Monitoring and Fault-Diagnosis with Digital Clocks
Karine Altisen, Franck Cassez, Stavros Tripakis

To cite this version:

HAL Id: inria-00363034
https://hal.inria.fr/inria-00363034
Submitted on 20 Feb 2009

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Monitoring and Fault-Diagnosis with Digital Clocks

Karine Altisen
Verimag Laboratory†
Karine.Altisen@imag.fr

Franck Cassez
IRCCyN Laboratory‡
Franck.Cassez@cnrs.irccyn.fr

Stavros Tripakis
Verimag Laboratory and Cadence Berkeley Labs§
Stavros.Tripakis@imag.fr, tripakis@cadence.com

Abstract

We study the monitoring and fault-diagnosis problems for dense-time real-time systems, where observers (monitors and diagnosers) have access to digital rather than analog clocks. Analog clocks are infinitely-precise, thus, not implementable. We show how, given a specification modeled as a timed automaton and a timed automaton model of the digital clock, a sound and optimal (i.e., as precise as possible) digital-clock monitor can be synthesized. We also show how, given plant and digital clock modeled as timed automata, we can check existence of a digital-clock diagnoser and, if one exists, how to synthesize it. Finally, we consider the problem of existence of digital-clock diagnosers where the digital clock is unknown. We show that there are cases where a digital clock, no matter how precise, does not exist, even though the system is diagnosable with analog clocks. Finally, we provide a sufficient condition for digital-clock diagnosability.

1. Introduction

Monitoring and Fault-Diagnosis. In this paper we study the problems of monitoring and fault diagnosis in the context of real-time systems. In both problems the objective is to synthesize an observer, that is, a device that observes a certain system (or plant) and infers some information about this system.

In the monitoring problem, we want to know whether the system satisfies a given specification. Here, the system is black-box, that is, we have no information about how the system behaves. Therefore, we cannot check that all behaviors of the system satisfy the specification. Rather, we can observe the system during its execution and attempt to check whether the observed behavior satisfies the specification. This is the objective of the observer, which in this case is called a monitor. Our goal is to synthesize a monitor automatically from the specification.

In the fault-diagnosis problem, we have a model of the system, for instance, in the form of an (untimed or timed) automaton. We also know that the system may produce some faults. However, these faults are not directly observable, thus, their occurrence must be deduced from other observations (this can be seen as a grey-box setting). The objective of the observer, which in this case is called a diagnoser, is to detect whether a fault occurred or not, and this as soon as possible after the fault happened. In this case, before we attempt to synthesize a diagnoser, we must first check existence of a diagnoser, called diagnosability. Indeed, a diagnoser may not exist in cases where the system can produce two behaviors, one faulty and the other non-faulty, which appear the same to an external observer.

Monitoring and fault-diagnosis have been extensively studied in “untimed” settings, for instance, where specifications and plants are given as finite automata. Then, synthesizing a monitor simply means determinizing the system (possibly after “hiding” unobservable events). Fault-diagnosis has been first introduced in [10], where it was shown how to check diagnosability and, in the case it holds, synthesize a diagnoser.

More recently, these problems have also been studied in a real-time setting, where specifications and plants are given as timed automata [2]. In particular, the monitoring problem has been studied, as a special case of the conformance testing problem in [8, 9]. The fault-diagnosis problem has been studied in [11, 5].

Implementability & Digital Clocks. Most of the above works, however, consider analog-clock observers, that is, observers that are capable of observing time as precisely...
In the fault-diagnosis problem, the plant is given as a timed automaton \( A \) over \( \Sigma_0 \cup \{\tau, f\} \). \( \Sigma_0 \) models the observable events, \( \tau \) the unobservable events, and \( f \) the (unobservable) fault. We consider three problems: (1) given plant \( A \), digital clock \( A_{DC} \) and time bound \( \Delta \), check whether there is a diagnoser that can detect any fault within \( \Delta \) time units; (2) given \( A \) and \( A_{DC} \) check whether there is a diagnoser for some \( \Delta \); (3) given \( A \), check whether there is a diagnoser for some digital clock \( A_{DC} \) and some \( \Delta \). We show how to solve problems (1) and (2). Problem (3) is open. However, we give an example that shows that existence of an analog-clock diagnoser does not imply existence of a digital-clock diagnoser, and this for any digital-clock, no matter how precise it is. We also provide a sufficient (but not necessary) condition for existence of a digital-clock diagnoser.

Related Work. The distinction between analog and digital clocks has been made in [6] and subsequent work on digitization of timed automata. Digitization studies how the “sampling” semantics of timed automata (that is, the semantics where only particular time delays are allowed, e.g., multiples of \( \frac{1}{n} \)) are related to its dense-time semantics. Although related, this is not the same as our problem, where we study the observational capabilities of digital-clocks.

Digital clocks have been used for monitoring and testing in [8, 3]. We borrow from these works the idea to model digital clocks as timed automata. Soundness and optimality of monitors is not considered in these works, neither are the problems of fault-diagnosis and synthesis of digital clocks. Finally, digital clocks have been also used in [1] in the context of timed automata implementation.

Fault diagnosis with digital clocks has recently been considered independently in [7]. This work is restricted in several ways compared to ours. First, it only considers digital clocks that “tick” every \( \lfloor \Delta + \delta \rfloor \) time units. In our framework digital clocks are modeled as timed automata, which can capture the above clocks and more (see Figure 2). Second, in [7] the non-faulty behavior of the plant is modeled as a deterministic timed automaton, whereas we allow the plant to be non-deterministic (we also allow unobservable events other than faults). Finally, the synthesis of a digital clock (problem (3) above) is not considered in [7].

Outline of the paper. In section 2, we recall the basic definitions of timed words, timed automata and monitors and diagnosers. In section 3, we study digital-clock monitoring. Section 4 is devoted to digital-clock fault-diagnosis. Section 5 concludes the paper.
2. Preliminaries

2.1. Clock constraints, timed words

Let \( \mathbb{N} \) be the set of natural numbers, \( \mathbb{Z} \) the set of integers, \( \mathbb{Q} \) the set of rationals and \( \mathbb{R} \) the set of non-negative reals.

