# Dynamic Observers for Fault Diagnosis of Timed Systems

Franck Cassez, *Member, IEEE*

***Abstract*— In this paper we extend the work on *dynamic observers* for fault diagnosis [1], [2], [3] to timed automata. We study sensor minimization problems with static observers and then address the problem of computing the most permissive dynamic observer for a system given by a timed automaton.**

## I. Introduction

Discrete-event systems [4] (DES) can be modelled by finite automata over an alphabet of actions/events $\Sigma$. The fault diagnosis problem [5] for DES consists in detecting *faulty* sequences in the system. A *faulty* sequence is a sequence of the DES containing an occurrence of a special event $f$. It is assumed that an external *observer* which has to detect faults, knows the specification/model of the DES, but can partially observe the system at runtime: it is able to observe sequences of *observable* events in $\Sigma_o \subseteq \Sigma$. Based on this knowledge, it has to announce whether an observation (in $\Sigma_o^*$) stems from a faulty sequence (in $(\Sigma \cup \{\tau, f\})^*$). Checking diagnosability of DES can be done in PTIME and computing a diagnoser amounts to determinizing the DES (EXPTIME) [5], [6], [7].

***Fault Diagnosis for Timed Automata.*** The fault diagnosis problem for Timed Automata (TA) has been introduced and solved by S. Tripakis in [8], where he proved that checking diagnosability of a timed automaton is PSPACE-complete. In the timed case however, the diagnoser may be a Turing machine. In a subsequent work by P. Bouyer and F. Chevalier [9], the problem of checking whether a timed automaton is diagnosable using a diagnoser which is a *deterministic* timed automaton (DTA) was studied, and they proved that this problem was 2EXPTIME-complete.

***Our Contribution and Related Work.*** In [1], [2] (and [3] for an extended version), we have introduced *dynamic observers* for fault diagnosis of DES. In this framework, an observer can choose dynamically which events it is going to observe and make a new choice after each occurrence of any (currently) observable event. In [1], [3] we have shown how to compute (2EXPTIME) a *most permissive observer* which represents all the the dynamic observers that ensures that a DES is diagnosable. In [2] we have furthermore introduced a notion of *cost* of an observer, and proved that an optimal observer could also be computed in 2EXPTIME.

In this paper, we extend the previous results for systems given by timed automata. Proofs are omitted and can be found in [10].

Franck Cassez is with National ICT Australia & CNRS, Locked Bag 6016, The University of New South Wales, Sydney NSW 1466, Australia. franck.cassez@cnrs.irccyn.fr,Franck.Cassez@nicta.com.au

Author supported by a Marie Curie International Outgoing Fellowship within the 7th European Community Framework Programme.

## II. Preliminaries

$\Sigma$ denotes a finite alphabet and $\Sigma_\tau = \Sigma \cup \{\tau\}$ where $\tau \notin \Sigma$ is the *unobservable* action. $\mathbb{B} = \{\text{TRUE}, \text{FALSE}\}$ is the set of boolean values, $\mathbb{N}$ the set of natural numbers, $\mathbb{Z}$ the set of integers and $\mathbb{Q}$ the set of rational numbers. $\mathbb{R}$ is the set of real numbers and $\mathbb{R}_{\geq 0}$ is the non-negative real numbers.

### A. Clock Constraints

Let $X$ be a finite set of variables called *clocks*. A *clock valuation* is a mapping $v : X \to \mathbb{R}_{\geq 0}$. We let $\mathbb{R}_{\geq 0}^X$ be the set of clock valuations over $X$. We let $\mathbf{0}_X$ be the *zero* valuation where all the clocks in $X$ are set to 0 (we use $\mathbf{0}$ when $X$ is clear from the context). Given $\delta \in \mathbb{R}$, $v + \delta$ denotes the valuation defined by $(v + \delta)(x) = v(x) + \delta$. We let $\mathcal{C}(X)$ be the set of *convex constraints* on $X$, i.e., the set of conjunctions of constraints of the form $x \bowtie c$ with $c \in \mathbb{Z}$ and $\bowtie \in \{\leq, <, =, >, \geq\}$. Given a constraint $g \in \mathcal{C}(X)$ and a valuation $v$, we write $v \models g$ if $g$ is satisfied by $v$. Given $R \subseteq X$ and a valuation $v$, $v[R]$ is the valuation defined by $v[R](x) = v(x)$ if $x \notin R$ and $v[R](x) = 0$ otherwise.

### B. Timed Words

The set of finite (resp. infinite) words over $\Sigma$ is $\Sigma^*$ (resp. $\Sigma^\omega$) and we let $\Sigma^\infty = \Sigma^* \cup \Sigma^\omega$. We let $\varepsilon$ be the empty word. A *language* $L$ is any subset of $\Sigma^\infty$. A finite (resp. infinite) *timed word* over $\Sigma$ is a word in $(\mathbb{R}_{\geq 0}.\Sigma)^*.\mathbb{R}_{\geq 0}$ (resp. $(\mathbb{R}_{\geq 0}.\Sigma)^\omega$). *Dur*$(w)$ is the duration of a timed word $w$ which is defined to be the sum of the durations (in $\mathbb{R}_{\geq 0}$) which appear in $w$; if this sum is infinite, the duration is $\infty$. Note that the duration of an infinite word can be finite, and such words which contain an infinite number of letters, are called *Zeno* words.

*TW*$^*(\Sigma)$ is the set of finite timed words over $\Sigma$, *TW*$^\omega(\Sigma)$, the set of infinite timed words and *TW*$(\Sigma) =$ *TW*$^*(\Sigma) \cup$ *TW*$^\omega(\Sigma)$. A *timed language* is any subset of *TW*$(\Sigma)$.

In this paper we write timed words as $0.4\ a\ 1.0\ b\ 2.7\ c \cdots$ where the real values are the durations elapsed between two letters: thus $c$ occurs at global time 4.1. We let *Unt*$(w)$ be the *untimed* version of $w$ obtained by erasing all the durations in $w$, e.g., *Unt*$(0.4\ a\ 1.0\ b\ 2.7\ c) = abc$. Given a timed language $L$, we let *Unt*$(L) = \{$*Unt*$(w) \mid w \in L\}$.

