Fault Diagnosis using Timed Automata

Patricia Bouyer[1 ⋆], Fabrice Chevalier[1 ⋆], and Deepak D'Souza[2 ⋆⋆]

[1] LSV – CNRS UMR 8643 & ENS de Cachan
61, avenue du Président Wilson, 94230 Cachan, France.
e-mails: {bouyer,chevalie}@lsv.ens-cachan.fr
[2] Dept. of Computer Science & Automation
Indian Institute of Science, Bangalore, India.
e-mail: deepakd@csa.iisc.ernet.in

Abstract. Fault diagnosis consists in observing behaviours of systems, and in detecting *online* whether an error has occurred or not. In the context of discrete event systems this problem has been well-studied, but much less work has been done in the timed framework. In this paper, we consider the problem of diagnosing faults in behaviours of timed plants. We focus on the problem of synthesizing fault diagnosers which are realizable as deterministic timed automata, with the motivation that such diagnosers would function as efficient online fault detectors. We study two classes of such mechanisms, the class of deterministic timed automata (DTA) and the class of event-recording timed automata (ERA). We show that the problem of synthesizing diagnosers in each of these classes is decidable, provided we are given a bound on the resources available to the diagnoser. We prove that under this assumption diagnosability is 2EXPTIME-complete in the case of DTA's whereas it becomes PSPACE-complete for ERA's.

1 Introduction

The problem of fault diagnosis involves detecting whether a given system (which we call a *plant*) has undergone a fault, based on a particular external observation of an execution of the plant [SSL+95,SSL+96]. More precisely we are given a detailed model of the plant – say as a finite state machine – based on internal unobservable events as well as externally observable events of the plant. Some of the internal actions correspond to *faults*. A *diagnoser* for such a plant is a function which given a sequence of observable events generated by the plant, tells us whether an internal fault happened or not. Not all plants are *diagnosable* (in the sense that such a function may not exist) – for example a plant which produces the two behaviours aub and afb, where u and f are internal events with f being the faulty one, and a and b are observable events, is *not* diagnosable since from the observable sequence ab it is impossible to tell whether f happened or not.

Our interest in this paper lies in the fault diagnosis problem for *timed* plants. Here we are given a plant modelled as a timed automaton. The timed automata of Alur and Dill [AD94] are a popular model for time-dependent systems that extend classical finite

⋆ Work partly supported by ACI Cortos, a program of the french ministry of research.
⋆⋆ Part of this work was done during a visit to LSV, ENS Cachan.

state machines with real-time clocks. These clocks can record the passage of time in states, and can be used to guard the occurrence of transitions. A timed automaton generates timed sequences of events – *i.e.* an alternating sequence of real-valued delays and events. The fault diagnosis problem for timed plants is thus to detect faulty behaviours from a given timed sequence of observable events of the plant.

This problem is considerably more difficult in the timed case than in the discrete case. In the discrete case one deals with classical regular languages which have robust closure properties and relatively efficient algorithms for determinization and checking emptiness. Thus one can obtain a diagnoser by essentially determinizing the model of the plant. In the timed setting the problem is compounded by the fact that timed automata are a very expressive formalism. While their language emptiness problem is decidable, they are not determinizable, nor closed under complementation [AD94].

The problem in the timed setting has been studied by Tripakis in [Tri02] where a variety of results have been shown. In particular, it is shown to be decidable to check whether a given timed plant is diagnosable or not, and a diagnoser can be constructed as an online algorithm whenever the plant is indeed diagnosable. The diagnosis algorithm in [Tri02] is based on state estimation; it is somewhat complex, since it involves keeping track of several possible control states and *zones* that the clock values can be in, with every observable action or time delay of the plant. A natural question that one may ask here is: when is there a diagnoser which is realizable as a *deterministic* timed automaton (DTA)? Such a DTA would lead to a more efficient online diagnosis algorithm, since with each observable event or time delay there is a *single* deterministic move in the DTA.

In this paper we consider two deterministic mechanisms namely general DTA's and Event Recording automata (ERA) [AFH94]. For general DTA's we show that it is decidable to check whether a given timed plant has a diagnoser realizable as a DTA, *provided* we are given a bound on the resources (*i.e.* the number of clocks and set of constants) available to the diagnoser. Whenever such a diagnoser exists, we are able to synthesize one. The technique used is to relate the existence of a DTA diagnoser to a winning strategy for a player in a classical state-based two player game.

The decision procedure runs in 2EXPTIME in the size of the plant. We show that this high complexity is unavoidable in that the problem is 2EXPTIME-complete. The completeness argument is based on a reduction from the halting problem of an alternating Turing machine which uses exponential space.

We also look at the problem for a restricted class of DTA's called Event Recording Automata [AFH94]. These are a determinizable subclass of timed automata, in which there is an implicit clock attached to each action. We show that the problem of deciding whether there is a diagnoser realizable as an ERA – again given a bound on the resources we allow for the diagnoser – is decidable in PSPACE. Once again the problem is shown to be complete for PSPACE.

Other recent works show an increasing interest in partial observability (e.g. learning [GJL04]); this increases complexity of systems as several control problems become undecidable under partial observability [DM02,BDMP03] . However it seems useful to combine this issue with on-the-fly analysis (for example monitoring [KT04] or run-time model-checking [KPA03]). This work puts together these two aspects: the framework is fault diagnosis of partially observable systems and the deterministic mechanisms we consider fit the online constraint.

The plan of the paper is as follows. After introducing notations and definitions in section 2, we will present the problem of fault diagnosis (section 3), starting with results from [Tri02] and going on to the problem we look at. We then present our result on the class of deterministic timed automata (section 4) and then on the class of event-recording automata (section 5). The paper contains only sketches of proof. Detailed proofs can be found in [Che04] (written in french).

2 Preliminaries

For a set Γ, let Γ^* be the set of *finite* sequences of elements in Γ.

Timed words. We consider a finite set of *actions* Σ and as time domain the set $\mathbb{Q}_{\geq 0}$ of non-negative rationals. A *timed word* over Σ is a sequence in $(\Sigma \cup \mathbb{Q}_{\geq 0})^*$, *i.e.* a finite sequence $\rho = \gamma_1, \gamma_2, \ldots$ where each γ_i is either an event in Σ or a delay in $\mathbb{Q}_{\geq 0}$.[3] A set of timed words will be called a *timed language*. If ρ is a timed word, we define $\mathsf{time}(\rho)$ to be the sum of all delays in ρ. If $\Sigma' \subseteq \Sigma$ and if ρ is a timed word, we denote by $\pi_{\Sigma'}(\rho)$ its projection over the alphabet Σ', which means that we erase actions not in Σ'. For example, if $\Sigma' = \{a, c\} \subseteq \{a, b, c\} = \Sigma$, then $\pi_{\Sigma'}(0.5, a, 0.7, b, 0.3, c) = 0.5, a, 0.7, 0.3, c$ which reduces naturally to the timed word $0.5, a, 1, c$. This operation extends in a natural way to timed languages.