Let \( X \) be a set of variables taking values in \( \mathbb{R} \). In the context of timed automata, a variable in \( X \) is called a clock. An atomic clock constraint over \( X \) is an expression of the form \( x \not\in c \), where \( x \in X \), \( c \) is a rational constant and \( \not\in \{<,\leq,=,\geq,>\} \). A convex clock constraint over \( X \) is a conjunction of atomic clock constraints over \( X \). A clock constraint over \( X \) is a boolean expression of atomic clock constraints over \( X \). A valuation over \( X \) is a function \( v : X \to \mathbb{R} \), assigning to each clock a value. Given \( r \subseteq X \), \( v[r := 0] \) denotes the valuation \( v' \) such that for all \( x \in r \), \( v'(x) = 0 \) and for all \( y \in X - r \), \( v'(y) = v(y) \). \( \overline{0} \) denotes the valuation assigning zero to each clock. A valuation \( v \) satisfies a clock constraint \( g \), written \( v \models g \), if substituting \( v(x) \) for every \( x \) appearing in \( g \) yields a valid constraint.

Let \( \Sigma \) be a finite alphabet. A timed word over \( \Sigma \) is a finite sequence of delays in \( \mathbb{R} \) and letters in \( \Sigma : \rho \in (\Sigma \cup \mathbb{R})^* \). Every such sequence can be put in a canonical form by summing up consecutive delays and adding initial and final zero delays if necessary: \( \rho = t_0 \cdot a_1 \cdot t_1 \cdot a_2 \cdot t_2 \cdot \cdots \cdot a_n \cdot t_n \).

We will define a set of projection functions on timed words. The first can be seen as a projection onto \( \mathbb{R} \): \( \text{Time}(\rho) \) denotes the total amount of time spent in \( \rho \), that is, \( \text{Time}(\rho) = \sum_{a_i = a_1, \ldots, a_n} t_n \). The second is the untrimming projection: \( \text{Unt}(\rho) \in \Sigma^* \) is the sequence of letters \( a_1 \cdots a_n \). Conversely, given a sequence of letters \( a_1 \cdots a_n \), \( \text{Unt}^{-1}(a_1 \cdots a_n) \) is the set of timed words \( t_0, a_1, t_1, \cdots, a_n, t_n \) with \( t_i \in \mathbb{R} \). Finally, given \( \Sigma' \subseteq \Sigma \), \( \Pi_{\Sigma'}(\rho) \) is the timed word over \( \Sigma' \) obtained from \( \rho \) by erasing all events not in \( \Sigma' \). For example, if \( \Sigma = \{a, b\} \), \( \Sigma' = \{a\} \) and \( \rho = a \cdot 1 \cdot 2 \cdot b \cdot 1 \cdot b \) then \( \Pi_{\Sigma'}(\rho) = a \cdot 1 \cdot 3 \).

Projections \( \text{Unt} \) and \( \Pi \) can be naturally extended to sets of timed words in the usual way.

Given two timed words \( \rho_1 \in (\Sigma_1 \cup \mathbb{R})^* \) and \( \rho_2 \in (\Sigma_2 \cup \mathbb{R})^* \) the parallel composition \( \rho_1 || \rho_2 \) is the set of words \( \rho \in (\Sigma_1 \cup \Sigma_2 \cup \mathbb{R})^* \) s.t. \( \Pi_{\Sigma_1}(\rho) = \rho_1 \) and \( \Pi_{\Sigma_2}(\rho) = \rho_2 \). For example \( 2.a || 3.b = \{2.a.1.b\} \) and \( 2.a || 2.b = \{2.a.2,b.a\} \). Given two sets of timed words \( L_1 \subseteq (\Sigma_1 \cup \mathbb{R})^* \) and \( L_2 \subseteq (\Sigma_2 \cup \mathbb{R})^* \), the parallel composition of \( L_1 \) and \( L_2 \) is \( L_1 || L_2 = \{\rho_1 || \rho_2 | \rho_1 \in L_1, \rho_2 \in L_2\} \).

2.2. Timed automata

Let \( \Sigma \) be a finite alphabet. A timed automaton over \( \Sigma \) is a tuple \( A = (\Sigma, Q, Q_0, Q_I, X, I, E) \), where:

- \( Q \) is a finite set of locations, \( Q_0 \subseteq Q \) is the set of initial locations and \( Q_I \subseteq Q \) is the set of final locations.
- \( X \) is a finite set of clocks.
- \( I \) is the invariant function, associating to each location \( q \in Q \) a clock constraint over \( X \). We assume that \( \overline{0} \in I(q_0) \) for all \( q_0 \in Q_0 \).
- \( E \) is a finite set of edges. Each edge is a tuple \((q, q', a, g, r)\), where \( q \) and \( q' \) are the source and destination locations, \( a \in \Sigma \) is the label of the edge, \( g \) is a clock constraint over \( X \), called the guard of the edge, and \( r \subseteq X \) is the set of clocks to reset to zero when the edge is crossed.

A state of \( A \) is a pair \((q, v)\), where \( q \in Q \) and \( v \) is a valuation over \( X \) such that \( v \models I(q) \). The set of initial states of \( A \) is \( S_0 = \{(q_0, \overline{0}) | q_0 \in Q_0\} \). The set of final states of \( A \) is \( S_f = \{(q_f, v) | q_f \in Q_I\} \). Let \( S_A \) denote the set of all states of \( A \).