Let $\boldsymbol{\pi}_{/\Sigma'}$ be the projection of timed words of *TW*$(\Sigma)$ over timed words of *TW*$(\Sigma')$. When projecting a timed word $w$ on a sub-alphabet $\Sigma' \subseteq \Sigma$, the durations elapsed between two events are set accordingly: for instance $\boldsymbol{\pi}_{/\{a,c\}}(0.4\ a\ 1.0\ b\ 2.7\ c) = 0.4\ a\ 3.7\ c$ (projection erases some letters but keep the time elapsed between two letters). Given $\Sigma' \subseteq \Sigma$, $\boldsymbol{\pi}_{/\Sigma'}(L) = \{\boldsymbol{\pi}_{/\Sigma'}(w) \mid w \in L\}$.

### C. Timed Automata

Timed automata (TA) are finite automata extended with real-valued clocks to specify timing constraints between occurrences of events. For a detailed presentation of the fundamental results for timed automata, the reader is referred to the seminal paper of R. Alur and D. Dill [11].

*Definition 1 (Timed Automaton):* A *Timed Automaton* $A$ is a tuple $(L,\ l_0,\ X, \Sigma_\tau, E, \mathit{Inv}, F, R)$ where: $L$ is a finite set of *locations*; $l_0$ is the *initial location*; $X$ is a finite set of *clocks*; $\Sigma$ is a finite set of *actions*; $E \subseteq L \times \mathcal{C}(X) \times \Sigma_\tau \times 2^X \times L$ is a finite set of *transitions*; for $(\ell, g, a, r, \ell') \in E$, $g$ is the *guard*, $a$ the *action*, and $r$ the *reset* set; $\mathit{Inv} \in \mathcal{C}(X)^L$ associates with each location an *invariant*; as usual we require the invariants to be conjunctions of constraints of the form $x \preceq c$ with $\preceq \in \{<, \leq\}$. $F \subseteq L$ and $R \subseteq L$ are respectively the *final* and *repeated* sets of locations. ■

A *state* of $A$ is a pair $(\ell, v) \in L \times \mathbb{R}^X_{\geq 0}$. A *run* $\varrho$ of $A$ from $(\ell_0, v_0)$ is a (finite or infinite) sequence of alternating *delay* and *discrete* moves:

$$\begin{aligned}\varrho \quad = \quad & (\ell_0, v_0) \xrightarrow{\delta_0} (\ell_0, v_0 + \delta_0) \xrightarrow{a_0} (\ell_1, v_1) \cdots \\ & \cdots \xrightarrow{a_{n-1}} (\ell_n, v_n) \xrightarrow{\delta_n} (\ell_n, v_n + \delta_n) \cdots\end{aligned}$$

s.t. for every $i \geq 0$:

- $v_i + \delta \models \mathit{Inv}(\ell_i)$ for $0 \leq \delta \leq \delta_i$;
- there is some transition $(\ell_i, g_i, a_i, r_i, \ell_{i+1}) \in E$ s.t. : $(i)$ $v_i + \delta_i \models g_i$ and $(ii)$ $v_{i+1} = (v_i + \delta_i)[r_i]$.

The set of finite (resp. infinite) runs from a state $s$ is denoted $\mathit{Runs}^*(s, A)$ (resp. $\mathit{Runs}^\omega(s, A)$) and we define $\mathit{Runs}^*(A) = \mathit{Runs}^*((l_0, \mathbf{0}), A)$, $\mathit{Runs}^\omega(A) = \mathit{Runs}^\omega((l_0, \mathbf{0}), A)$ and finally $\mathit{Runs}(A) = \mathit{Runs}^*(A) \cup \mathit{Runs}^\omega(A)$. If $\varrho$ is finite and ends in $s_n$, we let $\mathit{last}(\varrho) = s_n$. Because of the denseness of the time domain, the transition graph of $A$ is infinite (uncountable number of states and delay edges). The *trace*, $\mathit{tr}(\varrho)$, of a run $\varrho$ is the timed word $\pi_{/\Sigma}(\delta_0 a_0 \delta_1 a_1 \cdots a_n \delta_n \cdots)$. We let $\mathit{Dur}(\varrho) = \mathit{Dur}(\mathit{tr}(\varrho))$. For $V \subseteq \mathit{Runs}(A)$, we let $\mathit{Tr}(V) = \{\mathit{tr}(\varrho) \mid \varrho \in V\}$.

A finite (resp. infinite) timed word $w$ is *accepted* by $A$ if it is the trace of a run of $A$ that ends in an $F$-location (resp. a run that reaches infinitely often an $R$-location). $\mathcal{L}^*(A)$ (resp. $\mathcal{L}^\omega(A)$) is the set of traces of finite (resp. infinite) timed words accepted by $A$, and $\mathcal{L}(A) = \mathcal{L}^*(A) \cup \mathcal{L}^\omega(A)$ is the set of timed words accepted by $A$. In the sequel we often omit the sets $R$ and $F$ in TA and this implicitly means $F = L$ and $R = \varnothing$.

A timed automaton $A$ is *deterministic* if there is no $\tau$ labelled transition in $A$, and if, whenever $(\ell, g, a, r, \ell')$ and $(\ell, g', a, r', \ell'')$ are transitions of $A$, $g \wedge g' \equiv \text{FALSE}$. $A$ is *complete* if from each state $(\ell, v)$, and for each action $a$, there is a transition $(\ell, g, a, r, \ell')$ such that $v \models g$. We note DTA the class of deterministic timed automata.

### D. Region Graph of a TA

The *region graph* $\mathit{RG}(A)$ of a TA $A$ is a finite quotient of the infinite graph of $A$ which is time-abstract bisimilar to $A$ [11]. It is a finite automaton (FA) on the alphabet $E' = E \cup \{\tau\}$. The states of $\mathit{RG}(A)$ are pairs $(\ell, r)$ where $\ell \in L$ is a location of $A$ and $r$ is a *region* of $\mathbb{R}^X_{\geq 0}$. More generally, the edges of the graph are tuples $(s, t, s')$ where $s, s'$ are states of $\mathit{RG}(A)$ and $t \in E'$. Genuine unobservable moves of $A$ labelled $\tau$ are labelled by tuples of the form $(s, (g, \tau, r), s')$ in $\mathit{RG}(A)$. An edge $(g, \lambda, R)$ in the region graph corresponds to a discrete transition of $A$ with guard $g$, action $\lambda$ and reset set $R$. A $\tau$ move in $\mathit{RG}(A)$ stands for a delay move to the time-successor region. The initial state of $\mathit{RG}(A)$ is $(l_0, \mathbf{0})$. A final (resp. repeated) state of $\mathit{RG}(A)$ is a state $(\ell, r)$ with $\ell \in F$ (resp. $\ell \in R$). A fundamental property of the region graph [11] is:

*Theorem 1 ([11]):* $\mathcal{L}(\mathit{RG}(A)) = \mathit{Unt}(\mathcal{L}(A))$.

