Optimal Constructions for Active Diagnosis *

Stefan Haar[1], Serge Haddad[1], Tarek Melliti[2], and Stefan Schwoon[1]

1 **LSV (CNRS & ENS Cachan) & INRIA, France**
2 **IBISC (Université d'Evry Val-d'Essonne), France**

Abstract

The task of *diagnosis* consists in detecting, without ambiguity, occurrence of faults in a partially observed system. Depending on the degree of observability, a discrete event system may be *diagnosable* or not. *Active* diagnosis aims at controlling the system in order to make it diagnosable. Solutions have already been proposed for the active diagnosis problem, but their complexity remains to be improved. We solve here the active diagnosability decision problem and the active diagnoser synthesis problem, proving that (1) our procedures are optimal w.r.t. to computational complexity, and (2) the memory required for the active diagnoser produced by the synthesis is minimal. We then focus on the delay between the occurrence of a fault and its detection by the diagnoser. We construct a memory-optimal diagnoser whose delay is at most twice the minimal delay, whereas the memory required for a diagnoser with optimal delay may be highly greater.

1998 ACM Subject Classification F.4.3 Formal Languages

Keywords and phrases Diagnosis, Control theory, Automata theory, Games

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2013.527

1 Introduction

In monitoring discrete event systems, one of the central tasks is that of *diagnosis*: Given a finite labeled transition system $\mathcal{A}$ (also called "plant") whose events are partially observable, our task is to decide – based on the stream of observation labels – whether or not particular unobservable events, called *faults*, have occurred. More precisely, the system is considered k-diagnosable iff at most k events after the occurrence of a fault, the observation is sufficient to detect that occurrence with certainty, i.e. all possible system runs compatible with the partial observation collected so far are faulty. The system $\mathcal{A}$ is *diagnosable* iff there exists $k \geq 1$ such that $\mathcal{A}$ is k-diagnosable. As the system may be insufficiently observable, or the observation not discriminating enough, *diagnosability* verification has received considerable attention since the seminal paper by Sampath et al [13]; see also [4, 3]. Those works construct a dedicated deterministic version of the original plant, a so-called *diagnoser*; the absence of indeterminate cycles in this auxiliary automaton is equivalent to diagnosability.

On the other hand, once a system has been shown to be undiagnosable – in a sense that we will formalize later – several actions can follow, such as complete redesign of the system, or adding further sensors to enhance observability. Sampath et al [12] have initiated a different approach, that of *active diagnosis*: if the given plant $\mathcal{A}$ is not diagnosable, synthesize a partial-observation controller $\mathcal{C}$ that forces $\mathcal{A}$ to stay within a diagnosable subset of its behaviors (or, equivalently, such that the controlled plant $\mathcal{A}_C$ is diagnosable). The pair consisting of the controller and the diagnoser is called an *active diagnoser*. Later, Chanthery and Pencolé [5] have proposed a planning-based approach via a twin plant construction.

* This work has been supported by project ImpRo ANR-2010-BLAN-0317 and the European Union Seventh Framework Programme [FP7/2007-2013] under grant agreement 257462 HYCON2 NOE.

Our contributions. We follow the approach of Sampath et al [12], but via a different method based on automata and game theory. This allows us to improve the construction of diagnosers and moreover establish complexity results, which were not treated in previous work:

1. We build a deterministic Büchi automaton that accepts the sublanguage of infinite *unambiguous* observable sequences, i.e. those that are either (i) triggered by a set of correct runs or (ii) triggered by a set of faulty runs. Its size is upper-bounded by $2^{\mathcal{O}(n)}$, where n is the number of states, which is better than all previous constructions. In addition we show the optimality of our construction proving that there is a family of systems for which any corresponding deterministic Büchi automaton must have a size in $2^{\Omega(n)}$.
2. We then design a Büchi game, where a winning strategy yields an active diagnoser for the system, and vice versa. We thus solve the active diagnosis problem by deciding whether there exists a winning strategy, and the synthesis problem by giving an active diagnoser associated with a positional strategy. The size of the active diagnoser is singly exponential w.r.t. the size of the system, while that of [12] is doubly exponential. We also prove that the decision procedure is EXPTIME-complete and that the synthesis procedure is optimal w.r.t. the number of states of the active diagnoser (still in $2^{\mathcal{O}(n)}$).
3. We then study the delay between a fault and its detection by an active diagnoser. We first present a family of systems for which a minimal-delay diagnoser must have $2^{\Omega(n \log(n))}$ states. However, refining our earlier construction yields an active diagnoser with size $2^{\mathcal{O}(n)}$, whose delay is at most twice the minimal possible delay. In addition, we sketch the construction of a minimal-delay active diagnoser with at most $2^{\mathcal{O}(n^2)}$ states.

Organization. Section 2 recalls notions related to diagnosis and active diagnosis. In Section 3, we establish the lower bounds related to the computational complexity, the memory requirements and the index. Section 4.1 presents the construction of the deterministic Büchi automaton. Then in Section 4.2, we solve the decision and the synthesis problems for active diagnosis. After that, Section 4.3 refines the synthesis problem w.r.t. the delay. Section 5 gives some perspectives of this work. A long version with all proofs is available [7].

2 The active diagnosis problem

Labeled transition systems

When dealing with discrete event systems (DES) diagnosis, systems are often modeled using labeled transition systems (LTS). So we define LTS, their properties and languages.

▶ **Definition 1.** A labeled transition system is a tuple $\mathcal{A} = \langle Q, q_0, \Sigma, T \rangle$ where:

- Q is a set of states with $q_0 \in Q$ the initial state;
- Σ is a finite set of events;
- $T \subseteq Q \times \Sigma \times Q$ is the set of transitions.

We note $q \xrightarrow{a} q'$ for $(q, a, q') \in T$; this transition is then said to be *enabled* in q. A *run* over the word $\sigma = a_1 a_2 \ldots \in \Sigma^\omega$ is a sequence of states $(q_i)_{i \geq 0}$ 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 of an LTS). Let $\mathcal{A} = \langle Q, q_0, \Sigma, T\rangle$ be an LTS. The finite language $\mathcal{L}^*(\mathcal{A}) \subseteq \Sigma^*$ of $\mathcal{A}$ and the infinite language $\mathcal{L}^\omega(\mathcal{A}) \subseteq \Sigma^\omega$ of $\mathcal{A}$ are defined by:

$$\mathcal{L}^*(\mathcal{A}) = \{\, w \in \Sigma^* \mid \exists q : q_0 \stackrel{w}{\Rightarrow} q \,\} \qquad \mathcal{L}^\omega(\mathcal{A}) = \{\, \sigma \in \Sigma^\omega \mid q_0 \stackrel{\sigma}{\Rightarrow} \,\}$$

An LTS $\mathcal{A}$ is *live* if for any reachable state there exists a transition enabled in that state. An LTS $\mathcal{A}$ is *deterministic* if for every pair $q \in Q, a \in \Sigma$ there is at most one q' such that $q \xrightarrow{a} q'$. For a deterministic automaton we write $T(q,a) = q'$ if $q \xrightarrow{a} q'$.