Clocks, operations on clocks. We consider a finite set X of variables, called *clocks*. A *clock valuation* over X is a mapping $v : X \to \mathbb{Q}_{\geq 0}$ that assigns to each clock a time value. We use $\mathbf{0}$ to denote the valuation which sets each clock $x \in X$ to 0. If $t \in \mathbb{Q}_{\geq 0}$, the valuation $v + t$ is defined as $(v + t)(x) = v(x) + t$ for all $x \in X$. If Y is a subset of X, the valuation $v[Y \leftarrow 0]$ is defined as: for each clock x, $(v[Y \leftarrow 0])(x) = 0$ if $x \in Y$ and is $v(x)$ otherwise.

The set of *constraints* (or *guards*) over a set of clocks X, denoted $\mathcal{G}(X)$, is given by the syntax "$g ::= (x \sim c) \mid (g \wedge g)$" where $x \in X$, $c \in \mathbb{Q}_{\geq 0}$ and $\sim$ is one of $<$, $\leq$, $=$, $\geq$, or $>$. We write $v \models g$ if the valuation v satisfies the clock constraint g, and is given by $v \models (x \sim c)$ if $v(x) \sim c$ and $v \models (g_1 \wedge g_2)$ if $v \models g_1$ and $v \models g_2$. The set of valuations over X which satisfy a guard $g \in \mathcal{G}(X)$ is denoted by $[\![g]\!]_X$, or just $[\![g]\!]$ when X is clear from the context.

Symbolic alphabet and timed automata. Let Σ be an alphabet of actions, and X a finite set of clocks. A *symbolic alphabet* Γ based on (Σ, X) is a finite subset of $\mathcal{G}(X) \times \Sigma \times 2^X$. As used in the framework of timed automata [AD94], a symbolic word $\gamma = (g_i, b_i, Y_i)_{1 \leq i \leq k} \in \Gamma^*$ gives rise to a set of timed words, denoted *tw*(γ). We interpret the symbolic action (g, b, Y) to mean that action b can happen if the guard g is satisfied, with the clocks in Y being reset after the action. Formally, let $\rho = d_0, a_1, d_1, a_2, \ldots$ be a timed word. Then $\rho \in$ *tw*(γ) if there exists a sequence $v = (v_i)_{i \geq 1}$ of valuations such

[3] Following [Tri02], we use this definition of timed words, a more classical definition of timed words as in [AD94] could be used as well.

that for all $i \geq 1$, $a_i = b_i$, $v_{i-1} + d_{i-1} \models g_i$ and $v_i = (v_{i-1} + d_{i-1})[Y_i \leftarrow 0]$ (with the convention $v_0 = \mathbf{0}$).

A *timed automaton* (TA for short) is a tuple $\mathcal{A} = (\Sigma, X, Q, q_0, F, \longrightarrow, \mathit{Inv})$ where Σ is a finite alphabet of actions, X is a finite set of clocks, Q is a finite set of states, $q_0 \in Q$ is the initial state, $F \subseteq Q$ is the set of final states, $\longrightarrow \subseteq Q \times \Gamma \times Q$ is a finite set of transitions over some symbolic alphabet Γ based on (Σ, X), and $\mathit{Inv} : Q \to \mathcal{G}(X)$ is an invariant function. The timed automaton $\mathcal{A}$ is said to be *deterministic* if, for every state, the set of symbolic actions enabled at that state is time-deterministic, *i.e.* do not contain distinct symbolic actions (g, a, Y) and (g', a, Y') with $[\![g]\!] \cap [\![g']\!] \neq \emptyset$. The class of deterministic timed automata is denoted DTA. An *event-recording automaton* (ERA for short) [AFH94] is a timed automaton $(\Sigma, X, Q, q_0, F, \longrightarrow, \mathit{Inv})$ where $X = \{x_a \mid a \in \Sigma\}$ and $q \xrightarrow{g,a,Y} q'$ implies $Y = \{x_a\}$. Informally the clock x_a stores the time elapsed since the last occurrence of action a. We extend the above definitions to allow ε-transitions (or silent transitions) in our timed automata [BDGP98].

For convenience, we will assume that all guards in timed automata are compatible with invariants in the following sense. If $q \xrightarrow{g,a,Y} q'$ is a transition, we want that $[\![g]\!]$ be included in $[\![\mathit{Inv}(q)]\!]$, and $[Y \leftarrow 0][\![g]\!]$ be included in $[\![\mathit{Inv}(q')]\!]$. If it is not the case, it is easy to transform the timed automaton so that this condition holds.

A *path* in a TA $\mathcal{A}$ is a finite sequence of consecutive transitions:

$$q_0 \xrightarrow{g_1,a_1,Y_1} q_1 \ldots q_{k-1} \xrightarrow{g_k,a_k,Y_k} q_k,\ \text{s.t}\ \forall 1 \leq i \leq k,\ (q_{i-1}, g_i, a_i, Y_i, q_i) \in \longrightarrow$$

The path is said to be *accepting* in $\mathcal{A}$ if it ends in a final state $q_k \in F$.

A timed automaton $\mathcal{A}$ can then be interpreted as a classical finite automaton on the symbolic alphabet Γ. Viewed as such, $\mathcal{A}$ accepts (or generates) a language of symbolic words, $L_{sym}(\mathcal{A}) \subseteq \Gamma^*$, constituted by the labels of the accepting paths in $\mathcal{A}$. We will be more interested in the timed language generated by $\mathcal{A}$, denoted $L(\mathcal{A})$, and defined by $L(\mathcal{A}) = \mathit{tw}(L_{sym}(\mathcal{A}))$.

