Predictability of Event Occurrences in Timed Systems

Franck Cassez[1] and Alban Grastien[2]

[1] NICTA* and UNSW, Sydney
[2] NICTA and ANU, Canberra
Australia

Abstract. We address the problem of predicting events' occurrences in partially observable timed systems modelled by timed automata. Our contribution is manyfold: 1) we give a definition of bounded predictability, namely k-predictability, that takes into account the minimum delay between the prediction and the actual event's occurrence; 2) we show that 0-predictability is equivalent to the original notion of predictability of S. Genc and S. Lafortune; 3) we provide a necessary and sufficient condition for k-predictability (which is very similar to k-diagnosability) and give a simple algorithm to check k-predictability; 4) we address the problem of predictability of events' occurrences in timed automata and show that the problem is PSPACE-complete.

1 Introduction

Monitoring and fault diagnosis aim at detecting defects that can occur at run-time. The monitored system is partially observable but a formal model of the system is available which makes it possible to build (offline) a monitor or a diagnoser. Monitoring and fault diagnosis for discrete event systems (DES) have been have been extensively investigated in the last two decades [1,2,3]. Fault diagnosis consists in detecting a fault *as soon as possible* after it occurred. It enables a system operator to stop the system in case something went wrong, or reconfigure the system to drive it to a safe state. *Predictability* is a strong version of diagnosability: instead of detecting a fault after it occurred, the aim is to *predict* the fault before its occurrence. This gives some time to the operator to choose the best way to stop the system or to reconfigure it.

In this paper, we address the problem of predicting event occurrences in partially observable timed systems modelled by timed automata.

The Predictability Problem. A timed automaton [4] (TA) generates a timed language which is a set of timed words which are sequences of pairs (event, time-stamp). Only a subset of the events generated by the system is observable. The objective is to predict occurrences of a particular event (observable or not) based on the sequences of

* NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

observable events. Automaton G, Fig. 1, is a timed version of the example of automaton G_1 of [5]. The set of observable events is $\{a, b, c\}$. We would like to predict event f without observing event d. First consider the untimed version of G by ignoring the

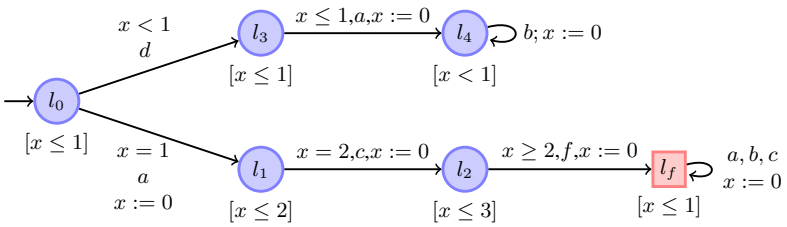


Fig. 1. Example G from [5].

constraints on clock x. The untimed automaton can generate two types of events' sequences: $d.a.b^*$ and $a.c.f.\{a, b, c\}^*$. Because d is unobservable, after observing a we do not know whether the system is in location l_4 or l_1 and cannot predict f as, according to our knowledge, it is not bound to occur in all possible futures from locations l_4 or l_1. However, after the next observable event, b or c, we can make a decision: if we observe $a.c$, G must be in l_2 and thus f is going to happen next. After observing $a.c$ we can predict event f. Note that there is no quantitative duration between occurrences of events in discrete event systems and thus we can predict f at a *logical* time which is before f occurs. The time that separates the prediction of f from the actual occurrence of f is measured in the number of discrete steps G can make. In this sense G is 0-predictable as when we predict f, it is the next event to occur. The untimed version of G is an abstraction of a real system, and in the real system, it could be that f is going to occur 5 seconds after we observe c.

Timed automata enable us to capture quantitative aspects of real-time systems. We can use *clocks* (like x) to specify constraints between the occurrences of events. Moreover *invariants* (like $[x \leq 1]$) ensure that G changes location when the upper bound of the invariant is reached. In the timed automaton G, the (infinite) sequences with no f are of the form $(d, \delta_d)(a, \delta_a)(b, \delta_b) \cdots$ with $\delta_d < 1, \delta_a \leq 1$ and $\delta_b < 2$. The sequences with event f are of the form $(a, 1)(c, 3)(f, \delta_f)$ with $5 \leq \delta_f \leq 6$. Thus if we do *not* observe a "b" within the first two time units, we know that the system is in location l_1. This implies that f is going to occur, and we know this at time 2. But f will not occur before $1 + 2$ time units, the time for c to occur (from time 2) and the minimum time for f to occur after c. G is thus 3-predictable. In the sequel we formally define the previous notions and give efficient algorithms to solve the predictability problem.

Related Work. Predictability for discrete event systems was first proposed by S. Genc and S. Lafortune in [6]. Later in [5] they gave two algorithms to decide the predictability problem, one of them is a polynomial decision procedure. T. Jéron, H. Marchand, S. Genc and S. Lafortune [7] extended the previous results to occurrences of *patterns* (of events) rather than a single event. L. Brandán Briones and A. Madalinski in [8]

studied *bounded* predictability without relating it to the notion defined by S. Genc and S. Lafortune.

Predictability is closely related to *fault diagnosis* [1,2,3]. The objective of fault diagnosis is to detect the occurrence of a special event, a fault, which is unobservable, as soon as possible after it occurs. Fault diagnosis for timed automata has first been studied by S. Tripakis in [9] and he proved that the diagnosis problem is PSPACE-complete. P. Bouyer, F. Chevalier and D. D'Souza [10] later studied the problem of computing a diagnoser with *fixed* resources (a deterministic TA) and proved that this problem is 2EXPTIME-complete. To the best of our knowledge the predictability problem for TA has not been investigated yet.

Our Contribution. We give a new characterization of bounded predictability and show it is equivalent to the definition of S. Genc and S. Lafortune. This new characterization is simple and dual to the one for the diagnosis problem; we can derive easily algorithms to decide predictability, bounded predictability, and to compute the largest anticipation delay to predict a fault. We also study the bounded predictability problem for TA and prove it is PSPACE-complete. We investigate implementability issues, i.e., how to build a *predictor*, and solve the *sampling predictability* problem which ensures an implementable predictor exists. We show how to compute bounded predictability with UPPAAL [11].

Organization of the Paper. The paper is organized as follows: the next section recalls some definitions: timed words, timed automata. Section 3 states the predictability problems for TA and Finite Automata (FA) and presents a necessary and sufficient condition for bounded predictability. Section 4 compares our definition of predictability with the original one (by S. Genc and S. Lafortune) and provides an algorithm (for finite automata) to solve the bounded predictability problem and compute the largest bound. Section 5 studies the bounded predictability problem for TA and implementation issues related to the construction of a predictor. An example is also solved with UPPAAL. Omitted proofs are given in Appendix.

2 Preliminaries

$\mathbb{B} = \{\text{TRUE}, \text{FALSE}\}$ 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 set of non-negative reals.

2.1 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 $\mathbb{R}^X_{\geq 0}$ 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 which is the set of conjunctions of constraints of the form $x \bowtie c$ with $c \in \mathbb{N}$ 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.

2.2 Timed Words

A finite (resp. infinite) *timed word* over Σ is a word in $\mathbb{R}_{\geq 0}.(\Sigma.\mathbb{R}_{\geq 0})^*$ (resp. $(\mathbb{R}_{\geq 0}.\Sigma)^\omega$). 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 events: thus c occurs at global time 4.1. We let $\mathit{Dur}(w)$ be 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 ∞. Note that the duration of an infinite word can be finite, and such words which still contain an infinite number of events, are called *Zeno* words. An infinite timed word w is *time-divergent* if $\mathit{Dur}(w) = \infty$. We let $\mathit{Unt}(w)$ be the *untimed* version of w obtained by erasing all the durations in w, e.g., $\mathit{Unt}(0.4\ a\ 1.0\ b\ 2.7\ c\ 0) = abc$. Given w a timed word and $a \in \Sigma$, $|w|_a$ is the number of occurrences of a in w (∞ if a occurs infinitely often in w.)

