INSTITUTO DE COMPUTAÇÃO
UNIVERSIDADE ESTADUAL DE CAMPINAS

On Almost Deterministic Timed Automata

A. V. Moura   *G. A. Pinto*

Technical Report - IC-01-006 - Relatório Técnico

May - 2001 - Maio

On Almost Deterministic Timed Automata

Arnaldo V. Moura and Guilherme A. Pinto
{arnaldo,guialbu}@ic.unicamp.br

Abstract

Timed Automata (TAs) are a generalization of ω-automata that has been largely studied both from the practical point of view of verification of real-time systems, and from the theoretical perspective of formal languages. Universality for deterministic TAs is PSPACE-complete but, surprisingly, it was shown to be Π^1_1-hard for nondeterministic TAs. The exact position of this problem in the analytical hierarchy is still open. In this paper we consider the more restricted class of almost deterministic TAs. In our main contribution we show that the restriction to almost deterministic TAs characterizes this decision problem as Π^1_1-complete and we also show that, in contrast to ω-automata, almost deterministic TAs define a proper subclass of nondeterministic TAs. These results give new insights regarding the role of nondeterminism in TAs and reveal some surprising aspects of the universality problem for nondeterministic TAs.

1 Introduction

Timed automata (TAs) were proposed in [3] as a formalism for the verification of real-time systems. Being a natural extension of ω-automata to "real-time", it has attracted much attention from practitioners and theorists. Some interesting aspects of TAs are their characterization through timed expressions [8], the power of silent transitions [6], some pumping lemmas [5] and analogies to ω-regular languages [13].

Nondeterminism and decision problems for TAs have been constant issues since the universality and the language inclusion problems were shown to be Π^1_1-hard for nondeterministic TAs [3]. The question of a more precise positioning of these problems in the analytical hierarchy is still open. See [15] for a comprehensive introduction to the hierarchies of the undecidable. For deterministic TAs the universality and language inclusion problems are PSPACE-complete [1]. Other results relating TAs and the hierarchies of the undecidable are the generalizations and reachability problems considered in [4, 7, 14].

In this paper, we consider the effect of limiting the nondeterminism in TAs by imposing syntactical restrictions on the formalism. With these restrictions, the automata are allowed to make only finitely many nondeterministic choices. Independently, a similar idea was considered in the context of probabilistic verification of ω-automata [11, 16] under the name of almost determinism. Almost deterministic ω-automata define the same class of languages as nondeterministic ω-automata and have not been studied outside that probabilistic context. For TAs, the situation is much richer. In our main contribution we show that the universality problem for almost deterministic TAs is Π^1_1-complete and we also show that almost deterministic TAs define a proper subclass of nondeterministic TAs. Because almost deterministic TAs impose a weighty restriction on nondeterministic TAs, one would be surprised if the universality problem for the latter turns out to be Π^1_1-complete too.

The paper is organized as follows. The next section introduces the TAs formalism. In Section 3 we define almost deterministic TAs and prove that they generate a proper subclass of nondeterministic TAs. The results about the universality problem and the analytical hierarchy are presented is Section 4. The following section presents a more comprehensive view of the theory by including also the Rabin and Streett acceptance conditions. We show that they give rise to the same classes as do the Muller and Büchi acceptance conditions. We conclude by discussing the significance of our results, stressing some of its surprising aspects in connection to the question of the universality problem for nondeterministic TAs.

2 Timed Automata

Informally, a TA is a finite-state ω-automaton [10] together with a finite set of clock variables whose values increase with the passage of time. Every transition of the automaton has a constraint on the values of the clocks and can be taken only if the clocks satisfy the constraint. In addition, a transition may reset some of the clocks. A TA accept timed words instead of ω-words. A *timed word* ρ, over a finite alphabet of symbols Σ, is a pair $(\overline{\sigma}, \overline{\tau})$ where $\overline{\sigma} = \sigma_1\sigma_2\cdots$ is a ω-word over Σ and $\overline{\tau} = \tau_1\tau_2\cdots$ is a strictly increasing sequence of positive time values $\tau_i \in \mathbb{R}$ satisfying the *progress* property: for every $t \in \mathbb{R}$, there is some $i \geq 1$ such that $\tau_i > t$. Let Σ^{t} denote the set of all timed words over Σ.

