagnoser either detect a fault or allow the production to run to completion. This concept generalizes the liveness requirement made in [6], where all runs must be infinite. We remark that a similar concept called *marked* states is discussed for controllability problems in [2]. However, in our case quiescence may be a temporary state of the system from which it may re-awaken, e.g. following a lazy event triggered by the environment.

3 Definitions

This section serves to provide formal notations related to discrete-event systems and active diagnosis. As mentioned in Section 2, we shall study discrete-event systems whose events, here called *actions*, have additional properties, such as being observable, controllable, or eager, and we will successively introduce the notions related to these concepts. Given a set X, we use the standard notations X^* (resp. X^+, X^ω) to denote the finite (resp. non empty finite, infinite) sequences over X; the empty sequence is denoted ε, and the length of a finite sequence σ by $|\sigma|$. Given a sequence $\sigma = a_1 \cdots a_n \in X^+$, $last(\sigma)$ denotes a_n. Given a sequence $\sigma = a_1 \cdots \in X^* \cup X^\omega$ and $1 \leq i \leq j \leq |\sigma|$, $\sigma[i]$ denotes a_i and $\sigma[i,j]$ denotes $a_i \cdots a_j$.

Labeled transition systems

Our main model will be an extension of labeled transition systems:

Definition 1 *A* labeled transition system *(LTS) is a tuple $\mathcal{A} = \langle Q, q_0, \Sigma, T \rangle$ where Q is a set of states with initial state $q_0 \in Q$, Σ is a finite set of actions, and $T \subseteq Q \times \Sigma \times Q$ is the set of transitions. An LTS $\mathcal{A}$ is deterministic if for every pair $q \in Q, a \in \Sigma$ there is at most one q' such that $\langle q, a, q' \rangle \in T$.*

If $\langle q, a, q' \rangle \in T$, we write $q \xrightarrow{a} q'$ and say that a is *enabled* in q; for a deterministic automaton we also write $T(q,a) = q'$. The set of enabled actions in state q is denoted $en(q)$. An infinite *run* over the word $\sigma = a_1 a_2 \ldots \in \Sigma^\omega$ is an alternating sequence of states and actions $(q_i a_{i+1})_{i \geq 0}$ such that $q_i \xrightarrow{a_{i+1}} q_{i+1}$ for all $i \geq 0$, and we write $q_0 \overset{\sigma}{\Rightarrow}$ if such a run exists. A finite run over $w \in \Sigma^*$ is defined analogously, and we write $q \overset{w}{\Rightarrow} q'$ if such a run ends at state q'. A state q is *reachable* if there exists a run $q_0 \overset{w}{\Rightarrow} q$ for some w.

Definition 2 (languages) *Let $\mathcal{A} = \langle Q, q_0, \Sigma, T \rangle$ be an LTS. The finite and infinite language of $\mathcal{A}$ are defined by $\mathcal{L}^*(\mathcal{A}) = \{ w \in \Sigma^* \mid \exists q : q_0 \overset{w}{\Rightarrow} q \}$ and $\mathcal{L}^\omega(\mathcal{A}) = \{ \sigma \in \Sigma^\omega \mid q_0 \overset{\sigma}{\Rightarrow} \}$.*

Partially observable controllable systems

We now define partially observable controllable systems (POCS) on which we shall perform active diagnosis. Syntactically, a POCS $\mathcal{S}$ is an LTS $\mathcal{A}$ enlarged with three binary partitions: an action may be (1) *observable* (in Σ_o) or *unobservable* (in Σ_{uo}), (2) *controllable* (in Σ_c) or *uncontrollable* (in Σ_{uc}), and (3) *eager* (in Σ_e) or *lazy* (in Σ_ℓ). We require that unobservable actions are uncontrollable. Below, we shall define the semantics of a POCS as a new LTS $\mathcal{S}_{cont}$ depending on a controller *cont*. Intuitively, during an execution of system $\mathcal{A}$, a controller may forbid a subset of the controllable actions based on the observable actions seen so far, thereby restricting the behaviour of $\mathcal{A}$. This implies that $\mathcal{A}$ must be *convergent*, i.e. there is no infinite run with a suffix of unobservable actions: $\mathcal{L}^\omega(\mathcal{A}) \cap \Sigma^*\Sigma_{uo}^\omega = \emptyset$. Occasionally, the control may lead to a situation in which the system reaches a state where all non-blocked actions are lazy. In this situation, either one such action occurs or the controller sees that "nothing is happening", represented by a special observation symbol δ. We say that the system is in a *quiescent* state (not to be confused with a deadlock state). The controller may then once again change the set of allowed actions. The set *Idle* indicates the states in which the system may legitimately remain quiescent forever; in particular, a correct controller should not block all eager actions indefinitely when $\mathcal{A}$ is in a state $q \notin Idle$. The next two definitions formalize POCS and their semantics.

Definition 3 *A* partially observable controllable system *(POCS) is a tuple* $\mathcal{S} = \langle \mathcal{A}, \Sigma_o, \Sigma_c, \Sigma_e, f, Idle \rangle$*, where* $\mathcal{A} = \langle Q, q_0, \Sigma, T \rangle$ *is a convergent LTS with* $\Sigma_o, \Sigma_e \subseteq \Sigma$ *such that* $Idle \subseteq Q$*,* $\Sigma_c \subseteq \Sigma_o$*, and* $f \in \Sigma \setminus \Sigma_o$ *is a distinguished action called* fault*.*

As previously discussed, we write $\Sigma_{uo}, \Sigma_{uc}, \Sigma_\ell$ for the complements of $\Sigma_o, \Sigma_c, \Sigma_e$. Let us denote $\Xi := \Sigma \cup \{\delta\}$ and $\Xi_o := \Sigma_o \cup \{\delta\}$ the set of (observable) actions extended with δ, and let $\sigma \in \Xi^*$. The projection $\mathcal{P}(\sigma)$ erases all letters not from Ξ_o, more precisely $\mathcal{P}(\varepsilon) = \varepsilon$, and $\mathcal{P}(\sigma a)$ equals $\mathcal{P}(\sigma)a$ if $a \in \Xi_o$ and $\mathcal{P}(\sigma)$ otherwise. For $\sigma \in \Xi^\omega$, its projection is the limit of the projections of its finite prefixes. When using the projection to another subset X, we write $\mathcal{P}_X$.

A *controller* for $\mathcal{S}$ is a mapping $cont : \Xi_o^* \to 2^{\Sigma_c}$.

Definition 4 (Controlled system) *Let* $\mathcal{S}$ *be a POCS and cont a controller. Then* $\mathcal{S}_{cont} := \langle Q_{cont}, q_{0cont}, \Xi, T_{cont} \rangle$ *is defined as the smallest LTS satisfying:*

(i) $q_{0cont} := \langle \varepsilon, q_0 \rangle \in Q_{cont}$*;*

(ii) *if* $\langle \sigma, q \rangle \in Q_{cont}$*,* $a \in cont(\sigma) \cup \Sigma_{uc}$*, and* $q \xrightarrow{a} q'$*, then* $\langle \sigma\mathcal{P}(a), q' \rangle \in Q_{cont}$ *and* $\langle \langle \sigma, q \rangle, a, \langle \sigma\mathcal{P}(a), q' \rangle \rangle \in T_{cont}$*;*

(iii) *if* $\langle \sigma, q \rangle \in Q_{cont}$ *and* $cont(\sigma) \cap en(q) \subseteq \Sigma_\ell$ *then* $\langle \sigma\delta, q \rangle \in Q_{cont}$ *and* $\langle \langle \sigma, q \rangle, \delta, \langle \sigma\delta, q \rangle \rangle \in T_{cont}$*.*

An *observable sequence* is an item of Ξ_o^*. An *observed sequence* is an item of $\Lambda(cont) := \{ \sigma \mid \langle \sigma, q \rangle \in Q_{cont} \}$.

Ambiguity

A finite or infinite word σ over Σ (resp. Ξ) is *faulty* if it contains an occurrence of f; otherwise it is called *correct*. Given an observed sequence σ, the aim of diagnosis is to determine whether a fault has surely occurred. The ambiguous sequences are exactly the observed sequences where diagnosis is not yet possible.

Definition 5 (ambiguous and surely faulty sequence) *Let $\mathcal{S}_{cont}$ be a controlled system, $\sigma_1, \sigma_2 \in \mathcal{L}^\omega(\mathcal{S}_{cont})$ be two sequences and $\sigma \in \Xi_o^\omega$ such that: (i) $\mathcal{P}(\sigma_1) = \mathcal{P}(\sigma_2) = \sigma$, (ii) σ_1 is correct, and (iii) σ_2 is faulty. Then σ is called* ambiguous *in $\mathcal{S}_{cont}$, and the pair $\langle \sigma_1, \sigma_2 \rangle$ is a* witness *for the ambiguity of σ. Ambiguous finite sequences are defined analogously. A sequence $\sigma' \in \Xi_o^*$ is* surely faulty *in $\mathcal{S}_{cont}$ for all $\sigma \in \mathcal{L}^*(\mathcal{S}_{cont})$ such that $\mathcal{P}(\sigma) = \sigma'$, σ is faulty.*

Active diagnosability

In the diagnosis framework, the goal of the controller is to make the system diagnosable, and to perform diagnosis. Thus, an *active diagnoser* is a controller equipped with a diagnosis function. The active dignoser must (1) eliminate ambiguity, (2) detect fault and (3) does not leave the system stuck forever in a non-idle quiescent state.