$\mathit{TW}^*(\Sigma)$ is the set of finite timed words over Σ, $\mathit{TW}^\omega(\Sigma)$, the set of infinite timed words and $\mathit{TW}^\infty(\Sigma) = \mathit{TW}^*(\Sigma) \cup \mathit{TW}^\omega(\Sigma)$. We use Σ^* and Σ^ω for the corresponding sets of untimed words. A *timed language* is any subset of $\mathit{TW}^\infty(\Sigma)$. For $L \subseteq \mathit{TW}^\infty(\Sigma)$, we let $\mathit{Unt}(L) = \{\mathit{Unt}(w) \mid w \in L\}$.

For $w \in \mathit{TW}^*(\Sigma)$ and $w' \in \mathit{TW}^\infty(\Sigma)$, $w.w'$ is the concatenation of w and w'. A finite timed word w is a prefix of $w' \in \mathit{TW}^\infty(\Sigma)$ if $w' = w.w''$ for some $w'' \in \mathit{TW}^\infty(\Sigma)$. In the sequel we also the prefix operator and $\overline{L}$ is the set of finite words that are prefixes of words in L.

Let $\Sigma_1 \subseteq \Sigma$. $\boldsymbol{\pi}_{/\Sigma_1}$ is the projection of timed words of $\mathit{TW}^\infty(\Sigma)$ over timed words of $\mathit{TW}^\infty(\Sigma_1)$. When projecting a timed word w on a sub-alphabet $\Sigma_1 \subseteq \Sigma$, the durations elapsed between two events are set accordingly: $\boldsymbol{\pi}_{/\{a,c\}}(0.4\ a\ 1.0\ b\ 2.7\ c) = 0.4\ a\ 3.7\ c$ (projection erases some events but preserves the time elapsed between the non-erased events). It follows that $\boldsymbol{\pi}_{/\Sigma_1}(w) = \boldsymbol{\pi}_{/\Sigma_1}(w')$ implies that $\mathit{Dur}(w) = \mathit{Dur}(w')$. For $L \subseteq \mathit{TW}^\infty(\Sigma)$, $\boldsymbol{\pi}_{/\Sigma_1}(L) = \{\boldsymbol{\pi}_{/\Sigma_1}(w) \mid w \in L\}$.

2.3 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 [4]. As usual we use the symbol ε to denote the silent (invisible) action in an automaton.

Definition 1 (Timed Automaton). *A* Timed Automaton A *is a tuple* $(L, l_0, X, \Sigma \cup \{\varepsilon\}, E, \mathit{Inv}, F, R)$ *where:* L *is a finite set of* locations*;* l_0 *is the* initial location*;* X *is a finite set of* clocks*;* Σ *is a finite set of* events*;* $E \subseteq L \times \mathcal{C}(X) \times \Sigma \cup \{\varepsilon\} \times 2^X \times L$ *is a finite set of* transitions*; for* $(\ell, g, a, r, \ell') \in E$*,* g *is the* guard*,* a *the* event*, and* r *the* reset *set;* $\mathit{Inv} : L \to \mathcal{C}(X)$ *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* ϱ of A from (ℓ_0, v_0) is a (finite or infinite) sequence of alternating *delay* and *discrete* moves:

$$\varrho = (\ell_0, v_0) \xrightarrow{\delta_0} (\ell_0, v_0 + \delta_0) \xrightarrow{a_1} (\ell_1, v_1) \cdots \xrightarrow{a_n} (\ell_n, v_n) \xrightarrow{\delta_n} (\ell_n, v_n + \delta_n) \cdots$$

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

- $v_i + \delta \models \mathit{Inv}(\ell_i)$ for $0 \leq \delta \leq \delta_i$ (Def. 1 implies that $v_i + \delta_i \models \mathit{Inv}(\ell_i)$ is equivalent);
- there is a transition $(\ell_i, g_i, a_{i+1}, r_i, \ell_{i+1}) \in E$ s.t. : (i) $v_i + \delta_i \models g_i$ and (ii) $v_{i+1} = (v_i + \delta_i)[r_i]$ (by the previous condition we have $v_{i+1} \models \mathit{Inv}(\ell_{i+1})$.)

If ϱ is finite and ends in s_n, we let $\mathit{tgt}(\varrho) = s_n$. We say that event $a \in \Sigma \cup \{\varepsilon\}$ is *enabled* in $s = (\ell, v)$, written $a \in \mathit{en}(s)$, if there is a transition $(\ell, g, a, R, \ell') \in E$ s.t. $v \models g$ and $v[R] \models \mathit{Inv}(\ell')$. 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 $\mathit{Runs}^*(A) = \mathit{Runs}^*((l_0, \mathbf{0}), A)$ and $\mathit{Runs}^\omega(A) = \mathit{Runs}^\omega((l_0, \mathbf{0}), A)$.

We make the following *boundedness* assumption on timed automata: time-progress in every location is bounded. This is not a restrictive assumption as every timed automaton that does not satisfy this requirement can be transformed into a language-equivalent one that is bounded [12]. This implies that every infinite run has an infinite number of events. We further assume[3] that every infinite run has an infinite number of discrete transitions with $a \neq \varepsilon$.

The *trace*, $\mathit{tr}(\varrho)$, of a run ϱ is the timed word $\delta_0 a_1 \delta_1 a_2 \cdots a_n \delta_n \cdots$ where ε is removed (and durations are updated accordingly). We let $\mathit{Dur}(\varrho) = \mathit{Dur}(\mathit{tr}(\varrho))$. For $V \subseteq \mathit{Runs}^*(A) \cup \mathit{Runs}^\omega(A)$, we let $\mathit{Tr}(V) = \{\mathit{tr}(\varrho) \mid \varrho \in V\}$.

A finite (resp. infinite) timed word w is *accepted* by A if $w = \mathit{tr}(\varrho)$ for some $\varrho \in \mathit{Runs}^*(A)$ that ends in an F-location (resp. for some $\varrho \in \mathit{Runs}^\omega(A)$ 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. In the sequel we often omit the sets R and F in TA and this implicitly means $F = L$ and $R = L$.

2.4 Product of Timed Automata

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

- *either $\sigma \neq \varepsilon$, 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) $r_{1,2} = r_1 \cup r_2$;*
- *or $\sigma = \varepsilon$ and for $k \in \{1, 2\}$, (i) $(\ell_k, g_k, \sigma, r_k, \ell_k') \in E_k$; (ii) $g_{1,2} = g_k$, (iii) $r_{1,2} = r_k$ and (iv) $\ell_{3-k}' = \ell_{3-k}$;*

$\mathit{Inv}(\ell_1, \ell_2) = \mathit{Inv}(\ell_1) \wedge \mathit{Inv}(\ell_2)$, $F = F_1 \times F_2$ and R is defined[4] such that $\mathcal{L}^\omega(A_1) \cap \mathcal{L}^\omega(A_2) = \mathcal{L}^\omega(A_1 \times A_2)$. ■

[3] Otherwise the trace of an infinite word can have a finite number of events in Σ but still infinite duration which cannot be defined in our setting. This is not a compulsory assumption and can be removed at the price of longer (not more complex) proofs.

[4] The product of Büchi automata requires an extra variable to keep track of the automaton that repeated its state. For the sake of simplicity we ignore this and assume the set R can be defined to ensure $\mathcal{L}^\omega(A_1) \cap \mathcal{L}^\omega(A_2) = \mathcal{L}^\omega(A_1 \times A_2)$.

2.5 Finite Automata

A finite automaton (FA) is a TA with $X = \varnothing$: guards and invariants are vacuously true and time elapsing transitions do not exist.

We write $A = (L, l_0, \Sigma \cup \{\varepsilon\}, E, F, R)$ for a FA. A run of a FA A is thus a sequence of the form: $\varrho = \ell_0 \xrightarrow{a_1} \ell_1 \cdots \quad \cdots \xrightarrow{a_n} \ell_n \cdots$ where for each $i \geq 0$, $(\ell_i, a_{i+1}, \ell_{i+1}) \in E$. Definitions of traces and languages are inherited from TA but the duration of a run ϱ is the number of steps (including ε-steps) of ϱ: if ϱ is finite and ends in ℓ_n, $\mathit{Dur}(\varrho) = n$ and otherwise $\mathit{Dur}(\varrho) = \infty$. The product definition also applies to finite automata.