Example 1 The sequence $(a, 3.1)(b, 6)(a, 6.2)(b, 7.5)$ represents a finite prefix of a timed word ρ_s over the alphabet $\{a, b\}$. $\Box$

Given a finite set X of clock variables, a *clock constraint* δ over X is defined inductively by $\delta := x \leq c \,|\, x \geq c \,|\, \neg\delta \,|\, \delta_1 \wedge \delta_2$, where $x \in X$ and $c \in \mathbb{Q}$ is a non-negative rational constant. The set of all clock constraints over X is denoted by $\Phi(X)$. A *timed table* is a tuple $\mathcal{T} = \langle \Sigma, Q, Q_0, X, T \rangle$, where

- Σ is a finite alphabet of symbols;
- Q is a finite set of locations;
- $Q_0 \subseteq Q$ is a set of start locations;
- X is a finite set of clocks;
- $T \subseteq Q \times Q \times \Sigma \times 2^X \times \Phi(X)$ is a set of transitions. For a transition $\langle q, q', a, \lambda, \delta \rangle$ from location q to location q' on input symbol a, δ gives the constraint to be satisfied and λ gives the set of clocks to be reset. We call $\mathrm{Const}(T)$ the set of all constants that appear in some clock constraint in T.

A *clock interpretation for* X is a function from X to $\mathbb{R}$ yielding a particular reading of the clocks in X. A *state* of $\mathcal{T}$ has the form (q, ν), where q is a location and ν is a clock interpretation for X. For $t \in \mathbb{R}$, we write $\nu + t$ for the clock interpretation which maps every clock x to the new value $\nu(x) + t$. A clock interpretation ν for X *satisfies* a clock constraint δ over X iff δ evaluates to *true* when each clock x is replaced by $\nu(x)$ in δ. A *run* r of $\mathcal{T}$, over a timed word $\rho = (\overline{\sigma}, \overline{\tau})$ is a pair $(\overline{q}, \overline{\nu})$, where $\overline{q} = q_0 q_1 q_2 \cdots$ is an infinite sequence of locations of Q and $\overline{\nu} = \nu_0 \nu_1 \nu_2 \cdots$ is an infinite sequence of clock interpretations for X satisfying:

- $q_0 \in Q_0$, and $\nu_0(x) = 0$ for all $x \in X$;
- for all $i \geq 1$, there exists a transition $e = \langle q, q', \sigma, \lambda, \delta \rangle \in T$ such that $q = q_{i-1}$, $q' = q_i$, $\sigma = \sigma_i$, $(\nu_{i-1} + \tau_i - \tau_{i-1})$ satisfies δ, and $\nu_i(x) = 0$ if $x \in \lambda$, otherwise $\nu_i(x) = \nu_{i-1}(x) + \tau_i - \tau_{i-1}$. We assume $\tau_0 = 0$. In the sequel, we call e the i-th transition of r.

Given a run $r = (\overline{q}, \overline{\nu})$ over a timed word $\rho = (\overline{\sigma}, \overline{\tau})$, let *inf*$(r)$ be the set of locations such that $s \in$ *inf*(r) iff $s = q_i$ for infinitely many $i \geq 1$. A *timed Büchi automaton* (TBA) $\mathcal{A}$ is a tuple $\langle \Sigma, Q, Q_0, X, T, F \rangle$, where $\langle \Sigma, Q, Q_0, X, T \rangle$ is a timed table, and $F \subseteq Q$ is a set of accepting locations of $\mathcal{A}$. The run r over ρ is called an *accepting* run of $\mathcal{A}$ iff *inf*$(r) \cap F \neq \emptyset$. The language accepted by the TBA is defined by the set $L(\mathcal{A}) = \{\rho \in \Sigma^{\mathrm{t}} \mid \mathcal{A}$ has an accepting run over $\rho\}$.