A discrete transition of \( A \) is a triple \((s, a, s')\), where \( s, s' \in S_A \) and \( a \in \Sigma \), such that \( s = (q, v), s' = (q', v') \) and there is an edge \( e = (q, q', a, g, r) \in E \), such that \( v \models g \) and \( v' = v[r := 0] \). If such a transition exists we write \( s \xleftarrow{a} s' \). A time transition of \( A \) is a triple \((s, t, s')\), where \( s, s' \in S_A \) and \( t \in \mathbb{R} \), such that \( s = (q, v), s' = (q, v + t) \) and for all \( 0 \leq t' \leq t, v + t' \in I(q) \). If such a transition exists we write \( s \xrightarrow{t} s' \). Notice that for all states \( s \in S_A \), \( s \not\xrightarrow{t} s \).

A run of \( A \) is a finite sequence of transitions: \( \sigma = s_0 \xrightarrow{t_0} s_1 \xrightarrow{t_1} \cdots \xrightarrow{t_n} s_{n+1} \), where \( n \geq 0 \). We say that the run starts from state \( s_0 \) and reaches state \( s_{n+1} \). If \( s_0 \in S_0 \) and \( s_{n+1} \in S_f \) then the run is accepting and the corresponding timed word is accepted by \( A \). The set of all timed words accepted by \( A \) is the (timed) language of \( A \), denoted \( L(A) \).

The untimed language of \( A \), denoted \( \text{Unt}(A) \), is \( L(U(T(A))) \).

A state \( s \) of \( A \) is called non-Zeno if for all \( t \in \mathbb{R} \) there exists a run \( \sigma \) starting at \( s \) such that \( \text{Time}(\text{tw} (\sigma)) \geq \tau \). A is called non-Zeno if all its reachable states are non-Zeno. Non-Zenoness means that \( A \) cannot “block time”. Note that this does not mean that all runs of \( A \) let time progress (i.e., are non-Zeno) but rather that there is always the possibility of letting time progress.

Given two timed automata \( A_1 = (\Sigma_1, Q_1, Q_0, Q_I, X_1, I_1, E_1) \) and \( A_2 = (\Sigma_2, Q_2, Q_0, Q_I, X_2, I_2, E_2) \), the parallel composition of \( A_1 \) and \( A_2 \) denoted \( A_1 || A_2 \), is the timed automaton \( A = (\Sigma, Q, Q_0, Q_I, X, I, E) \) defined as follows:\footnote{We assume neither \( \Sigma_1 \) nor \( \Sigma_2 \) contains the silent action \( \tau \).}

\[ \Sigma = \Sigma_1 \cup \Sigma_2. \]
• \( Q = Q_1 \times Q_2, Q^0 = Q^0_1 \times Q^0_2 \) and \( Q^f = Q^f_1 \times Q^f_2 \).

• \( X = X_1 \cup X_2 \).

• \( I(q_1, q_2) = I(q_1) \wedge I(q_2) \).

• \( E \) contains the following transitions:
  - For each \( a \in \Sigma_1 \cap \Sigma_2 \), if \( (q_1, q_1', a, q_1, r_1) \in E_1 \) and \( (q_2, q_2', a, q_2, r_2) \in E_2 \) then \( e = ((q_1, q_2), (q_1', q_2'), a, g_1 \wedge g_2, r_1 \cup r_2) \in E \). That is, the two automata synchronize on \( \Sigma_1 \cap \Sigma_2 \).
  - For each \( a \in \Sigma_1 \setminus \Sigma_2 \), if \( (q_1, q_1', a, q_1, r_1) \in E_1 \) then \( e = ((q_1, q_2), (q_1, q_2'), a, q_1, r_1) \in E \).
  - For each \( a \in \Sigma_2 \setminus \Sigma_1 \), if \( (q_2, q_2', a, q_2, r_2) \in E_2 \) then \( e = ((q_1, q_2), (q_1', q_2), a, g_2, r_2) \in E \).

It can be checked that \( L(A_1 || A_2) = L(A_1) || L(A_2) \).

3. Monitoring

3.1. Digital-clock automata: models of digital clocks

A digital-clock automaton (DC-automaton for short) \( A \) is a non-Zeno timed automaton \( (\{\text{tick}\}, Q, Q^0, Q^f, \Sigma, X, I, E) \). The tick event is a special event representing one “tick” of the digital clock.

The idea of using timed automata to model digital clocks has been introduced in [8] where it has been used for real-time testing. The idea has also been used in [1] for implementation of timed automata.

To illustrate the concept, we borrow some examples of digital-clock automata from the above works. These are shown in Figure 2. \( A_{DC}^1 \) models a perfectly periodic digital clock with period 1. \( A_{DC}^2(\epsilon) \) is an automaton parameterized by \( \epsilon \), and models a clock with non-perfect period \( 1 \pm \epsilon \). In this model errors may accumulate, so that the \( i \)-th tick of the clock may occur anywhere in the interval \([1 - \epsilon)i, 1 + \epsilon)i\]. \( A_{DC}^1(\epsilon) \) models a more restricted behavior where errors do not accumulate: the \( i \)-th tick occurs in the interval \([i - \epsilon, i + \epsilon]\), for all \( i \).

3.2. Digital-clock monitors

Let \( \Sigma_o \) be a finite alphabet such that \( \text{tick} \notin \Sigma_o \). A digital-clock monitor over alphabet \( \Sigma_o \) is a function \( D : (\Sigma_o \cup \{\text{tick}\})^* \to \{0, 1\} \). The untimed language of \( D \) is defined as \( \text{Unt}(D) = \{ \pi \in (\Sigma_o \cup \{\text{tick}\})^* \mid D(\pi) = 1 \} \). The timed language of \( D \) is defined as \( L(D) = \text{Unt}^{-1}(\text{Unt}(D)) \).

Digital-clock monitors accept or reject untimed words in \((\Sigma_o \cup \{\text{tick}\})^*\). Such words represent observations that the monitor receives during its execution. These observations are sequences of: (a) observable events received by the monitored plant; and (b) tick events received by the (digital) clock of the system (i.e., computer) where the monitor executes. More precisely, if \( \rho \) is a timed behavior generated by the system under observation, and if \( \sigma \) is a timed behavior generated by the digital clock, then \( D \) receives an untimed observation in \( \Pi_{\Sigma_o}(\text{Unt}(\rho(\sigma))) \). Indeed, \( \rho' \in \rho(\sigma) \) is a timed behavior corresponding to some (non-deterministically chosen) interleaving of \( \rho \) and \( \sigma \). \( \pi = \Pi_{\Sigma_o}(\text{Unt}(\rho')) \) is the observation received by the monitor when \( \rho' \) occurs: real-time delays and unobservable events are removed from \( \rho' \) in order to obtain \( \pi \).