Definition 6 (Active Diagnoser) *Let $\mathcal{S}$ be a POCS and $h = \langle cont, diag \rangle$, where cont is a controller and diag a mapping from $\Lambda(cont)$ to $\{\bot, \top\}$. We call h* active diagnoser *for $\mathcal{S}$ iff:*

1. *$\mathcal{S}_{cont}$ does not contain any infinite ambiguous sequence;*
2. *$diag(\sigma) = \top$ if and only if σ is surely faulty in $\mathcal{S}_{cont}$;*
3. *For all infinite runs $(s_i a_{i+1})_{i \geq 0}$ of $\mathcal{S}_{cont}$, if there exists i_0 with $a_i = \delta$ for all $i \geq i_0$ then $s_{i_0} = \langle \sigma', q \rangle$ for some σ' and $q \in Idle$.*

*For $k \geq 1$, h is called a k-*active diagnoser *if for all $\sigma = \sigma' f \sigma'' \in \mathcal{L}^*(\mathcal{S}_{cont})$ with $|\mathcal{P}(\sigma'')| \geq k$, $diag(\mathcal{P}(\sigma)) = \top$, i.e. every fault is diagnosed after at most k observations. The minimal k s.t. h is a k-active diagnoser is called the* delay *of h. We call $\mathcal{S}$ (k-)*actively diagnosable *if a (k-)active diagnoser exists, and the minimal such k the* minimal delay of the language recognized by $\mathcal{S}$.

An active diagnoser does not necessarily have a finite delay [6]. However, we will see that if $\mathcal{S}$ is actively diagnosable, there does exist a k-active diagnoser for some k.

We are now in a position to formally state the relevant problems for active diagnosis. Let $\mathcal{S}$ be a POCS with finitely many states. We are interested in:

- the *active diagnosis decision problem*, i.e. decide whether $\mathcal{S}$ is actively diagnosable;
- the *synthesis problem*, i.e. decide whether $\mathcal{S}$ is actively diagnosable and in the positive case build an active diagnoser.
- the *minimal-delay synthesis problem*, i.e. decide whether $\mathcal{S}$ is actively diagnosable and in the positive case build an active diagnoser with minimal delay.

We introduce the notion of *pilot* as a finite representation of an active diagnoser.

Definition 7 (pilot) *Let $\mathcal{S}$ be a POCS. Then $\mathcal{C} = \langle \mathcal{B}_\mathcal{C}, cont_\mathcal{C}, diag_\mathcal{C} \rangle$ is called* pilot *for $\mathcal{S}$ if $\mathcal{B}_\mathcal{C} = \langle Q^c, q_0^c, \Xi_o, T^c \rangle$ is a deterministic LTS, $\langle cont_\mathcal{C}, diag_\mathcal{C} \rangle : Q^c \to 2^{\Sigma_c} \times \{\bot, \top\}$, are labellings such that for all $q \in Q^c$, and for all $a \in cont_\mathcal{C}(q) \cup \{\delta\} \cup \Sigma_{uc} \setminus \Sigma_{uo}$, there is an outgoing edge labelled by a. Let $h_\mathcal{C} = \langle cont, diag \rangle$ associated with $\mathcal{C}$ be defined by $cont(\sigma) = cont_\mathcal{C}(q)$ and $diag(\sigma) = diag_\mathcal{C}(q)$ for all $\sigma \in \Lambda(cont)$, where q is the unique state such that $q_0^c \stackrel{\sigma}{\Rightarrow} q$. Then $\mathcal{C}$ is a (k-)active diagnoser for $\mathcal{S}$ if $h_\mathcal{C}$ is one.*

4 Diagnoser construction

We simultaneously solve the decision and synthesis problems. We shall try to construct a pilot-based active diagnoser for a POCS $\mathcal{S}$. The construction succeeds iff $\mathcal{S}$ is actively diagnosable. According to Definition 6, the main challenges in building an active diagnoser are to ensure that (i) the controlled system does not get stuck forever in a non-idle quiescent state, (ii) the controller excludes the ambiguous sequences, and (iii) diagnosis information is provided.

The approach in [6] consisted of two stages. First one builds a Büchi automaton that accepts the infinite unambiguous observed sequences of $\mathcal{A}$. Then using this automaton, one builds a Büchi game where the *Control* player chooses the allowed controllable actions and then the *Environment* player selects the next observable action. The correctness of this approach partly relies on the fact that given two controls *cont* and *cont'* and an observed sequence σ of both $\mathcal{S}_{cont}$ and $\mathcal{S}_{cont'}$, σ is ambiguous in $\mathcal{S}_{cont}$ iff it is ambiguous in $\mathcal{S}_{cont'}$. This is no longer the case here. For instance, suppose that in Figure 1 (b) both c, d are controllable. Then $b\delta$ is ambiguous if the controller blocks both c and d, and unambiguous if the controller blocks only c.

Here, our solution consists in directly building a generalized Büchi game and taking into account the control that has already been performed to specify the relevant information that must be memorized to define the winning states.

Definition 8 (game) *A game (between two players called Control and Environment) is a tuple $\mathcal{G} = \langle V_C, V_E, E, v_0, \mathcal{P}_F \rangle$, where V_C are the vertices owned by Control, V_E are the vertices owned by Environment; $V_\mathcal{G} = V_C \uplus V_E$ denotes all vertices, and $v_0 \in V_C$ is an* initial vertex. *$E \subseteq V_\mathcal{G} \times V_\mathcal{G}$ is a set of directed edges such that for all $v \in V_C$ there exists $(v, w) \in E$, and $\mathcal{P}_F \subseteq 2^{V_\mathcal{G}}$ is a* winning condition.
A play *ρ is a sequence of $V_\mathcal{G}^\omega$ such that $\rho[0] = v_0$ and $\langle \rho[i], \rho[i+1] \rangle \in E$ for all $i \geq 0$; we call $\rho[0, k]$, for some $k \geq 0$, a* partial play *if $\rho[k] \in V_C$, and define $state(\rho[0, k]) := \rho[k]$. We write $Play^*(\mathcal{G})$ for the set of partial plays of $\mathcal{G}$. A play ρ is called* winning *(for Control) if, for all $V \in \mathcal{P}_F$, $\rho[i] \in V$ for infinitely many i.*

Definition 9 (strategy) *Let $\mathcal{G} = \langle V_C, V_E, E, v_0, \mathcal{P}_F \rangle$ be a game. A* strategy *(for Control) is a function $\theta \colon Play^*(\mathcal{G}) \to V_\mathcal{G}$ such that $\langle state(\xi), \theta(\xi) \rangle \in E$ for all $\xi \in Play^*(\mathcal{G})$. A play ρ* adheres *to θ if $\rho[i] \in V_C$ implies $\rho[i+1] = \theta(\rho[0, i])$ for all $i \geq 0$. A strategy is called* winning *if every play ρ that adheres to θ is winning.*

Let $M = \langle Q_M, q_M, V_\mathcal{G}, T_M \rangle$ be a complete deterministic LTS, where $FM(\xi)$ denotes the state $q \in Q_M$ such that $q_M \stackrel{\xi}{\Rightarrow} q$, and let $\alpha \colon Q_M \times V_C \to V_\mathcal{G}$ such that for all $q \in Q_M$ and $v \in V_C$

we have $\langle q, \alpha(q,v)\rangle \in E$. *The strategy* $\theta_{M,\alpha}$ *defined by* $\theta(\xi) = \alpha(\mathrm{FM}(\xi), state(\xi))$ *is called* finite-memory strategy *if* Q_M *is finite; it is called* one-bit strategy *if* $Q_M = \{0,1\}$.

In the games that we have defined, a play can only be stuck in a state of Environment and considered as losing for this player. Thus we do not consider finite maximal plays for defining the winning strategies of Control.

The controller we are looking for will memorize a tuple of subsets of states $\langle U, V, W, X\rangle$ with the following meaning. Whatever the subset, it represents possible states that have been reached *after the last observable action or that corresponds to a quiescent state w.r.t. the current control*. U represents the possible states reached by a *correct* run, and $V \uplus W$ represent the possible states reached by a *faulty* run. Among the latter, W represents the states for which the controller tries to solve the ambiguity with U, while V is some "waiting room"; the ambiguity between U and V will be resolved later. X represents a *subset* of the possible non-idle states reached by a run for which no action has been performed between the two last observations. The controller tries to discard these states either by observing that they could not occur or by allowing an urgent action. The other states with the same features will be handled later. We denote $S = \{\langle U, V, W, X\rangle \mid U, V, W \subseteq Q \wedge X \subseteq (U \cup V \cup W) \setminus Idle \wedge U \cup V \cup W \neq \emptyset \wedge V \cap W = \emptyset\}$, $Solved_1 = \{\langle U, V, W, X\rangle \in S \mid U = \emptyset \vee W = \emptyset\}$, $Solved_2 = \{\langle U, V, W, X\rangle \in S \mid X = \emptyset\}$, and $Reach(\langle U, V, W, X\rangle) = U \cup V \cup W$. The next definition describes how the controller updates its tuple once an observed action occurs (including the quiescent signal δ). The range of this function also includes the tuple $\langle\emptyset,\emptyset,\emptyset,\emptyset\rangle$ in order to capture the impossible observations.