3 Predictability Problems

Predictability problems are defined on partially observable TA. Given a TA $A = (L, \ell_0, X, \Sigma, E, \mathit{Inv}, L, L)$, $\Sigma_o \subseteq \Sigma$ a set of *observable* events, and a *bound* $\Delta \in \mathbb{N}$, we want to predict the occurrences of event $f \in \Sigma$ at least Δ time units before they occur. Without loss of generality, we assume 1) that the target location of the f-transitions is l_f, and they all reset a dedicated clock of A, x, which is only used on f-transitions; 2) A has transitions $(l_f, \mathrm{TRUE}, a, \{x\}, l_f)$ for every $a \in \Sigma_o$. We let $\mathit{Inv}(l_f) = x \leq 1$. In the remaining of this paper, Σ_o is fixed and we use $\boldsymbol{\pi}$ for $\boldsymbol{\pi}_{/\Sigma_o}$.

We again make the assumption that every infinite run of A contains infinitely many Σ_o events: this is not compulsory but simplifies some of the proofs.

3.1 Δ-Predictability

A run ρ of A is *non-faulty* if $\mathit{Unt}(\mathit{tr}(\rho))$ does not contain event f; otherwise it is *faulty*. We write $\mathit{NonFaulty}(s, A)$ for the non-faulty runs from s and define $\mathit{NonFaulty}(A) = \mathit{NonFaulty}((l_0, \mathbf{0}), A)$. Let $\varrho \in \mathit{NonFaulty}(A)$ be a finite non-faulty run:

$$\varrho = (l_0, v_0) \xrightarrow{\delta_0} (l_0, v_0 + \delta_0) \xrightarrow{a_1} (l_1, v_1) \cdots \xrightarrow{a_n} (l_n, v_n) \xrightarrow{\delta_n} (l_n, v_n + \delta_n).$$

ϱ is *Δ-prefaulty*, if it can be extended by a run ϱ' as follows:

$$\varrho'' = (l_0, v_0) \xrightarrow{\delta_0} \cdots \xrightarrow{\delta_n} \underbrace{\mathit{tgt}(\varrho) \xrightarrow{\delta'_0} s'_1 \xrightarrow{a'_1\delta'_1} \cdots \xrightarrow{a'_k\delta'_k} \cdots \xrightarrow{a'_j\delta'_j} s_j}_{\text{run } \varrho'}$$

where the extended run $\varrho'' \in \mathit{NonFaulty}(A)$ satisfies: (i) $f \in \mathit{en}(s_j)$ and (ii) $\mathit{Dur}(\rho') \leq \Delta$ (i.e., $\sum_{k=0}^{j} \delta'_k \leq \Delta$.) In words, f can occur within Δ time units from $\mathit{tgt}(\varrho)$. We let $\mathit{PreFaulty}_{\leq\Delta}(A)$ be the set of Δ-prefaulty runs of A. Note that if $\Delta \leq \Delta'$ then $\mathit{PreFaulty}_{\leq 0}(A) \subseteq \mathit{PreFaulty}_{\leq\Delta}(A) \subseteq \mathit{PreFaulty}_{\leq\Delta'}(A)$.

We want to predict the occurrence of event f at least Δ time units before it occurs and it makes sense only if $\Delta \leq \kappa(A)$ where $\kappa(A)$ is the minimum duration to reach a state where f is enabled. If f is never enabled, we let $\kappa(A) = \infty$. If $\kappa(A)$ is finite, let $0 \leq \Delta \leq \kappa(A)$ and define the following timed languages:

$$L^{\omega}_{\neg f} = \mathcal{L}^{\omega}(A) \cap \mathit{Tr}(\mathit{NonFaulty}(A)) \tag{1}$$

$$L_f^{-\Delta} = \mathit{Tr}(\mathit{PreFaulty}_{\leq\Delta}(A)). \tag{2}$$

If $\kappa(A) = \infty$ then we let $L_f^{-\Delta} = \varnothing$. $L^\omega_{\neg f}$ contains the infinite non-faulty traces of A. $L_f^{-\Delta}$ contains the finite traces w of A that can be extended into $w.x.f$ with f occurring less then Δ time units after w.

A Δ-*Predictor* is a device that predicts the occurrence of f at least Δ time units before it occurs. It should do it observing only the projection $\boldsymbol{\pi}(w)$ of the current trace w. Thus for every word $w \in L_f^{-\Delta}$, the predictor predicts f by issuing a 1. On the other hand, if a trace w can be extended as an infinite trace without any event f, i.e., it is in $\overline{L^\omega_{\neg f}}$, the predictor must not predict f and thus should issue a 0. For a trace which is in $L_f^{-\Delta'}$ with $\Delta' > \Delta$ and not in $\overline{L^\omega_{\neg f}}$, we do not require anything from the predictor: it can predict f or not and this is why we define a predictor as a partial mapping.

Definition 3 (Δ-Predictor). *A* Δ-predictor *for A is a partial mapping $P : TW^*(\Sigma_o) \longrightarrow \{0, 1\}$ such that:*

- $\forall w \in L_f^{-\Delta}, P(\boldsymbol{\pi}(w)) = 1$,
- $\forall w \in \overline{L^\omega_{\neg f}}, P(\boldsymbol{\pi}(w)) = 0$.

A is Δ-predictable *if there exists a Δ-predictor for A and is* predictable *if there is some Δ such that A is Δ-predictable.* ■

It follows that if f is never enabled in A, A is Δ-predictable for any Δ: a predictor is a mapping $P(\cdot) = 0$. In the sequel we assume that A contains a state where f is enabled and thus $\kappa(A)$ is finite.[5]

In the dual problem of *diagnosability* [9], it is required that the infinite words in $L^\omega_{\neg f}$ be *non-Zeno*. This is required by the problem statement that time must advance beyond any bound. For predictability, this is not a requirement and we could accept non time-divergent runs in $L^\omega_{\neg f}$. However for realistic systems we should add this requirement. This can be easily done and we discuss how to do this in section 5.2.

3.2 PSPACE-Hardness of Bounded Predictability

We are interested in the two following problems:

Problem 1 (Δ-Predictability (Bounded Predictability))
INPUT: *A TA $A = (L, \ell_0, X, \Sigma, E, Inv)$ and $\Delta \in \mathbb{N}$.*
PROBLEM: *Is A Δ-predictable?*

Problem 2 (Predictability)
INPUT: *A TA $A = (L, \ell_0, X, \Sigma, E, Inv)$.*
PROBLEM: *Is A predictable?*

Notice that predictability problems for finite automata are defined using the number of steps in the automaton A (including unobservable steps) for the duration of a run. A first result is the PSPACE-hardness of the Bounded Predictability problem. This is obtained

[5] Checking whether a state where f is enabled is reachable and the computation of $\kappa(A)$ can be done in PSPACE [13] for TA and linear time for FA.

by reducing the *reachability problem* for TA to the Bounded Predictability problem. The *location reachability problem* for TA asks, given a location l, whether (l, v) (for some valuation v) is reachable from the initial state of A. This problem is PSPACE-complete for TA [4].

Theorem 1. *The Bounded Predictability problem is PSPACE-hard for TA.*

Proof. We can reduce the location reachability problem for bounded TA to the predictability problem as follows (the reduction is similar to [9]): let A be a bounded TA and l a location of A. We can build A' by adding transitions to A: let END by a new location. We add a transition $(l, \text{TRUE}, f, \{x\}, \text{END})$, and another one $(l, \text{TRUE}, u, \{x\}, \text{END})$ with u unobservable, assuming A has at least one clock x. We then add loops on location END $(\text{END}, x = 1, a, \{x\}, \text{END})$, for each $a \in \Sigma$. Moreover $\mathit{Inv}(\text{END}) = x \leq 1$. It follows from our definition of predictability that l is reachable in A iff A' is not predictable, and A' has size polynomial in A.

3.3 Necessary and Sufficient Condition for Δ-Predictability

We now give a necessary and sufficient condition (NSC) for Δ-predictability which is similar in form to the condition used for Δ-*diagnosability* [9].