3.3. Soundness of digital-clock monitors

The monitor is supposed to check conformance to a specification. The latter is modeled as a timed automaton \( A \) over \( \Sigma \supseteq \Sigma_o \). A crucial property for monitors is soundness: if \( \rho \) is a behavior in \( L(A) \), then the monitor should not reject it (i.e., announce non-conformance whereas the timed word is conforming). A first attempt to capture soundness for digital-clock monitors is given by the definition below. The definition can be read as follows: “if \( \rho \) is a timed word of the specification \( L(A) \), then the untimed observable version of \( \rho \) must be accepted by the digital-clock monitor \( D \) when the latter executes in parallel with the digital clock automaton \( A_{DC}^\bullet \).”

Definition 1 (A first attempt to define soundness) Given a timed automaton \( A \), a digital-clock automaton \( A_{DC} \) and a monitor \( D \), the predicate sound\( _1(A, A_{DC}, D) \) holds iff:

\[
\Pi_{\Sigma_o}(L(A)) \subseteq \Pi_{\Sigma_o}(L(D)||L(A_{DC}))
\]

This definition is not satisfactory as demonstrated by the following example:

Example 1 Let \( \Sigma_o = \Sigma = \{a\} \), \( L(A) = \{2 \cdot a \cdot t \mid t \in \mathbb{R}\} \) (i.e., \( t \) occurs at time 2) and \( L(A_{DC}) = \{t_1 \cdot \text{tick} \cdot t_2 \mid t_1 \in [1, 3), t_2 \in \mathbb{R}\} \) i.e. \( \text{tick} \) occurs either at time 1 or at time 3. Consider the monitor \( D \) such that \( \text{Unt}(D) = \{\text{tick} \cdot a\} \).

We claim that \( D \) is sound, according to predicate sound\( _1 \). Indeed, \( L(D) = \{t_1 \cdot \text{tick} \cdot t_2 \cdot a \cdot t_3 \mid t_i \in \mathbb{R}\} \). Thus, \( L(D)||L(A_{DC}) = \{t_1 \cdot \text{tick} \cdot t_2 \cdot a \cdot t_3 \mid t_1 \in [1, 3), t_2, t_3 \in \mathbb{R}\} \) and \( \Pi_{\Sigma_o}(L(D)||L(A_{DC})) = \{t_1 \cdot t_2 \cdot a \cdot t_3 \mid t_1 \in [1, 3), t_2, t_3 \in \mathbb{R}\} = \{t_1 \cdot t_2 \cdot t_3 \mid t_1 \geq 1, t_2, t_3 \in \mathbb{R}\} \). (i.e., \( \text{tick} \) occurs at some point later than time 1). Therefore, we have \( \Pi_{\Sigma_o}(L(A)) \subseteq \Pi_{\Sigma_o}(L(D)||L(A_{DC})) \), as required by predicate sound\( _1 \). However, the above monitor does not conform to our intuition of soundness (page 4). Indeed, consider the timed word of \( A_{DC} \), \( \rho_{A_{DC}} = 3 \cdot \text{tick} \). Executed together with the only timed word of \( A \), \( \rho = 2 \cdot a \), this produces the the untimed digital observation \( \Pi_{\Sigma_o}(\text{Unt}(\rho(\rho_{A_{DC}}))) = \{a \cdot \text{tick}\} \). The monitor \( D \) rejects it whereas it should have accepted it if it were “sound” as stated p. 4. Note also that
taking 1·tick for \( \rho_{\text{DC}} \) leads the monitor to accept \( \rho = 2 \cdot a \): this lights on why sound\(_1\) does not capture what we want.

Actually the definition of sound\(_1\) captures the following fact: whenever a timed word \( \rho \) is in \( L(A) \), there is a behavior of \( A_{\text{DC}} \) that generates tick so that the digital monitor will accept the untimed digital behavior. This means that if we have a timed automaton \( A_{\text{DC}} \) that generates a unique timed word, sound\(_1\) would be sufficient. But of course, we want to model non deterministic and drifting clocks and the previous definition is not what we are looking for. We want our digital monitor to be robust against the disturbances generated by the digital clock, i.e. that for any of its timed words, the monitor has a constant answer (in case of acceptance). To remedy this problem, we provide a robust definition of soundness.

**Definition 2 (Robust definition of soundness)** Given a timed automaton \( A \), a digital-clock automaton \( A_{\text{DC}} \) and a monitor \( D \), the predicate sound\((A, A_{\text{DC}}, \epsilon)\) holds iff:

\[
\forall \rho \in L(A). \forall \rho' \in L(A_{\text{DC}}). (\text{Unt}(\rho)||\text{Unt}(\rho')) \subseteq \text{Unt}(D)
\]

This definition states precisely what we want, namely, that for any behavior \( \rho \) which conforms to the specification \( A \), and for any possible behavior \( \rho' \) of the digital clock, the monitor \( D \) will accept any observation resulting from these two behaviors. It can be shown that sound is strictly stronger than sound\(_1\).

**Proposition 1** For any timed automaton \( A \), for any digital-clock automaton \( A_{\text{DC}} \) such that \( L(A_{\text{DC}}) \neq \emptyset \), and for any monitor \( D \)

\[
\text{sound}(A, A_{\text{DC}}, D) \Rightarrow \text{sound}_1(A, A_{\text{DC}}, D).
\]

The following results are immediate consequences of Definition 2: the first one states that if a monitor is sound for a certain digital-clock then it remains sound for a “more deterministic” digital-clock; the second one that a sound monitor should at least accept the untimed language of the product of \( A \) and \( A_{\text{DC}} \).