The (maximum) size of the region graph is exponential in the number of clocks and in the maximum constant of the automaton $A$ (see [11]): $|\mathit{RG}(A)| = |L| \cdot |X|! \cdot 2^{|X|} \cdot K^{|X|}$ where $K$ is the largest constant used in $A$.

### E. Product of TA

*Definition 2 (Product of two TA):* Let $A_i = (L_i, l_0^i, X_i, \Sigma_\tau^i, E_i, \mathit{Inv}_i)$ for $i \in \{1, 2\}$, be two TA s.t. $X_1 \cap X_2 = \varnothing$. The *product* of $A_1$ and $A_2$ is the TA $A_1 \times A_2 = (L, l_0, X, \Sigma_\tau, E, \mathit{Inv})$ given by: $L = L_1 \times L_2$; $l_0 = (l_0^1, l_0^2)$; $\Sigma = \Sigma^1 \cup \Sigma^2$; $X = X_1 \cup X_2$; and $E \subseteq L \times \mathcal{C}(X) \times \Sigma_\tau \times 2^X \times L$ and $((\ell_1, \ell_2), g_{1,2}, \sigma, r, (\ell_1', \ell_2')) \in E$ if:

- either $\sigma \in (\Sigma_1 \cap \Sigma_2) \setminus \{\tau\}$, and $(i)$ $(\ell_k, g_k, \sigma, r_k, \ell_k') \in E_k$ for $k = 1$ and $k = 2$; $(ii)$ $g_{1,2} = g_1 \wedge g_2$ and $(iii)$ $r = r_1 \cup r_2$;
- or for $k = 1$ or $k = 2$, $\sigma \in (\Sigma_k \setminus \Sigma_{3-k}) \cup \{\tau\}$, and $(i)$ $(\ell_k, g_k, \sigma, r_k, \ell_k') \in E_k$; $(ii)$ $g_{1,2} = g_k$ and $(iii)$ $r = r_k$;

and finally $\mathit{Inv}(\ell_1, \ell_2) = \mathit{Inv}(\ell_1) \wedge \mathit{Inv}(\ell_2)$. ■

## III. Fault Diagnosis Problems & Known Results

### A. The Model

To model timed systems with faults, we use timed automata on the alphabet $\Sigma_{\tau,f} = \Sigma_\tau \cup \{f\}$ where $f$ is the *faulty* (and unobservable) event. We only consider one type of fault, but the results we give are valid for many types of faults $\{f_1, f_2, \cdots, f_n\}$: indeed solving the many types diagnosability problem amounts to solving $n$ one type diagnosability problems [7]. The observable events are given by $\Sigma_o \subseteq \Sigma$ and $\tau$ is always unobservable.

The system we want to supervise is given by a TA $A = (L, l_0, X, \Sigma_{\tau,f}, E, \mathit{Inv})$. Fig. 1 gives an example of such a system. Invariants in the automaton $\mathcal{A}$ are written within square brackets as in $[x \leq 3]$.

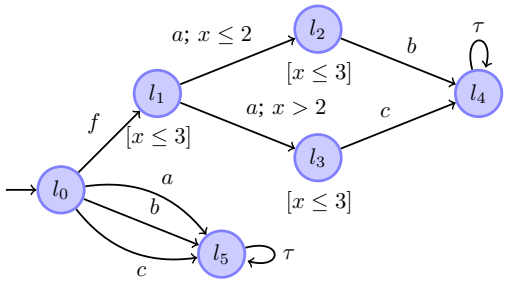



Figure 1. The Timed Automaton $\mathcal{A}$

Let $\Delta \in \mathbb{N}$. A run of $A$

$$\begin{aligned}\varrho \;=\; & (\ell_0, v_0) \xrightarrow{\delta_0} (\ell_0, v_0+\delta_0) \xrightarrow{a_0} (\ell_1, v_1)\cdots \\ & \cdots \xrightarrow{a_{n-1}} (\ell_n, v_n) \xrightarrow{\delta_n} (\ell_n, v_n+\delta)\cdots\end{aligned}$$

is $\Delta$-faulty if: (1) there is an index $i$ s.t. $a_i = f$ and (2) the duration of the run $\varrho' = (\ell_i, v_i) \xrightarrow{\delta_i} \cdots \xrightarrow{\delta_n} (\ell_n, v_n + \delta_n)\cdots$ is larger than $\Delta$. We let $\mathit{Faulty}_{\geq\Delta}(A)$ be the set of $\Delta$-faulty runs of $A$. Note that by definition, if $\Delta' \geq \Delta$ then $\mathit{Faulty}_{\geq\Delta'}(A) \subseteq \mathit{Faulty}_{\geq\Delta}(A)$. We let $\mathit{Faulty}(A) = \cup_{\Delta\geq 0}\mathit{Faulty}_{\geq\Delta}(A) = \mathit{Faulty}_{\geq 0}(A)$ be the set of faulty runs of $A$, and $\mathit{NonFaulty}(A) = \mathit{Runs}(A) \setminus \mathit{Faulty}(A)$ be the set of non-faulty runs of $A$. Moreover we let $\mathit{Faulty}^{tr}_{\geq\Delta}(A) = \mathit{Tr}(\mathit{Faulty}_{\geq\Delta}(A))$ and $\mathit{NonFaulty}^{tr}(A) = \mathit{Tr}(\mathit{NonFaulty}(A))$ which are the traces[1] of $\Delta$-faulty and non-faulty runs of $A$.