Lemma 1. *A is Δ-predictable iff $\boldsymbol{\pi}(L_f^{-\Delta}) \cap \boldsymbol{\pi}(\overline{L_{\neg f}^{\omega}}) = \varnothing$.*

Proof. Only If. Assume A is Δ-predictable. There exists a partial mapping P s.t. $\forall w \in L_f^{-\Delta}, P(\boldsymbol{\pi}(w)) = 1, \forall w \in \overline{L_{\neg f}^{\omega}}, P(\boldsymbol{\pi}(w)) = 0$. Assume $w \in \boldsymbol{\pi}(L_f^{-\Delta}) \cap \boldsymbol{\pi}(\overline{L_{\neg f}^{\omega}}) \neq \varnothing$. Then $w = \boldsymbol{\pi}(w_1) = \boldsymbol{\pi}(w_2)$ with $w_1 \in L_f^{-\Delta}$ and $w_2 \in \overline{L_{\neg f}^{\omega}}$. By definition of P we must have $P(w) = P(\boldsymbol{\pi}(w_1)) = 1$ and $P(w) = P(\boldsymbol{\pi}(w_2)) = 0$ which is a contradiction.
If. If $\boldsymbol{\pi}(L_f^{-\Delta}) \cap \boldsymbol{\pi}(\overline{L_{\neg f}^{\omega}}) = \varnothing$ define $P(w) = 1$ if $w \in \boldsymbol{\pi}(L_f^{-\Delta})$ and $P(w) = 0$ otherwise. If P does not exist, we must have $w = \boldsymbol{\pi}(w_1) = \boldsymbol{\pi}(w_2)$ with $w_1 \in L_f^{-\Delta}$ and $w_2 \in \overline{L_{\neg f}^{\omega}}$. In this case $w \in \boldsymbol{\pi}(L_f^{-\Delta}) \cap \boldsymbol{\pi}(\overline{L_{\neg f}^{\omega}})$ which is a contradiction. □

From Lemma 1 we can prove the following Proposition and Theorem:

Proposition 1. *if $\Delta \leq \Delta'$ and A is Δ'-predictable, then A is Δ-predictable.*

Proof. $L_f^{-\Delta} \subseteq L_f^{-\Delta'}$ and thus $\boldsymbol{\pi}(L_f^{-\Delta}) \cap \boldsymbol{\pi}(\overline{L_{\neg f}^{\omega}}) \subseteq \boldsymbol{\pi}(L_f^{-\Delta'}) \cap \boldsymbol{\pi}(\overline{L_{\neg f}^{\omega}})$. □

Theorem 2. *A is predictable iff A is 0-predictable.*

In the next section, we focus on the Δ-predictability problem for finite automata and discuss how it generalizes the previous notion introduced by S. Genc and S. Lafortune in [5]. Section 5 tackles the Δ-predictability problem for TA.

4 Predictability for Discrete Event Systems

In this section, we address the predictability problems for discrete event systems specified by FA. We first show that the definition of predictability (Def. 3) we introduced in Section 3 is equivalent to the original definition of predictability by S. Genc and S. Lafortune in [5].

4.1 Original Definition of Predictability (S. Genc and S. Lafortune)

Let $L_f = Tr(PreFaulty_{\leq 0}(A))$ be the set of non-faulty traces that can be extended with a fault in one step, and $L_{\neg f} = \overline{Tr(NonFaulty(A))}$ be the set of finite prefixes of non-faulty traces. S. Genc and S. Lafortune originally defined predictability for discrete event systems in [5] and we refer to GL-predictability for this definition. GL-predictability is defined as follows[6]:

$$\exists n \in \mathbb{N}, \forall w \in L_f, \exists t \in \overline{w} \text{ such that } \mathbf{P}(t) \tag{3}$$

with $\mathbf{P}(t)$ defined by:

$$\mathbf{P}(t) : \forall u \in L_{\neg f}, \forall v \in \mathcal{L}(A)/u, \boldsymbol{\pi}(u) = \boldsymbol{\pi}(t) \wedge |v| \geq n \implies |v|_f > 0.$$

According to [5], A is GL-predictable iff Equation (3) is satisfied. GL-predictability as defined by Equation (3) is equivalent to our notion of predictability:

Theorem 3. *A is GL-predictable iff A is 0-predictable.*

4.2 Checking k-Predictability

To check whether A is k-predictable, $0 \leq k \leq \kappa(A)$, we can use the NSC we established in Lemma 1: A is k-predictable iff $\boldsymbol{\pi}(L_f^{-k}) \cap \boldsymbol{\pi}(\overline{L_{\neg f}^{\omega}}) = \varnothing$. To check this condition, it suffices to build a *twin plant* (similar to [5] and to what is defined for fault diagnosis [2]). We define two automata $A_1(k)$ and A_2 that accept $\boldsymbol{\pi}(L_f^{-k})$ and $\boldsymbol{\pi}(\overline{L_{\neg f}^{\omega}})$ and synchronize them to check whether the intersection is empty. The first automaton $A_1(k)$ accepts finite words which are in $\boldsymbol{\pi}(L_f^{-k})$ and is defined as follows:

1. in A, we compute the set of states F_k that can reach a state where f is enabled within k steps (this can be done in linear time using a backward breadth-first search from states where f is enabled.)
2. $A_1(k)$ is a copy of A where the set of final states is F_k, and every $a \notin \Sigma_o$ is replaced by ε.

It follows that $A_1(k)$ accepts $\boldsymbol{\pi}(L_f^{-k})$.

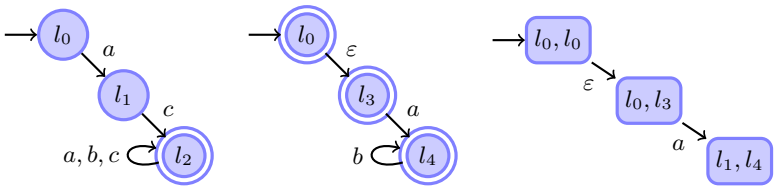
The second automaton A_2 accepts $\boldsymbol{\pi}(\overline{L_{\neg f}^{\omega}})$. To compute it, we merely need to compute the states from which there is an infinite path without any state where f is enabled. This can be done in linear time again (e.g., computing the states that satisfy the CTL formula $\mathsf{EG}\neg en(f)$.) A_2 is defined as follows:

1. let $F_{\neg f}$ be the set of states in A from which there exists an infinite path with no states where f is enabled.
2. A_2 is a copy of A restricted to the set of states $F_{\neg f}$, and every $a \notin \Sigma_o$ is replaced by ε (this implies that the target state of the f transitions cannot be in A_2).

[6] Technically S. Genc and S. Lafortune let w range over $L_f.f$ and impose that $|t|_f = 0$; the definition we give in Equation (3) is equivalent to Definition 1 of [5].

From the previous construction with sets of accepting states F_k for $A_1(k)$ and $F_{\neg f}$ for A_2 (every state in A_2 is accepting), $\mathcal{L}^*(A_1(k) \times A_2) = \boldsymbol{\pi}(L_f^{-k}) \cap \boldsymbol{\pi}(\overline{L_{\neg f}^{\omega}})$ and we can check k-predictability in quadratic time in the size of A.

Example 1. For the untimed version of Automaton G (Fig. 1, page 2), we obtain $G_1(0)$ and G_2 as depicted on Fig. 2. Recall that d is unobservable.



(a) $G_1(0)$ (b) G_2 (c) $G_1(0) \times G_2$

Fig. 2. Construction of G_1 and G_2 for automaton G ((Fig. 1)

Computing the largest k such that A is k-predictable can also be done in quadratic time. In A, we can compute, in linear time[7], the shortest distance $d_f(q)$ (going backwards) from q to a state where f is enabled (it is ∞ if q is unreachable going backwards in A). In the product $A_1(k) \times A_2$, if there is a run from the initial state to (s_1, s_2) and $d(s_1) = k', k' \leq k$, this implies that A is not k'-predictable. To determine the largest k such that A is k-predictable, it suffices to perform the following steps:

1. compute the shortest distance $d_f(q)$ to an f-enabled state for each $q \in Q$;
2. build the product $A_1(0) \times A_2$;
3. let S be the set of reachable states in $A_1(0) \times A_2$ and $M = \min_{(s_1,s_2)\in S} d_f(s_1)$.

The largest k such that A is k-predictable is $M - 1$.

Example 2. On automaton G of Fig. 1: $d(l_2) = 0$, $d(l_1) = 1$, $d(l_0) = 2$, $d(l_3) = d(l_4) = \infty$. The minimum value reachable in $G_1(0) \times G_2$ is obtained for l_1 and is $d(l_1) = 1$. Thus G is 0-predictable.