**Lemma 1** For any timed automaton \( A \) and digital-clock automata \( A_{\text{DC}}^1 \) and \( A_{\text{DC}}^2 \), and for any monitor \( D \)

\[
\begin{align*}
\text{sound}(A, A_{\text{DC}}^1, D) \\
\wedge L(A_{\text{DC}}^1) \subseteq L(A_{\text{DC}}^2)
\end{align*}
\Rightarrow \text{sound}(A, A_{\text{DC}}^2, D).
\]

**Lemma 2** Let \( D \) be such that sound\((A, A_{\text{DC}}, D)\). Then \( \Pi_{\text{Unt}}(\text{Unt}(L(A||A_{\text{DC}}))) \subseteq \text{Unt}(D) \).

Now that we have a definition of sound digital-clock monitors, it appears straightforward to define the following monitoring problem.

**Problem 1 (First definition of a monitoring problem)**

Given a timed automaton \( A \) and a digital-clock automaton \( A_{\text{DC}} \), synthesize a monitor \( D \) such that sound\((A, A_{\text{DC}}, D)\).

This problem, however, has a trivial solution, namely, the monitor that accepts all behaviors: \( D(\sigma) = 1 \) for all \( \sigma \). Therefore, we need to find some way to exclude such trivial monitors. We do this by introducing, in the section that follows, orders that allow us to speak of “better” or “worse” monitors, and also to synthesize optimal monitors w.r.t. those orders.

### 3.4. Orders on monitors

Our aim is to synthesize “optimal” monitors. For this, we need to introduce an order which captures that a monitor is “better” than another monitor. We explore some possible orders in this section.

A first possible choice is to compare the languages accepted by the monitors using standard subset relation. Given two monitors \( D \) and \( D' \),

\[
D \subseteq D' \equiv \text{Unt}(D) \subseteq \text{Unt}(D').
\]

Notice that this order is independent of the “application” in question, namely, the specification \( A \) and the digital clock \( A_{\text{DC}} \). We proceed by defining an alternative order which depends on \( A \) and \( A_{\text{DC}} \).

![Digital-clock Automata](image-url)
We define $D \leq_{A_{DC}} D'$ iff for any $\rho \in L(A)$ and any $\sigma \in L(A_{DC})$, we have
\[
\Pi_{\Sigma_o}(\text{Unt}(\rho \mid \sigma)) \cap \text{Unt}(D') = \emptyset \Rightarrow \\
\Pi_{\Sigma_o}(\text{Unt}(\rho \mid \sigma)) \cap \text{Unt}(D) = \emptyset
\]

The above formula states that an observation rejected by $D'$ is also rejected by $D$, provided this observation can be generated by some behavior $\rho$ of $A$ and some behavior $\sigma$ of $A_{DC}$.

It can be easily shown that if $D \subseteq D'$ then for any $A$ and $A_{DC}$, $D \leq_{A_{DC}} D'$.

**Definition 3 (Minimal and optimal monitors)** Let $A \prec$ class of monitors and let $A$ respect to order $D$ generated by some behavior $D_\sigma$ optimal called $D_A$. Given a timed automaton $A$, this automaton introduced above.

We now present a solution to the optimal digital-clock monitoring problem, namely, we show how to construct a monitor such that $D$ is minimal or optimal with respect to $A_{DC}$.

Consider a timed automaton $A$ and a digital-clock automaton $A_{DC}$. Let $D$ be a monitor such that $D \leq_{A_{DC}} D'$. Then $D_0 \subseteq D$ and $D_0 \leq_{A_{DC}} D$

**Remark 1** Proposition 3 does not hold for predicate sound. Indeed, consider the monitor $D$ described in Example 1, which is sound w.r.t. sound. We have $L(D_0) = \{\text{tick-a, a-tick}\}$, whereas $L(D) = \{\text{tick-a}\}$. Thus, $D \subseteq D_0$, which means that $D_0$ is not optimal w.r.t. $\subseteq$ and sound.

**Remark 2** The automaton $A_0$ described above can be determined using the usual subsets construction. This way, $D_0$ is just a Moore version of $A_0$ where the final states of $A_0$ are labeled with 1 and the other states with 0.

4. Fault diagnosis

The fault-diagnosis problem has been introduced and studied in the untimed setting of discrete-event systems in [10] and extended to the timed automata setting with analog-clock (i.e., infinite-precision) diagnosers in [11]. Here, we study the problem in the timed automata setting but with digital-clock diagnosers.

Fault diagnosis is similar to monitoring. The main differences are two. First, the plant under observation is not entirely black-box. A model of the plant is available, but this model contains unobservable actions. Some of these unobservable actions model faults that may occur in the plant. The goal of the diagnoser is to detect whether a fault has occurred and, in the case where there are many different types of faults, to identify which fault occurred. The second difference with monitoring is that the diagnoser must announce a fault within a bounded, albeit unknown a-priori, delay after the fault occurred. Thus, the plant is supposed to continue execution forever, and the diagnoser’s task is to detect faults if possible, and as soon as possible.

These two differences imply that, contrary to monitoring, synthesis of a diagnoser is not always possible. This is the case when the plant contains two distinct behaviors, one faulty and the other non-faulty, which produce the same
observation as far as the diagnoser is concerned. Thus, the first task is to check \textit{ diagnosability}, that is, the existence of a diagnoser. If diagnosability holds, then \textit{a diagnoser can be synthesized.}

It should then be clear that knowing how to solve the monitoring problem does not imply we can solve the diagnosis problem: the latter requires us to check diagnosability, which does not exist in monitoring. On the other hand, one might think that the monitoring problem can be reduced to the diagnosis problem, by building a plant where non-faulty behaviors correspond to the “positive” monitoring specification and faulty behaviors to its negation. A first problem with this is that it assumes we can complement the specification, which is not always possible, neither easy, for timed automata. Even when the complement is available, the above construction may yield a plant automaton with non-accepting states: this is not allowed in our diagnosis framework (see below). For the above reasons, it is justified to treat the two problems separately.