Observations

In order to formalize problems related to diagnosis, we partition Σ into two disjoint sets Σ_o and Σ_{uo}, the sets of *observable* and of *unobservable events*, respectively. Moreover, we distinguish a special *fault* event $f \in \Sigma_{uo}$. Let σ be a finite word; its length is denoted $|\sigma|$. For $\Sigma' \subseteq \Sigma$, define $\mathcal{P}_{\Sigma'}(\sigma)$ inductively by: $\mathcal{P}_{\Sigma'}(\varepsilon) = \varepsilon$; for $a \in \Sigma'$, $\mathcal{P}_{\Sigma'}(\sigma a) = \mathcal{P}_{\Sigma'}(\sigma)a$; and $\mathcal{P}_{\Sigma'}(\sigma a) = \mathcal{P}_{\Sigma'}(\sigma)$ for $a \notin \Sigma'$. Write $|\sigma|_{\Sigma'}$ for $|\mathcal{P}_{\Sigma'}(\sigma)|$, and for $a \in \Sigma$, write $|\sigma|_a$ for $|\sigma|_{\{a\}}$. When σ is an infinite word, its projection is the limit of the projections of its finite prefixes. This projection can be either finite or infinite. As usual the projection is extended to languages. $\mathcal{P}_{\Sigma_o}$ will be more simply denoted by $\mathcal{P}$.

An LTS $\mathcal{A}$ is *convergent* if $\mathcal{L}^\omega(\mathcal{A}) \cap \Sigma^*\Sigma_{uo}^\omega = \emptyset$ (i.e. no infinite sequence of unobservable events from any reachable state). When $\mathcal{A}$ is convergent, then for all $\sigma \in \mathcal{L}^\omega(\mathcal{A})$, one has $\mathcal{P}(\sigma) \in \Sigma_o^\omega$. We shall assume that the system under diagnosis is live and convergent.

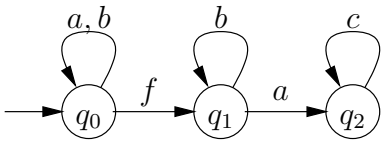

Figure 1 An LTS.

▶ **Example 3.** Figure 1 shows a live and convergent LTS with $\Sigma_o = \{a, b, c\}$ and $\Sigma_{uo} = \{f\}$.

Diagnosability

A finite (resp. infinite) sequence σ is *correct* if it belongs to $(\Sigma \setminus \{f\})^*$ (resp. $(\Sigma \setminus \{f\})^\omega$). Otherwise σ is called *faulty*. An observation sequence may be the projection of both a correct and a faulty sequence, hence ambiguous.

▶ **Definition 4** (ambiguous and surely faulty sequence). Let $\mathcal{A}$ be an LTS, $\sigma_1, \sigma_2 \in \mathcal{L}^\omega(\mathcal{A})$ be two sequences and $\sigma \in \Sigma_o^\omega$ be an observable sequence such that:

(1) $\mathcal{P}(\sigma_1) = \mathcal{P}(\sigma_2) = \sigma$, (2) σ_1 is correct and (3) σ_2 is faulty.

Then σ is called *ambiguous* and the pair (σ_1, σ_2) is a *witness* for the ambiguity of σ. Ambiguous finite observable sequences are defined analogously.
A sequence $\sigma' \in \mathcal{P}(\mathcal{L}^*(\mathcal{A}))$ is *surely faulty* iff $\mathcal{P}^{-1}(\sigma') \cap \mathcal{L}^*(\mathcal{A}) \subseteq \Sigma^* f \Sigma^*$.

▶ **Definition 5** (Diagnosability). Let $k \in \mathbb{N}$. An LTS $\mathcal{A}$ is k-diagnosable if:

$$\forall \sigma = \sigma' f \sigma'' \in \mathcal{L}^*(\mathcal{A})\ |\sigma''|_{\Sigma_o} \geq k \Rightarrow \mathcal{P}(\sigma) \text{ is a surely faulty sequence,}$$

Furthermore, $\mathcal{A}$ is *diagnosable* if there exists a k such that $\mathcal{A}$ is k-diagnosable.

Our definition of diagnosability is a slight variation of the one given in [13]. Indeed the number k above is related to observable events while in former works, it is related to any kind of events. However for finite-state convergent systems (which are the ones addressed by both works) the definitions of diagnosability coincide.

▶ **Example 6.** The LTS of Figure 1 is not diagnosable since the correct infinite trace b^ω and the faulty infinite trace fb^ω have the same projection.

Active diagnosability

We suppose that Σ_o is partitioned into subsets $\Sigma_c \subseteq \Sigma_o$ of *controllable* and $\Sigma_{uc} = \Sigma_o \backslash \Sigma_c$ of *uncontrollable* actions. Intuitively, a controller may forbid a subset of the controllable actions based on the observations made so far, thereby restricting the behaviour of $\mathcal{A}$.

▶ **Definition 7** (Controller)**.** Let $\mathcal{A}$ be an LTS. A *controller* for $\mathcal{A}$ is a mapping $cont : \mathcal{P}(\mathcal{L}^*(\mathcal{A})) \to 2^\Sigma$ such that for all σ, $\Sigma_{uc} \cup \Sigma_{uo} \subseteq cont(\sigma)$. The controlled LTS $\mathcal{A}_{cont} = \langle Q_{cont}, q_{0cont}, \Sigma, T_{cont} \rangle$ is defined by:

- Q_{cont} is the smallest subset of $\Sigma_o^* \times Q$ such that
  1. $(\varepsilon, q_0) \in Q_{cont}$;
  2. $(\sigma, q) \in Q_{cont} \wedge a \in cont(\sigma) \wedge q \xrightarrow{a} q'$ implies $(\mathcal{P}(\sigma a), q') \in Q_{cont}$.
- $q_{0cont} = (\varepsilon, q_0)$
- $((\sigma, q), a, (\sigma', q')) \in T_{cont}$ iff $q \xrightarrow{a} q' \wedge a \in cont(\sigma) \wedge \sigma' = \mathcal{P}(\sigma a)$

In the diagnosis framework, the goal of our controller is to make the system diagnosable, and to perfom diagnosis. However, one requires that the control cannot introduce deadlocks.