5 Predictability for Timed Automata

In this section we address the predictability problems for TA. We first rewrite the NSC of Lemma 1 using infinite languages. This enables us 1) to deal with time-divergent runs and 2) to design an algorithm to solve the predictability problems for TA.

[7] e.g., standard breadth-first search [14] on A.

5.1 Checking Δ-Predictability

We can reformulate Lemma 1 without the prefix operator by extending $L_f^{-\Delta}$ into an equivalent language of infinite words: let $L_f^{\omega,-\Delta} = L_f^{-\Delta}.(\Sigma_o \times \mathbb{R}_{\geq 0})^\omega$.

Lemma 2. $\boldsymbol{\pi}(L_f^{-\Delta}) \cap \boldsymbol{\pi}(\overline{L_{\neg f}^{\omega}}) = \varnothing \iff \boldsymbol{\pi}(L_f^{\omega,-\Delta}) \cap \boldsymbol{\pi}(L_{\neg f}^{\omega}) = \varnothing.$

To check Δ-predictability we build a product of timed automata $A_1(\Delta) \times A_2$, and reduce the problem to Büchi emptiness on this product. This construction is along the lines of the *twin plant* introduced in [2,9]. The difference in the predictability problem lies in the construction of $A_1(\Delta)$ which is detailed later. The twin plant idea is the following:

- $A_1(\Delta)$ accepts $\boldsymbol{\pi}(L_f^{\omega,-\Delta})$ i.e., (projections of) infinite timed words of the form $w.(\mathbb{R}_{\geq 0} \times \Sigma_o)^\omega$ with $w \in L_f^{-\Delta}$;
- A_2 accepts $\boldsymbol{\pi}(L_{\neg f}^{\omega})$ i.e., (projections of) infinite non-faulty timed words in $L_{\neg f}^{\omega}$;
- the product $A_1(\Delta) \times A_2$ accepts the language $\boldsymbol{\pi}(L_f^{\omega,-\Delta}) \cap \boldsymbol{\pi}(L_{\neg f}^{\omega})$;
- thus checking Δ-predictability of A reduces to Büchi emptiness checking on the product $A_1(\Delta) \times A_2$.

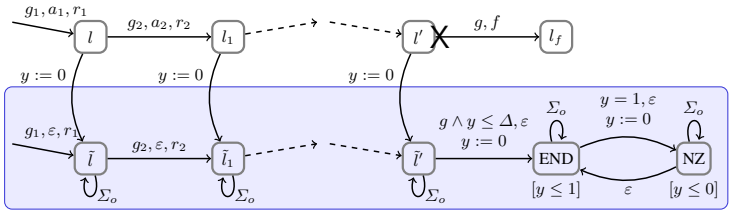


Fig. 3. Construction of Automaton A_1

$A_1(\Delta)$ itself is made of two copies of A: the original A and a twin copy (see Fig. 3). A_1 starts in the initial location of A, ℓ_0, and at some point in time switches to the twin copy (grey area on Fig. 3). The purpose of the twin copy is to extend the previously formed timed word with a timed word of duration less than Δ time units that reaches a state where f is enabled. The actions performed in the copy do not matter as we only have to check that f is reachable within Δ time units since we switched to the copy. In this case the timed word built in the original A is in $L_f^{-\Delta}$.

$A_1(\Delta) = (L \cup \tilde{L} \cup \{\text{END}\}, l_0^1, X \cup \{y\}, \Sigma \cup \{\varepsilon\}, E_1, \mathit{Inv}_1, \varnothing, \{\text{END}\})$ is formally defined as follows[8] (see Fig. 3):

- $\tilde{L} = \{\tilde{\ell}, \ell \in L\}$ is the set of twin locations;

[8] For now ignore the NZ location in the Figure and the invariants $[y \leq k]$. Their sole purposes is to ensure time-divergence.

- $l_0^1 = l_0$; $A_1(\Delta)$ starts in the same initial state as A.
- $\mathit{Inv}_1(\ell) = \mathit{Inv}_1(\tilde{\ell}) = \mathit{Inv}(\ell)$; invariants are the same as in the original automaton A including the twin locations;
- the transition relation is defined as follows:
  - original transitions of A: $(\ell, g, a', R, \ell') \in E_1$ iff $(\ell, g, a, R, \ell') \in E$ and $a \in \Sigma_o \setminus \{f\}$; $a' = a$ if $a \in \Sigma_o$ and $a' = \varepsilon$ otherwise; this renaming hides the unobservable events by renaming them in ε.
  - transitions to the twin locations: $(\ell, \text{TRUE}, \varepsilon, \{y\}, \tilde{\ell}) \in E_1$ for each $\ell \in L$; A_1 can switch to the twin copy at any time and doing so preserves the values for the clocks in X but resets y;
  - equivalent unobservable transitions inside the twin copy: $(\tilde{\ell}, g, \varepsilon, R, \tilde{\ell}_1) \in E_1$ iff $(\ell, g, a, R, \ell_1) \in E$ for some $a \neq f$;
  - equivalent of f-transitions in the twin copy: $(\tilde{\ell}', g \wedge y \leq \Delta, \varepsilon, R, \text{END}) \in E_1$ iff $(\ell', g, f, R, l_f) \in E$.
  - loop transitions on observable events in the twin copy: $(\tilde{\ell}, \text{TRUE}, a, \varnothing, \tilde{\ell}) \in E_1$ for each $a \in \Sigma_o$. This enables A_2 (defined below) to synchronize with A_1 on Σ_o after A_1 has chosen to switch to the twin copy of A.

Finally, A_2 is simply of copy of A without the f-transitions and the clocks are renamed to be local to A_2. Every location in A_2 is a *repeated* location. Notice that the only repeated location in $A_1(\Delta)$ is END. By definition of the synchronized product, $\mathcal{L}^\omega(A_1(\Delta) \times A_2) = \mathcal{L}^\omega(A_1(\Delta)) \cap \mathcal{L}^\omega(A_2)$.

Lemma 3. $\boldsymbol{\pi}(L_f^{\omega,-\Delta}) \cap \boldsymbol{\pi}(L_{\neg f}^\omega) = \mathcal{L}^\omega(A_1(\Delta) \times A_2)$.

Theorem 4. *Problems 1 and 2 are PSPACE-complete.*

Proof. PSPACE-easiness of Problem 1 is established as follows: checking Büchi emptiness for timed automata is in PSPACE [4]. The product $A_1(\Delta) \times A_2$ has size polynomial in the size of A and thus checking Büchi emptiness of the product is in PSPACE as well. Problem 1 is thus in PSPACE. By Theorem 2, Problem 2 is in PSPACE as well.

Theorem 1 states PSPACE-hardness for Problem 1. As 0-predictability i.e., Problem 1, is equivalent to Problem 2, it is PSPACE-hard as well. □

5.2 Restriction to Time-Divergent Runs of $L_{\neg f}^\omega$

To deal with time-divergence and enforce the runs in $L_{\neg f}^\omega$ to have infinite duration (see Remark **??**), we can add another automaton in the product with a Büchi condition that enforces time-divergence (this is how this kind of requirements is usually addressed). In our setting, we can re-use the fresh clock y of $A_1(\Delta)$ after location END is visited: it is not useful anymore to check whether a timed word is in $L_f^{-\Delta}$. The modifications to $A_1(\Delta)$ required to ensure time-divergence in A_2 are the following:

- add a new location NZ, which is now the repeated location of $A_1(\Delta)$;
- add two transitions as depicted on Fig. 3 between END and NZ.

This way infinite timed words accepted by $A_1(\Delta)$ must be time-divergent and with the synchronization with A_2 this forces the runs of A_2 to be time-divergent.

Finally, once we know how to solve Problem 1, we can compute the optimal (maximum) anticipation delay by performing a binary search on the possible values of $0 \leq \Delta \leq \kappa(A)$.