### B. Diagnosers

The purpose of fault diagnosis is to detect a fault as soon as possible. Faults are unobservable and only the events in $\Sigma_o$ can be observed as well as the time elapsed between these events. Whenever the system generates a timed word $w$, the observer can only see $\boldsymbol{\pi}_{/\Sigma_o}(w)$. If an observer can detect faults in this way it is called a *diagnoser*. A diagnoser must detect a fault within a given delay $\Delta \in \mathbb{N}$.

*Definition 3 ($(\Sigma_o, \Delta)$-Diagnoser):* Let $A$ be a TA over the alphabet $\Sigma_{\tau,f}$, $\Sigma_o \subseteq \Sigma$ and $\Delta \in \mathbb{N}$. A $(\Sigma_o, \Delta)$-*diagnoser* for $A$ is a mapping $D : \mathit{TW}^*(\Sigma_o) \to \{0, 1\}$ such that:

- for each $\varrho \in \mathit{NonFaulty}(A)$, $D(\boldsymbol{\pi}_{/\Sigma_o}(\varrho)) = 0$,
- for each $\varrho \in \mathit{Faulty}_{\geq\Delta}(A)$, $D(\boldsymbol{\pi}_{/\Sigma_o}(\varrho)) = 1$. ■

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

*Example 1:* The TA $\mathcal{A}$ in Fig. 1 with $\Sigma = \Sigma_o = \{a, b, c\}$ is $(\Sigma, 3)$-diagnosable. If $\Sigma_o = \{b\}$, it is not. □

### C. Classical Diagnosis Problems

Assume $A = (L, \ell_0, X, \Sigma_{\tau,f}, E, \mathit{Inv})$ is a TA . The classical fault diagnosis problems are the following:

*Problem 1 (Bounded or $\Delta$-Diagnosability):*
INPUTS: A TA $A$, $\Sigma_o \subseteq \Sigma$, and $\Delta \in \mathbb{N}$.
PROBLEM: Is $A$ $(\Sigma_o, \Delta)$-diagnosable?

*Problem 2 (Diagnosability):*
INPUTS: A TA $A$ and $\Sigma_o \subseteq \Sigma$.
PROBLEM: Is $A$ $\Sigma_o$-diagnosable?

*Problem 3 (Maximum delay):*
INPUTS: A TA $A$ and and $\Sigma_o \subseteq \Sigma$.
PROBLEM: If $A$ is $\Sigma_o$-diagnosable, what is the minimum $\Delta$ s.t. $A$ is $(\Sigma_o, \Delta)$-diagnosable ?

According to Definition 3, $A$ is $\Sigma_o$-diagnosable, iff, there is some $\Delta \in \mathbb{N}$ s.t. $A$ is $(\Sigma_o, \Delta)$-diagnosable. Thus $A$ is not $\Sigma_o$-diagnosable iff $\forall\Delta \in \mathbb{N}$, $A$ is not $(\Sigma_o, \Delta)$-diagnosable. Moreover a trace based definition of $(\Sigma_o, \Delta)$-diagnosability can be stated as[2]: $A$ is $(\Sigma_o, \Delta)$-diagnosable iff

$$\boldsymbol{\pi}_{/\Sigma_o}(\mathit{Faulty}^{tr}_{\geq\Delta}(A)) \cap \boldsymbol{\pi}_{/\Sigma_o}(\mathit{NonFaulty}^{tr}(A)) = \varnothing. \quad (1)$$

[1]Notice that $tr(\varrho)$ erases $\tau$ and $f$.

[2]This definition does not take into account *Zeno* runs; this is not difficult to add and the reader is referred to [12] for more details.

This gives a necessary and sufficient condition for non $\Sigma_o$-diagnosability:

$$A \text{ is not } \Sigma_o\text{-diagnosable} \Longleftrightarrow \begin{cases} \forall\Delta \in \mathbb{N}, \\ \quad \exists\rho \in \mathit{NonFaulty}(A) \\ \quad \exists\rho' \in \mathit{Faulty}_{\geq\Delta}(A) \text{ s.t.} \\ \quad\quad \boldsymbol{\pi}_{/\Sigma_o}(\rho) = \boldsymbol{\pi}_{/\Sigma_o}(\rho'), \end{cases} \quad (2)$$

or in other words, $A$ is $\Sigma_o$-diagnosable iff there is no pair of runs $(\rho_1, \rho_2)$ with $\rho_1 \in \mathit{Faulty}_{\geq\Delta}(A)$, $\rho_2 \in \mathit{NonFaulty}(A)$ the $\Sigma_o$-traces of which are equal.

Complexity results for the diagnosis problems on timed automata were established in [8] (see [12] for a comprehensive study) and Problems 1–3 are PSPACE-complete (note that PSPACE-completeness already holds for $\Sigma_o = \Sigma$).

## IV. Sensor Minimization with Static Observers

In this section, we extend the results of [1] to systems given by TA.

*Problem 4 (Minimum Cardinality Set):*
INPUTS: A TA $A = (L, \ell_0, X, \Sigma_{\tau,f}, E, \mathit{Inv})$ and $n \in \mathbb{N}$.
PROBLEM:

(A) Is there any set $\Sigma_o \subseteq \Sigma$, with $|\Sigma_o| = n$ s.t. $A$ is $\Sigma_o$-diagnosable ?
(B) If the answer to (A) is "yes", compute the minimum value for $n$.

*Theorem 2:* Problem 4 is PSPACE-complete.

The previous results also hold in a more general setting using *masks* (see the extended version [10]).

## V. Sensor Minimization with Dynamic Observers

The use of *dynamic observers* was already advocated for DES in [1], [3]. We start with an example that shows that dynamically choosing what to observe can be even more efficient using timing information.

*Example 2:* Let $\mathcal{A}$ be the automaton of Figure 1. To diagnose $\mathcal{A}$, we can use a *dynamic observer* that switches $a$, $b$ and $c$-sensors on/off. If we do not measure time, to be able to detect faults in $\mathcal{A}$, we have to switch the $a$ sensor on at the beginning. When an $a$ has occurred, we must be ready for either an $b$ or a $c$ and therefore, switch on the $b$ and $c$ sensors on. A dynamic observer must thus first observe $\{a\}$ and after an occurrence of $a$, observe $\{b, c\}$.