▶ **Definition 8** (Pilot and Active Diagnoser)**.** Let $\mathcal{A}$ be an LTS. We call $h = \langle cont, diag \rangle$ a *pilot* for $\mathcal{A}$ if *cont* is a controller and *diag* is a mapping from $\mathcal{P}(\mathcal{L}^*(\mathcal{A}_{cont}))$ to $\{\bot, \top\}$. Moreover, h is called an *active diagnoser* if:

1. $\mathcal{A}_{cont}$ is live;
2. $\mathcal{P}(\mathcal{L}^\omega(\mathcal{A}_{cont}))$ does not contain any ambiguous sequence;
3. $diag(\sigma) = \top$ if and only if σ is a surely faulty sequence for $\sigma \in \mathcal{P}(\mathcal{L}^*(\mathcal{A}_{cont}))$.

For $k \geq 1$, we say that h is a *k-active diagnoser*, if for all $\sigma = \sigma' f \sigma'' \in \mathcal{L}^*(\mathcal{A}_{cont})$ with $|\sigma''|_{\Sigma_o} \geq k$, $diag(\mathcal{P}(\sigma)) = \top$, i.e. every fault is diagnosed after at most k observations. The minimal k such that h is a k-active diagnoser is called the *delay* of h. We call $\mathcal{A}$ *(k-)actively diagnosable* if a (k-)active diagnoser exists, and the minimal such k the *index* of $\mathcal{A}$.

▶ **Example 9.** In the LTS of Figure 1, assume that $\Sigma_c = \{a, b\}$. Let $h_n = \langle cont_n, diag \rangle$, with $n \geq 1$, be the pilot defined by:

- $cont_n(\sigma b^n) = \{a, c, f\}$ for $\sigma \in \Sigma_c^*$ and $cont_n(\sigma) = \Sigma$ otherwise;
- $diag(\sigma) = \top$ iff $\sigma \in \Sigma_o^* c \Sigma_o^*$.

Then h_n is an active diagnoser with delay $n + 2$.

Notice that an active diagnoser does not necessarily have a finite delay. For instance, in Figure 1, there is an active diagnoser that admits the sequence $bab^2ab^3a\cdots$ and is not an k-active diagnoser for any k. However, we will see that if $\mathcal{A}$ is actively diagnosable, there does exist a k-active diagnoser (for some k). We come back to this point in Section 4.3.

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

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

We introduce the notion of *state-based pilot* as finite representation of an active diagnoser.

▶ **Definition 10** (state-based pilot)**.** A *state-based pilot* $\mathcal{C} = \langle \mathcal{B}, cont_\mathcal{C}, diag_\mathcal{C} \rangle$ consists of a deterministic LTS $\mathcal{B} = \langle Q^c, q_0^c, \Sigma_o, T^c \rangle$ and labellings $cont_\mathcal{C}, diag_\mathcal{C} : Q^c \to 2^\Sigma \times \{\bot, \top\}$, such that for all $q \in Q^c$, $\Sigma_{uc} \cup \Sigma_{uo} \subseteq cont_\mathcal{C}(q)$. The pilot $h_\mathcal{C} = \langle cont, diag \rangle$ associated with $\mathcal{C}$ is

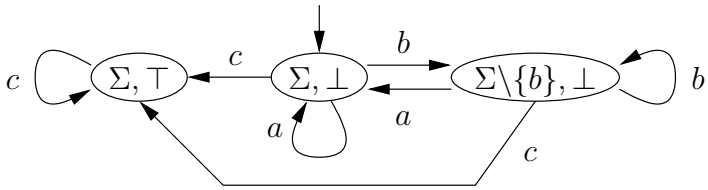

Figure 2 A state-based pilot.

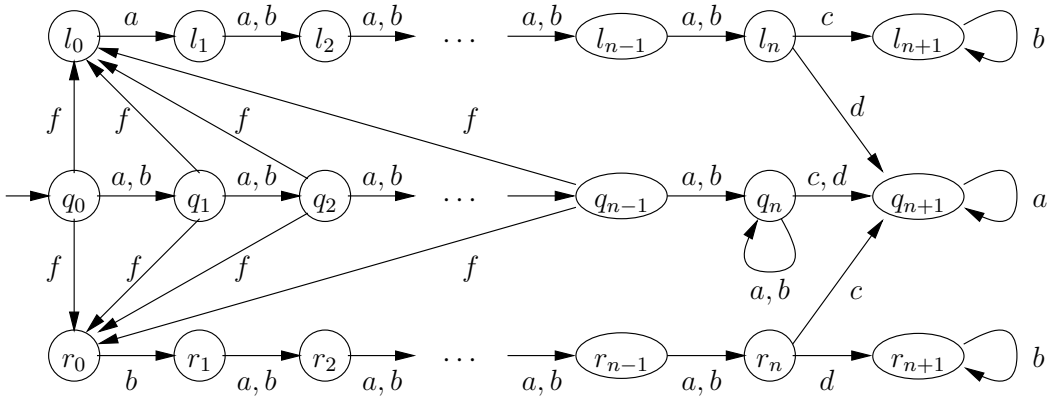

Figure 3 An LTS $\mathcal{A}_n$ with $\Sigma_o = \{a, b, c, d\}$, $\Sigma_c = \{c, d\}$ used in Theorem 14.

given by $cont(\sigma) = cont_{\mathcal{C}}(q)$ and $diag(\sigma) = diag_{\mathcal{C}}(q)$ for all $\sigma \in \mathcal{P}(\mathcal{L}^*(\mathcal{A}))$, where q is the unique state such that $q_0^c \stackrel{\sigma}{\Rightarrow} q$.

▶ **Example 11.** Figure 2 shows a state-based pilot for the LTS of Figure 1. Observe that there is an outgoing transition b from the rightmost state (to fulfil the language inclusion requirement) but b is disabled in this state (in order to implement the active diagnoser h_1).

3 Lower bounds

We first establish that the active diagnosis decision problem is EXPTIME-hard. The proof [7] relies on a reduction from safety games with imperfect information [1].

▶ **Theorem 12** (hardness). *The active diagnosis decision problem is EXPTIME-hard.*

The next theorems focus on the memory required for synthesis problems related to active diagnosis. We start with the language of unambiguous sequences of an LTS.