Definition 10 (Knowledge update) *Let* $\mathcal{S}$ *be a POCS. Then* Δ, *the* knowledge transition partial function *from* $S \times 2^{\Sigma_c} \times (\Sigma_o \cup \{\delta\})$ *to* $S \cup \{\langle\emptyset,\emptyset,\emptyset,\emptyset\rangle\}$, *is defined for* $s = \langle U, V, W, X\rangle$, $\Sigma' \in 2^{\Sigma_c}$, $a \in \Sigma' \cup \{\delta\} \cup \Sigma_{uc} \setminus \Sigma_{uo}$ *by* $\Delta(s, \Sigma', a) := \langle U', V', W', X'\rangle$ *as follows.*
• *When* $a \neq \delta$, *let* $V_a = \{\, q' \mid q \in V, q \overset{\sigma a}{\Rightarrow} q', \sigma \in \Sigma_{uo}^* \,\} \cup \{\, q' \mid q \in U, q \overset{\sigma a}{\Rightarrow} q', \sigma \in \Sigma_{uo}^* f \Sigma_{uo}^* \,\}$. *Then:*

- $U' = \{\, q' \mid q \in U, q \overset{\sigma a}{\Rightarrow} q', \sigma \in (\Sigma_{uo} \setminus \{f\})^* \,\}$;
- *If* $W = \emptyset$ *then* $W' = V_a$
  else $W' = \{\, q' \mid q \in W, q \overset{\sigma a}{\Rightarrow} q', \sigma \in \Sigma_{uo}^* \,\}$;
- *If* $W = \emptyset$ *then* $V' = \emptyset$ *else* $V' = V_a \setminus W'$;
- $X' = \emptyset$.

• *When* $a = \delta$, *let* $Quiet_{\Sigma'} = \{\, q \in Q \mid en(q) \cap \Sigma_e \subseteq \Sigma_c \setminus \Sigma' \,\}$ *and* $V_\delta = \{\, q' \in Quiet_{\Sigma'} \mid q \in V, q \overset{\sigma}{\Rightarrow} q', \sigma \in \Sigma_{uo}^* \,\} \cup \{\, q' \in Quiet_{\Sigma'} \mid q \in U, q \overset{\sigma}{\Rightarrow} q', \sigma \in \Sigma_{uo}^* f \Sigma_{uo}^* \,\}$. *Then:*

- $U' = \{\, q' \in Quiet_{\Sigma'} \mid q \in U, q \overset{\sigma}{\Rightarrow} q', \sigma \in (\Sigma_{uo} \setminus \{f\})^* \,\}$;
- *If* $W = \emptyset$ *then* $W' = V_\delta$
  else $W' = \{\, q' \in Quiet_{\Sigma'} \mid q \in W, q \overset{\sigma}{\Rightarrow} q', \sigma \in \Sigma_{uo}^* \,\}$;
- *If* $W = \emptyset$ *then* $V' = \emptyset$ *else* $V' = V_\delta \setminus W'$;
- *If* $X = \emptyset$ *then* $X' = (Quiet_{\Sigma'} \cap Reach(s)) \setminus Idle$
  else $X' = Quiet_{\Sigma'} \cap X$.

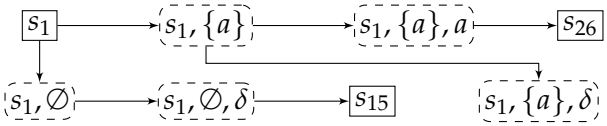

Figure 3: Excerpt of the Büchi game for the POCS of Figure 1 (c). Environment states are shown with dashed boundary. We use $s_1 := \langle\{1\},\emptyset,\emptyset,\emptyset\rangle$, $s_{15} := \langle\{1\},\emptyset,\{5\},\{1\}\rangle$, and $s_{26} := \langle\{2\},\emptyset,\{6\},\emptyset\rangle$.

Before formally defining the game, we discuss its intuition: Control and Environment play alternatingly; the Control chooses the allowed controllable actions based on its knowledge of the possible states, and the Environment chooses an action among those permitted by the Control. The controller states V_C are the tuples $\langle U, V, W, X\rangle$. Once the controller chooses a set of actions, the play moves to a state in $V_C \times 2^{\Sigma_c} \subseteq V_E$, where Environment selects an allowed observable action in Ξ_o and reaches a game state in $V_C \times 2^{\Sigma_c} \times \Xi_o \subseteq V_E$. This state has (1) either a single successor, a controller state whose tuple is given by the above update function when the tuple is *not empty*, (2) or none at all, when the action leads to an empty tuple, meaning that the behaviour was not possible. The generalized Büchi condition is given by $\{Solved_1, Solved_2\}$.

Definition 11 (controller-synthesis game) *Let $\mathcal{S}$ be a POCS. Then $\mathcal{G}(\mathcal{S}) := \langle V_C, V_E, E, v_0, \mathcal{P}_F\rangle$ denotes a game, where $V_C = S$, $v_0 = \langle\{q_0\},\emptyset,\emptyset,\emptyset\rangle$, $V_E = (V_C \times 2^{\Sigma_c}) \cup (V_C \times 2^{\Sigma_c} \times \Xi_o)$, and $E = E_1 \cup E_2 \cup E_3$ with*

- $E_1 = \{\, \langle s, \langle s, \Sigma'\rangle\rangle \mid s \in S,\ \Sigma' \in 2^{\Sigma_c} \,\}$;
- $E_2 = \{\, \langle\langle s, \Sigma'\rangle, \langle s, \Sigma', a\rangle\rangle \mid s \in S,$ $\Sigma' \in 2^{\Sigma_c}, a \in \Sigma' \cup \{\delta\} \cup \Sigma_{uc} \setminus \Sigma_{uo} \}$;
- $E_3 = \{\, \langle\langle s, \Sigma', a\rangle, s'\rangle \mid s' = \Delta(s, \Sigma', a) \}$.

and $\mathcal{P}_F = \{Solved_1, Solved_2\}$.

Example 1 *Figure 3 depicts an excerpt of the game for for POCS of Figure 1 (c). From the initial state, we have represented two control decisions: either Control allows $\{a\}$ or disallows all controllable actions. If Control chooses $\{a\}$, and Environment chooses action δ, the next state has no successor, since under this control, a is the single observable action, so Environment loses immediately. If Control chooses $\emptyset$ and Environment chooses action δ, the tuple of the reached state is $\langle\{1\},\emptyset,\{5\},\{1\}\rangle$ as f is lazy and may have occurred or not.*

We can now address the decision and synthesis problems. To this aim, we shall mainly exploit the following facts: (1) Generalized Büchi games can be solved in polynomial time (see, e.g., [5]), (2) a one-bit winning strategy (when the number of winning conditions is two) can always be chosen for Control if it wins and (3) there is a tight correspondence between winning strategies and active diagnosers.

Let $\xi \in Play^*(\mathcal{G}(\mathcal{S}))$ be a partial play. We define $word(\xi)$ as the observable actions played along ξ, i.e. $word(\varepsilon) = \varepsilon$, $word(\xi v) = word(\xi)$ if $v \notin V_C \times 2^{\Sigma_c} \times \Xi_o$, and $word(\xi\langle v, \Sigma', a\rangle) =$

$word(\xi)a$. In a similar way, $states(\xi)$ are the states of V_C touched along ξ, formally $states(\xi) = \mathcal{P}_{V_C}(\xi)$. We naturally extend these notions to plays ρ.

Definition 12 (from control to strategy and play) *Let cont be a controller for $\mathcal{S}$. The strategy θ_{cont} of the game $\mathcal{G}(\mathcal{S})$ is defined as follows. Let $\xi \in Play^*(\mathcal{G}(\mathcal{S}))$ be a partial play ending in a state of the controller. Then $\theta_{cont}(\xi) = cont(word(\xi))$. Let σ be an observed sequence of $\mathcal{S}_{cont}$. Then the play $\xi_{cont}(\sigma)$ is inductively defined by:*
• *If $\sigma = \varepsilon$ then $\xi_{cont}(\sigma) = \langle \{q_0\}, \emptyset, \emptyset, \emptyset \rangle$.*
• *If $\sigma = \sigma' a$, then $\xi_{cont}(\sigma) = \xi_{cont}(\sigma')\langle l, \Sigma' \rangle \langle l, \Sigma', a \rangle s'$, where $l := last(\xi_{cont}(\sigma'))$, $\Sigma' := cont(\sigma')$, and s' is the single successor of $\langle l, \Sigma', a \rangle$ in $\mathcal{G}(\mathcal{S})$.*

The following observation is straightforward. Let ξ be an infinite play that adheres to strategy θ_{cont}. Then $word(\xi)$ is an observed sequence of $\mathcal{S}_{cont}$.

Proposition 1 *If cont is a controller for $\mathcal{S}$ and σ an observed sequence of $\mathcal{S}_{cont}$, the following are equivalent:*

1. *the play $\xi_{cont}(\sigma)$ is winning for the controller;*
2. *σ is unambiguous and for all $\sigma' = q_0 a_1 \cdots a_n (q_n \delta)^\omega$ such that $\mathcal{P}(a_1 \cdots) = \sigma$, $q_n \in Idle$.*

Proof. Fix an observed sequence σ of $\mathcal{S}_{cont}$ and note:
$states(\xi_{cont}(\sigma)) = \langle U_0, V_0, W_0, X_0 \rangle) \cdots \langle U_i, V_i, W_i, X_i \rangle \cdots$.