If the observer can measure time using a clock, say $y$, it can first switch the $a$ sensor on. If an $a$ occurs when $y \leq 2$, then switch the $b$ sensor on and if $y > 2$ switch the $c$ sensor on. This way the observer never has to observe more than event at each point in time. □

### A. Dynamic Observers

The choice of the events to observe can depend on the choices the observer has made before and on the observations (event, time-stamp) it has made. Moreover an observer may have *unbounded* memory. The following definition extends the notion of observers introduced in [1] to the timed setting.

*Definition 4 (Observer):* An *observer Obs* over $\Sigma$ is a *deterministic and complete* timed automaton $\mathit{Obs} = (N, n_0, Y,$

$\Sigma, \delta, \mathit{Inv}_{\text{TRUE}})$ together with a mapping $O : N \to 2^{\Sigma}$, where $N$ is a (possibly infinite) set of locations, $n_0 \in N$ is the initial location, $\Sigma$ is the set of observable events, $\delta : N \times \Sigma \times \mathcal{C}(Y) \to N \times 2^Y$ is the transition function (a total function), and $O$ is a labeling function that specifies the set of events that the observer wishes to observe when it is at location $n$. The invariant[3] $\mathit{Inv}_{\text{TRUE}}$ maps every location to TRUE, implying that an observer cannot prevent time from elapsing. We require that, for any location $n$ and any $a \in \Sigma$, if $a \notin O(n)$ then $\delta(n, a, \cdot) = (n, \varnothing)$: this means the observer does not change its location nor resets its clocks when an event it has chosen not to observe occurs. ■

As an observer is deterministic we let $\delta(n_0, w)$ denote the state $(n, v)$ reached after reading the timed word $w$ and $O(\delta(n_0, w))$ is the set of events *Obs* observes after $w$.

An observer defines a *transducer* which is a mapping $[\![\mathit{Obs}]\!] : \mathit{TW}^*(\Sigma) \to \mathit{TW}^*(\Sigma)$. Given a word $w$, $[\![\mathit{Obs}]\!](w)$ is the output of the transducer on $w$. It is called the *observation* of $w$ by the observer *Obs*.

### B. Diagnosability with Dynamic Observers

*Definition 5 ($(\mathit{Obs}, \Delta)$-diagnoser):* Let $A$ be a TA over $\Sigma_{\tau,f}$ and *Obs* be an observer over $\Sigma$. $D : \mathit{TW}^*(\Sigma) \to \{0, 1\}$ is an $(\mathit{Obs}, \Delta)$-*diagnoser* for $A$ if:

- $\forall \rho \in \mathit{NonFaulty}(A)$, $D([\![\mathit{Obs}]\!](tr(\rho))) = 0$ and
- $\forall \rho \in \mathit{Faulty}_{\geq \Delta}(A)$, $D([\![\mathit{Obs}]\!](tr(\rho))) = 1$. ■

$A$ is $(\mathit{Obs}, \Delta)$-diagnosable if there is an $(\mathit{Obs}, \Delta)$-diagnoser for $A$. $A$ is *Obs*-diagnosable if there is some $\Delta$ such that $A$ is $(\mathit{Obs}, \Delta)$-diagnosable.

We now show how to check *Obs*-diagnosability when the observer *Obs* is a DTA.

*Problem 5 (Deterministic Timed Automata Observers):*
INPUTS: A TA $A = (L, \ell_0, X, \Sigma_{\tau,f}, E, \mathit{Inv})$ and an observer given by a DTA $\mathit{Obs} = (N, n_0, Y, \Sigma, \delta, O)$.
PROBLEM:

(A) Is $A$ *Obs*-diagnosable?
(B) If the answer to (A) is "yes", compute the minimum $\Delta \in \mathbb{N}$ s.t. $A$ is $(\mathit{Obs}, \Delta)$-diagnosable.

*Theorem 3:* Problem 5 is PSPACE-complete.

### C. Synthesis of the Most Permissive Dynamic Diagnoser

In this section we address the problem of *synthesizing* a DTA dynamic observer which ensures diagnosability. Following [3], we want to compute a *most permissive* observer ($\varnothing$ if none exists), which gives a representation of all the good observers. Indeed, checking whether there exists a DTA observer *Obs* s.t. $A$ is *Obs*-diagnosable is not an interesting problem: it suffices to check that $A$ is $\Sigma$-diagnosable as the DTA observer which observes $\Sigma$ continuously will be a solution.

When synthesizing (deterministic) timed automata, an important issue is the amount of *resources* the timed automaton can use: this can be formally defined [13] by the (number of) clocks, $Z$, that the automaton can use, the maximal constant max, and a *granularity* $\frac{1}{m}$. As an example, a TA of resource $\mu = (\{c, d\}, 2, \frac{1}{3})$ can use two clocks, $c$ and $d$, and the clocks constraints using the rationals $-2 \leq k/m \leq 2$ where $k \in \mathbb{Z}$ and $m = 3$. A *resource* $\mu$ is thus a triple $\mu = (Z, \max, \frac{1}{m})$ where $Z$ is finite set of clocks, $\max \in \mathbb{N}$ and $\frac{1}{m} \in \mathbb{Q}_{>0}$ is the *granularity*. $\text{DTA}_\mu$ is the class of DTA of resource $\mu$.

*Remark 1:* Notice that the number of locations of the DTA in $\text{DTA}_\mu$ is not bounded and hence this family has an infinite (yet countable) number of elements.

We now focus on the following problem :

*Problem 6 (Most Permissive Dynamic $\Delta$-Diagnoser):*
INPUTS: A TA $A = (L, \ell_0, X, \Sigma_{\tau,f}, E, \mathit{Inv})$, $\Delta \in \mathbb{N}$, and a resource $\mu = (Z, \max, \frac{1}{m})$.
PROBLEM: Compute the set $O$ of all observers in $\text{DTA}_\mu$, s.t. $A$ is $(\mathit{Obs}, \Delta)$-diagnosable iff $\mathit{Obs} \in O$.

For DES, the previous problem can be solved by computing a most permissive observer, and we refer to [3] section 5.5 for the formal definition of the most permissive observer. This can be done in 2EXPTIME [3], and the solution is a reduction to a safety control problem under partial observation. For the timed case, we cannot use the same solution as controller synthesis under partial observation is undecidable [13]. The solution we present for Problem 6 is a modification of an algorithm originally introduced in [9].