Synchronized product. Let $\mathcal{A}_i = (\Sigma_i, X_i, Q_i, q_0^i, F_i, \longrightarrow_i, \mathit{Inv}_i)$ be two timed automata. Without loss of generality we assume that X_1 and X_2 are disjoint. The *synchronized product* of $\mathcal{A}_1$ and $\mathcal{A}_2$ is defined as the timed automaton $\mathcal{A}_1 \parallel \mathcal{A}_2 = (\Sigma, X, Q, q_0, F, \longrightarrow, \mathit{Inv})$ where $\Sigma = \Sigma_1 \cup \Sigma_2$, $X = X_1 \cup X_2$, $Q = Q_1 \times Q_2$, $q^0 = (q_1^0, q_2^0)$, $F = F_1 \times F_2$ and $(q_1, q_2) \xrightarrow{g,a,Y} (q_1', q_2')$ whenever one of the following conditions holds:

- $a \in \Sigma_1 \cap \Sigma_2$ and there exist $q_1 \xrightarrow{g_1,a,Y_1}_1 q_1'$ and $q_2 \xrightarrow{g_2,a,Y_2}_2 q_2'$ with $g = g_1 \wedge g_2$ and $Y = Y_1 \cup Y_2$
- $a \in \Sigma_i \setminus \Sigma_j$ (with $i \neq j$), there exists $q_i \xrightarrow{g,a,Y}_i q_i'$ and $q_j' = q_j$

This synchronized product is the classical composition where both components synchronize on common actions.

Region automata. Region automata have been proposed in [AD94] for abstracting timed behaviours. Regions are classes of an equivalence relation over valuations which satisfy the nice property that two equivalent valuations have equivalent (time and discrete) successors. The region automaton construction is the core of the decidability proof for checking emptiness of timed automata. We will make use of the region automaton in Lemma 2. In the following if $\mathcal{A}$ is a timed automaton, we will denote its region automaton by $\mathcal{R}(\mathcal{A})$.

Granularity. In the following, we will consider models with bounded resources, for example the subclass of DTA's using 5 clocks and integer constants smaller than 7. Fixing resources of models has been frequently done in the past in several contexts: satisfiability of timed μ-calculus [LLW95], controller synthesis [DM02,BDMP03], testing [KT04]. In all cases, fixing the resources helps in getting decidability results, and it is quite natural when our aim is to synthesize a system (with given physical resources).

We formalize this notion by defining a measure of the clocks and constants used in a set of constraints. A *granularity* is a tuple $\mu = (X, m, max)$ where X is a set of clocks, m is a positive integer and $max : X \longrightarrow \mathbb{Q}_{\geq 0}$ a function which associates with each clock of X a positive rational number. The granularity of a finite set of constraints is the tuple (X, m, max) where X is the exact set of clocks mentioned in the constraints, m is the least common multiple of the denominators of the constants mentioned in the constraints, and max records for each $x \in X$ the largest constant it is compared with. A granularity $\nu = (X', m', max')$ is said to be *finer* than a granularity $\mu = (X, m, max)$ (or equivalently μ is said to be *coarser* than ν) if $X \subseteq X'$, m divides m' and for all $x \in X$ $max'(x) \geq max(x)$. A constraint g is called μ-*granular* if it belongs to some set of constraints of granularity μ (note that a μ-granular constraint is also ν-granular for any granularity ν finer than μ). We denote the set of all μ-granular constraints by $\mathcal{G}(\mu)$. A constraint $g \in \mathcal{G}(\mu)$ is called $\mu-$*atomic* if for all $g' \in \mathcal{G}(\mu)$, either $[\![g]\!]_X \subseteq [\![g']\!]_X$ or $[\![g]\!]_X \cap [\![g']\!]_X = \emptyset$. Let $atoms_\mu$ denote this set of μ-atomic constraints. By the granularity of a timed automaton, we will mean the granularity of the set of constraints used in it. For such an automaton, the granularity μ represents its *resources* in terms of clocks and constants. We denote by DTA$_\mu$ (resp. ERA$_\mu$) the class of DTA's (resp. ERA's) whose granularity is coarser than μ. Let $\mu = (X, m, max)$ be a granularity over Σ, we denote by Γ_μ the symbolic alphabet over μ (*i.e.* the set $atoms_\mu \times \Sigma \times 2^X$) and $\mathcal{U}_\mu$ the universal single-state automaton over symbolic alphabet Γ_μ.

Let $\mu = (X, m, max)$ be a granularity over the alphabet Σ and $\nu = (X', m', max')$ be a granularity finer than μ over $\Sigma' \supseteq \Sigma$. For $(g', a', Y') \in atoms_\nu \times \Sigma' \times X'$ a symbolic letter over ν, we define the projection $(g', a', Y')\lceil\mu$ as follows: let g be the unique μ-atomic constraint such that $[\![g']\!]_{X'} \subseteq [\![g]\!]_{X'}$, $Y = Y' \cap X$, then $(g', a', Y')\lceil\mu$ is defined to be (g, a', Y) if $a' \in \Sigma$ and ε (the empty word) if $a' \notin \Sigma$. If μ is a granularity and $\mathcal{A}$ a TA whose granularity is finer than μ, we denote by $\mathcal{A}\lceil\mu$ the TA in which every transition label is replaced by its projection on μ.

3 The Fault Diagnosis Problem

In this section, we present the problem of fault diagnosis for timed systems. First we recall basic definitions and existing work and then we present our approach which involves fault diagnosis by timed automata.

3.1 Existing Work

In this section we present the basic notions and the main results from [Tri02].

For the rest of the paper, Σ_o denotes an alphabet of *observable* events while Σ_u denotes an alphabet of *unobservable* events. We assume that Σ_o and Σ_u are disjoint. Given a timed word ρ, its *observation* is its projection over Σ_o, *i.e.* $\pi_{\Sigma_o}(\rho)$. In what follows, we will simply write π instead of π_{Σ_o}. A run $\rho = \beta_1, \beta_2, \ldots, \beta_p$ is called *faulty* if there exists $i \in \mathbb{N}$ such that $\beta_i = f$. It is called *Δ-faulty* if for one such i, $\mathsf{time}(\beta_{i+1}, \ldots, \beta_p) \geq \Delta$.

A *plant* is a tuple $\mathcal{P} = (\Sigma_o, \Sigma_u, Q, q_0, \longrightarrow, X, \mathit{Inv})$ where $(\Sigma_o \cup \Sigma_u, X, Q, q_0, Q, \longrightarrow, \mathit{Inv})$ is a TA (thus a plant has all states final). A *run* of the plant is simply a timed word generated by the plant. Given a plant $\mathcal{P}$, we denote by $L_{\Delta f}(\mathcal{P})$ the set of Δ-faulty runs of $\mathcal{P}$ and $L_{\neg f}(\mathcal{P})$ the set of non-faulty runs of $\mathcal{P}$. From now on, when there is no ambiguity, $L_{\Delta f}(\mathcal{P})$ (resp. $L_{\neg f}(\mathcal{P})$) will be denoted $L_{\Delta f}$ (resp. $L_{\neg f}$).