▶ **Definition 13** (Büchi automaton). A *Büchi automaton* over Σ is a tuple $\mathcal{B} = \langle \mathcal{B}', F \rangle$, where $\mathcal{B}' = \langle S, s_0, \Sigma, \delta \rangle$ is an LTS such that S is finite, and $F \subseteq S$ an *acceptance condition*. A *run* $(q_i)_{i \geq 0}$ is *accepting* if $q_i \in F$ for infinitely many values of i. The language $\mathcal{L}(\mathcal{B})$ consists of all words in $\mathcal{L}^\omega(\mathcal{B}')$ for which there exists an accepting run. A Büchi automaton is called deterministic (live) if its underlying LTS is.

▶ **Theorem 14** (lower bound for determinization). *There exists a family $(\mathcal{A}_n)_{n \geq 1}$ of LTS with the size of $\mathcal{A}_n$ in $\mathcal{O}(n)$ such that any deterministic Büchi automaton recognizing the unambiguous sequences of A_n has at least 2^n states.*

The family of LTS $(\mathcal{A}_n)_{n\geq 1}$ is depicted in Figure 3. During the n first steps a fault can occur leading to the upper (resp. lower) "branch" of the LTS when followed by a (resp. b). However the corresponding observable sequence becomes definitively ambiguous if n steps later the LTS performs d (resp. c). So any deterministic automaton should lead to different states when reading two different words of length n. With an appropriate choice of controllable events, this family also provides a lower bound for a state-based active diagnoser.

▶ **Theorem 15** (lower bound for pilots). *There exists a family $(\mathcal{A}_n)_{n\geq 1}$ of actively diagnosable LTS with the size of $\mathcal{A}_n$ in $\mathcal{O}(n)$ such that the LTS of any state-based pilot $\mathcal{C}$, where $h_{\mathcal{C}}$ is an active diagnoser for $\mathcal{A}$, has at least 2^n states.*

We shall now see that the lower bound is even higher when one tries to minimize the fault-detection delay. The LTS $\mathcal{A}_n$ of Figure 4 contains the observable (and uncontrollable) observation sequence $a_{\pi(1)} \dots a_{\pi(n)}$, where π is a permutation. Such a sequence is ambiguous since a fault may have occurred before any observable event. To remove ambiguity with minimal delay (i.e. $n+2$) an active diagnoser must disallow at time $n+i$ all events in B except $b_{\pi(i)}$ that forces the potential faulty sequence to reach state s where only c is possible. Thus, any active diagnoser for $\mathcal{A}_n$ must remember the permutation π.

▶ **Theorem 16** (minimal-delay diagnoser). *There exists a family $(A_n)_{n\geq 1}$ of $f(n)$-actively diagnosable LTS (for some function f) with $\mathcal{O}(n)$ states such that the LTS of any state-based pilot $\mathcal{C}$, where $h_{\mathcal{C}}$ is an $f(n)$-active diagnoser for $\mathcal{A}$, has at least $n!$ states.*

Note that in Figure 4 the alphabet size depends on n; however Theorem 16 also holds for a fixed-size alphabet [7]. While the previous examples exhibit an index linear w.r.t. the size of the LTS, this index may be exponential in the worst case (and no more as shown in the Section 4). An example for this is shown in [7].

▶ **Theorem 17** (lower bound for index). *There exists a family $(A_n)_{n\geq 1}$ of actively diagnosable LTS with $\mathcal{O}(n)$ states such that the index of $\mathcal{A}_n$ is at least 2^n.*

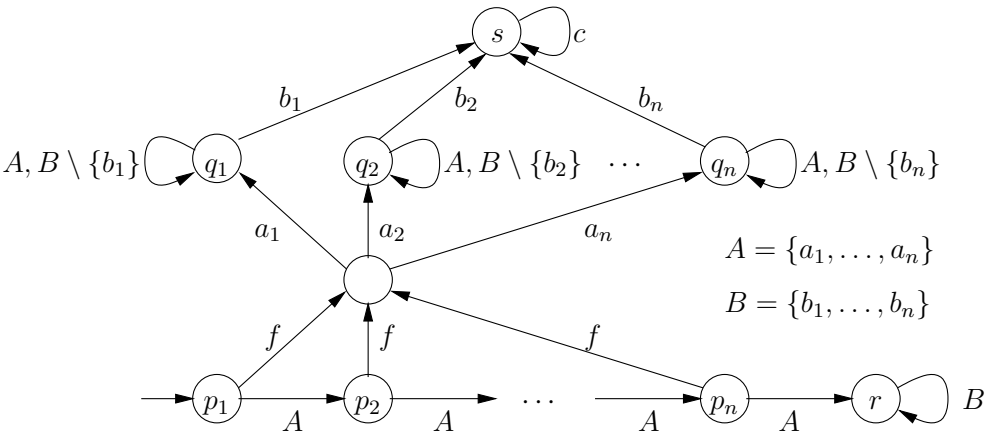

Figure 4 An LTS $\mathcal{A}_n$ with $\Sigma_o = A \cup B \cup \{c\}$, $\Sigma_c = B$ whose minimal-delay active diagnoser requires at least $\mathcal{O}(n!)$ states.

4 Size-Optimal Controller

4.1 Characterization of unambiguous sequences

In this section, we characterize the infinite unambiguous sequences in an efficient way. Fix a finite-state live, convergent LTS $\mathcal{A} = \langle Q, q_0, \Sigma, T\rangle$ for the rest of the section. We build a Büchi automaton $\mathcal{B} = (\mathcal{B}', F)$ that accepts the unambiguous observation sequences. Since $\mathcal{B}$ is the base of the active diagnoser constructed in Section 4.2, we want $\mathcal{B}$ to be deterministic.

A potential procedure for obtaining a deterministic automaton accepting unambiguous sequences is as follows: First, build a non-deterministic Büchi automaton that accepts observable sequences explainable by both a correct and a faulty sequence. This leads to a quadratic blow up w.r.t. the size of $\mathcal{A}$. Then, determinize it by the Safra procedure [11], yielding a deterministic Rabin automaton, and complement it so it accepts the unambiguous sequences. However, we now provide the construction of a simpler and smaller deterministic Büchi automaton. More precisely, the automaton that we build has the following properties:

- $\mathcal{B}'$ is deterministic;
- $\mathcal{B}'$ "reads" the observable sequences of $\mathcal{A}$, i.e. $\mathcal{L}^\omega(\mathcal{B}') = \mathcal{P}(\mathcal{L}^\omega(\mathcal{A}))$;
- $\mathcal{B}$ accepts exactly the unambiguous observation sequences.