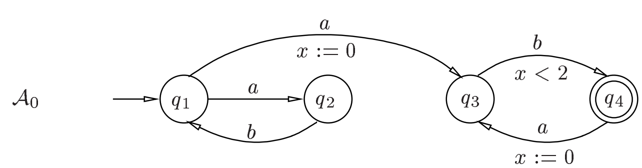

Figure 1: Expressing the *convergent bounded response property* with TAs

Given a timed table $\langle \Sigma, Q, Q_0, X, T \rangle$, a location $q \in Q$ is *deterministic* iff given any two transitions $\langle q_1, q_1', a_1, \lambda_1, \delta_1 \rangle$ and $\langle q_2, q_2', a_2, \lambda_2, \delta_2 \rangle$ in T, if $q_1 = q_2 = q$ and $a_1 = a_2$, then $\delta_1 \wedge \delta_2$ is an unsatisfiable clock constraint. A set $R \subseteq Q$ of locations is *deterministic* iff all locations in R are deterministic. A TBA $\langle \Sigma, Q, Q_0, X, T, F \rangle$ is *deterministic* (DTBA) iff $|Q_0| = 1$ and Q is deterministic. This definition implies the intended property that every DTBA has at most one run over any timed word. A timed table $\langle \Sigma, Q, Q_0, X, T \rangle$ is called *complete* iff given any state (q, ν) and any symbol $a \in \Sigma$ there is a transition $\langle q_1, q_1', a_1, \lambda_1, \delta_1 \rangle \in T$ such that $q_1 = q$, $a_1 = a$ and ν satisfies δ_1.

Example 2 The TBA $\mathcal{A}_0$ in Fig. 1 expresses the convergent bounded response property: "symbols a and b alternate and eventually always the time difference between an a and the next b is less than 2 time units" [3]. Clock x is reset only in the transitions from q_1 to q_3 and from q_4 to q_3. The transition from q_3 to q_4 is the only one with a clock constraint different than *true*. The automaton is nondeterministic, since the transitions from q_1 to q_3 and from q_1 to q_2 are taken on the same input symbol and the clock conditions on these transitions are simultaneously satisfied. The language accepted by $\mathcal{A}_0$ is $\{((ab)^\omega, \overline{\tau}) \mid \exists i \forall j [(j \geq i) \Rightarrow (\tau_{2j} < \tau_{2j-1} + 2)]\}$. Given the timed word ρ_s defined in Example 1, two possible finite prefixes of runs of $\mathcal{A}_0$ over ρ_s are given by $(q_1,0)(q_2,3.1)(q_1,6)(q_2,6.2)(q_1,7.5)$ and $(q_1,0)(q_2, 3.1)(q_1,6)(q_3,0)(q_4,1.3)$. $\square$

3 Almost Deterministic Timed Automata

The concept of almost-determinism appeared independently in the context of probabilistic verification [11], where a concurrent probabilistic program is tested against a specification given by an ω-automaton. The first step of the verification algorithms is to obtain an equivalent deterministic automaton, or an equivalent almost deterministic automaton, the latter being much smaller. An almost deterministic automaton has the property that every accepting run makes a finite number of nondeterministic choices. In [16, 11] it is shown that nondeterministic and almost deterministic Büchi ω-automata are equivalent, by direct translations.

For a timed table $\langle \Sigma, Q, Q_0, X, T\rangle$ and a set $R \subseteq Q$, let $\text{Reach}(R) \subseteq Q$ be the set of locations s for which there is a sequence $s_1, s_2, \ldots, s_k$, $k \geq 1$, such that $s_1 \in R$, $s_k = s$ and for every $1 \leq i < k$ there is $\langle q, q', a, \lambda, \delta\rangle$ in T such that $q = s_i$ and $q' = s_{i+1}$. A TBA $\langle \Sigma, Q, Q_0, X, T, F\rangle$ is an *almost deterministic* TBA (ADTBA) iff $\text{Reach}(F)$ is deterministic.

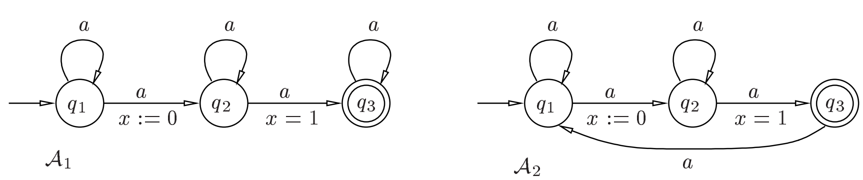

Figure 2: The automata $\mathcal{A}_1$ and $\mathcal{A}_2$

The TBA $\mathcal{A}_0$ in Fig. 1 is an example of an ADTBA. Other example is the TBA $\mathcal{A}_1$ in Fig. 2. It accepts every timed word over $\{a\}$ for which there is a pair of a's separated by a difference of exactly 1 time unit, that is, $L(\mathcal{A}_1) = \{(a^\omega, \overline{\tau}) \mid \exists i \exists j[(i < j) \wedge (\tau_j = \tau_i + 1)]\}$. While $L(\mathcal{A}_1)$ can be accepted by an ADTBA, it cannot be accepted by a DTBA [3]. The intuitive reason is that, since the number of a's that can happen in a time unit is unbounded, a DTBA would need an unbounded number of clocks to correctly recognize such a pair of a's. Since a DTBA cannot recognize one such pair, an ADTBA should not be able to recognize an infinite number of such pairs, otherwise it would have to do so deterministically from a certain point on. Infinite occurrences of such pairs, however, can be recognized by a TBA. In fact, for the simple TBA $\mathcal{A}_2$ in Fig. 2 we have $L(\mathcal{A}_2) = \{(a^\omega, \overline{\tau}) \mid \forall k \exists i \exists j[(k < i < j) \wedge (\tau_j = \tau_i + 1)]\}$.