Fault diagnosis aims at computing a function which, given an observation, decides if a fault has occurred or not, and which always announces faults at most Δ time units after it has occurred. Such a function should announce a fault on all Δ-faulty runs and should not announce a fault on non-faulty runs; this is captured by the next definition, which is an equivalent reformulation of Tripakis' notion of diagnosability.

Definition 1. *A plant $\mathcal{P}$ is called* Δ-diagnosable *if there exists a recursive language L such that*

$$\pi(L_{\Delta f}) \subseteq L \subseteq \pi(L_{\neg f})^c \ .$$

This definition raises the following computational problem where $\Delta \in \mathbb{Q}_{\geq 0}$:

Problem 1 (Δ-diagnosability) *Given a plant $\mathcal{P}$, decide whether $\mathcal{P}$ is Δ-diagnosable or not.*

This problem is solved in [Tri02]:

Theorem 1 ([Tri02]). *Δ-diagnosability is PSPACE-complete.*

3.2 Diagnosability by Automata

The problem solved in [Tri02] is very general: the diagnoser is only supposed to be recursive, which, in practice, may be a complex algorithm. The algorithm proposed in [Tri02] is based on state estimation in a TA with ε-transitions, its complexity to diagnose faults from an observation is doubly exponential in the size of the plant and in the size of the observation, though an algorithm based on regions (and no more on zones) with a complexity exponential in both the size of the plant and of the observation could be proposed as well.

This high complexity in the size of the observation is not satisfactory if we want to perform "*online diagnosis*", *i.e.* if we want the diagnoser to detect faults from real-time observations of the system.

This has motivated the definition of diagnosability using timed automata: we are no more looking for a diagnoser which may be a general algorithm but for a diagnoser which will be a timed automaton. With such a diagnoser, the complexity of detecting faults *online* will no more be (doubly) exponential in the length of the observation since after each observable action the diagnoser has just to make a single deterministic move. We formalize this notion of diagnosability using timed automata as follows.

Definition 2. *Let $\mathcal{C}$ be a class of timed automata. Let $\mathcal{P}$ be a plant. We say that $\mathcal{P}$ is* Δ-$\mathcal{C}$-diagnosable *whenever there exists some $\theta \in \mathcal{C}$ such that*

$$\pi(L_{\Delta f}) \subseteq L(\theta) \subseteq \pi(L_{\neg f})^c \ .$$

We call such a θ a Δ-$\mathcal{C}$-diagnoser *for $\mathcal{P}$.*

The sets of diagnosers which will be of interest to us are deterministic mechanisms like DTA's and ERA's. In the sequel we will study the following problem, where $\Delta \in \mathbb{Q}_{\geq 0}$ and $\mathcal{C}$ is a class of automata:

Problem 2 (Δ-$\mathcal{C}$-diagnosability) *Given a plant $\mathcal{P}$, decide whether $\mathcal{P}$ is Δ-$\mathcal{C}$-diagnosable or not.*

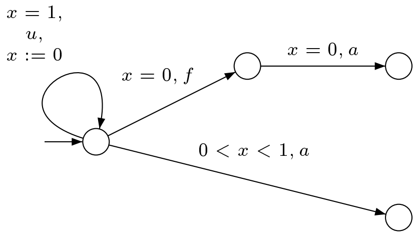


Fig. 1. Plant diagnosable but not DTA-diagnosable

We first notice that this problem is distinct from problem 1: every DTA-diagnosable plant is diagnosable, but some diagnosable plants are not DTA-diagnosable as illustrated by the plant in Fig. 1. Indeed, a diagnoser will announce a fault if action a happens at an integer date (this can not be expressed by a DTA, as shown in [BDGP98]).

4 Diagnosability with Deterministic Timed Automata

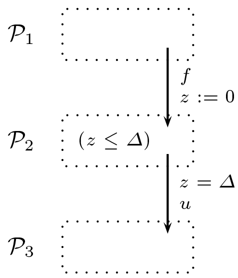
We do not consider the general problem of Δ-DTA-diagnosability but we restrict ourselves to the case when the resources of the diagnoser are fixed. We thus consider in this section the Δ-DTA$_\mu$-diagnosability problem: we aim at constructing diagnosers which are DTA's with a fixed granularity μ over Σ_o. In this framework, our main theorem is the following.

Theorem 2. *Let μ be a granularity over observable events and $\Delta \in \mathbb{Q}_{\geq 0}$. The Δ-DTA$_\mu$-diagnosability problem is 2EXPTIME-complete.*

Before presenting the proof of this theorem, let us first state the two following useful lemmas: the first lemma states that we can construct timed automata recognizing non-faulty and Δ-faulty runs while the second one explains how behaviours of the plant can be seen "through" the granularity μ.

Lemma 1. *Let $\mathcal{P}$ be a plant and $\Delta \in \mathbb{Q}_{\geq 0}$. We can construct in polynomial time timed automata with ε-transitions $\mathcal{P}_{\neg f}$ and $\mathcal{P}_{\Delta f}$ such that $L(\mathcal{P}_{\neg f}) = \pi(L_{\neg f})$ and $L(\mathcal{P}_{\Delta f}) = \pi(L_{\Delta f})$.*

Proof (Sketch). $\mathcal{P}_{\neg f}$ is constructed from $\mathcal{P}$ as follows: erase transitions labelled by f (to prevent $\mathcal{P}$ from making faults), replace all transitions labelled by $u \in \Sigma_u$ by ε-transitions and make all states final.
Before constructing $\mathcal{P}_{\Delta f}$, we modify the plant $\mathcal{P}$, and construct a new plant $\mathcal{P}'$, which has the same observations as $\mathcal{P}$, and in which information on whether the current run is Δ-faulty or not is stored in the current state of $\mathcal{P}'$. $\mathcal{P}'$ is constructed as three copies of $\mathcal{P}$, say $\mathcal{P}_1, \mathcal{P}_2$ and $\mathcal{P}_3$: doing a fault in $\mathcal{P}_1$ leads to $\mathcal{P}_2$; in $\mathcal{P}_2$ the automaton behaves like $\mathcal{P}$ for Δ time units before switching to $\mathcal{P}_3$ by an unobservable action u. This can easily be formalized using a fresh clock z which is reset when a fault is done in $\mathcal{P}_1$ (leading to plant $\mathcal{P}_2$), adding an invariant $z \leq \Delta$ in all locations of $\mathcal{P}_2$ and as soon as $z = \Delta$, a transition labelled by u leads to $\mathcal{P}_3$. In the following we will assume that the plant is already given as $\mathcal{P}'$ and will call states of $\mathcal{P}_1$ "non-faulty" and states of $\mathcal{P}_3$ "Δ-faulty".
To get $\mathcal{P}_{\Delta f}$, we just replace transitions labelled by $u \in \Sigma_u$ by ε-transitions in $\mathcal{P}'$ and mark all states of $\mathcal{P}_3$ as final. It is not difficult to check that this automaton recognizes $\pi(L_{\Delta f})$.