For simplicity, we are going to consider the case of a single type of faults. The results can be easily extended to the case of multiple different types of faults.

### 4.1. Digital-clock diagnosers

The plant is modeled as a timed automaton $A$ over $\Sigma = \Sigma_o \cup \{\tau, f\}$, where $\tau, f \notin \Sigma_o$: $\tau$ models unobservable events which are not faults; $f$ models the faults, which are also unobservable; $\Sigma_o$ models the observable events. We assume that all locations of $A$ are accepting. That is, the language of $A$ is prefix-closed. This is in accordance with the interpretation given above, namely, that the plant is expected to continue execution and the objective is to detect faults after some bounded delay.

Let $\rho$ be a timed word in $(\Sigma \cup \mathbb{R})^*$. $\rho$ is said to be \textit{non-faulty} if the letter $f$ does not appear in $\rho$, that is, $\Pi(f)(\rho) = \text{Time}(\rho)$. Otherwise, $\rho$ is said to be \textit{faulty}. Let $\Delta \in \mathbb{N}$. $\rho$ is said to be $\Delta$-\textit{faulty} if there exist $\rho_1 \in ((\Sigma \setminus \{f\}) \cup \mathbb{R})^*$ and $\rho_2 \in (\Sigma \cup \mathbb{R})^*$ such that $\rho = \rho_1 \cdot f \cdot \rho_2$ and $\text{Time}(\rho_2) \geq \Delta$. That is, if $\rho$ is $\Delta$-faulty then at least $\Delta$ time units have elapsed after the occurrence of the first fault in $\rho$.

**Definition 4 (Digital-clock diagnosability)** Let $A_{DC}$ be a digital-clock automaton and let $\Delta \in \mathbb{N}$. $A_{DC}$ is a total function

$$D : (\Sigma_o \cup \{\text{tick}\})^* \rightarrow \{0, 1\}$$

such that:

- for any $\pi, \pi' \in (\Sigma_o \cup \{\text{tick}\})^*$,\footnote{If $D(\pi) = 1$ then $D(\pi \cdot \pi') = 1$, and}
- for any $\rho \in L(A)$ and any $\sigma \in L(A_{DC})$, \footnote{If $\text{Time}(\rho) = \text{Time}(\sigma)$, then}

In other words, a $(A_{DC}, \Delta)$-diagnoser must announce 0 (i.e., “no fault detected”) for any behavior that is non-faulty, no matter what the behavior of the digital clock is: this is a soundness requirement. On the other hand, a $(A_{DC}, \Delta)$-diagnoser must announce 1 (i.e., “fault detected”) for any behavior that is faulty, provided at least $\Delta$ time units have elapsed after the first fault: this is a liveness requirement. No requirement is made for faulty behaviors where less than $\Delta$ time units have elapsed after the fault. However, the first requirement ensures that the diagnoser does not “change its mind” once it has announced a fault.

$A$ is said to be $(A_{DC}, \Delta)$-\textit{diagnosable} if there exists a $(A_{DC}, \Delta)$-diagnoser for $A$. $A$ is said to be $A_{DC}$-\textit{diagnosable} if there exists $\Delta \in \mathbb{N}$ such that $A$ is $(A_{DC}, \Delta)$-diagnosable. $A$ is said to be digital-clock diagnosable if there exists $A_{DC}$ such that $A$ is $A_{DC}$-diagnosable.

The following result is the counterpart of Lemma 1 for diagnosis, and the proof is straightforward.

**Lemma 3** For any timed automaton $A$, for any digital-clock automata $A_{DC}^1$ and $A_{DC}^2$, for any $\Delta_1, \Delta_2$, if $D$ is a $(A_{DC}^1, \Delta_1)$-diagnoser for $A$, $L(A_{DC}^1) \subseteq L(A_{DC}^2)$ and $\Delta_2 \geq \Delta_1$, then $D$ is also a $(A_{DC}^2, \Delta_2)$-diagnoser for $A$.

Consider the digital-clock automaton $A_{DC}^p$ shown in Figure 3. $A_{DC}^p$ is a generalization of automaton $A_{DC}$ of Figure 2. $A_{DC}^p$ models a perfectly-periodic clock with period $p$. We say that $A$ is $p$-\textit{diagnosable} if $A$ is $A_{DC}^p$-diagnosable. The lemma below states that a periodic clock that “ticks” $k$ times faster than another clock is better for diagnosability, that is, if diagnosability holds for the slower clock then it will also hold for the fast clock.

**Lemma 4** Consider the digital-clock automaton $A_{DC}^p$ shown in Figure 3. For any timed automaton $A$, for any $p_1, p_2$, if $p_1 = k \cdot p_2$ for some $k \in \mathbb{N}$, and $A$ is $p_1$-diagnosable, then $A$ is also $p_2$-diagnosable.

With these definitions, we can define a set of problems.

**Problem 3 ((A_{DC}, \Delta)-diagnosability problem)** Given $A$ and $A_{DC}, \Delta$, check whether there exists a $(A_{DC}, \Delta)$-diagnoser for $A$. 

Problems 4 (\(A_{DC}\)-diagnosability problem) Given \(A\) and \(A_{DC}\), check whether there exists \(\Delta\) such that there exists a \((A_{DC}, \Delta)\)-diagnoser for \(A\).

Problem 5 (Diagnosability problem) Given \(A\), check whether there exist \(A_{DC}\) and \(\Delta\) such that there exists a \((A_{DC}, \Delta)\)-diagnoser for \(A\).

In each of the problems above, in the case where a diagnoser exists, we would also like to synthesize one.