5.3 Implementability of the Δ-Predictor

In the previous sections, we defined a predictor as a mapping from timed words to $\{0, 1\}$. To build an implementation of this mapping (an actual predictor) we still have some key problems to address: 1) we have to recognize when a timed word is in $L_f^{-\Delta}$; and 2) we have to detect that a timed word is in $L_f^{-\Delta}$ as soon as possible. S. Tripakis addressed similar problems in [9] in the context of fault diagnosis where a *diagnoser* is given as an algorithm that computes a state estimate of the system after reading a timed word w. The diagnoser updates its status after the occurrence of an observable event or after a *timeout* (TO) has occurred, which means some timed elapsed since the last update and no observable event occurred. The value of the timeout period (TO) is required to be less than the minimum delay between two observable events to ensure that the diagnoser works as expected. However, point 2) above still poses problem in our context, as demonstrated by the TA $\mathcal{B}$ of Fig. 4.
The set of observable events is $\{a\}$ and $\mathcal{B}$ is 4-predictable. To see this, define the predictor P as follows: for a timed word $w = \delta.w'$ with $\delta \geq 2$, $P(w) = 1$ and otherwise $P(w) = 0$. Indeed if 2 time units elapse and we see no observable events, for sure the system is still is l_0 and thus a fault f is bound to happen, but not before 4 time units. An implementation of a 4-predictor has to observe the state of the system *exactly* at time 2 otherwise it cannot predict the fault 4 time units in advance.

Now assume the platform on which we implement the predictor can make an observation every $\frac{3}{5}$ time units. The first observation of the predictor occurs at time $\frac{3}{5}$; the third at $\frac{9}{5}$ and we cannot predict the fault as we still don't know whether the system is in l_0 or has made a silent move to l_1. The next observation is at $\frac{12}{5}$: if we have seen no a so far, for sure the system is in l_0 and we can predict the fault. However the fault may now occur in $\frac{18}{5}$ time units i.e., less than 4 time units from the current time. Such a platform cannot implement the 4-predictor.

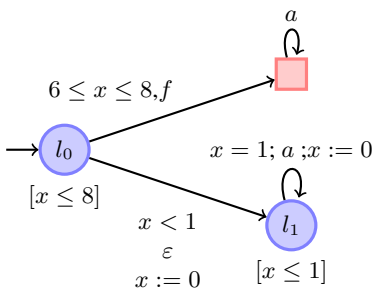


Fig. 4. The Timed Automaton $\mathcal{B}$

The maximal anticipation delay we computed in the previous section is thus an *ideal* maximum that can be achieved by an *ideal* predictor that could monitor the system *continuously*. In a realistic system, there is a *sampling rate*, or at least a minimum amount of time between two observations [15]. In the sequel we address the *sampling predictability problem* that takes into account the speed of the platform.

5.4 Sampling Predictability

Let $\alpha \in \mathbb{Q}$ and L be a timed language. We let $L \bmod \alpha$ be the set of timed words in L with a duration multiple of α: $L \mod \alpha = \{w \in L, \exists k \in \mathbb{N}, \mathit{Dur}(w) = k \cdot \alpha\}$.

Given a *sampling rate* $\alpha \in \mathbb{Q}$, the *sampling* predictability problem is defined by refining the definition of a Δ-predictor: an (α, Δ)-*predictor* for A is a partial mapping $P : \mathit{TW}^*(\Sigma_o) \bmod \alpha \longrightarrow \{0, 1\}$ such that:

- $\forall w \in L_f^{-\Delta} \bmod \alpha, P(\boldsymbol{\pi}(w)) = 1$,
- $\forall w \in \overline{L_{\neg f}^{\omega}} \bmod \alpha, P(\boldsymbol{\pi}(w)) = 0$.

A timed automaton A is (α, Δ)-predictable if there exists a (α, Δ)-predictor for A and is α-predictable is there is some Δ such that A is (α, Δ)-predictable.

Remark 1. The problem of deciding whether there exists a sampling rate α such that A is α-predictable is also interesting but very likely to be undecidable as the existence of a *sampling rate* s.t. a location is reachable in a TA is undecidable [16].

The solution to the sampling predictability problem is a simple adaptation of the solution we presented in Section 5: in the construction of automaton $A_1(\Delta)$ (Fig. 3, page 11), it suffices to restrict the transitions from the original A to the twin copy (those resetting y) to happen at time points multiple of α. This can be achieved by adding a *sampler* timed automaton, and a common fresh clock, s, that *sampler* resets every α time units. The transitions resetting y in A_1 are now guarded by $s = 0$.

We can now safely define an implementation for an (α, Δ)-predictor along the lines of the *diagnoser* defined in [9]. The implementation performs an observation every α time units. It computes a state estimate of the system. If one of the states in the state estimate can reach a state where f is enabled within Δ time units, the predictor predicts f and issue a 1. Otherwise it issues 0. Computing a representation of the state estimate as a set of polyedra is a standard operation and can be done given an observed timed word w, and the timed automaton model A. Checking that one of the states in the estimate can reach an f-enabled state within Δ time units can also be done using standard reachability algorithm. It can be performed on-line or off-line by computing a polyedral representation of this set of states.

5.5 A Simple Example

The example of Fig. 4 can be analyzed using UPPAAL [11]. UPPAAL cannot check for Büchi emptiness but in this example there is no *Zeno* non-faulty behaviours; thus we can restrict to a sufficiently large horizon to check the condition of Lemma 2.

The construction of the product $\mathcal{B}_1(\Delta) \times \mathcal{B}_2$ defined in Section 5.1 for $\mathcal{B}$ is depicted on Fig. 5. Assume the sampling rate is $\alpha = \frac{q}{p}$. The rational rates must be encoded by scaling up the constants in a network of TA as UPPAAL only accepts integers to compare clocks against. We use the variables `qsRate` and `psRate` in the UPPAAL model for these two constants. To obtain a network of TA with integers, and *sampling rate* α, we multiply all the constants by p (this is standard in TA and scales up time such that one

time unit in the original automaton is p time units in the scaled up one). We add one automaton *sampler* that resets the clock s every q time units. The transitions in $\mathcal{B}_1(\Delta)$ that reset y are now guarded by $s = 0$ which implies there can only be taken at points in time which are multiples of q. As mentioned earlier we cannot check a Büchi condition with UPPAAL and replace it by a reachability condition on a sufficiently large horizon. Note also that the Δ (D in the UPPAAL model) is multiplied by q in the guard leading to END. Synchronization is realized with a broadcast channel for each observable event.

Given a value of D, the property we check is P: "Can we reach END in the product with global time larger than $M * p$"? $M = 10$ is enough for our example. If the answer is "yes" then the system is not $(D \cdot \frac{p}{q})$-predictable, otherwise it is.

For a sampling rate $\alpha = \frac{3}{5}$, we get as expected that the maximum D for which $\mathcal{B}$ is predictable is 6. Which means that the actual maximal anticipation delay is $\Delta = 6 \cdot \frac{3}{5} = \frac{18}{5}$ time units. And indeed, the first time we can check that more than 2 time units have elapsed is $\frac{12}{5}$ and thus an interval of $\frac{18}{5}$ before f can occur. If we set $\alpha = 1$ we get $D = \Delta = 4$ meaning we can ideally predict the fault 4 time units in advance.

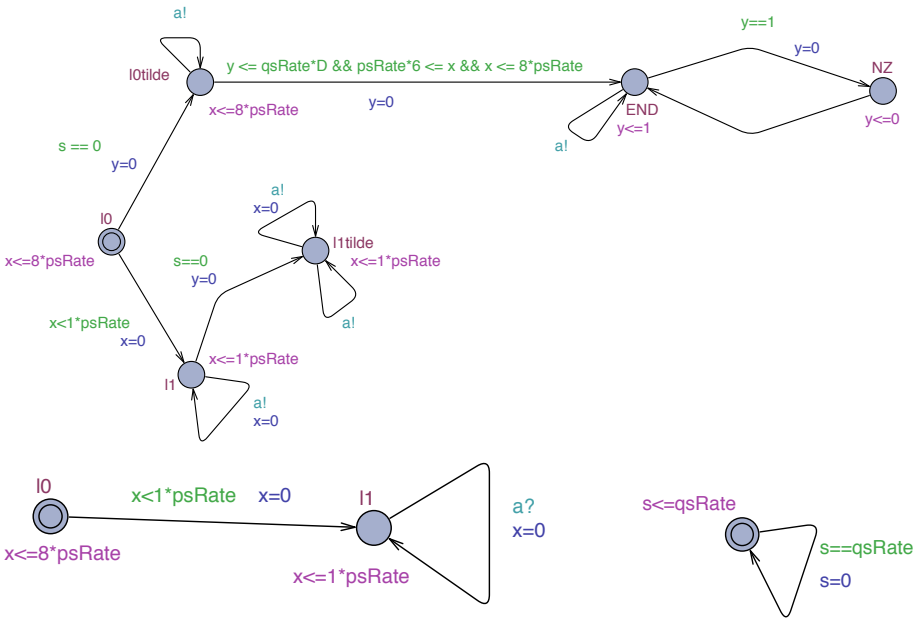


Fig. 5. UPPAAL Models for $\mathcal{B}$ of Fig. 4.