□

The following lemma is a consequence of the region automata construction:

Lemma 2. *Let $\mathcal{A}$ be a timed automaton and μ a granularity. The region automaton $\mathcal{R}(\mathcal{A} \parallel \mathcal{U}_\mu)\lceil\mu$ recognizes the set*

$$L_{sym}\left(\mathcal{R}(\mathcal{A} \parallel \mathcal{U}_\mu)\lceil\mu\right) = \{\gamma \in \Gamma_\mu^* \mid \exists \rho \in L(\mathcal{A}) \text{ s.t. } \pi(\rho) \in tw(\gamma)\}$$

4.1 Δ-DTA$_\mu$-Diagnosability is in 2EXPTIME

In [BDMP03], the control problem under partial observability is proved to be in 2EXPTIME using a timed game construction. A similar construction can be carried out, but we present here a direct construction which gives more intuition.

Lemma 3. *Δ-DTA$_\mu$-diagnosability is in 2EXPTIME.*

Proof (Sketch). Let $\mathcal{P} = (\Sigma_o, \Sigma_u, Q, q_0, \longrightarrow, X, Inv)$ be a plant and $\mu = (Y, m, max)$ a granularity over Σ_o. We will construct a classical (untimed) safety game $\mathcal{G}_{\mathcal{P},\mu,\Delta}$: it is a two-player turn-based perfect information game over a finite graph where one player wants to stay in the "safe" states, whereas the other player wants to enforce an "unsafe" state. We refer to [GTW02] for basics results on games. In our case, the two players are the "*diagnoser*" and the "*environment*", and player "*diagnoser*" will have a winning strategy in game $\mathcal{G}_{\mathcal{P},\mu,\Delta}$ if and only if $\mathcal{P}$ is Δ-DTA$_\mu$-diagnosable.

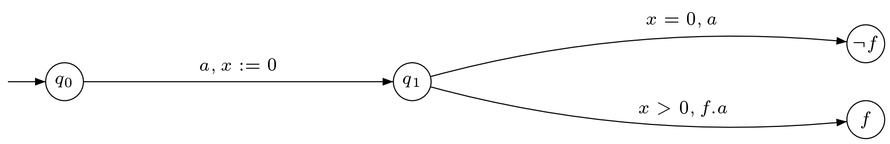
The arena of the game is constructed as follows: we first compute the region automaton $\mathcal{R}$ of $\mathcal{P}_{\Delta f} \parallel \mathcal{U}_\mu$. Its granularity is finer than μ, because it takes into account clocks and constraints from $\mathcal{U}_\mu$. To express that not everything can be observed, we project this automaton over the granularity μ to get $\mathcal{R}\lceil\mu$ (intuitively this represents how runs can be seen "through" the granularity μ). This automaton (considered as a finite automaton over the alphabet Γ_μ) is not deterministic and has ε-transitions, we thus determinize it as a classical finite automaton by the usual subset construction and denote $\mathcal{K}$ the result. A state of $\mathcal{K}$ is a set $\{(q_1, R_1), \cdots, (q_k, R_k)\}$ where q_i's are states of $\mathcal{P}$ and R_i's regions; being in such a state means that according to the observation the plant can be in a state (q_i, v_i) with $v_i \in R_i$.

Finally $\mathcal{G}_{\mathcal{P},\mu,\Delta}$ is obtained from $\mathcal{K}$ by splitting every transition $q \xrightarrow{g,a,Y} q'$ of $\mathcal{K}$ into two transitions $q \xrightarrow{g,a} (q, g, a)$ and $(q, g, a) \xrightarrow{Y} q'$ where (q, g, a) is a new state. The intuition behind this split is that player "*environment*" chooses which action is done and at what time this action is done (state q will thus be an "*environment*" state) whereas player "*diagnoser*" chooses which clocks are reset (the state (q, g, a) is a "*diagnoser*" state). The forbidden states of this safety game are those states of $\mathcal{K}$ which contain both "non-faulty" and "Δ-faulty" states of $\mathcal{R}\lceil\mu$.

Using lemma 2, we can prove that player "*diagnoser*" has a winning strategy in the game $\mathcal{G}_{\mathcal{P},\mu,\Delta}$ to avoid the forbidden states if and only if $\mathcal{P}$ is Δ-DTA$_\mu$-diagnosable. $\mathcal{G}_{\mathcal{P},\mu,\Delta}$ is a simple untimed game with a safety objective, it is easy to synthesize positional winning strategies when some exist. Such a winning strategy can be obtained from $\mathcal{G}_{\mathcal{P},\mu,\Delta}$ by erasing some of the reset transitions (*i.e.* transitions labelled by some subset Y). From this automaton, we can easily synthesize a diagnoser for $\mathcal{P}$ (by taking as final those states where all regions are Δ-faulty and merging $q \xrightarrow{g,a} (q, g, a)$ with $(q, g, a) \xrightarrow{Y} q'$ into $q \xrightarrow{g,a,Y} q'$).

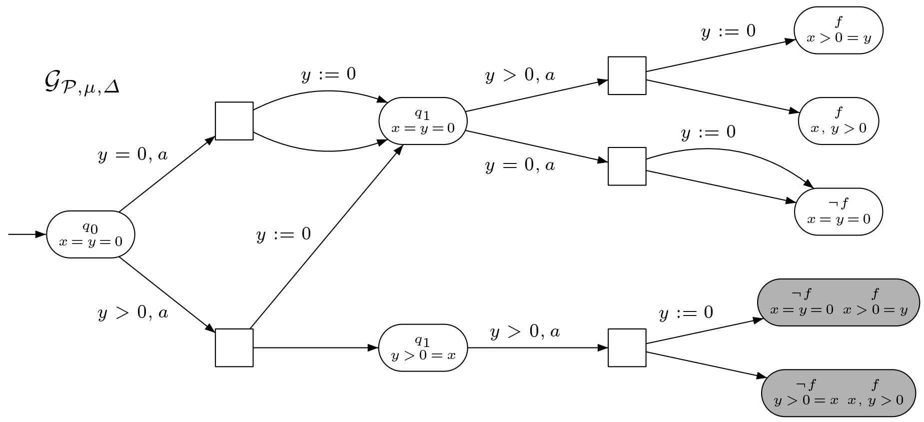
The complexity of extracting winning strategies from safety (untimed) games is linear in the size of the arena, the complexity of deciding whether a Δ-DTA$_\mu$-diagnoser exists (and constructing it) is thus doubly exponential because the size of $\mathcal{R}$ is exponential in the size of $\mathcal{P}$ and μ (see [AD94]) and the size of $\mathcal{K}$ is exponential in the size of $\mathcal{R}$ thus doubly exponential in the size of $\mathcal{P}$ and μ. $\square$

Example 1. We illustrate the proof on a small example: consider the following plant, the granularity $\mu = (\{y\}, 1, 0)$ and the delay $\Delta = 0$.