4.2. Solution to the \((A_{DC}, \Delta)\)-diagnosability problem

Proposition 4 (Necessary and sufficient condition for \((A_{DC}, \Delta)\)-diagnosability) Let \(A\) be any timed automaton, let \(A_{DC}\) be a digital-clock automaton and let \(\Delta \in \mathbb{N}\). \(A\) is \((A_{DC}, \Delta)\)-diagnosable if and only if for all non-faulty \(\rho \in L(A)\), \(\Delta\)-faulty \(\rho' \in L(A)\), \(\sigma, \sigma' \in L(A_{DC})\) the following holds:

\[
\begin{align*}
\text{Time}(\rho) &= \text{Time}(\sigma) \land \text{Time}(\rho') = \text{Time}(\sigma') \\
\Pi_{\Sigma_{\infty}}(\text{Unt}(\rho||\sigma)) \land \Pi_{\Sigma_{\infty}}(\text{Unt}(\rho'||\sigma')) &= \emptyset
\end{align*}
\]

We now present an algorithmic method to check the necessary and sufficient condition given above. First, let \(\text{Obs}(\Delta)\) be the timed automaton of Figure 4. It is an automaton parameterized by \(\Delta\) and the accepting state is \textit{bad}.

Second, let \(A = (\Sigma, Q_0, Q, X, I, E)\). We define the timed automaton \(A_f\) over \(\Sigma\) as follows:

1. the locations of \(A_f\) are \(\{q_f | q \in Q\} \cup \{q_{-f} | q \in Q\}\); Let \(Q_f\) be the set of faulty locations \((q_f)\) locations and \(Q_{-f}\) be the set of non-faulty locations \((q_{-f})\) locations. The idea is that locations in \(Q_f\) encode the fact that a fault has occurred. The initial locations of \(A_f\) are the non-faulty locations. The accepting locations of \(A_f\) are the faulty locations.
2. the set of clocks of \(A_f\) is \(X\);
3. the initial state of \(A_f\) is \(q_0_{-f}\);
4. the transition function is defined as follows: for each edge \((q, q_f', a, g, r) \in E\) with \(a \neq f\) we create two edges in \(A_f\): \((q_f, q_f', a, g, r)\) and \((q_{-f}, q_{-f}', a, g, r)\). If \(a = f\) we create \((q_f, q_f', f, g, r)\) and \((q_{-f}, q_{-f}', f, g, r)\) (the target location must be a faulty location);
5. the invariant of \(A_f\) for \(q_f\) is the same as for \(q\) in \(A\).

Third, let \(P = (A_f | \text{Obs}(\Delta))|\text{Unt}(P)\). That is, \(P\) is the parallel composition of \(A_f\), \(\text{Obs}(\Delta)\) and \(A_{DC}\), where \(A_f\) and \(\text{Obs}(\Delta)\) synchronize on the \(f\) label, while all three automata synchronize on the passage of time. \(P\) accepts all the interleavings of (1) \(\Delta\)-faulty timed words of \(A\) and (2) timed words of \(A_{DC}\).

Fourth, let \(A_{-f}\) be a copy of \(A\) with all \(f\)-labeled transitions removed. Finally, let \(P' = A_{-f}|A_{DC}\). \(P'\) accepts all the interleavings of (1) non-faulty timed words of \(A\) and (2) timed words of \(A_{DC}\).

Proposition 5 \(A\) is \((A_{DC}, \Delta)\)-diagnosable if and only if \(\Pi_{\Sigma_{\infty}}(\text{Unt}(P)) \land \Pi_{\Sigma_{\infty}}(\text{Unt}(P')) = \emptyset\)

The condition of Proposition 5 can be checked algorithmically. Indeed, \(\text{Unt}(P)\) and \(\text{Unt}(P')\) are regular languages accepted by finite-state automata which can be constructed as explained in subsection 3.5. The projection \(\Pi_{\Sigma_{\infty}}(L)\) of a regular language \(L\) is regular and it is accepted by an automaton obtained from the automaton accepting \(L\) by replacing \(\tau\) and \(f\) labels by \(\epsilon\).

4.3. Solution to the \(A_{DC}\)-diagnosability problem

The above algorithm works for a given \(\Delta\). However, as mentioned in the introduction, \(\Delta\) is a-priori unknown. In this section, we show how to check \(A_{DC}\)-diagnosability, where \(\Delta\) is unknown. To do this, we need some definitions first.

An infinite timed word over \(\Sigma\) is an infinite sequence of delays in \(\mathbb{R}\) and letters in \(\Sigma\): \(\rho = \rho_1 \cdot \rho_2 \cdot \rho_3 \cdots\). Every such sequence can be put in a canonical form where delays and letters alternate: \(\rho = \tau_0 \cdot a_1 \cdot \tau_1 \cdot a_2 \cdot \tau_2 \cdots\). Terminology and operators that we introduced for finite words can be extended to infinite words in a straightforward way. For instance, we can speak about faulty infinite timed words, if \(f\) appears in them. We can also extend the projection and untiming operators.

Consider a timed automaton \(A\). An infinite run of \(A\) is an infinite sequence of transitions, \(\sigma = s_0 \xrightarrow{t_0} s_1 \xrightarrow{a_1} s_2 \xrightarrow{t_1} \cdots\), such that every finite prefix of \(\sigma\) is a run of \(A\) and \(s_0\) is an initial state of \(A\). The run is called non-Zeno if \(\sum t_i = \infty\). The run is called accepting if it visits accepting locations infinitely often. The set of all infinite timed words corresponding to infinite, non-Zeno, accepting runs of \(A\) is denoted \(L^\infty(A)\).

Proposition 6 (Necessary and sufficient condition for \(A_{DC}\)-diagnosability) Let \(A\) be a non-Zeno timed automaton and let \(A_{DC}\) be a digital-clock automaton. \(A\) is \(A_{DC}\)-diagnosable if and only if there do not exist \(\rho, \rho' \in L^\infty(A)\) and \(\sigma, \sigma' \in L^\infty(A_{DC})\) such that the following hold:
Proof: Let $\Delta \geq 2$. Consider a digital-clock automaton $A_{DC}$ and let $\rho' \in L(A_{DC})$ such that $\text{Time}(\rho') = \Delta$. Let $n$ be the number of tick events appearing in $\rho'$ until time 1 (including time 1) and let $m$ be the number of observable events in $\rho'$ strictly later than time 1. There are two cases to consider: either $m = 0$, that is, no tick appears after time 1 in $\rho'$; or $m > 0$.