To prove that (1) implies (2), assume first that σ is ambiguous; we show that the play is losing. Let (σ', σ'') be a pair of witnesses for σ. Because of σ', we have $U_i \neq \emptyset$ for all $i \geq 0$. Moreover, we will show that σ'' implies the existence of some i_0 such that for all $i \geq i_0$ we have $W_i \neq \emptyset$. These two facts together mean that the play is losing, since $\langle U_i, V_i, W_i, X_i \rangle \notin Solved_1$ for all $i \geq i_0$. So, let w be the minimal prefix of σ'' containing f, and let $|w|_{\Xi_o} = j$. Then clearly, $V_i \cup W_i \neq \emptyset$ for all $i > j$.

- Either $W_j = \emptyset$, i.e. the watchlist is empty after j observations; then the faulty run of $\mathcal{S}_{cont}$ for σ'' will be recorded in the watchlist after the next observation, and remain there; we take $i_0 := j + 1$.
- Or $W_j \neq \emptyset$; then the possibility of a fault will be recorded in the "waiting room". Then either the watchlist becomes empty at a later time, i.e. $W_{j'} = \emptyset$ for some $j' > j$, in which case the state associated with the faulty run for σ'' is transferred to the watchlist in the next step, and we take $i_0 := j' + 1$; or the watchlist never becomes empty (in which case there exists another faulty sequence with same observation), and we take $i_0 := j$.

Assume now that there is $\sigma' = q_0 a_1 \cdots a_n (q_n \delta)^\omega$ such that $\mathcal{P}(a_1 \cdots) = \sigma$ and $q_n \notin Idle$. Consider $\langle U_{i_0}, V_{i_0}, W_{i_0}, X_{i_0} \rangle$ reached in the game after the partial play such that its word is $\mathcal{P}(a_1 \cdots a_n \delta)$. Now if for all $i \geq i_0$, $X_i \neq \emptyset$ then $\langle U_i, V_i, W_i, X_i \rangle \notin Solved_2$ and so the play is losing. Otherwise let $i_1 \geq i_0$ such that $X_{i_1} = \emptyset$, then for all $i > i_1$, $q_n \in X_i$ and the play is losing.

For the other direction, assume that this play is losing. Let us note: $word(\xi_{cont}(\sigma)) = a_1 \cdots a_i \cdots$. Since the play is losing there exists an i_0 such that either (1) for all $i \geq i_0$, $\langle U_i, V_i, W_i, X_i \rangle \notin Solved_1$ or (2) for all $i \geq i_0$, $\langle U_i, V_i, W_i, X_i \rangle \notin Solved_2$ meaning that $X_i \neq \emptyset$.

In the first case, the specification of Δ implies (using König's lemma) (i) the existence of some non-faulty prefix that can reach a state of U_{i_0} and continue from there (without faults), and (ii) the existence of a faulty prefix that can reach a state from W_{i_0} that can continue forever (with or without faults). Thus σ is ambiguous.

In the second case, since $\{X_i\}_{i \geq i_0}$ is a decreasing sequence of finite non empty sets, it must stabilize to a non empty set meaning that there is some non-idle state $q \in \bigcap_{i \geq i_0} X_i$. In addition $\sigma = a_1 \cdots a_{i_0-1}\delta^\omega$. So q is reached by a run whose observation sequence is $a_1 \cdots a_{i_0-1}$ such that the controller never allows an urgent action from q after the i_o^{th} observation. This concludes the proof.

■

Lemma 1 *Let $h = \langle cont, diag \rangle$ be an active diagnoser for $\mathcal{S}$. Then there exists a winning strategy θ_h in the game $\mathcal{G}(\mathcal{S})$. Moreover, there also exists a winning one-bit strategy θ in $\mathcal{G}(\mathcal{S})$.*

Proof. Suppose that $h = \langle cont, diag \rangle$ is any active diagnoser for $\mathcal{S}$. Then, *cont* defines a (not necessarily finite memory) strategy θ_h in $\mathcal{G} = \mathcal{G}(\mathcal{S})$: for $\xi \in Play^*(\mathcal{G})$, let $\theta_h(\xi) = \langle state(\xi), cont(word(\xi)) \rangle$.

Now, for any play ρ that adheres to θ_h we have that $word(\rho) \in \mathcal{P}(\mathcal{L}^\omega(\mathcal{S}_{cont}))$, thus by Definition 6, $word(\rho)$ is not ambiguous and if $word(\rho) \in \Sigma_o^* \delta^\omega$ any sequence σ with $\mathcal{P}(\sigma) = word(\rho)$ ends in an idle state. Hence by Proposition 1, ρ is winning, which means that θ_h is a winning strategy.

Finally, the existence of θ_h implies the existence of a winning one-bit strategy (due to well-known results of game theory, see, e.g., [5], as in this case the generalized Büchi game has two winning conditions). ■

For the reverse direction, we show how to define a pilot from a one-bit strategy and we prove that this pilot is an active diagnoser if the strategy is winning. State *useless* is added only to stick to Definition 7.

Definition 13 *Let $\theta_{M,\alpha}$ be a one-bit strategy in $\mathcal{G}(\mathcal{S})$. Then $\mathcal{C}(\theta_{M,\alpha}) := \langle \mathcal{B}_\mathcal{C}, cont_\mathcal{C}, diag_\mathcal{C} \rangle$ denotes a pilot where $\mathcal{B}_\mathcal{C} = \langle Q^c, q_0^c, \Xi_o, T^c \rangle$ with $Q^c = \{0,1\} \times V_C \cup \{useless\}$ and $q_0^c = \langle b_0, v_0 \rangle$, where b_0 is the initial state of M. Moreover, for any $\langle b, s \rangle \in \{0,1\} \times V_C$, if $\langle s, \Sigma' \rangle = \alpha(b, s)$ then:*

- *$T^c((b,s),a)$ is either $\langle b', s' \rangle$ where s' is the single state reached from $\langle s, \Sigma', a \rangle$ when $a \in \Sigma' \cup \{\delta\} \cup \Sigma_{uc} \setminus \Sigma_{uo}$ and $b \xrightarrow{\langle s,\Sigma' \rangle \langle s,\Sigma',a \rangle s'} b'$ in M, or useless if such a state does not exist;*
- *$cont_\mathcal{C}(b,s) := \Sigma'$ where $\langle s, \Sigma' \rangle = \alpha(b,s)$;*
- *$diag_\mathcal{C}(b,s) := \top$ iff s has the form $\langle \emptyset, V, W, X \rangle$.*

Finally, $T^c(useless, a) = useless$ for $a \in \{\delta\} \cup \Sigma_{uc} \setminus \Sigma_{uo}$; moreover $cont_\mathcal{C}(useless) := \emptyset$ and $diag_\mathcal{C}(useless) := \bot$.

Lemma 2 *Let $\theta_{M,\alpha}$ be a winning one-bit strategy in $\mathcal{G}(\mathcal{S})$. Then $\mathcal{C} = \mathcal{C}(\theta_{M,\alpha})$ is an active diagnoser for $\mathcal{S}$.*

Proof. We show that $h_{\mathcal{C}}$ fulfills the three conditions of Definition 6, which shows that $\mathcal{S}$ is actively diagnosable.

1. Let $\sigma \in \mathcal{P}(\mathcal{L}^{\omega}(\mathcal{S}_{cont}))$, let $\rho = (s_i a_{i+1})_{i\geq 0}$ its unique run in $\mathcal{B}_{\mathcal{C}}$, and denote by $w_i = \sigma[1,i]$, for $i \geq 0$. By construction of $\mathcal{S}_{cont}$, we have that $w_{i+1} = w_i a_{i+1}$ implies $a_{i+1} \in cont(s_i) \cup \{\delta\} \cup \Sigma_{uc} \setminus \Sigma_{uo}$. Let $s_i = \langle b_i, v_i \rangle$. By construction of $\mathcal{B}_{\mathcal{C}}$, $\alpha(s_i) = \langle v_i, cont_{\mathcal{C}}(s_i) \rangle$ and $v_{i+1} = \Delta(v_i, cont_{\mathcal{C}}(s_i), a)$. Therefore $v_0, \langle v_0, cont_{\mathcal{C}}(s_0)\rangle, \langle v_0, cont_{\mathcal{C}}(s_0), a_1\rangle, v_1, \ldots$ is a play of $\mathcal{G}(\mathcal{S})$ that adheres to $\theta_{M,\alpha}$. Since by assumption any such play is winning, $v_i \in Solved_1$ for infinitely many i, so by Proposition (1), σ is unambiguous.

2. For $v = \langle U, V, W, X\rangle$ such that $v_0 \xrightarrow{\sigma} v$ for some observed sequence of $\mathcal{S}_{cont}$, we have by construction of $\mathcal{G}(\mathcal{S})$ that (a) $q \in U$ iff there exists $w \in \mathcal{P}^{-1}(\sigma) \cap \mathcal{L}^*(\mathcal{S}_{cont}) \cap (\Sigma \setminus \{f\})^*$ and (b) $q \in V \cup W$ iff there exists $w \in \mathcal{P}^{-1}(\sigma) \cap \mathcal{L}^*(\mathcal{S}_{cont}) \cap \Sigma^* f \Sigma^*$. Now, $diag(\sigma) = diag_{\mathcal{C}}(v) = \top$ iff $U = \emptyset$, which by the above and Definition 5 is equivalent to saying that σ is surely faulty.