### D. Fault Diagnosis with DTA [9]

In case a TA $A$ is $\Sigma_o$-diagnosable, the diagnoser is a mapping [8] which performs a state estimate of $A$ after a timed word $w$ is read by $A$. For DES, it is obtained by *determinizing* the system, but we cannot always determinize a TA $A$ (see [11]). And unfortunately testing whether a timed automaton is determinizable is undecidable [14], [15].

P. Bouyer and F. Chevalier in [9] considers the problem of deciding whether there exists a diagnoser which is a DTA using resources in $\mu$:

*Problem 7 ($\text{DTA}_\mu$ $\Delta$-Diagnoser [9]):*
INPUTS: A TA $A = (L, \ell_0, X, \Sigma_{\tau,f}, E, \mathit{Inv})$, $\Delta \in \mathbb{N}$, and a resource $\mu = (Z, \max, \frac{1}{m})$.
PROBLEM: Is there any $D \in \text{DTA}_\mu$ s.t. $A$ is $(D, \Delta)$-diagnosable ?

*Theorem 4 ([9]):* Problem 7 is 2EXPTIME-complete.

The solution to the previous problem is based on the construction of a *two-player game*, the solution of which gives the *set* of all $\text{DTA}_\mu$ diagnosers (the most permissive diagnosers) which can diagnose $A$ (or $\varnothing$ is there is none).

We recall here the construction of the two-player game.

Let $A = (L, \ell_0, X, \Sigma_{\tau,f}, \to, \mathit{Inv})$ be a TA, $\Sigma_o \subseteq \Sigma$. Define $A(\Delta) = (L_1 \cup L_2 \cup L_3, \ell_0^1, X \cup \{z\}, \Sigma_{\tau,f}, \to_\Delta, \mathit{Inv}_\Delta)$ as follows:

- $L_i = \{\ell^i, \ell \in L\}$, for $i \in \{1, 2, 3\}$, i.e., $L_i$ elements are copies of the locations in $L$,
- $z$ is (new) clock not in $X$,
- for $\ell \in L$, $\mathit{Inv}(\ell^1) = \mathit{Inv}(\ell)$, $\mathit{Inv}(\ell^2) = \mathit{Inv}(\ell) \wedge z \leq \Delta$, and $\mathit{Inv}(\ell^3) = \text{TRUE}$,
- the transition relation is given by:
  - for $i \in \{1, 2, 3\}$, $\ell^i \xrightarrow{(g,a,R)}_\Delta \ell'^i$ if $a \neq f$ and $\ell \xrightarrow{(g,a,R)} \ell'$,

[3] In the sequel, we omit the invariant when a TA is an observer, and replace it by the mapping $O$.

– for $i \in \{2,3\}$, $\ell^i \xrightarrow{(g,f,R)}_\Delta \ell'^i$ if $a \neq f$ and $\ell \xrightarrow{(g,f,R)} \ell'$,
– $\ell^1 \xrightarrow{(g,f,R\cup\{z\})}_\Delta \ell'^2$ if $a \neq f$ and $\ell \xrightarrow{(g,f,R)} \ell'$,
– $\ell^2 \xrightarrow{(z=\Delta,\tau,\varnothing)}_\Delta \ell^3$.

The previous construction creates 3 copies of $A$: the system starts in copy 1, when a fault occurs it switches to copy 2, resetting the clock $z$, and when in copy 2 (a fault has occurred) it can switch to copy 3 after $\Delta$ time units. We can then define $L_1$ as the non-faulty locations, and $L_3$ as the $\Delta$-faulty locations.

Given a resource $\mu = (Y, \max, \frac{1}{m})$ ($X \cap Y = \varnothing$), a *minimal guard* for $\mu$ is a guard which defines a region of granularity $\mu$. We define the (symbolic) *universal automaton* $\mathcal{U} = (\{0\}, \{0\}, Y, \Sigma, E_\mu, \mathit{Inv}_\mu)$ by:

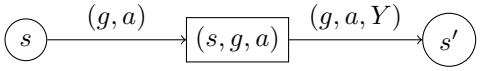
- $\mathit{Inv}_\mu(0) = \text{TRUE}$,
- $(0, g, a, R, 0) \in E_\mu$ for each $(g, a, R)$ s.t. $a \in \Sigma$, $R \subseteq Y$, and $g$ is a minimal guard for $\mu$.

$\mathcal{U}$ is finite because $E_\mu$ is finite. Nevertheless $\mathcal{U}$ is not deterministic because it can choose to reset different sets of clocks $Y$ for a pair "(guard, letter)" $(g, a)$. To diagnose $A$, we have to find when a set of clocks has to be reset. This can provide enough information to distinguish $\Delta$-faulty words from non-faulty words.

The algorithm of [9] requires the following steps:

1) define the region graph $\mathit{RG}(A(\Delta) \times \mathcal{U})$,
2) compute a *projection* of this region graph:
   - let $(g, a, R)$ be a label of an edge in $\mathit{RG}(A(\Delta)\times\mathcal{U})$,
   - let $g'$ be the unique minimal guard s.t. $[\![g]\!] \subseteq [\![g']\!]$;
   - define the projection $p_\mathcal{U}(g, a, R)$ by $(g', \lambda, R \cap Y)$ with $\lambda = a$ if $a \in \Sigma_o$ and $p_\mathcal{U}(g, a, R) = \tau$ otherwise.

   The projected automaton $p_\mathcal{U}(\mathit{RG}(A(\Delta) \times \mathcal{U}))$ is the automaton $\mathit{RG}(A(\Delta) \times \mathcal{U})$ where each label $\alpha$ is replaced by $p_\mathcal{U}(\alpha)$.
3) determinize $p_\mathcal{U}(\mathit{RG}(A(\Delta) \times \mathcal{U}))$ (removing $\tau$ actions) and obtain $H_{A,\Delta,\mu}$,
4) build a two-player safety game $G_{A,\Delta,\mu}$ as follows:
   - each transition $s \xrightarrow{(g,a,Y)} s'$ in $H_{A,\Delta,\mu}$ yields a transition in $G_{A,\Delta,\mu}$ of the form:




   - the round-shaped state are the states of Player 1, whereas the square-shaped states are Player 0 states (the choice of the clocks to reset).
   - the Bad states (for Player 0) are the states of the form $\{(\ell_1, r_1), (\ell_2, r_2), \cdots, (\ell_k, r_k)\}$ with both a $\Delta$-faulty (in $L_3$) and a non-faulty (in $L_1$) location.