We first give some intuition about the way $\mathcal{B}$ works. Its states are triples $\langle U, V, W\rangle$, where $U, V, W \subseteq Q$. The states in U represent states reachable by non-faulty traces in $\mathcal{A}$, whereas $V \cup W$ are states reachable by committing a fault. Let $\sigma = a_1 a_2 \ldots \in \Sigma_o^\omega$ be an observation sequence. An ambiguous prefix of σ will lead to a state in which both U and $V \cup W$ are non-empty, and if σ is ambiguous, then its run will eventually remain in such states forever. Unfortunately, the reverse implication is not true, as the example from Figure 1 shows: every finite prefix of the sequence a^ω is ambiguous, but a^ω is not. In order to distinguish ambiguous sequences from those that merely have infinitely many ambiguous prefixes, V and W assume different functions: W represents a "watchlist", initially empty. Suppose that the observation $a_1 \ldots a_j$, for some j, corresponds to some faulty execution. Then we put the state reachable by that faulty execution into W and trace its successor states there while making further observations. If W never becomes empty, then indeed there exists a faulty element of $\mathcal{P}(\sigma)$ in $\mathcal{L}^\omega(\mathcal{A})$. On the other hand, if some observation $a_{j'}$, for $j' > j$, is impossible in all states of W, then we can conclude that no fault has occurred before a_j. In the meantime, V serves as a "waiting room": it stores states that can be reached by faulty sequences where the fault has occurred between observations a_j and $a_{j'}$. When W becomes empty, those states are shifted from V to W to form the new watchlist.

Let $S' \subseteq S$, $a \in \Sigma_o$, and $\mathcal{L} \subseteq \Sigma_{uo}^*$ be a language of unobservable actions. We denote $\delta_{\mathcal{L}}(S', a) := \{\, q \in Q \mid \exists q' \in S', w \in \mathcal{L} : q' \xRightarrow{wa} q \,\}$, and introduce the abbreviations

- δ_n for $\mathcal{L} = (\Sigma_{uo} \setminus \{f\})^*$ (non-faulty executions),
- δ_f for $\mathcal{L} = \Sigma_{uo}^* f \Sigma_{uo}^*$ (faulty executions),
- and δ_* for $\mathcal{L} = \Sigma_{uo}^*$ (arbitrary executions).

We can now state the formal construction of $\mathcal{B} = \langle\langle S, s_0, \Sigma_o, \delta\rangle, F\rangle$ as follows:

- $S = 2^Q \times 2^Q \times 2^Q \setminus \{\langle\emptyset, \emptyset, \emptyset\rangle\}$ and $s_0 = \langle\{q_0\}, \emptyset, \emptyset\rangle$;
- $F = \{\, \langle\emptyset, S_1, S_2\rangle, \langle S_1, S_2, \emptyset\rangle \mid S_1, S_2 \subseteq Q \,\}$;
- for $s = \langle U, V, W\rangle \in S$ and $a \in \Sigma_o$ such that $\delta_*(U \cup V \cup W, a) \neq \emptyset$, let $\Delta := \delta_f(U, a) \cup \delta_*(V, a)$; then

$$\delta(s, a) = \begin{cases} \langle \delta_n(U, a), \emptyset, \Delta\rangle & \text{if } W = \emptyset; \\ \langle \delta_n(U, a), \Delta \setminus \delta_*(W, a), \delta_*(W, a)\rangle & \text{otherwise.} \end{cases}$$

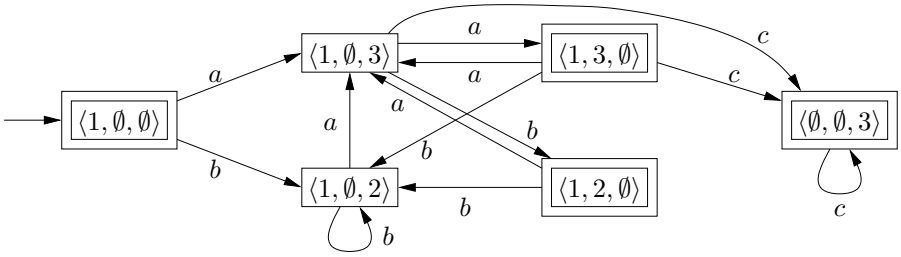

Figure 5 Büchi automaton resulting from Figure 1; accepting states have double frames.

Observe that disregarding the acceptance condition, the sequences read by $\mathcal{B}$ exactly correspond to observable sequences of $\mathcal{A}$, i.e. $\mathcal{P}(\mathcal{L}^\omega(\mathcal{A}))$.

▶ **Theorem 18.** *A sequence of observations $\sigma \in \Sigma_o^\omega$ is accepted by $\mathcal{B}$ iff it is unambiguous.*

▶ **Example 19.** Figure 5 shows the result of the construction on the system from Figure 1. Since all non-empty sets are singletons we have represented them by their item. Notice that any sequence ending in b^ω is ambiguous in Figure 1 and hence not accepted in Figure 5. On the other hand, e.g., sequence a^ω is accepted: while every prefix a^i, for $i \geq 1$, is ambiguous, we always know after $i+1$ observation that no fault has occurred before the i-th observation.

We briefly discuss the relationship of our determinization construction with other standard constructions in diagnosis and automata theory. In [16], diagnosability of an LTS $\mathcal{A}$ is decided by building two automata: one is a modification of $\mathcal{A}$ that accepts the projections all non-faulty sequences, the other accepts the projections of all faulty sequences, remembering whether a fault has occurred in the current state. The cross product of these two is a non-deterministic Büchi automaton of size $2n^2$ (for $|Q| = n$) that accepts all ambiguous sequences. A direct determinization [11] of that cross product would yield a Rabin automaton of size $2^{\mathcal{O}(n^2 \log n)}$. However, given that the cross product is *weak* in the sense that all its strongly connected components are either fully accepting or fully non-accepting, one could apply the *breakpoint construction* of Miyano and Hayashi [9] to obtain a deterministic Büchi automaton of its complement language, of size 3^{2n^2}. Our construction, while similar in spirit to that of [9], is more efficient than that: for a reachable Büchi state $\langle U, V, W\rangle \in S$, any LTS state $q \in Q$ may or may not appear in U, and it may appear in at most one of V or W, but not in both. Thus, the number of reachable states in $\mathcal{B}$ is bounded by $2^n \cdot 3^n = 6^n = 2^{\mathcal{O}(n)}$. Theorem 14 shows that an exponential blowup in n is unavoidable in general, i.e. our construction is optimal up to a constant factor in the exponent.