6 Conclusion and Future Work

In this paper we have proved some new results for predictability of events' occurrences for timed automata. We also contributed a new and simpler definition of bounded predictability for finite automata. The natural extensions of our work are as follows:

- in [10], P. Bouyer, F. Chevalier and D. D'Souza proposed an algorithm to decide the existence of a diagnoser with fixed resources (number of clocks and constants). The very same question arises for the existence of a predictor in timed systems.
- *dynamic* observers [17] have been proposed in the context of fault diagnosis and *opacity* [18]; in [19] it is shown how to compute a most permissive observer that ensures diagnosability (or opacity [20]) and also how to compute an optimal observer [21] (w.r.t. to a given criterion). We can define the same problems for predictability.
- given the similarities between the fault diagnosis and predictability problems, it would be interesting to state these two problems in a similar and unified way and design an algorithm that can solve the unified version.

References

1. Sampath, M., Sengupta, R., Lafortune, S., Sinnamohideen, K., Teneketzis, D.: Diagnosability of discrete event systems. IEEE Transactions on Automatic Control **40**(9) (September 1995)
2. Yoo, T.S., Lafortune, S.: Polynomial-time verification of diagnosability of partially-observed discrete-event systems. IEEE Transactions on Automatic Control **47**(9) (September 2002) 1491–1495
3. Jiang, S., Huang, Z., Chandra, V., Kumar, R.: A polynomial algorithm for testing diagnosability of discrete event systems. IEEE Transactions on Automatic Control **46**(8) (August 2001)
4. Alur, R., Dill, D.: A theory of timed automata. Theoretical Computer Science **126** (1994) 183–235
5. Genc, S., Lafortune, S.: Predictability of event occurrences in partially-observed discrete-event systems. Automatica **45**(2) (2009) 301–311
6. Genc, S., Lafortune, S.: Predictability in discrete-event systems under partial observation. In: IFAC Symposium on Fault Detection, Supervision and Safety of Techical Processes, Beijing, China, IEEE (2006)
7. Jéron, T., Marchand, H., Genc, S., Lafortune, S.: Predictability of sequence patterns in discrete event systems. In: IFAC World Congress, Seoul, Korea (July 2008) 537–453
8. Brandán Briones, L., Madalinski, A.: Bounded predictability for faulty discrete event systems. In: 30th International Conference of the Chilean Computer Science Society (SCCC-11). (2011)
9. Tripakis, S.: Fault diagnosis for timed automata. In Damm, W., Olderog, E.R., eds.: Proceedings of the International Conference on Formal Techniques in Real Time and Fault Tolerant Systems (FTRTFT'02). Volume 2469 of LNCS., Springer (2002) 205–224
10. Bouyer, P., Chevalier, F., D'Souza, D.: Fault diagnosis using timed automata. In Sassone, V., ed.: FoSSaCS. Volume 3441 of LNCS., Springer (2005) 219–233
11. Larsen, K.G., Pettersson, P., Yi, W.: Uppaal in a nutshell. STTT **1**(1-2) (1997) 134–152

12. Behrmann, G., Fehnker, A., Hune, T., Larsen, K.G., Pettersson, P., Romijn, J., Vaandrager, F.W.: Minimum-cost reachability for priced timed automata. In Benedetto, M.D.D., Sangiovanni-Vincentelli, A.L., eds.: HSCC. Volume 2034 of LNCS., Springer (2001) 147–161
13. Courcoubetis, C., Yannakakis, M.: Minimum and maximum delay problems in real-time systems. Formal Methods in System Design **1**(4) (1992) 385–415
14. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms (3. ed.). MIT Press (2009)
15. Wulf, M.D., Doyen, L., Raskin, J.F.: Almost asap semantics: From timed models to timed implementations. In Alur, R., Pappas, G.J., eds.: HSCC. Volume 2993 of LNCS., Springer (2004) 296–310
16. Cassez, F., Henzinger, T.A., Raskin, J.F.: A Comparison of Control Problems for Timed and Hybrid Systems. In: Proc. of the Workshop on Hybrid Systems: Computation and Control (HSCC'02). Volume 2289 of LNCS., Springer (March 2002) 134–148
17. Cassez, F., Tripakis, S.: Fault diagnosis with static and dynamic diagnosers. Fundamenta Informaticae **88**(4) (November 2008) 497–540
18. Cassez, F., Dubreil, J., Marchand, H.: Synthesis of opaque systems with static and dynamic masks. Formal Methods in System Design **40**(1) (2012) 88–115
19. Cassez, F., Tripakis, S., Altisen, K.: 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) 90–99
20. Cassez, F., Dubreil, J., Marchand, H.: Dynamic observers for the synthesis of opaque systems. In Liu, Z., Ravn, A.P., eds.: ATVA. Volume 5799 of LNCS., Springer (2009) 352–367
21. Cassez, F., Tripakis, S., Altisen, K.: Synthesis of optimal-cost 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) 316–325

A Proof of Theorem 3

Proof. if Part. Assume there exists a 0-predictor P for A and Equation (3) does not hold. Then $\forall n, \exists w \in L_f, \forall t \in \overline{w}$, $\mathbf{P}(t)$ does not hold. Let $t = w^{-0} = w$. As $\mathbf{P}(t)$ does not hold: $\exists u \in L_{\neg f}, \exists v \in \mathcal{L}^*(A)/u, \boldsymbol{\pi}(u) = \boldsymbol{\pi}(t) \wedge |v| \geq n$ but $|v|_f = 0$. Assume we have $n \geq |Q|$, the number of states of A. Then v has a cycle (pumping Lemma) and can be written $v = x.y.z$ with $|x.y^j.z|_f = 0, x.y^j.z \in \mathcal{L}^*(A)/u, \forall j \geq 0$ and thus we can build $v' = x.y^\omega \in \mathcal{L}(A)^\omega/u$ s.t. $|v'|_f = 0$. It follows that $u\ v' \in \mathcal{L}^\omega_{\neg f}$. We have: 1) $w \in L_f^{-0}$, 2) $w' \in L^\omega_{\neg f}$. Moreover $\boldsymbol{\pi}(w^{-0}) = \boldsymbol{\pi}(t)$ and $P(\boldsymbol{\pi}(w^{-0})) = 1$ because P is a 0-predictor. But $\boldsymbol{\pi}(t) = \boldsymbol{\pi}(u)$ and $u \in \overline{L^\omega_{\neg p}}$ which entails $P(\boldsymbol{\pi}(u)) = 0$ which is a contradiction.
Only if. Assume Equation (3) holds. Define the mapping P as follows:

- $\forall w \in L_f^{-0}, P(\boldsymbol{\pi}(w)) = 1$ and
- $\forall w \in \overline{L^\omega_{\neg p}}$, $P(\boldsymbol{\pi}(w)) = 0$.