3. Let $\sigma' \in \Xi^* \delta^{\omega}$ be a sequence of $\mathcal{S}_{cont}$. Since the strategy $\theta_{M,\alpha}$ is winning, the play in $\mathcal{G}(\mathcal{S})$ corresponding to this sequence is winning. Applying Proposition (1), any run of $\mathcal{S}_{cont}$ that corresponds to σ' ends up in an idle state.

■

We can now state the main result of this section:

Theorem 1 *Let $\mathcal{S}$ be a POCS with n states and m controllable actions. The active-diagnosis decision and synthesis problems for $\mathcal{S}$ can be solved in $2^{\mathcal{O}(n+m)}$ time. If $\mathcal{S}$ is actively diagnosable, then one can synthesize a pilot $\mathcal{C}$ with at most $2 \cdot 11^n$ states, where $\mathcal{C}$ is an active diagnoser for $\mathcal{S}$.*

Proof. Lemma 1 and Lemma 2 imply that $\mathcal{S}$ is actively diagnosable iff there is a winning one-bit strategy for s_0 in $\mathcal{G}(\mathcal{B})$, and the second part of the theorem follows from Lemma 2. As for the complexity statement, we note that the game $\mathcal{G}(\mathcal{S})$ has $\mathcal{O}(11^n \cdot 2^m)$ vertices and edges, and a winning strategy can be computed in polynomial time in the size of the game [5], which gives the result. ■

By a straightforward adaptation of the proof of Theorem 1 in [7], we can prove EXPTIME-hardness of the decision problem. So we get the following corollary.

Corollary 1 *The active diagnosis decision problem is EXPTIME-complete.*

Observe that in any play of a winning one-bit strategy, one can visit at most twice (one per bit value) the same state while U and W remain non empty since otherwise, a losing play could be built ending with a loop. So once a fault has occurred, there are two possible situations: (1) either W remains non empty and after at most $2|V_C| + 1$ observations, U becomes empty, or (2) while U remains non empty, W becomes empty after at most $2|V_C| + 1$

observations and filled again by at least the state in V that corresponds to the faulty run and then after at most $2|V_C|$ additional observations U becomes empty. Summarizing, the delay achieved by our active diagnoser is at most $4|V_C| + 1 \leq 4 \cdot 11^n + 1 = 2^{\mathcal{O}(n)}$. The active diagnoser that we synthesize does not necessarily have minimal delay. However Theorem 6 of the next section shows that $2^{\mathcal{O}(n)}$ states are not enough for obtaining such an active diagnoser, and Theorem 4 shows that for some systems, the minimal achievable delay is indeed exponential. The following result provides a construction of an active diagnoser with minimal delay.

Theorem 2 *Let $\mathcal{S}$ be an actively diagnosable POCS with n states. One can construct a pilot $\mathcal{C}$ with $2^{\mathcal{O}(n^2)}$ states such that $\mathcal{C}$ is an active diagnoser for $\mathcal{S}$ with minimal delay.*

Proof. One iteratively builds a Büchi game $\mathcal{G}_i(\mathcal{S})$ parametrized by increasing values of i. A controller state of this game is defined by (U, d, X) where U is the set of states reached by a correct sequence while d (defined when $U \neq \emptyset$) associates with every state s reached by a faulty sequence a duration $d(s) \leq i + 1$ since the occurrence of the earliest fault that would lead to s, and X is defined as previously. As in $\mathcal{G}(\mathcal{S})$ the controller selects a subset of observable actions letting the environment select an action among them. There is a single winning condition $X = \emptyset$. Furthermore the controller has to avoid states with some $d(s) = i + 1$ for which one *artificially* fix X to some non empty set and define themselves are their single successors. The first i for which $\mathcal{G}_i(\mathcal{S})$ has a winning strategy is the minimal delay for $\mathcal{S}$ and the winning strategy yields an active diagnoser with minimal delay. Observe that since the minimal delay is bounded by $2^{\mathcal{O}(n)}$, in the worst case the final game has $2^{\mathcal{O}(n^2)}$ states. ■

5 Lower bounds

In this section, we establish that the active diagnoser built in the proof of Theorem 1 is almost optimal w.r.t. the number of states and the delay before fault detection provided by any diagnoser (both in $2^{\mathcal{O}(n)}$ for our construction). We remark that the lower bounds in this section match those shown in [6], despite the extended capabilities of our diagnosers here. The examples demonstrating the lower bounds are inspired by those in [6] but had to be carefully adapted to deal with the capacity of our controllers to detect quiescence.

Definition 14 *For an observation sequence σ, let us denote by $U(\sigma)$ (uncertainty set) the set of states that POCS can be in after observing σ.*

In the figures that follow (see Figure 5 for an example), we annotate certain states with $[X]$, for some $X \subseteq \Sigma_c$. Remark that this annotation is used exclusively on states that are reachable only without committing a fault. Let v be such a state. Then $[X]$ is a shorthand notation for saying that for every $x \in X$, v possesses an outgoing transition with x to some special state z, which can go with either f or u to another special state z', which in turn can loop with any observable action. Thus, when a controller cannot exclude the possibility that the system is currently in v, then it must disallow all actions in X, otherwise the controlled

system may become non-diagnosable. We say also say for short that the controller will be *punished* for X in v.

5.1 Size of minimal active-diagnoser

Theorem 3 (lower bound for active-diagnoser) *For a fixed alphabet $\Sigma = \{a, b, c, d, u, f\}$ there exists a family $(\mathcal{A}_n)_{n\geq 1}$ of actively diagnosable POCS such that the size of $\mathcal{A}_n$ is in $\mathcal{O}(n)$ and the LTS of any state-based active diagnoser $\mathcal{C}$ for $\mathcal{A}_n$ has at least 2^n states.*

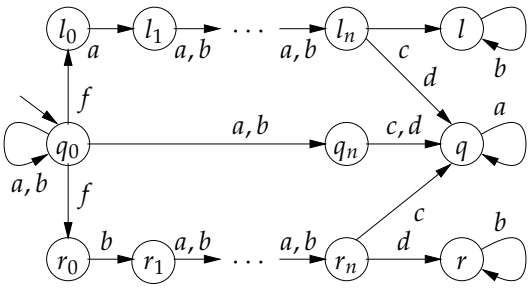

Figure 4: A POCS with $\Sigma_o = \{a, b, c, d\}$, $\Sigma_c = \{c, d\}$, $\Sigma_\ell = Idle = \emptyset$.

Proof. The family of POCS $(\mathcal{A}_n)_{n\geq 1}$ is depicted in Figure 4. A possible active diagnoser memorizes the last n observations. Before the n^{th} observation, it allows both c and d. When n observations have been memorized, the diagnoser allows c (resp. d) if the oldest memorized observation is a (resp. b). It detects a fault when it observes b after either c or d. One deduces the correctness of this diagnoser from the fact that no run can go from l_n or r_n to q with this control.

Assume there exist an active diagnoser with fewer than 2^n states. So there are two words $v, w \in \{a, b\}^n$ such that after observing them the diagnoser is in the same state. Let i be the first position on which v and w differ with $v[i] = a$ and $w[i] = b$. Consider the correct run $\rho = (q_0 v[i])_{i\leq n}(q_0 a)^{i-1} q_n$ and the faulty runs $\rho_l = q_0 v[1] q_0 \dots v[i-1] q_0 f l_0 v[i] l_1 v[i+1] \dots v[n] l_{n-i+1} a \dots a l_n$ and $\rho_r = q_0 w[1] q_0 \dots w[i-1] q_0 f l_0 w[i] r_1 w[i+1] \dots w[n] r_{n-i+1} a \dots a r_n$. The diagnoser is in the same state after these runs. If it always blocks c and d, the (possible) fault is never detected. So consider the first instant when it allows one of these actions, say c (resp. d). Then ρ_r (resp. ρ_l) and ρ can reach q and the fault will never be detected. Therefore, such an active diagnoser cannot exist. ■

Thus an active diagnoser with $2^{\Omega(n)}$ states may be required for a POCS of size n.

5.2 Languages with exponential minimal delay.

Theorem 4 *There exists a family $(\mathcal{A}_n)_{n\geq 1}$ of actively diagnosable POCS such that the number of states of $\mathcal{A}_n$ belongs to $\mathcal{O}(n)$ and the minimal delay of the language recognized by $\mathcal{A}_n$ is $\geq 2^n + 2$.*

Proof. Consider the POCS of Figure 5.