In the case $m = 0$, we set $t' = 1.5$. In the case $m > 0$, let $t > 1$ be the first moment after time 1 that a tick event appears in $\rho'$ and let $1 < t' < t$. Consider the following two behaviors of the timed automaton of Figure 5:

$$\rho_1 = t' \cdot f \cdot a \cdot \Delta,$$

$$\rho_2 = 1 \cdot \tau \cdot a \cdot \Delta.$$  

We claim that:

$$\text{tick}^n \cdot a \cdot \text{tick}^m \in \Pi_{\Sigma_{\rho}}(\text{Unt}(\rho_1)) \cap \Pi_{\Sigma_{\rho}}(\text{Unt}(\rho_2)).$$

It is clear that $\text{tick}^n \cdot a \cdot \text{tick}^m \in \Pi_{\Sigma_{\rho}}(\text{Unt}(\rho_1))$. It is also true that $\text{tick}^n \cdot a \cdot \text{tick}^m \in \Pi_{\Sigma_{\rho}}(\text{Unt}(\rho_2))$. Indeed, even if the $n$-th tick occurs exactly at time 1, the semantics of || are such that both interleavings $a \cdot \text{tick} \cdot a$ and $a \cdot \text{tick} \cdot a$ are included.

The point is that the digital-clock diagnoser cannot tell whether $a$ occurred exactly at time 1 or at time strictly greater than 1, that is, it cannot distinguish between $\rho_1$ and $\rho_2$. Thus, according to Proposition 4, the timed automaton of Figure 5 is not $(A_{DC}, \Delta)$-diagnosable. 

We do not know whether Problem 5 is decidable. In the rest of this section, we provide a sufficient condition for existence of digital-clock diagnosers.

Consider a TA $A$ and let $\rho, \rho' \in L^\infty(A)$. Let $\Pi_{\Sigma_\rho}(\rho) = t_0 \cdot a_1 \cdot t_1 \cdot a_2 \cdot t_2 \cdot \ldots$ and $\Pi_{\Sigma_{\rho'}}(\rho') = t'_0 \cdot a'_1 \cdot t'_1 \cdot a'_2 \cdot t'_2 \cdot \ldots$.

Suppose $\Pi_{\Sigma_{\rho}}(\text{Unt}(\rho)) = \Pi_{\Sigma_{\rho'}}(\text{Unt}(\rho'))$, that is, $a_1 = a'_1, a_2 = a'_2, \ldots$. Let $\text{date}(i, \rho)$ denote the absolute time that the observable event $a_i$ occurs, that is, $\text{date}(i, \rho) = \sum_{k=0}^{i-1} t_k$. Similarly, $\text{date}(i, \rho') = \sum_{k=0}^{i-1} t'_k$.

Given $\epsilon > 0$, we define the following predicate:

$$\text{close}_\epsilon(\rho, \rho') = \forall i. |\text{date}(a_i, \rho) - \text{date}(a_i, \rho')| \leq \epsilon.$$  

That is, $\text{close}_\epsilon(\rho, \rho')$ holds if the corresponding observable events in $\rho$ and $\rho'$ are not separated by more than $\epsilon$ time units.

Proposition 9 (Sufficient condition for digital-clock diagnosability) A is digital-clock-diagnosable if the following condition holds: there exists $\epsilon \in \mathbb{R}, \epsilon > 0$, such that for all $\rho, \rho' \in L^\infty(A)$, if $\rho$ is non-faulty, $\rho'$ is non-faulty and $\Pi_{\Sigma_\rho}(\text{Unt}(\rho)) = \Pi_{\Sigma_{\rho'}}(\text{Unt}(\rho'))$, then $\neg \text{close}_\epsilon(\rho, \rho')$. In particular, $A$ is $\frac{\epsilon}{2}$-diagnosable.

If the condition of Proposition 9 is true, then “sampling” with a period $\frac{\epsilon}{2}$ is sufficient to diagnose $A$. Intuitively, this is because for every two behaviors that yield identical observations on $\Sigma_n$, there will be a tick that “separates” the observable events in the two behaviors, thus allowing to distinguish them.
Figure 6. The condition of Proposition 9 is not necessary for digital-clock diagnosability.

The condition of Proposition 9 is sufficient but not necessary. Indeed, consider the example of Figure 6, which is a slight modification of the example of Figure 5. This automaton is digital-clock diagnosable: it suffices to take a digital-clock that produces \( \text{tick} \) at time 1. Then, if \( a \cdot \text{tick} \) is observed, the diagnoser knows that no fault occurred; if \( \text{tick} \cdot a \) is observed, a fault occurred. However, the condition of Proposition 9 does not hold for this example. Indeed, for any \( \epsilon > 0 \), we can take \( \rho = \tau \cdot (1 - \frac{1}{2}) \cdot a \) and \( \rho' = f \cdot (1 + \frac{1}{2}) \cdot a \), such that \( \epsilon \cdot (\rho, \rho') \).

5. Conclusions and perspectives

We have studied monitoring and fault-diagnosis problems for real-time systems, where observers only have access to digital (i.e., finite-precision) clocks. We have presented a framework where digital clocks are modeled as timed automata, and so are specifications (for monitoring) or plants (for fault-diagnosis). We have shown how sound and optimal monitors can be automatically synthesized, given specification and digital-clock models. We have also shown how to check diagnosability and, in case it holds, automatically synthesize a diagnoser, for given plant and digital-clock models. Finally, we have shown that there are cases where no digital clock, no matter how precise, can be used to diagnose a plant, even though the latter is diagnosable with an analog-clock.

An interesting question remains, namely, whether the problem of checking existence of such a digital clock is decidable. Another research direction is to study controller synthesis with digital-clock controllers.

References