The notation $f.a$ is for an action f immediately followed by an a. This plant is Δ-DTA$_\mu$-diagnosable: a diagnoser resetting his clock when reading the first a will be able to diagnose $\mathcal{P}$ simply by checking the value of his clock when reading the second a.

The game constructed in the previous proof is depicted on the next picture:



The states that player diagnoser must avoid in the above game are the gray states which contain both faulty and non-faulty regions (informally states in which the diagnoser cannot know if the run of the plant is faulty or not). In the game, "circle"-states belong to the "environment" player while "square"-states belong to the "diagnoser" player. In the game-graph, it is easy to see that "diagnoser" has a winning strategy. The problematic state is the bottom-left-most square-state: if "diagnoser" plays action "$y := 0$", he wins; if he chooses the other transition, player "environment" can win by next playing "$y > 0, a$". This confirms what we have noticed: if a diagnoser resets his clock when reading the first a, he can diagnose correctly; but if he does not reset his clock, he will be unable to diagnose the plant.

4.2 Δ-DTA$_\mu$-Diagnosability is 2EXPTIME-Hard

The proof uses a reduction from the halting problem of alternating Turing machines using exponential space. We only sketch the reduction, details can be found in [Che04].

Let $\mathcal{M}$ be an alternating Turing machine using exponential space and let w_0 be an input for $\mathcal{M}$. We will construct a plant $\mathcal{P}$ such that there is a Δ-DTA$_\mu$-diagnoser for $\mathcal{P}$ (with $\mu = (\{t\}, 2, 1)$ and $\Delta = 1$) if and only if $\mathcal{M}$ accepts w_0. We somehow want to force a potential diagnoser θ for $\mathcal{P}$ to play the sequence of configurations which accepts w_0. The role of the plant is to give inputs to the diagnoser so that it can verify that the diagnoser really plays the accepting sequence of configurations. The diagnoser will have to play a sequence of configurations $C_0 \# C_1 \# \ldots \# C_k$ where each C_i is a configuration of $\mathcal{M}$, C_{i+1} is the successor of C_i and C_0 is the initial configuration encoding the input w_0. The behaviour of $\mathcal{P}$ is depicted on Fig. 2. $\mathcal{P}$ produces a's (on the figure, one line of a's corresponds to a configuration, *i.e.* to an exponential number of a's) and checks that the diagnoser plays the configurations of $\mathcal{M}$ correctly by performing one test. As the decision to perform the test is done in a *non-observable* way (u actions are non-observable), to be correct, a diagnoser cannot cheat and has to simulate $\mathcal{M}$. $\#$'s are observable and are indexed to represent alternation of $\mathcal{M}$ (in case of a universal configuration of $\mathcal{M}$, the diagnoser has to know which transition rule the plant wants to follow). As already said, a configuration needs an exponential number of a's, as the discrete structure of $\mathcal{P}$ cannot count, we use clocks for counting this exponential

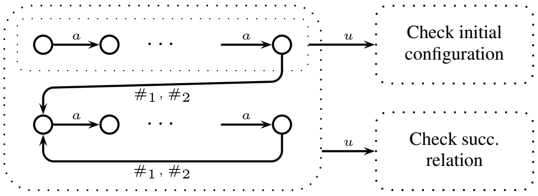


Fig. 2. Shape of the plant for the reduction

number of a's. We cannot give all details here and better refer the reader to [Che04]. Note that the two checks can be encoded by 3SAT-formulae (in conjunctive normal form [Pap94]). We will now explain how such formulae can be encoded.

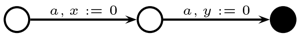


Fig. 3. Choice of a variable

Given a 3SAT-formula φ we want to construct a plant $\mathcal{P}$ such that $\mathcal{P}$ is Δ-DTA$_\mu$-diagnosable if and only if φ is satisfiable. We first explain how a diagnoser θ will choose the truth of a propositional variable p. Consider the plant in Fig. 3. When two a's have been done, θ will know that the plant is in the black state but it will know the value of at most one of the two clocks x and y because θ is supposed to have only one clock t. The choice *true* for p will be encoded by the fact that the clock t has the same value as the clock x (we will say in this case that θ *has stored* x). We now show how we can force a diagnoser θ to set p to *true*, *i.e.* to store clock x. Fig. 4 illustrates the construction. The main idea is that if θ stores x, after three a's, if the

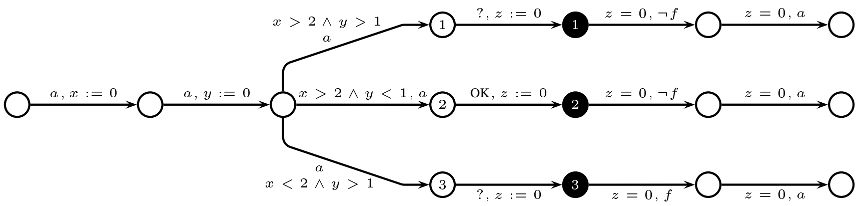


Fig. 4. Plant ensuring that p is set to *true*

constraint $x < 2$ holds he will know that $\mathcal{P}$ is in state 3 whereas if the constraint $x > 2$ holds he may not know if $\mathcal{P}$ is in state 1 or in state 2. Similarly if θ stores y, he may hesitate between state 1 and state 3, but if $\mathcal{P}$ is in state 2, θ knows it. To force θ storing x, $\mathcal{P}$ will give him one more information which will be an observable action "OK" or "?" with which θ will break the uncertainty between state 1 and state 2, but not the uncertainty between state 1 and state 3. Thus if θ stores x he will precisely know in which black state $\mathcal{P}$ is after having done four actions and will thus be able to diagnose $\mathcal{P}$ correctly, but if θ stores y he may be uncertain between states 1 and 3. and will thus not be able to diagnose correctly $\mathcal{P}$ because the execution after state 1 is non-faulty ($\neg f$ represents a non-observable non-faulty action) whereas the execution after state 3 is faulty. Of course a similar construction (breaking the uncertainty between 3 and 1) can be done for proposition $\neg p$, thus enforcing clock y to be stored by a diagnoser. The previous construction is extended to clauses of 3SAT by branching automata as the one on Fig. 4. It's better to explain the construction on an example. We choose the formula $p_1 \vee \neg p_3$ (thus ignoring p_2). The left-most frame of Fig. 5 corresponds to the previously