Let $\mathcal{ADTBA}$ and $\mathcal{TBA}$ denote the class of languages accepted, respectively, by an ADTBA or by a TBA. The next theorem asserts that $\mathcal{ADTBA}$ is a proper subclass of $\mathcal{TBA}$.

Theorem 1 *If $\mathcal{B}$ is an* ADTBA*, then* $L(\mathcal{B}) \neq L(\mathcal{A}_2)$.

Proof. We proceed by contradiction. Assume that $\mathcal{B} = \langle \Sigma, Q, Q_0, X, T, F\rangle$ is an ADTBA and that $L(\mathcal{B}) = L(\mathcal{A}_2)$. We first choose a special timed word $\rho^2 \in L(\mathcal{A}_2)$ and take any accepting run $r^2 = (\overline{q^2}, \overline{\nu^2})$ of $\mathcal{B}$ over ρ^2; then we perturb ρ^2 according to r^2, obtaining ρ^3, and show that $\mathcal{B}$ has a run $r^3 = (\overline{q^3}, \overline{\nu^3})$ over ρ^3, such that $\overline{q^3} = \overline{q^2}$. The contradiction is established when we note that $\rho_3 \notin L(\mathcal{A}_2)$.

Let $n = |\mathrm{Reach}(F)|$ and $k = |X|$. Let C_B be a natural constant such that $C_B > 1$ and $C_B > c$ for all $c \in \mathrm{Const}(T)$. Let $\varepsilon < 1$ be a rational constant such that for all $c \in \{\mathrm{Const}(T) \cup \{1\}\}$ there is some natural m, where $c = m\varepsilon$.

In order to construct ρ^2 we define two finite timed words. Let $p^0 = (a, \tau^0_1)(a, \tau^0_2) \ldots (a, \tau^0_{nk+1})$ consist of a sequence of $(nk+1)$ a's equally distributed between C_B and $C_B + \varepsilon$, that is, $\tau^0_i = C_B + \mu i$, where $\mu = \varepsilon/(nk+2)$. The upper part of Fig. 3 illustrates p^0. Given a location $\ell \in \mathrm{Reach}(F)$ and a clock interpretation ν, since $\mathrm{Reach}(F)$ is deterministic, there is at most one finite run $(q_0, \nu_0)(q_1, \nu_1) \ldots (q_{nk+1}, \nu_{nk+1})$ of $\mathcal{B}$ over p^0 such that $(q_0, \nu_0) = (\ell, \nu)$; and, since there are k clocks, at least $((n-1)k+1)$ transitions in this run are such that no clock is reset *for the last time* on them. Furthermore, since the value of any clock is greater than C_B when the first a occurs, exactly the same sequence of transitions will be taken for any ν, when ℓ is fixed. Thus, there is a fixed index j, $1 \le j \le (nk+1)$, such that for all $\ell \in \mathrm{Reach}(F)$ and for all ν, no clock is reset for the last time on the j-th transition of the run over p^0 starting at (ℓ, ν).