After observing a, the current state is q in case of a faulty run; otherwise it belongs to to $\{p, r_0 \dots r_{n-1}\}$. The sets of actions enabled from $q, p, r_0 \dots r_{n-1}$ are all equal to Γ_n. One

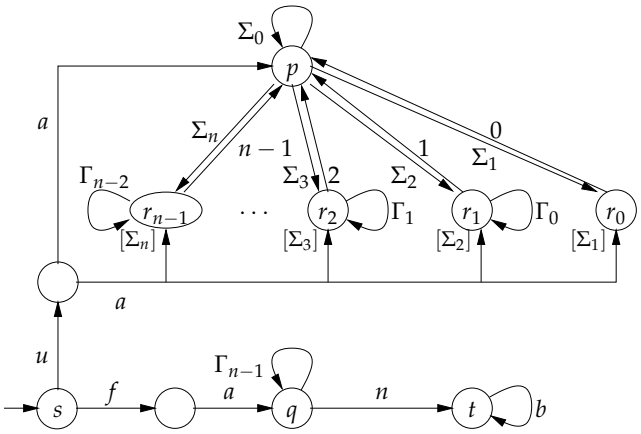

Figure 5: A system whose delay is $2^n + 2$. The observable actions are $\Sigma_o = \{a, b, 0, \ldots, n\}$, the controllable actions $\Sigma_c = \{0, \ldots, n\}$, and we denote $\Sigma_i := \{i, \ldots, n\}$ and $\Gamma_i := \{0, \ldots, i\}, \Sigma_\ell = \mathit{Idle} = \emptyset$.

observes b if and only if the run reaches state t. So an active diagnoser must enforce either (1) all the possible faulty runs to reach t while none of the correct runs reaches t or (2) vice versa; but a straightforward examination shows that a correct run can not reach t. So the active diagnoser has to obtain situation (1). To enforce such a situation, at some moment the controller has to enforce a move via action n, as this is the only observation which guarantees that a faulty run ends in t. On the other hand, since from states $r_0, \ldots, r_n$ action n is punished, the diagnoser must first ensure that all possible correct runs are in state p. Associate a counter *count* of n bits to the current uncertainty set of the diagnoser. Bit i is set if the uncertainty set includes state r_i. After the observation of a, all bits are set and so *count* is equal to $2^n - 1$. The intermediate goal of the controller consists in resetting *count* to zero.

Consider a situation when bit i of *count* is set and all bits j with $j < i$ are zero. Since there is a possible correct run in r_i, the diagnoser must forbid all actions in Σ_{i+1} since they would be punished. If the controller allows an action in Γ_{i-1} the correct runs that are in state r_j with $j \geq i$ may stay in their state, so *count* will not decrease. If the controller only allows action i, the correct runs in state r_i must go to p, while the correct runs in p may go to any r_j with $j < i$, and the correct runs in r_j with $j > i$ stay in their state, thus decrementing *count* by 1. So we have simultaneously established that there is a strategy that resets the *count* after $2^n - 1$ observations and that it is the optimal strategy w.r.t. the delay. Afterwards the controller only allows n, and then only allows b, observing δ if the actual run was correct and b if the actual run was faulty. ■

Thus an active diagnoser achieving delay $2^{\Omega(n)}$ may be required for a POCS of size n.

In Theorem 4, the POCS $\mathcal{A}_n$ has $\mathcal{O}(n)$ states and $\mathcal{O}(n^2)$ transitions, seeing as the size of the alphabet depends on n. Thus, strictly speaking, the size of $\mathcal{A}_n$ is quadratic. However, this result can be improved to the case where the alphabet is independent of n while the number of states remains in $\mathcal{O}(n)$, as shown by Theorem 5.

Theorem 5 *There exists a family $(\mathcal{A}_n)_{n\geq 1}$ of actively diagnosable POCS such that the size of $\mathcal{A}_n$ belongs to $\mathcal{O}(n)$ and the minimal delay of the language recognized by $\mathcal{A}_n$ is $\geq 2^n + 2$.*

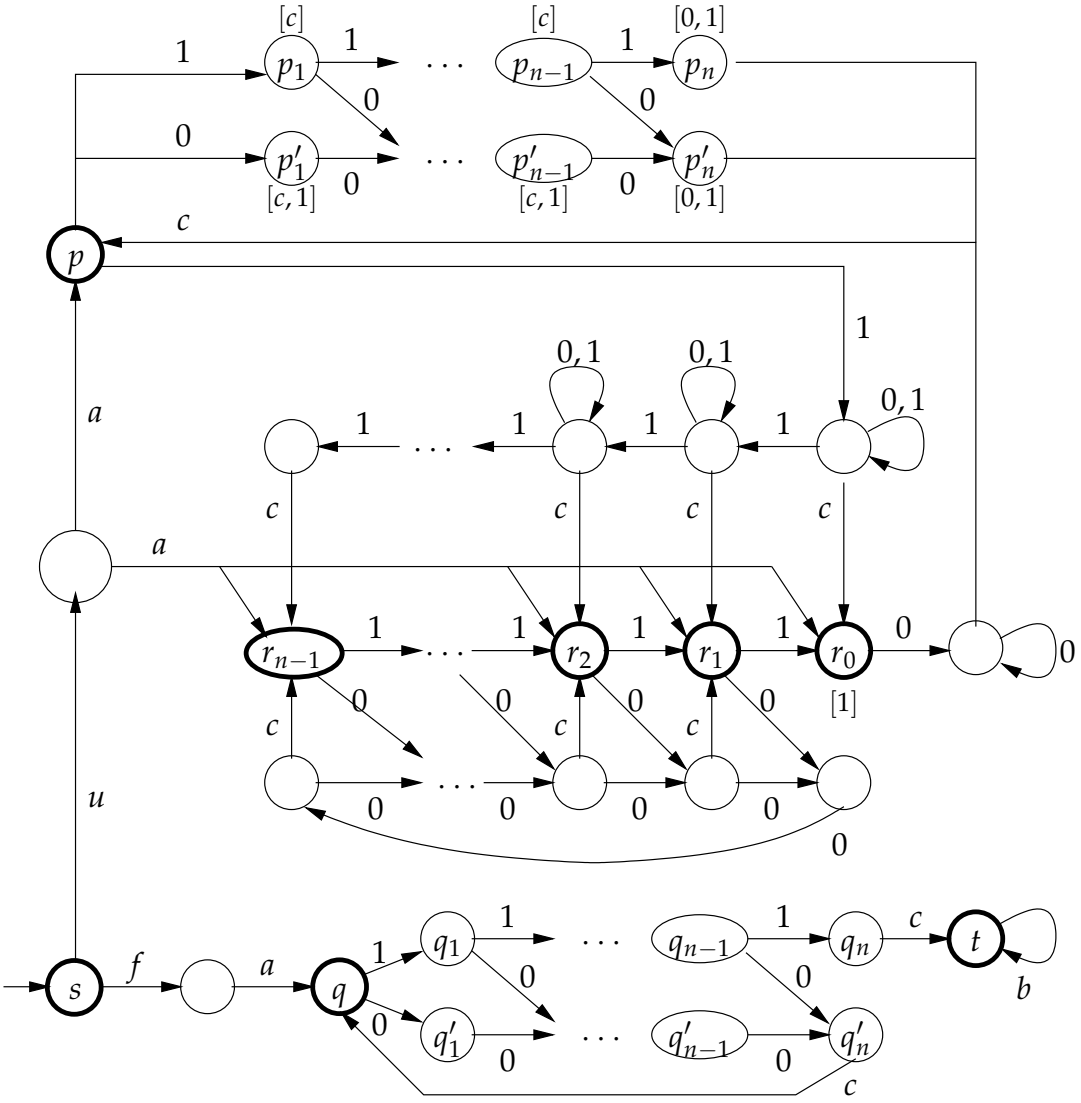

Figure 6: An LTS $\mathcal{A}'_n$ with $\Sigma_o = \Sigma_c = \{0, 1, a, b, c\}$ with delay $\Omega(2^n)$.

Proof. Consider the POCS of Figure 6, whose alphabet is $\Sigma' = \{0, 1, a, b, c, d, f, u\}$, hence independent of n.

Let $\mathcal{A}_n$ denote the system from Theorem 4 with variable-size alphabet Σ. Consider the system $\mathcal{A}'_n$ in Figure 6, which has $\mathcal{O}(n)$ states. We now prove that any controller performing active diagnosis for $\mathcal{A}'_n$ has a delay of $\Omega(2^n)$.

1. Let us denote by Q the states of $\mathcal{A}_n$. Then Q is a subset of the states in $\mathcal{A}'_n$. In Figure 6, the states in Q are marked with a thick border. $\mathcal{A}'_n$ works mostly like $\mathcal{A}_n$, but with observable letters encoded in unary. Let $code(a) = a$, $code(b) = b$, and $code(i) = 1^i 0^{n-i} c$, for $i = 0, \ldots, n$. The reader can convince himself that, modulo this encoding, $\mathcal{A}'_n$ "simulates" $\mathcal{A}_n$ in the following sense: Let $v, v' \in Q$ and x an observable action in $\mathcal{A}_n$.

Then there exists a sequence σ with $\mathcal{P}_\Sigma(\sigma) = x$ and $v \stackrel{\sigma}{\Rightarrow} v'$ in $\mathcal{A}_n$ if and only if there exists a sequence σ' with $\mathcal{P}_{\Sigma'}(\sigma') = code(x)$ such that $v \stackrel{\sigma'}{\Rightarrow} v'$ in $\mathcal{A}'_n$ and moreover, σ is faulty iff σ' is faulty.

2. For any sequence σ such that $\{p, r_k, q\} \subseteq U(\sigma)$ for some $k = 0, \ldots, n-1$, we see that the punishments on states $p_1, p'_1, \ldots, p_n, p'_n$ oblige the controller to enforce a valid encoding of $code(i)$, for some $i = 0, \ldots, n$. Moreover, the punishment of 1 on r_0 punishes $code(i)$, for $i > k$. Finally, the controller never gains any insight from trying to lead the system into a deadlock because p_j (resp. p'_j) has the same available actions as q_j (resp. q'_j) for all $j = 1, \ldots, n$.