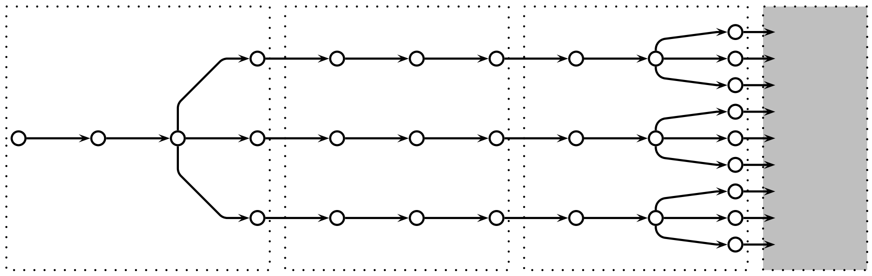


Fig. 5. Plant ensuring that $p_1 \vee \neg p_3$ is *true*

described choice for p_1, the second frame is for p_2 (p_2 is not used in the clause, thus no branching is needed, but it may be used by other clauses), there is no constraint on the choice for p_2, the third frame is for the choice for p_3. The last frame is for breaking the uncertainty between conflicting runs (adding "OK" and "?" labels followed by either a fault or no fault, as previously). It could be argued that there is no need of the frame for p_2 as p_2 is not used in the clause. However, as we have a set of clauses to be satisfied, we need to have this linear part for p_2. Indeed, for a formula $\varphi = \bigwedge_{i=1}^{n} \psi_i$ with ψ_i clause, the plant for φ will be as on Fig. 6. Non-observable action u's role is to hide what clause the plant is going to check. To be correct, θ must thus satisfy all clauses (with a unique valuation for the propositional variables). This plant can be diagnosed if and only if φ is satisfiable.

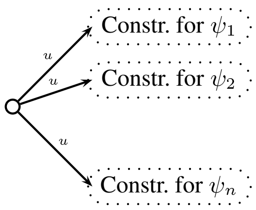


Fig. 6. Plant for a 3SAT-formula

This concludes the proof of 2EXPTIME-hardness of the Δ-DTA$_\mu$-diagnosability problem. Note that a similar construction could be done when the diagnoser can use an arbitrary (but fixed) number of clocks. $\square$

5 Diagnosability by Event-Recording Timed Automata

The class ERA [AFH94] appears as a natural class of automata for observing systems because clocks and thus timing information are dependent on events (thus precisely what is observed). The fundamental properties of ERA's we will use are the following:

- ERA's are *determinizable* [AFH94]
- ERA's are "*input-determined*" [DT04]: after having read a timed word, the truth of a guard is completely determined by the word itself. This implies in particular that if w is a timed word and μ a granularity for the observable events, there exists a unique symbolic word $\gamma \in \Gamma_\mu$ such that $w \in tw(\gamma)$.

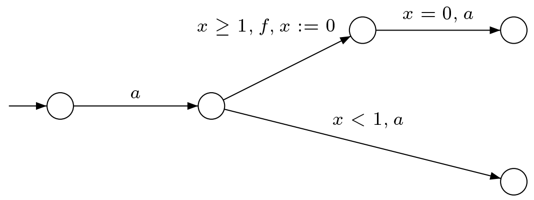


Fig. 7. Plant DTA-diagnosable but not ERA-diagnosable

As previously we restrict to ERA's with bounded resources and tackle the Δ-ERA$_\mu$-diagnosability problem for a fixed granularity μ. It is worth noticing first that ERA-diagnosability is less powerful than DTA-diagnosability as illustrated by the plant on Fig. 7: a DTA-diagnoser with one clock for this plant does not reset its clock when the first a occurs and checks the value of its clock when the second a occurs; it announces a fault only if the value is greater than 1. There is no ERA-diagnoser for this plant.

Deciding diagnosability for ERA's is much easier than for DTA's, as stated by the next theorem. Note that the PSPACE complexity is "optimal" for the diagnosability problem in the sense that there is no hope for finding interesting classes of diagnosers with a lower complexity.

Theorem 3. *Let μ be a granularity over observable events and $\Delta \in \mathbb{Q}_{\geq 0}$. The Δ-ERA$_\mu$-diagnosability problem is PSPACE-complete.*

Proof (Sketch). Let us first argue why Δ-ERA$_\mu$-diagnosability is PSPACE-hard. This can easily be shown by reducing the reachability problem in a timed automaton to Δ-ERA$_\mu$-diagnosability. Consider a timed automaton $\mathcal{A}$ over alphabet Σ and add two fresh unobservable actions u and f (the fault) which are done immediately after having reached some final state of $\mathcal{A}$. The modified automaton is noted $\mathcal{P}$. For every $\Delta > 0$, for every granularity μ, taking $\Sigma_o = \Sigma$ and $\Sigma_u = \{u, f\}$, we get that a final state is reachable in $\mathcal{A}$ if and only if there is no Δ-ERA$_\mu$-diagnoser for $\mathcal{P}$ (in case no final state is reachable, the diagnoser is trivial as it needs not accept anything).