The main results of [9] are:

- there is a TA $D \in \text{DTA}_\mu$ s.t. $A$ is $(D, \Delta)$-diagnosable iff Player 0 can win the safety game "avoid Bad" $G_{A,\Delta,\mu}$,
- it follows that Problem 7 can be solved in 2EXPTIME as $G_{A,\Delta,\mu}$ has size doubly exponential in $A$, $\Delta$ and $\mu$,
- the acceptance problem for Alternating Turing machines of exponential space can be reduced to Problem 7 and thus it is 2EXPTIME-hard.

### E. Problem 6 is in 2EXPTIME

We now show how to modify the previous algorithm to solve Problem 6, and obtain the following result:

*Theorem 5:* Problem 6 can be solved in 2EXPTIME.

*Remark 2:* In [9] it is also proved that for Event Recording Automata (ERA) [16] Problem 7 becomes PSPACE-complete. This result does not carry over in our case, as there is still an exponential step with the choice of the sets of events to be observed.

## VI. Optimal Dynamic Observers

In this section we extend the notion of *cost* defined for finite state observers in [3] to the case of timed observers.

### A. Weighted/Priced Timed Automata

Weighted/priced timed automata were introduced in [17], [18] and they extend TA with *prices/costs/weights* on the time elapsing and discrete transitions.

*Definition 6 (Priced Timed Automata):* A *priced timed automaton (PTA)* is a pair $(A, \mathit{Cost})$ where $A = (L, \ell_0, X, \Sigma_{\tau,f}, E, \mathit{Inv})$ is a timed automaton and *Cost* is a *cost function* which is a mapping from $L \cup E$ to $\mathbb{N}$. ■

Let

$$\begin{aligned}\varrho \;=\; & (\ell_0, v_0) \xrightarrow{\delta_0} (\ell_0, v_0 + \delta_0) \xrightarrow{a_0} (\ell_1, v_1) \cdots \\ & \cdots \xrightarrow{a_{n-1}} (\ell_n, v_n) \xrightarrow{\delta_n} (\ell_n, v_n + \delta_n)\end{aligned}$$

be a run of $A$. We denote by $e_i = (\ell_i, (g_i, a_i, R_i), \ell_{i+1})$ the discrete transition taken from $(\ell_i, v_i + \delta_i)$ to $(\ell_{i+1}, v_{i+1})$.

The *cost* of the run $\varrho$ is defined by:

$$\mathit{Cost}(\varrho) = \Sigma_{i\in 0..n}\mathit{Cost}(\ell_i) \cdot \delta_i + \Sigma_{i\in 0..n-1}\mathit{Cost}(e_i).$$

The *mean cost* of $\varrho$ is defined to be the cost per time unit and given[4] by $\overline{\mathit{Cost}}(\varrho) = \mathit{Cost}(\varrho)/\mathit{Dur}(\varrho)$. The cost of runs of duration $t \in \mathbb{R}_{>0}$ is defined by $\overline{\mathit{Cost}}(t) = \sup\{\overline{\mathit{Cost}}([\![\mathit{Obs}]\!](\varrho)) \mid \mathit{Dur}(\varrho) = t\}$. The *maximal mean cost* of $(A, \mathit{Cost})$ is $\overline{\mathit{Cost}}(A) = \limsup_{t\to\infty} \overline{\mathit{Cost}}(t)$. The minimal mean cost is defined dually and denoted $\underline{\mathit{Cost}}(A)$.

### B. Cost of an Observer

To select a best or optimal dynamic observer which ensures $\Delta$-diagnosability, we need to define a metric to compare them. We extend the one defined in [3] for DES to take into account (real) time elapsing.

Let $A$ be a TA and *Obs* a DTA observer. *Obs* is extended into a P(D)TA by associating costs with locations and transitions. The cost associated with the discrete transitions is the cost of switching on the sensors for a set of observable events, and the cost of a location is the cost per time unit of having a set of sensors activated.

Let $\varrho$ be a run of $A$. As *Obs* is deterministic (and complete) there is exactly one run of *Obs* the trace of which is $[\![\mathit{Obs}]\!](tr(\varrho))$. Given $\varrho$, let $[\![\mathit{Obs}]\!](\varrho)$ be this unique run. The average cost of the run $\varrho$ observed by *Obs* is $\overline{\mathit{Cost}}([\![\mathit{Obs}]\!](\varrho))$.

[4] Runs of duration 0 are not taken into account.

Given $t \in \mathbb{R}_{>0}$, the *maximal mean cost* of runs of duration $t$ is defined by:

$$\overline{Cost}(A, Obs, t) = \sup_{\varrho \in Runs^*(A) \wedge Dur(\varrho)=t} \{\overline{Cost}([\![Obs]\!](\varrho))\}.$$

The *maximal average cost* of the pair $< A, Obs >$ is defined

$$\overline{Cost}(< A, Obs >) = \limsup_{t \to \infty} \overline{Cost}(A, Obs, t).$$

We can then state the following problem:

*Problem 8 (Cost of an Observer):*
INPUTS: A TA $A$ and $(Obs, Cost)$ a PDTA observer.
PROBLEM: Compute $\overline{Cost}(< A, Obs >)$.

### C. Computing the Cost of a Given Timed Observer

The computation of optimal infinite schedules for TA has been addressed in [19]. The main result of [19] is:

*Theorem 6 (Minimal/Maximal Mean Cost [19]):* Given a PTA $A$, computing $\overline{Cost}$ and $\underline{Cost}$ is PSPACE-complete.

The definition of the cost of an observer is exactly the definition of the maximal mean cost in [19] and thus:

*Theorem 7:* Problem 8 is PSPACE-complete.