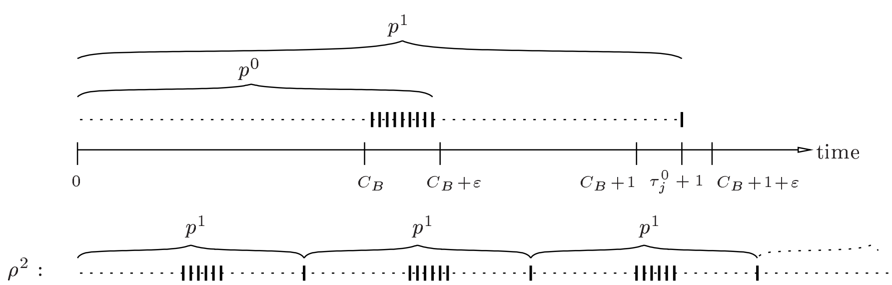

Figure 3: Constructing the timed word ρ^2

Now, let $p^1 = (a, \tau^1_1)(a, \tau^1_2) \ldots (a, \tau^1_{nk+2})$ where $\tau^1_i = \tau^0_i$ for $1 \le i \le (nk+1)$, and $\tau^1_{nk+2} = \tau^0_j + 1$. Then, $\rho^2 = (a^\omega, \overline{\tau^2})$ is the infinite concatenation of p^1, as illustrated in Fig 3. Formally, for any $i > 0$, let i^{d} and $i^{\mathrm{m}} < (nk+2)$ be naturals such that $i = i^{\mathrm{d}}(nk+2) + i^{\mathrm{m}}$. Thus, $\tau^2_i = i^{\mathrm{d}}\tau^1_{nk+2} + \tau^1_{i^{\mathrm{m}}}$. Define $\tau^1_0 = 0$. Clearly, $\rho^2 \in L(\mathcal{A}_2)$. Let $r^2 = (\overline{q^2}, \overline{\nu^2})$ be an accepting run of $\mathcal{B}$ over ρ^2. There must exist at least one such run since we assumed $L(\mathcal{B}) = L(\mathcal{A}_2)$. Let f be the smallest natural such that $f^{\mathrm{m}} = 0$ and $q^2_f \in \mathrm{Reach}(F)$. Note that r^2 is deterministic from the f-th transition on. Also, for every natural i, if $i \ge f$ and $i^{\mathrm{m}} = 0$ then for every clock $x \in X$ either x is not reset in the $(i+j)$-th transition of r^2, or x is reset in the $(i+j')$-th transition of r^2, for some j', $j < j' < nk+2$. Informally, this property makes the run r^2 insensitive to small perturbations in the occurrence times τ^2_i, for $i > f$ and $i^{\mathrm{m}} = 0$. We now obtain ρ^3 by perturbing ρ^2.

Let $\rho^3 = (a^\omega, \overline{\tau^3})$ be defined by letting $\tau_i^3 = \tau_i^2 - \mu/2$ if $i > f$ and $i^{\mathrm{m}} = 0$, otherwise $\tau_i^3 = \tau_i^2$. Thus, $\rho^3 \notin L(\mathcal{A}_2)$. We assume without loss of generality that $\mathcal{B}$ is complete. Let $r^3 = (\overline{q^3}, \overline{\nu^3})$ be the run of $\mathcal{B}$ over ρ^3 such that $(q_i^3, \nu_i^3) = (q_i^2, \nu_i^2)$ for every i, $0 \leq i \leq f$. Note that there is exactly one such run, since ρ_3 equals ρ_2 up to the f-th symbol, $\mathcal{B}$ is complete and r^3 must be deterministic from the f-th transition on.

We claim that r^3 and r^2 follow exactly the same sequence of transitions, which implies $\overline{q^3} = \overline{q^2}$. The proof is by induction on i. For $i > f$, if the g-th transition of r^3 equals the g-th transition of r^2 for all $g < i$, then the i-th transitions of r^3 and r^2 are equal. Let $\eta(\mathrm{h})$, $\mathrm{h} \in \{2, 3\}$, be an abbreviation for $\nu_{i-1}^{\mathrm{h}}(x) + \tau_i^{\mathrm{h}} - \tau_{i-1}^{\mathrm{h}}$. There are three cases:

1. If $i^{\mathrm{m}} = 1$, then for all $x \in X$, $\eta(2) > C_B$ and $\eta(3) > C_B$;
2. If $i^{\mathrm{m}} > 1$, then for all $x \in X$, either $\eta(2) > C_B$ and $\eta(3) > C_B$, or, $\eta(2) < \varepsilon$ and $\eta(3) < \varepsilon$;
3. If $i^{\mathrm{m}} = 0$, then for all $x \in X$, either $\eta(2) > C_B$ and $\eta(3) > C_B$, or for all natural m: (i) $\eta(2) \leq m\varepsilon$ iff $\eta(3) \leq m\varepsilon$ and, (ii) $\eta(2) \geq m\varepsilon$ iff $\eta(3) \geq m\varepsilon$, both hold.