4.2 Synthesizing the controller

We simultaneously solve the decision and synthesis problems. As before, we fix an LTS $\mathcal{A} = \langle Q, q_0, \Sigma, T\rangle$. We shall try to construct a state-based pilot $\mathcal{C}$ such that $h_\mathcal{C}$ is an active diagnoser for $\mathcal{A}$. The construction succeeds iff $\mathcal{A}$ is actively diagnosable. According to Definition 8, the main challenges in building an active diagnoser are to ensure that (i) the controlled system remains live, (ii) the controller excludes the ambiguous sequences, and (iii) diagnosis information is provided. For this, we introduce Büchi games.

▶ **Definition 20** (game)**.** A game $\mathcal{G}$ (between two players called Control and Environment) is a tuple $\langle V_C, V_E, E, v_0, V_F\rangle$, where V_C, V_E are the vertices owned by Control and Environment, respectively; $V_\mathcal{G}$ denotes all vertices, and $v_0 \in V_C$ is an *initial vertex*. $E \subseteq V_\mathcal{G} \times V_\mathcal{G}$

are directed edges such that for all $v \in V_C$ there exists some $(v, w) \in E$, and $V_F \subseteq V_\mathcal{G}$ is a *winning condition*. A *play* is a function $\rho\colon \mathbb{N} \to V_\mathcal{G}$ such that $\rho(0) = v_0$ and $\langle \rho_i, \rho_{i+1} \rangle \in E$ for all $i \geq 0$; we call $\rho^k := \rho(0) \cdots \rho(k)$, for some $k \geq 0$, a *partial play* if $\rho(k) \in V_C$, and set $\mathit{state}(\rho^k) := \rho(k)$. We write $\mathit{Play}^*(\mathcal{G})$ for the set of partial plays of $\mathcal{G}$. A play ρ is called *winning* (for Control) if $\rho(i) \in V_F$ for infinitely many i.

▶ **Definition 21** (strategy). Let $\mathcal{G} = \langle V_C, V_E, E, v_0, V_F \rangle$ be a game. A *strategy* (for Control) is a function $\theta\colon \mathit{Play}^*(\mathcal{G}) \to V_\mathcal{G}$ such that $\langle \mathit{state}(\xi), \theta(\xi) \rangle \in E$ for all $\xi \in \mathit{Play}^*(\mathcal{G})$. A play ρ *adheres* to θ if $\rho(i) \in V_C$ implies $\rho(i+1) = \theta(\rho^i)$ for all $i \geq 0$. A strategy is called *winning* if every play ρ that adheres to θ is winning. A *positional strategy* is a function $\theta'\colon V_C \to V_\mathcal{G}$ such that $\langle v, \theta'(v) \rangle \in E$ for all $v \in V_C$; we call θ' *winning* if the strategy θ with $\theta(\xi) = \theta'(\mathit{state}(\xi))$ is winning.

In the game that we have defined, a play can only be stuck in a state of Environment. Thus we do not consider finite maximal plays for defining the winning strategies of Control. Let $\mathcal{B} = \langle \mathcal{B}', F \rangle$, with $\mathcal{B}' = \langle S, s_0, \Sigma_o, \delta \rangle$, be the deterministic Büchi automaton constructed from $\mathcal{A}$ in Section 4.1. We shall take $\mathcal{B}'$ as the LTS component of $\mathcal{C}$. To determine $\mathit{cont}_\mathcal{C}$, we construct a Büchi game based on $\mathcal{B}$. The objective of Control is to obtain an accepting run by suitably restricting the possible actions, and any winning strategy will be a suitable candidate for $\mathit{cont}_\mathcal{C}$. Intuitively, a round of the game is played as follows:

1. Control restricts the set of possible actions to Σ'.
2. Environment chooses an action $a \in \Sigma'$ to determine the next state of $\mathcal{B}$.

The choices of Control are subject to some restrictions. Indeed, each state $s = \langle U, V, W \rangle$ represents Control's knowledge about the current potential states of $\mathcal{A}$. To ensure that the controlled system remains live, Σ' must not cause deadlocks in any state reachable by unobservable events from $U \cup V \cup W$. Also, Control cannot prevent the uncontrollable events. So we define the admissible sets and the game as follows.

▶ **Definition 22** (admissible action set). Let $s = \langle U, V, W \rangle$ be a state of $\mathcal{B}$. We call $\Sigma' \subseteq \Sigma_o$ *admissible* for s if (i) $\Sigma_{uc} \subseteq \Sigma'$ and (ii) for all states q' of $\mathcal{A}$ with $q \overset{w}{\Rightarrow} q'$ for some $q \in U \cup V \cup W$ and $w \in \Sigma_{uo}^*$, there exists $a \in \Sigma'$ and $q'' \in Q$ with $q' \xrightarrow{a} q''$. The admissible sets for s are denoted $\mathit{adm}(s)$.

▶ **Definition 23** (controller-synthesis game). Let $\mathcal{B} = \langle \langle S, s_0, \Sigma_o, \delta \rangle, F \rangle$ be a Büchi automaton. We denote $\mathcal{G}(\mathcal{B})$ the game $\langle V_C, V_E, E, s_0, F \rangle$, where $V_C = S$, $V_E = (S \times 2^{\Sigma_o}) \cup (S \times \Sigma_o)$, and $E = E_1 \cup E_2 \cup E_3$, where

- $E_1 = \{ \langle s, \langle s, \Sigma' \rangle \rangle \mid s \in S,\ \Sigma' \in \mathit{adm}(s) \}$;
- $E_2 = \{ \langle \langle s, \Sigma' \rangle, \langle s, a \rangle \rangle \mid s \in S,\ a \in \Sigma' \}$;
- $E_3 = \{ \langle \langle s, a \rangle, s' \rangle \mid \delta(s, a) = s' \}$.

The set E_3 is only introduced to record the sequence of observable actions that occur during a play. Furthermore Environment can be stuck in a vertex of E_3 meaning that the action chosen by Environment does not correspond to a possible behavior of the system.

▶ **Example 24.** Figure 6 depicts an excerpt of the game for Example 1. In the initial state, there are three possible admissible sets, all including c, the uncontrollable observable action. $\{c\}$ is not an admissible set as it blocks the system. If Environment chooses action c, it immediately loses since c is not possible initially even after a fault.