The facts proved in 1. and 2. imply that the optimal strategy of the controller is the same as in $\mathcal{A}_n$, modulo the encoding given by *code*, hence the minimal delay is $\Omega(2^n)$. ■

5.3 Diagnoser which guaranties minimal delay.

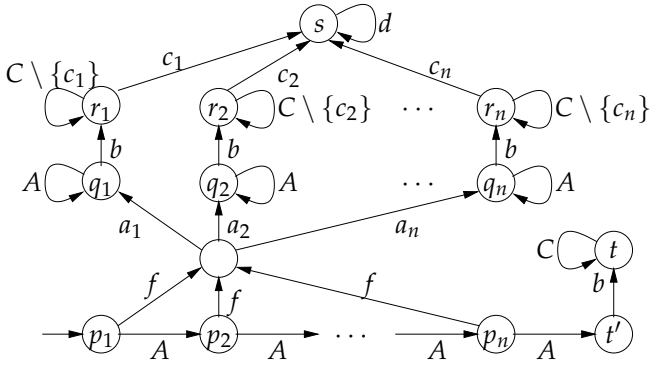

Figure 7: A POCS $\mathcal{A}_n$ whose minimal-delay active diagnoser requires at least $n!$ states. In the figure, $A := \{a_1, \ldots, a_n\}$, $C := \{c_1, \ldots, c_n\}$, $\Sigma_o = A \cup C \cup \{b, d\}$, $\Sigma_c = C, \Sigma_\ell = Idle = \emptyset$.

Theorem 6 (minimal-delay diagnoser) *There exists a family $(A_n)_{n\geq 1}$ of $f(n)$-actively diagnosable POCS (for some function f) with $\mathcal{O}(n)$ states such that the LTS of any state-based active diagnoser $\mathcal{C}$ that achieves the minimal delay $f(n)$ has at least $n!$ states.*

Proof. Consider the POCS of Figure 7. First we show that $f(n) \geq n + 3$. Consider an observed uncontrollable sequence $a_1^n b$. It corresponds either to a correct run that is in state t, or wlog to a run that started with f and is in state r_1 (there may be other faulty runs with the same observation). States t and r_1 have the same set of enabled actions; therefore, the diagnoser needs at least two additional observations to decide whether the run is faulty. This provides the claimed lower bound for the minimal delay of the language recognized by A_n .

Let us provide an active diagnoser that achieves delay $n + 3$. This diagnoser memorizes the first n observations. If b occurs in these observations, then the active diagnoser immediately concludes that the run is faulty. If the $(n + 1)$-th observation is not b (i.e. some a_k) then

it immediately detects a fault. Otherwise after this occurrence of b, it successively checks whether a fault has occurred just before any of the first n observations. Let us denote the observations $a_{\alpha(1)} \dots a_{\alpha(n)}$. To check whether a fault has occurred before $a_{\alpha(1)}$, the diagnoser only allows $c_{\alpha(1)}$. The possible earliest faulty run must go to s, while the other possible faulty runs either remain in their state (in case a fault occurred just before $a_{\alpha(i)}$ with $\alpha(i) \neq \alpha(1)$) or also go to s (in the other case). So one observes $c_{\alpha(1)}$ and then d if the actual run is this earliest faulty run. Any other control would achieve a longer delay. So as long as the diagnoser does not observe d, it only allows $c_{\alpha(i)}$, for i ranging from 2 to n. After observing $a_{\alpha(1)} \dots a_{\alpha(n)} c_{\alpha(1)} \dots c_{\alpha(n)} x$ with $x \neq d$, the diagnoser concludes that no fault has occurred. By the previous reasoning, its delay is at worst $n + 3$.

Let us suppose that there exists an active diagnoser with fewer than $n!$ states. Then there exist two permutations α and β of $\{1, \dots, n\}$ such that the diagnoser is in the same state after observed sequences $a_{\alpha(1)} \dots a_{\alpha(n)}$ and $a_{\beta(1)} \dots a_{\beta(n)}$. Let i be the first position on which α and β differ. Consider the correct run ρ ending in t after observation $a_{\alpha(1)} \dots a_{\alpha(n)} b$ and the faulty run ρ_α (resp. ρ_β) ending in $r_{\alpha(i)}$ (resp. $r_{\beta(i)}$). after observation $a_{\alpha(1)} \dots a_{\alpha(n)} b$ (resp. $a_{\beta(1)} \dots a_{\beta(n)} b$). The diagnoser is in the same state after these runs. In order to achieve the minimal delay w.r.t. the first possible fault it has no other choice than only allowing $c_{\alpha(1)}$ and then for the second possible fault $c_{\alpha(2)}$, etc. Thus the faulty run ρ_β will not be detected in at most $n + 3$ observations when observing $a_{\beta(1)} \dots a_{\beta(n)} b c_{\alpha(1)} \dots c_{\alpha(i)}$. So the delay of this active diagnoser is not minimal. ■

Thus an active diagnoser with $2^{\Omega(n \log(n))}$ states may be required for a POCS of size n for achieving minimal delay. As in Section 5.2, this result can be improved to an example whose alphabet size is fixed.

Theorem 7 (minimal-delay diagnoser) *There exists a family $(A_n)_{n \geq 1}$ of $f(n)$-actively diagnosable POCS (for some function f) of size $\mathcal{O}(n)$ such that the LTS of any state-based active diagnoser $\mathcal{C}$ that achieves the minimal delay $f(n)$ has at least $n!$ states.*

Proof. Let $\mathcal{A}_n$ denote the system from Theorem 6 with variable-size alphabet Σ. Consider the system $\mathcal{A}'_n$ in Figure 8, which has alphabet $\Sigma' = \{0, 1, a, b, c, d, f, u\}$ (independently of n) and $\mathcal{O}(n)$ states. We now prove that any controller performing active diagnosis for $\mathcal{A}'_n$ needs $\Omega(n!)$ states.

1. Let us denote by Q the states of $\mathcal{A}_n$. Then Q is a subset of the states in $\mathcal{A}'_n$. In Figure 8, the states in Q are marked with a thick border. Moreover, some states in Figure 8 are drawn with a light or a dark shade, and we call these sets Q_1 and Q_0, respectively. Observe that all states of $Q_1 \cup Q_0$ can do an a, either punishing the controller or going to a state in $Q \cup \{o\}$. Moreover, states in Q_1 can do 1, with a choice of going to a state in Q_1 or Q_0, and states in Q_0 will stay in Q_0 with 0.

2. $\mathcal{A}'_n$ works mostly like $\mathcal{A}_n$, but with observable letters encoded in unary. For $i = 1, \dots, n$, let $code(a_i) = 1^i 0^{n-i} a$, $code(b) = b$, $code(c_i) = 1^i 0^{n-i} c$, and $code(d) = d$. The reader can convince himself that, modulo this encoding, $\mathcal{A}'_n$ "simulates" $\mathcal{A}_n$ in the following sense: Let $v, v' \in Q$ and x an observable action in $\mathcal{A}_n$. Then there exists a sequence