### D. Optimal Synthesis Problem

Checking whether the mean cost of a given observer is less than $k$ requires that we have computed or are given such an observer. A more difficult version of Problem 8 is to check for the existence of cheap dynamic observer:

*Problem 9 (Bounded Cost Dynamic Observer):*
INPUTS: A TA $A = (L, \ell_0, X, \Sigma_{\tau,f}, E, Inv)$, $\Delta \in \mathbb{N}$, $\mu$ a resource and $k \in \mathbb{N}$.
PROBLEM:

(A) Is there a dynamic observer $D \in \text{DTA}_\mu$ s.t. $A$ is $(D, \Delta)$-diagnosable and $\overline{Cost}(< A, D >) \leq k$ ?
(B) If the answer to (A) is "yes", compute a witness dynamic observer?

We cannot provide of proof that Problem 9 is decidable. However, we give a lower bound for Problem 9 and later discuss the exact complexity.

*Theorem 8:* Problem 9 is 2EXPTIME-hard.

## VII. CONCLUSION

The results of the paper are summarized by the line "TA" in Table I below.

TABLE I
SUMMARY OF THE RESULTS

| | Static Observers | Dynamic Observers | |
|---|---|---|---|
| | Min. Cardinality | Most Perm. Obs. | Optimal Observer |
| DES | NP-Complete [1] | 2EXPTIME [1] | 2EXPTIME [2] |
| TA | PSPACE-Complete | 2EXPTIME | 2EXPTIME-hard |

The complexity/decidability status of Problem 9 is left open. A solution to this problem would be to solve the following optimization problem on the class of S-PTGA:

*Problem 10 (Optimal Infinite Schedule in S-PTGA):*
INPUTS: A S-PTGA $(A, Cost)$, a set of *Bad* states and $k \in \mathbb{N}$.
PROBLEM: Is there a strategy $f$ for Player 1 in $A$ s.t. $f(A)$ ($A$ controlled by $f$) avoids *Bad* and satisfies $\overline{Cost}(f(A)) \leq k$?

## REFERENCES


[1] F. Cassez, S. Tripakis, and K. Altisen, "Sensor minimization problems with static or dynamic observers for fault diagnosis," in *7th Int. Conf. on Application of Concurrency to System Design (ACSD'07)*. IEEE Computer Society, 2007, pp. 90–99.

[2] ——, "Synthesis of optimal dynamic observers for fault diagnosis of discrete-event systems," in *Proceedings of the 1st IEEE & IFIP International Symposium on Theoretical Aspects of Software Engineering (TASE'07)*. IEEE Computer Society, 2007, pp. 316–325.

[3] F. Cassez and S. Tripakis, "Fault diagnosis with static and dynamic diagnosers," *Fundamenta Informaticae*, vol. 88, no. 4, pp. 497–540, Nov. 2008.

[4] P. Ramadge and W. Wonham, "Supervisory control of a class of discrete event processes," *SIAM Journal of Control and Optimization*, vol. 25, no. 1, pp. 1202–1218, 1987.

[5] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis, "Diagnosability of discrete event systems," *IEEE Transactions on Automatic Control*, vol. 40, no. 9, Sept. 1995.

[6] S. Jiang, Z. Huang, V. Chandra, and R. Kumar, "A polynomial algorithm for testing diagnosability of discrete event systems," *IEEE Transactions on Automatic Control*, vol. 46, no. 8, Aug. 2001.

[7] T.-S. Yoo and S. Lafortune, "Polynomial-time verification of diagnosability of partially-observed discrete-event systems," *IEEE Transactions on Automatic Control*, vol. 47, no. 9, pp. 1491–1495, Sept. 2002.

[8] S. Tripakis, "Fault diagnosis for timed automata," in *Proceedings of the International Conference on Formal Techniques in Real Time and Fault Tolerant Systems (FTRTFT'02)*, ser. LNCS, W. Damm and E.-R. Olderog, Eds., vol. 2469. Springer Verlag, 2002, pp. 205–224.

[9] P. Bouyer, F. Chevalier, and D. D'Souza, "Fault diagnosis using timed automata," in *FoSSaCS*, ser. LNCS, V. Sassone, Ed., vol. 3441. Springer Verlag, 2005, pp. 219–233.

[10] F. Cassez, "Dynamic Observers for Fault Diagnosis of Timed Systems," Mar. 2010, 8 pages, CoRR/abs arXiv:1006.4681 [cs.FL].

[11] R. Alur and D. Dill, "A theory of timed automata," *Theoretical Computer Science*, vol. 126, pp. 183–235, 1994.

[12] F. Cassez, "A Note on Fault Diagnosis Algorithms," in *48th IEEE Conference on Decision and Control and 28th Chinese Control Conference*. Shanghai, P.R. China: IEEE Computer Society, Dec. 2009.

[13] P. Bouyer, D. D'Souza, P. Madhusudan, and A. Petit, "Timed control with partial observability," in *Proceedings of the 15th International Conference on Computer Aided Verification (CAV'03)*, ser. LNCS, W. A. Hunt, Jr and F. Somenzi, Eds., vol. 2725. Boulder, Colorado, USA: Springer, July 2003, pp. 180–192.

[14] O. Finkel, "On decision problems for timed automata," *Bulletin of the European Association for Theoretical Computer Science*, vol. 87, pp. 185–190, 2005.

[15] S. Tripakis, "Folk theorems on the determinization and minimization of timed automata," *Information Processing Letters*, vol. 99, no. 6, pp. 222–226, 2006.

[16] R. Alur, L. Fix, and T. A. Henzinger, "A determinizable class of timed automata," in *Proceedings of the 6th International Conference on Computer Aided Verification (CAV'94)*, ser. LNCS, vol. 818. Springer Verlag, 1994, pp. 1–13.

[17] G. Behrmann, A. Fehnker, T. Hune, K. G. Larsen, P. Pettersson, J. Romijn, and F. Vaandrager, "Minimum-cost reachability for priced timed automata," in *Proc. 4th International Workshop on Hybrid Systems: Computation and Control (HSCC'01)*, ser. LNCS, vol. 2034. Springer, 2001, pp. 147–161.

[18] R. Alur, S. La Torre, and G. J. Pappas, "Optimal paths in weighted timed automata," in *Proc. 4th Int. Work. Hybrid Systems: Computation and Control (HSCC'01)*, ser. LNCS, vol. 2034. Springer, 2001, pp. 49–62.

[19] P. Bouyer, E. Brinksma, and K. G. Larsen, "Optimal infinite scheduling for multi-priced timed automata," *Formal Methods in System Design*, vol. 32, no. 1, pp. 2–23, Feb. 2008.