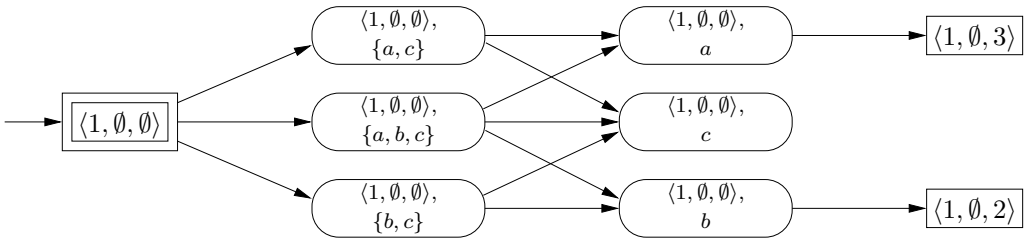

Figure 6 Excerpt of the Büchi game for Example 1.

We can now address the decision and synthesis problems. The next theorem is based on the following: (1) Büchi games can be solved in polynomial time, (2) a positional winning strategy can always be chosen for Control if it wins and (3) there is a tight correspondence between winning strategies and active diagnosers.

▶ **Theorem 25.** *Let $\mathcal{A}$ be an LTS with n states and m controllable actions. The active diagnosis decision and synthesis problems for $\mathcal{A}$ can be solved in $2^{\mathcal{O}(n+m)}$ time. Moreover, if $\mathcal{A}$ is actively diagnosable, then one can synthesize a state-based pilot $\mathcal{C}$ with at most 6^n states such that $h_{\mathcal{C}}$ is an active diagnoser for $\mathcal{A}$.*

We briefly discuss the relationship of our construction with that of [12]. There, an active diagnoser is built on the basis of a powerset construction that is similar to ours but without splitting the possibly faulty states into a 'watchlist' W and a 'waiting room' V. However, they then face the aforementioned problem of distinguishing sequences with infinitely many ambiguous prefixes (like a^ω in Example 1) from truly ambiguous sequences (like b^ω), which they resolve by examining each cycle of the automaton. Since the number of states in that automaton is 3^n,[1] and there can be exponentially many cycles, this procedure is doubly exponential in n. Our construction is only singly exponential in n.

Using Theorems 12 and 25, we get the following corollary.

▶ **Corollary 26.** *The active diagnosis decision problem is EXPTIME-complete.*

4.3 Index and waiting time

We assume that $\mathcal{A}$ is actively diagnosable and develop the construction of an active diagnoser with a delay close to the index of $\mathcal{A}$, and a computational complexity still in $2^{\mathcal{O}(n)}$. For simplicity, we denote the game $\mathcal{G}(\mathcal{B})$ by $\mathcal{G}$. Let $\mathcal{G}'$ be any game. Given a strategy θ for $\mathcal{G}'$, we denote by $Play^\omega_\theta(\mathcal{G}')$ the set of plays that adhere to strategy θ, and by $R(\theta)$ the subset of states of S that are visited by a play of $Play^\omega_\theta(\mathcal{G}')$. We are now in the position to introduce the main concept of this section, the *waiting time* of a strategy: the maximal number of states visited without encountering an accepting state.

▶ **Definition 27** (waiting time)**.** Let θ be a strategy for $\mathcal{G}$. Then the *waiting time* $K(\theta)$ is defined as $\sup(|\{k \mid i \leq k \leq j \wedge \rho(k) \in S\}| \mid \exists i, j\ \exists \rho \in Play^\omega_\theta(\mathcal{G})\ F \cap \{\rho(k)\}_{i \leq k \leq j} = \emptyset)$ with the convention $\sup(\emptyset) = 0$.

Observe that $K(\theta)$ may be infinite for a non-positional winning strategy. However, it is finite and strictly smaller than $|S|$ (since there is at least one accepting state) for a winning

[1] This is the result when only one fault type is considered; [12] actually provides for several fault types, which we omit here for sake of simplicity.

positional strategy. In fact, for θ a winning positional strategy, $K(\theta)$ can be computed in linear time (with appropriate data structures) w.r.t. the size of $\mathcal{G}$. In order to present it and for subsequent use, we introduce the following notation. Let s be a state of the Büchi automaton, $Out(s) := \{a \in \Sigma_o \mid \delta(s,a) \text{ is defined }\}$. First one computes, by increasing values, the minimal solution of the following equation system:

$$V_\theta(s) = \begin{cases} 0 & \text{if } s \in R(\theta) \cap F; \\ 1 + \max(V_\theta(\delta(s,a)) \mid a \in \Sigma' \cap Out(s) \text{ s.t. } (s,\Sigma') = \theta(s)) & \text{if } s \in R(\theta) \setminus F. \end{cases}$$

Then $K(\theta) = \max(V_\theta(s) \mid s \in R(\theta))$. Denote by $D(\theta)$ the delay of the active diagnoser related to strategy θ. Lemma 28 shows that $K(\theta)$ provides useful information about $D(\theta)$.

▶ **Lemma 28.** *Let θ be a strategy for game $\mathcal{G}$ with finite waiting time. Then:*

$$1 + K(\theta) \leq D(\theta) \leq 1 + 2K(\theta)$$

Intuitively, the upper bound is potentially due to a fault staying in the "waiting room" of $\mathcal{B}$ for at most $K(\theta)$ steps, then in the "watchlist" for at most $K(\theta)+1$ steps. The lower bound is due to the fact that along a subrun with a non-empty watchlist, a possible fault could have occurred before this subrun.

Define $K_\mathcal{A} = \min(K(\theta))$, where θ ranges over the winning strategies for $\mathcal{G}$. Since a positional such strategy exists, we know that $K_\mathcal{A}$ is finite and belongs to $2^{\mathcal{O}(n)}$. Let us note $D_\mathcal{A} = \min(D(\theta))$ the index of $\mathcal{A}$. The following corollary provides a tight frame for $D_\mathcal{A}$ and shows that the index is in $2^{\mathcal{O}(n)}$.