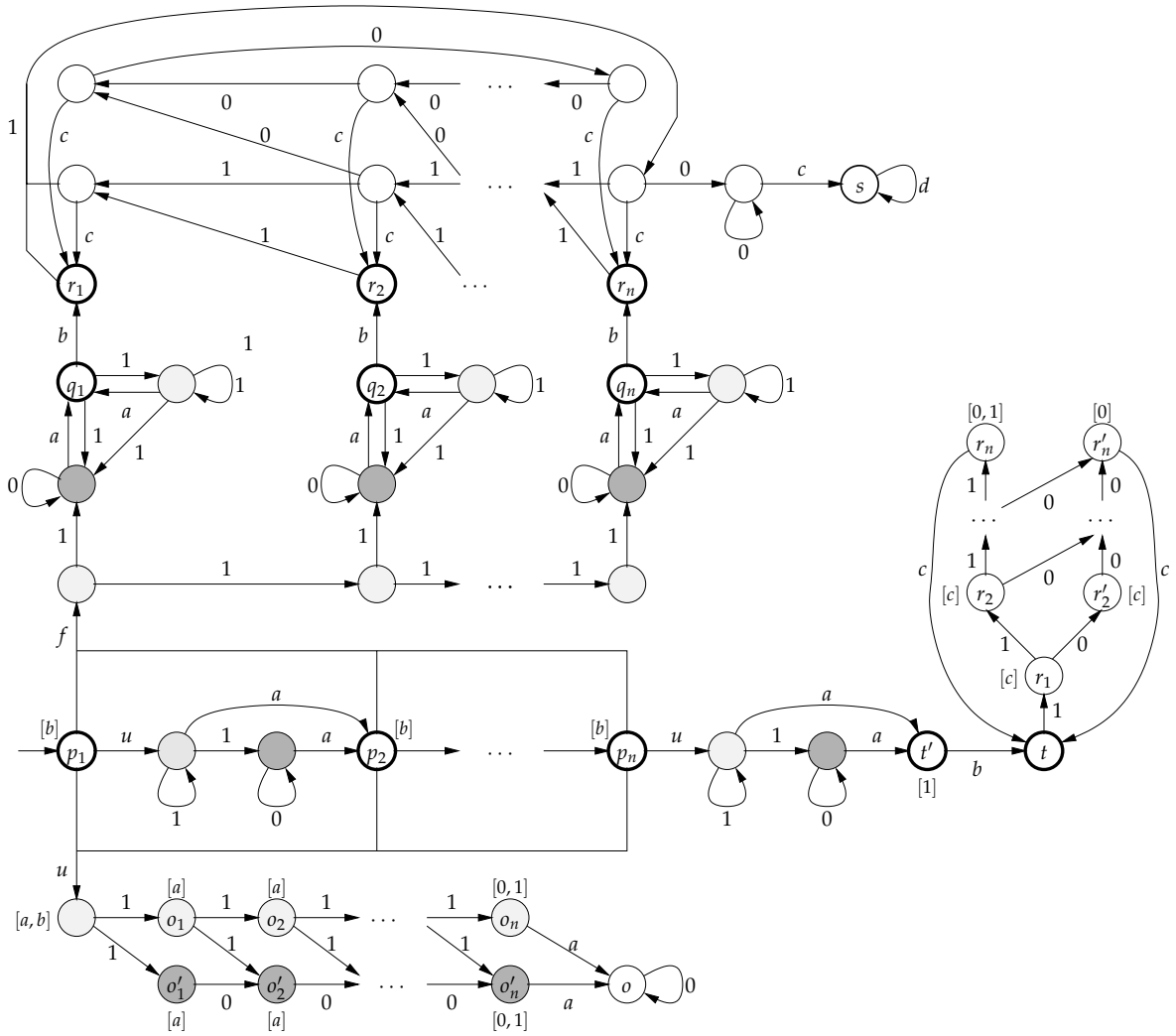

Figure 8: An LTS $\mathcal{A}'_n$ with $\Sigma' = \{0, 1, a, b, c, d, f, u\}$, $\Sigma'_o = \Sigma'_c = \{0, 1, a, b, c, d\}$ and all actions are eager (it is also possible to declare either f or u to be lazy, but not for both). . An LTS $\mathcal{A}'_n$ requires $\Omega(n!)$ states for optimal-delay diagnosis.

σ with $\mathcal{P}_\Sigma(\sigma) = x$ and $v \stackrel{\sigma}{\Rightarrow} v'$ in $\mathcal{A}_n$ if and only if there exists a sequence σ' with $\mathcal{P}_{\Sigma'}(\sigma') = code(x)$ such that $v \stackrel{\sigma'}{\Rightarrow} v'$ in $\mathcal{A}'_n$ and moreover, σ is faulty iff σ' is faulty.

Additionally, $\mathcal{A}'_n$ has some behaviours that do not correspond to a valid encoding, but as we shall see, such behaviours are either punishing (and must be avoided) or will lead to an immediate diagnosis result (but cannot be enforced). Therefore, the strategy of the controller is closely related to the strategy in $\mathcal{A}_n$, and the minimal delay achievable in $\mathcal{A}'_n$ is the same as in $\mathcal{A}_n$, modulo a factor of $\mathcal{O}(n)$ for the unary encoding.

We shall also see that the controller never benefits from blocking all available actions in any state, so in the interest of delivering a diagnosis verdict as quickly as possible, the controller should never attempt to induce a deadlock. (As a side effect, this makes $\mathcal{A}'_n$ also a valid example for the same lower bound in settings where inducing deadlocks

is not permitted [6].)

3. We first prove (by induction of n) that after observing $\sigma = code(a_{i_1}) \cdots code(a_{i_k})$, for $0 \leq k \leq n$, $\{p_{k+1}, o\} \subset U(\sigma) \subseteq \{p_{k+1}, q_1, \ldots, q_n, o\}$ (with $p_{n+1} := t'$), more precisely $U(\varepsilon) = \{p_1\}$, and $U(\sigma) = \{p_{i+1}, q_{i_1}, \ldots, q_{i_k}, o\}$ otherwise. Certainly the claim holds for $k = 0$ and $\sigma = \varepsilon$. Then, thanks to the simulation property mentioned above, it is easy to see that the claim also holds for $k > 0$.

   Additionally, let us show that for all $k < n$, the controller must admit $code(a_i)$ for any $i \in 1, \ldots, n$. To begin with, the possible actions in states $\{p_{i+1}, q_{i_1}, \ldots, q_{i_k}, o\}$ are $0, 1, b$, and b is punished in p_{i+1}, so the controller must disallow it. Action 0 is only possible in state o, so seeing 0 would allow to conclude that no fault has happened (and none will happen in the future), therefore blocking it is unproductive for the controller. However, if $\mathcal{A}'_n$ is in a state different from o and the controller blocks 1, then it will merely cause a deadlock without gaining any information because the only unobservable moves are with u and f from p_{i+1}, and b remains punished. So the only reasonable course of action for the controller is to allow $0, 1$ and block b.

   Let us assume that $\mathcal{A}'_n$ is not in state o. Then our next observation is 1 and $U(\sigma 1) \subseteq Q_1 \cup Q_0$, including one of o_1, o'_1, which will punish a. The only allowable actions are therefore $0, 1$. Blocking either of them will merely introduce an unnecessary delay, e.g., blocking 1 but allowing 0 will either lead us to observing 0 or a deadlock; in the latter case $U(\sigma 1)$ shrinks to a subset of Q_1, but the only possibility to progress is to allow 1 in the next move; the situation for 0 is symmetric. Thus, while $0, 1$ are both controllable, the controller has nothing to gain from exercising control over them.

   This situation is repeated n times (with states from Q_1 eventually excluded from the uncertainty set after observing some 0). After n observations, the uncertainty set includes at least one of o_n, o'_n. At this point $0, 1$ are punished, so the controller must block them and allow a. Note that whatever state of $U(\sigma)$ the automaton was in initially, the observation was of the form $code(a_i)$, for some $a_i \in A$.

4. Now, let the sequence σ observed so far be $code(a_{\pi(1)}) \cdots code(a_{\pi(n)})$, for some permutation π, so $U(\sigma) = \{t', o, q_1, \ldots, q_n\}$. (Recall that a permutation is the worst case for the controller, otherwise $U(\sigma)$ will be smaller.) Since t' punishes 1, the controller will allow 0 and b, which allows to distinguish between o and the other states. In the first case, as before, the controller can diagnose 'no fault' (for past and future), otherwise $U(\sigma b)$ becomes $\{t, r_{i_1}, \ldots, q_{i_n}\}$, as it would be in $\mathcal{A}_n$ for the observation $a_{i_1} \cdots a_{i_n} b$.

5. At this point, the controller can, like in $\mathcal{A}_n$, begin to successively eliminate states from its uncertainty set. The punishments of c respectively $0, 1$ in $r_1, \ldots, r_n, r'_n$ force the controller to admit some sequence $code(c_i)$, however this time the controller may freely choose the value of i. In analogy to $\mathcal{A}_n$, the controller's best course of action to obtain a minimal delay is to enforce $code(c_{\pi(1)}) \cdots code(c_{\pi(n)})$, which concludes our proof of the theorem.

■

6 Conclusion

We have extended the scope of active diagnosis in two directions: (1) enlarging the behaviour of the system with modalities for actions and (2) allowing the controller to observe quiescence of the system. We have addressed this problem by modeling it by a Büchi game and obtained almost optimal decision and synthesis procedures. We plan to study other extensions like action priorities and multiple faults. We also want to analyze the safe active diagnosis introduced in a probabilistic framework [1], where the active diagnoser is required to preserve correct behaviours as most as possible.

References

[1] N. Bertrand, E. Fabre, S. Haar, S. Haddad, and L. Hélouët. Active diagnosis for probabilistic systems. In *FOSSACS 2014, Grenoble, France*, volume 8412 of *LNCS*, pages 29–42, 2014.

[2] C. G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems - Second Edition*. Springer, 2008.

[3] F. Cassez and S. Tripakis. Fault diagnosis with static and dynamic observers. *Fundamenta Informaticae*, 88:497–540, 2008.

[4] E. Chanthery and Y. Pencolé. Monitoring and active diagnosis for discrete-event systems. In *Proc. SafeProcess'09*, pages 1545–1550, 2009.

[5] Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games*. LNCS 2500. Springer, 2002.

[6] S. Haar, S. Haddad, T. Melliti, and S. Schwoon. Optimal constructions for active diagnosis. In *Proc. FSTTCS*, volume 24 of *Leibniz International Proceedings in Informatics*, pages 527–539, 2013.

[7] Stefan Haar, Serge Haddad, Tarek Melliti, and Stefan Schwoon. Optimal constructions for active diagnosis. Research Report LSV-13-12, LSV, ENS Cachan, France, July 2013.

[8] M. Sampath, S. Lafortune, and D. Teneketzis. Active diagnosis of discrete-event systems. *IEEE Transactions on Automatic Control*, 43(7):908–929, July 1998.

[9] D. Thorsley and D. Teneketzis. Active acquisition of information for diagnosis and supervisory control of discrete-event systems. *Journal of Discrete Event Dynamic Systems*, 17:531–583, 2007.

[10] J. Tretmans. Conformance testing with labelled transition systems: Implementation relations and test generation. *Computer Networks and ISDN Systems*, 29(1):49–79, 1996.

[11] J. Tretmans. Test generation with inputs, outputs and repetitive quiescence. *Software - Concepts and Tools*, 17(3):103–120, 1996.

[12] J. Tretmans. Model based testing with labelled transition systems. In *Formal Methods and Testing*, volume 4949 of *LNCS*, pages 1–38. Springer, 2008.