We can show that P is a 0-predictor i.e., it is well-defined. On the contrary assume there exists $r = \boldsymbol{\pi}(w_1)$ with $w_1 \in L_f^{-0}$ and $r = \boldsymbol{\pi}(w_2)$ with $w_2 \in \overline{L^\omega_{\neg p}}$. We can show that Equation (3) cannot hold which is a contradiction. Take $n \in \mathbb{N}$. $w_1 \in L_f^{-0} = L_f$ and, by Equation (3), there must exist $t \in \overline{w_1}$ s.t. $\mathbf{P}(t)$ holds. But we can exhibit two words $u \in L_{\neg p}$ and $v \in \mathcal{L}^*(A)/u$ that falsify $\mathbf{P}(t)$. $t \in \overline{w_1}$. As $\boldsymbol{\pi}(w_1) = \boldsymbol{\pi}(w_2)$, there exists $w_2' \in \overline{w_2}$ s.t. $\boldsymbol{\pi}(t) = \boldsymbol{\pi}(w_2')$. $w_2' \in \overline{L^\omega_f}$ because $w_2 \in \overline{L^\omega_f}$. Take $u = w_2'$ and $v \in L_{\neg f}/w_2'$ with $|v| \geq n$ (exists as $w_2' \in \overline{L^\omega_f}$). We have $\boldsymbol{\pi}(u) = \boldsymbol{\pi}(t)$, $v \in \mathcal{L}^*(A)/u$, $|v| \geq n$ but $|v|_f = 0$ which contradicts Equation (3). □

B Proof of Lemma 2

Proof. If. Assume $\boldsymbol{\pi}(L_f^{-\Delta}) \cap \boldsymbol{\pi}(\overline{L^\omega_{\neg f}}) \neq \varnothing$. Let $w \in \boldsymbol{\pi}(L_f^{-\Delta}) \cap \boldsymbol{\pi}(\overline{L^\omega_{\neg f}})$. Then $w = \boldsymbol{\pi}(w_1) = \boldsymbol{\pi}(w_2)$ with $w_1 \in L_f^{-\Delta}$ and $w_2 \in \overline{L^\omega_{\neg f}}$. Moreover there exists some $w_2' \in L^\omega_{\neg f}$ such that $w_2.w_2' \in L^\omega_{\neg f}$. It follows that $\boldsymbol{\pi}(w_2.w_2')$ is an infinite timed word because by assumption every infinite timed has an infinite number of events in Σ_o. By definition of $L_f^{\omega,-\Delta}$, $w_1.w_2' \in L_f^{\omega,-\Delta}$. Moreover[9] $\mathit{Dur}(w_1) = \mathit{Dur}(w_2)$ and $\boldsymbol{\pi}(w_1) = \boldsymbol{\pi}(w_2)$ and thus $\boldsymbol{\pi}(w_1.w_2') = \boldsymbol{\pi}(w_1).\boldsymbol{\pi}(w_2') = \boldsymbol{\pi}(w_2.w_2')$. This entails $\boldsymbol{\pi}(L_f^{\omega,-\Delta}) \cap \boldsymbol{\pi}(L^\omega_{\neg f}) \neq \varnothing$.
Only If. Now assume $w \in \boldsymbol{\pi}(L_f^{\omega,-\Delta}) \cap \boldsymbol{\pi}(L^\omega_{\neg f}) \neq \varnothing$. We have $w = \boldsymbol{\pi}(w_1.w_1') = \boldsymbol{\pi}(w_2)$ with $w_1 \in L_f^{-\Delta}$, $w_2 \in L^\omega_{\neg f}$. Let w_2' be a prefix of w_2 such that $\boldsymbol{\pi}(w_2') = \boldsymbol{\pi}(w_1)$ (such a prefix exists because $\boldsymbol{\pi}(w_1.w_1') = \boldsymbol{\pi}(w_2)$.) Then $w_2' \in \overline{L^\omega_{\neg f}}$ and $w_1 \in L_f^{-\Delta}$ and $\boldsymbol{\pi}(w_2') = \boldsymbol{\pi}(w_1)$ which entails that $\boldsymbol{\pi}(L_f^{-\Delta}) \cap \boldsymbol{\pi}(\overline{L^\omega_{\neg f}}) \neq \varnothing$. □

C Proof of Lemma 3

Proof. $\boxed{\supseteq}$ Let $w \in \mathcal{L}^\omega(A_1(\Delta) \times A_2) = \mathcal{L}^\omega(A_1(\Delta)) \cap \mathcal{L}^\omega(A_2)$. Then $w = \boldsymbol{\pi}(\mathit{tr}(\rho_1))$ with $\rho_1 \in \mathit{Runs}^\omega(A_1(\Delta))$ and $w = \boldsymbol{\pi}(\mathit{tr}(\rho_2))$ with $\rho_2 \in \mathit{Runs}^\omega(A_2)$. We can write

[9] The condition $\mathit{Dur}(w_1) = \mathit{Dur}(w_2)$ is only needed for TA. For FA, it does not hold but is not necessary to concatenate the words.

$tr(\rho_1) = w_1.w_1'.w_1''$ with $w_1 \in L_f^{-\Delta}$, $\mathit{Dur}(w_1') \leq \Delta$ and $w_1'' \in (\mathbb{R}_{\geq 0} \times \Sigma_o)^\omega$ by construction of A_1 and its accepting condition. It follows that $w \in \boldsymbol{\pi}(L_f^{\omega,-\Delta}) \cap \boldsymbol{\pi}(L_{\neg f}^{\omega})$.

$\boxed{\subseteq}$ Let $w \in \boldsymbol{\pi}(L_f^{\omega,-\Delta}) \cap \boldsymbol{\pi}(L_{\neg f}^{\omega})$. Fig. 6 depicts the following proof. We can write $w = \boldsymbol{\pi}(w_1.w_1^+)$ with $w_1 \in L_f^{-\Delta}$, $w_1^+ \in (\mathbb{R}_{\geq 0} \times \Sigma_o)^\omega$. We also have $w = \boldsymbol{\pi}(w_2.w_2^+)$ for some $w_2.w_2^+ \in L_{\neg f}^{\omega}$ and such that $\boldsymbol{\pi}(w_1) = \boldsymbol{\pi}(w_2)$ and $\boldsymbol{\pi}(w_1^+) = \boldsymbol{\pi}(w_2^+)$. Note also that $\boldsymbol{\pi}(w_2.w_2^+) \in \mathcal{L}^\omega(A_2)$ because we assume every infinite timed word has an infinite number of Σ_o actions. As $w_1 \in L_f^{-\Delta}$, we can split w_1^+ into $w_1^+ = w_1'.w_1''$ with

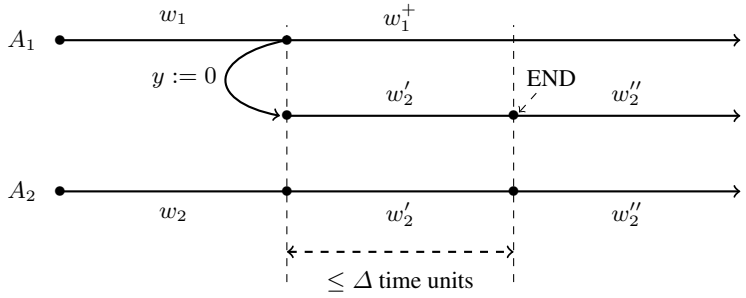


Fig. 6. Proof of Lemma 3, $\subseteq$.

$\mathit{Dur}(w_1') \leq \Delta$. We can split w_2^+ accordingly such that $w_2^+ = w_2'.w_2''$ and $\boldsymbol{\pi}(w_2') = \boldsymbol{\pi}(w_1')$ and $\boldsymbol{\pi}(w_2'') = \boldsymbol{\pi}(w_1'')$ and $\mathit{Dur}(w_2') = \mathit{Dur}(w_1') \leq \Delta$. Moreover $w_1.w_2'$ can be generated in $A_1(\Delta)$ as follows: start with w_1, after w_1 switch to the twin copy and reset y and generate w_2'. By playing w_1' in the original copy of A in $A_1(\Delta)$ we reach a state (l', v) where f is enabled: there is a transition $(l', g, f, R, l_f) \in E$ such that $v \models g$. By construction of $A_1(\Delta)$, playing w_2' in the twin copy in $A_1(\Delta)$ we reach an equivalent state $(\tilde{l}', v)$ and a twin transition $(\tilde{l}', g \wedge y \leq \Delta, \varepsilon, \{y\}, \text{END})$. As $\mathit{Dur}(w_2') \leq \Delta$, we must have $y \leq \Delta$ and this twin transition can be fired and END is reachable. We can subsequently read w_2'' in $A_1(\Delta)$. It follows that $\boldsymbol{\pi}(w_1.w_2^+) \in \mathcal{L}^\omega(A_1(\Delta))$ and $w \in \mathcal{L}^\omega(A_1(\Delta) \times A_2)$. $\square$