▶ **Corollary 29.** *Let $\mathcal{A}$ be actively diagnosable. Then:* $1 + K_\mathcal{A} \leq D_\mathcal{A} \leq 1 + 2K_\mathcal{A}$

Let us compute an active diagnoser or, equivalently, a strategy θ that achieves $K(\theta) = K_\mathcal{A}$. To this aim, we introduce a family of games $\{\mathcal{G}_i\}_{i\in\mathbb{N}}$ defined as follows. The set of vertices of $\mathcal{G}_i$ are: $V_{\mathcal{G}_i} = \{v^j \mid v \in V_\mathcal{G} \wedge 0 \leq j \leq i\} \cup \{lost\}$ where the subset of vertices owned by Control are $\{v^j \mid v \in V_C \wedge 0 \leq j \leq i\} \cup \{lost\}$, the initial vertex is s_0^0, and the set of accepting states are $\{s^0 \mid s \in F\}$. Its set of edges $E' = E'_1 \cup E'_2 \cup E'_3$ is defined by:

- for all $j \leq i$, $\langle v^j, w^j\rangle$ belongs to E'_1 iff $\langle v, w\rangle$ belongs to E_1;
- for all $j \leq i$, $\langle v^j, w^j\rangle$ belongs to E'_2 iff $\langle v, w\rangle$ belongs to E_2;
- for all $j \leq i$, $\langle \langle s,a\rangle^j, s'^0\rangle \in E'_3$ iff $\langle \langle s,a\rangle, s'\rangle \in E_3$ and $s' \in F$;
- for all $j < i$, $\langle \langle s,a\rangle^j, s'^{j+1}\rangle \in E'_3$ iff $\langle \langle s,a\rangle, s'\rangle \in E_3$ and $s' \notin F$;
- $\langle \langle s,a\rangle^i, lost\rangle$ belongs to E'_3 iff $\langle \langle s,a\rangle, s'\rangle$ belongs to E_3 and $s' \notin F$;
- $\langle lost, lost\rangle$ belongs to E'_3 and there is no other edge.

Game $\mathcal{G}_i$ has the following properties: an infinite play either ends up in *lost* or visits the accepting states infinitely often, with at most i visits of the set $\{v^j \mid v \in S \setminus F,\ 0 \leq j \leq i\}$ between two visits of accepting states. The following lemma relates strategies in $\mathcal{G}$ and $\mathcal{G}_i$. Based on it an efficient computation of an optimal strategy w.r.t. $K(\theta)$ can be performed.

▶ **Lemma 30.** *There is a winning strategy θ in $\mathcal{G}$ with $K(\theta) \leq i$ iff there is a winning strategy θ_i in $\mathcal{G}_i$. Moreover, in the positive case, θ can be chosen to be positional.*

▶ **Theorem 31.** *If $\mathcal{A}$ is actively diagnosable, there exists a positional strategy θ that fulfills $K(\theta) = K_\mathcal{A}$. Moreover, such a strategy can be computed in $2^{\mathcal{O}(n)}$.*

This construction represents a reasonable tradeoff, since, due to Theorem 16, an active diagnoser that realizes a delay equal to the index of $\mathcal{A}$ may need to be much larger, i.e. $2^{\Omega(n\log(n))}$. We sketch the construction of a controller with minimal delay once one knows

that the system is actively diagnosable. One iteratively builds a safety game $\mathcal{G}'_i$ parametrized by increasing values of i. A controller state of this game is defined by (U, d) where U is the set of states reached by a correct sequence while d 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. As in the previous games the controller selects a subset of observable actions letting the environment select an action among them. The aim of the controller is to avoid states with some $d(s) = i + 1$. The first i for which $\mathcal{G}'_i$ has a winning strategy is the index and the winning strategy yields an active diagnoser with minimal delay. Observe that since the index is bounded by $2^{\mathcal{O}(n)}$, in the worst case the final game has $2^{\mathcal{O}(n^2)}$ states.

5 Conclusion and Perspectives

We have developed an active-diagnosis method for finite-state systems and shown it to be optimal w.r.t. several criteria. For instance, our work allows to minimize the delay between the occurrence of a fault and its detection. In general, striving for a minimal delay may lead to an overly restrictive controller, e.g., in Figure 1 the controller could completely forbid the action b. We have therefore undertaken work to allow for *parametrized* active diagnosis, which constructs the most permissive controller that respects a user-specified delay. This work, not included here for space reasons, is contained in the long version [7].

Future work has some research leads to address. First, it remains to determine the precise memory requirements for the minimal-delay diagnoser since we showed that it lies between $2^{\Theta(n \log(n))}$ and $2^{\Theta(n^2)}$. Second, the control for active diagnosis could be refined into a *safe* control, i.e. one that does not "encourage" the faulty behaviours. Last we aim at addressing infinite-state systems or systems with quantitative features, as for passive diagnosability in pushdown systems [10], Petri nets [2], timed [17, 15] and probabilistic systems [14].

References

1 D. Berwanger and L. Doyen. On the power of imperfect information. In *Proc. FSTTCS*, volume 2 of *LIPICS*, Bangalore, India, 2008.

2 M.P. Cabasino, A. Giua, S. Lafortune, and C. Seatzu. Diagnosability analysis of unbounded Petri nets. In *CDC09: 48th IEEE Conf. on Decision and Control*, pages 1267–1272, 2009.

3 C. G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems – Second Edition*. Springer, 2008.

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

5 E. Chanthery and Y. Pencolé. Monitoring and active diagnosis for discrete-event systems. In *Proc. SafeProcess'09*, 2009.

6 A. Cimatti, C. Pecheur, and R. Cavada. Formal verification of diagnosability via symbolic model checking. In *Proceedings of IJCAI*, pages 363–369. Morgan Kaufmann, 2003.

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

8 R. Küsters. *Memoryless Determinacy of Parity Games*, pages 95–106. LNCS 2500. 2002.

9 S. Miyano and T. Hayashi. Alternating finite automata on ω-words. *Theoretical Computer Science*, 32:321–330, 1984.

10 C. Morvan and S. Pinchinat. Diagnosability of pushdown systems. In *Proceedings of the Haifa Verification Conference*, LNCS 6405, 2009.

11 S. Safra. On the complexity of omega-automata. In *FOCS*, pages 319–327. IEEE, 1988.

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

13 M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Diagnosability of discrete-event systems. *IEEE Trans. Aut. Cont.*, 40(9):1555–1575, 1995.

14 D. Thorsley and D. Teneketzis. Diagnosability of stochastic discrete-event systems. *IEEE Transactions on Automatic Control*, 50(4):476–492, 2005.

15 S. Xu, S. Jiang, and R. Kumar. Diagnosis of dense-time systems using digital clocks. *IEEE Transactions on Automation Science and Engineering*, 7(4):870–878, 2010.

16 T-S. Yoo and S. Lafortune. Polynomial-time verification of diagnosability of partially observed discrete-event systems. *IEEE Trans. Automat. Contr.*, 47(9):1491–1495, 2002.

17 S. Hashtrudi Zad, R.H. Kwong, and W.M. Wonham. Fault diagnosis in discrete-event systems: Incorporating timing information. *Trans. Aut. Cont.*, 50(7):1010–1015, 2005.