We will now sketch the proof of PSPACE-membership of the problem. Let $\mathcal{P} = (\Sigma_o, \Sigma_u, Q, q_0, \longrightarrow, X, \mathit{Inv})$ be a plant and $\mu = (Y, m, max)$ a granularity over Σ_o. Let $S_{\Delta f} = \mathcal{R}(\mathcal{P}_{\Delta f} \parallel \mathcal{U}_\mu)\lceil\mu$ and $S_{\neg f} = \mathcal{R}(\mathcal{P}_{\neg f} \parallel \mathcal{U}_\mu)\lceil\mu$. Informally $S_{\Delta f}$ (resp. $S_{\neg f}$) recognizes all observations that may come from Δ-faulty (resp. non-faulty) runs. The result can finally be deduced from lemma 2 and from the following lemma:

Lemma 4. *The following properties are equivalent:*

(i) $\mathcal{P}$ *is* Δ*-ERA*$_\mu$*-diagnosable,*
(ii) $\{\gamma \in \Gamma_\mu^* \mid \exists \rho_1 \in L_{\neg f}$ *and* $\rho_2 \in L_{\Delta f}$ *s.t.* $\pi(\rho_1), \pi(\rho_2) \in \mathit{tw}(\gamma)\}$ *is empty,*
(iii) $L(S_{\Delta f}) \cap L(S_{\neg f})$ *is empty.*

□

Note that if $\mathcal{P}$ is Δ-ERA$_\mu$-diagnosable, the proof provides a diagnoser: one can prove that $\pi(L_{\Delta f}) \subseteq L(S_{\Delta f}) \subseteq L(S_{\neg f})^c \subseteq \pi(L_{\neg f})^c$, $S_{\Delta f}$ is a Δ-ERA$_\mu$-diagnoser for $\mathcal{P}$. Moreover, $S_{\Delta f}$ is an *optimal* diagnoser in the sense that it is the smallest diagnoser (for language inclusion). This property is specific to the model of ERA's; such a property does not hold for the class DTA$_\mu$.

6 Conclusion

We have shown that diagnosability using DTA's and ERA's is a decidable problem when resources of the diagnoser are fixed. Moreover if a diagnoser exists, it is possible to construct one: the size of such a diagnoser is doubly exponential in the granularity and in the size of the plant. Thus if we assume that the diagnoser can be pre-computed, diagnosing online becomes exponential in the granularity and in the plant, but only **linear** in the length of the observation. The use of deterministic mechanisms thus allows to construct diagnosers with short response time, which is crucial in fault detection.

We have also pointed out a significant complexity jump between two classes of potential diagnosers: existence of diagnoser in the class DTA (with bounded resources) is 2EXPTIME-complete whereas it is PSPACE-complete for the class ERA (with bounded resources). The class ERA thus appears as a natural and useful class of diagnosers.

This work is related to conformance testing where the aim is to generate testers for a given specification. Such a problem has for example been considered in [KT04] where an algorithm for building small testers (with fixed resources) is proposed. We think that our approach (based on games) could be applied in such a framework as well.

As future work we aim at studying the diagnosability problem in the classes DTA and ERA but without bounding the resources of the diagnoser. We also want to explore more precisely the links between control and diagnosability: even if we can reduce diagnosability to control, the result of [BDMP03] together with our 2EXPTIME-completeness result show that diagnosability is as difficult as control for some classes of diagnosers/controllers, which may appear intriguing.

Acknowledgment: We thank Thierry Cachat for his remarks on a draft of this paper.

References

[AD94] R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science (TCS)*, 126(2):183–235, 1994.

[AFH94] R. Alur, L. Fix, and T.A. Henzinger. A determinizable class of timed automata. In *Proc. 6th Int. Conf. on Computer Aided Verification (CAV'94)*, vol. 818 of *LNCS*, pp. 1–13. Springer, 1994.

[BDGP98] B. Bérard, V. Diekert, P. Gastin, and A. Petit. Characterization of the expressive power of silent transitions in timed automata. *Fundamenta Informaticae*, 36(2–3):145–182, 1998.

[BDMP03] P. Bouyer, D. D'Souza, P. Madhusudan, and A. Petit. Timed control with partial observability. In *Proc. 15th Int. Conf. Computer Aided Verification (CAV'2003)*, vol. 2725 of *LNCS*, pp. 180–192. Springer, 2003.

[Che04] F. Chevalier. Détection d'erreurs dans les systèmes temporisés. Master's thesis, DEA Algorithmique, Paris, 2004.

[DM02] D. D'Souza and P. Madhusudan. Timed control synthesis for external specifications. In *Proc. 19th Int. Symp. Theoretical Aspects of Computer Science (STACS'02)*, vol. 2285 of *LNCS*, pp. 571–582. Springer, 2002.

[DT04] D. D'Souza and N. Tabareau. On timed automata with input-determined guards. In *Proc. Joint Conf. Formal Modelling and Analysis of Timed Systems and Formal Techniques in Real-Time and Fault Tolerant System (FORMATS+FTRTFT'04)*, vol. 3253 of *LNCS*, pp. 68–83. Springer, 2004.

[GJL04] O. Grinchtein, B. Jonsson, and M. Leucker. Learning of event-recording automata. In *Proc. Joint Conf. Formal Modelling and Analysis of Timed Systems and Formal Techniques in Real-Time and Fault Tolerant System (FORMATS+FTRTFT'04)*, vol. 3253 of *LNCS*, pp. 379–395. Springer, 2004.

[GTW02] E. Grädel, W. Thomas, and T. Wilke, eds. *Automata, Logics, and Infinite Games: A Guide to Current Research*, vol. 2500 of *LNCS*. Springer, 2002.

[KPA03] K.J. Kristoffersen, C. Pedersen, and H.R. Andersen. Runtime verification of timed LTL using disjunctive normalized equation systems. In *Proc. 3rd Int. Work. Runtime Verification*, Electronic Notes in Computer Science. Elsevier, 2003.

[KT04] M. Krichen and S. Tripakis. Real-time testing with timed automata testers and coverage criteria. In *Proc. Joint Conf. Formal Modelling and Analysis of Timed Systems and Formal Techniques in Real-Time and Fault Tolerant System (FORMATS+FTRTFT'04)*, vol. 3253 of *LNCS*, pp. 134–151. Springer, 2004.

[LLW95] F. Laroussinie, K.G. Larsen, and C. Weise. From timed automata to logic – and back. In *Proc. 20th Int. Symp. Mathematical Foundations of Computer Science (MFCS'95)*, vol. 969 of *LNCS*, pp. 529–539. Springer, 1995.

[Pap94] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[SSL+95] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. C. Teneketzis. Diagnosability of discrete event systems. *IEEE Transactions on Automatic Control*, 40(9):1555–1575, 1995.

[SSL+96] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. C. Teneketzis. Failure diagnosis using discrete event systems. *IEEE Transactions on Control Systems Technology*, 4(2):105–124, 1996.

[Tri02] S. Tripakis. Fault diagnosis for timed automata. In *Proc. 7th Int. Symp. Formal Techniques in Real-Time and Fault Tolerant Systems (FTRTFT'02)*, vol. 2469 of *LNCS*, pp. 205–224. Springer, 2002.

[Tri03] S. Tripakis. Folk theorems on the determinization and minimization of timed automata. In *Proc. 1st Int. Work. on Formal Modeling and Analysis of Timed Systems (FORMATS'03)*, vol. 2791 of *LNCS*, pp. 182–188. Springer, 2003.