All cases imply that $(\nu_{i-1}^3 + \tau_i^3 - \tau_{i-1}^3)$ satisfies δ iff $(\nu_{i-1}^2 + \tau_i^2 - \tau_{i-1}^2)$ satisfies δ, for any clock constraint δ in T. Therefore, the i-th transitions of r^3 and r^2 will be the same. □

3.1 Closure Properties

The class $\mathcal{ADTBA}$ is closed under union and intersection. Given a set $\{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k\}$ of ADTBAs, $\bigcup_{i=1}^k L(\mathcal{C}_i)$ is accepted by the disjoint union of $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k$, which is an ADTBA. For $\bigcap_{i=1}^k L(\mathcal{C}_i)$, we simply note that the product construction in [3, p. 197] when applied on $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k$, yields an ADTBA. In the next section, we will show that the universality problem for ADTBA is Π_1^1-hard. This fact leads to the non-closure of $\mathcal{ADTBA}$ under complementation, as shown below:

Theorem 2 ([3]) *$\mathcal{ADTBA}$ is not closed under complementation.*

Proof. The Π_1^1-hardness of the universality problem implies the Π_1^1-hardness of the inclusion problem, since an ADTBA $\mathcal{C}$ is universal iff $L(\mathcal{U}) \subseteq L(\mathcal{C})$, where $\mathcal{U}$ is a given universal ADTBA.

Given ADTBAs $\mathcal{C}_1$ and $\mathcal{C}_2$, $L(\mathcal{C}_1) \subseteq L(\mathcal{C}_2)$ iff $L(\mathcal{C}_1) \cap \overline{L(\mathcal{C}_2)} = \emptyset$. Assume that $\mathcal{ADTBA}$ *is* closed under complementation. Then, one can easily show that $L(\mathcal{C}_1) \not\subseteq L(\mathcal{C}_2)$ iff there is $\mathcal{A}$ such that $L(\mathcal{C}_1) \cap L(\mathcal{A}) \neq \emptyset$ and $L(\mathcal{C}_2) \cap L(\mathcal{A}) = \emptyset$. But then, since intersection and emptiness are decidable, and since the set of all ADTBAs has a recursive indexing $\mathcal{A}_0, \mathcal{A}_1, \ldots$, the complement of the inclusion problem would be recursively enumerable or, equivalently, would be in Σ_1^0: $\exists k[L(\mathcal{C}_1) \cap L(\mathcal{A}_k) \neq \emptyset \wedge L(\mathcal{C}_2) \cap L(\mathcal{A}_k) = \emptyset]$. This is a contradiction, because the complement of a Π_1^1-hard problem cannot be in Σ_1^0 [15]. □

4 Universality for $\mathcal{ADTBA}$

Before we consider ADTBAs, let us show that the universality problem for TBAs is in Π_2^1. Let $\mathcal{B}_0, \mathcal{B}_1, \ldots$ be a recursive indexing of all TBAs. The universality problem for TBAs,

$U_{\text{TBA}} \subset \mathbb{N}$, is defined as the set $U_{\text{TBA}} = \{ z \mid \forall \rho \in \Sigma^{\text{t}} \, \exists r \, [\, r$ is an accepting run of $\mathcal{B}_z$ over $\rho \,] \}$. First we note that it suffices to consider only rational timed words. Let $\Sigma^{\text{rt}} \subset \Sigma^{\text{t}}$ be defined as $\Sigma^{\text{rt}} = \{(\overline{\sigma}, \overline{\tau}) \in \Sigma^{\text{t}} \mid \tau_i \text{ rational, for all } i \geq 1\}$.

Lemma 1 *Let $\mathcal{B}$ be a TBA. Given $(\overline{\sigma}, \overline{\tau}) \in \Sigma^t$, there is $(\overline{\sigma'}, \overline{\tau'}) \in \Sigma^{rt}$ such that $(\overline{\sigma}, \overline{\tau}) \in L(\mathcal{B})$ iff $(\overline{\sigma'}, \overline{\tau'}) \in L(\mathcal{B})$.*

Proof. The intended timed word can be defined inductively as in the proof of Theorem 3.17 in [3]. For the sake of completeness, we provide a definition. Let ε be the constant defined as in the proof of Theorem 1. Let $\tau'_0 = \tau_0 = 0$. If $\tau_i = \tau_j + m\varepsilon$ for some $0 \leq j < i$ and some natural m, take $\tau'_i = \tau'_j + m\varepsilon$. Otherwise, take a rational τ'_i such that $(\tau'_i - \tau'_j) < m\varepsilon$ iff $(\tau_i - \tau_j) < m\varepsilon$ for all $0 \leq j < i$ and for all natural m. □

Corollary 1 *For any natural z, $z \in U_{TBA}$ iff*

$$\forall \rho \in \Sigma^{\text{rt}} \, \exists r [\, r \textit{ is an accepting run of } \mathcal{B}_z \textit{ over } \rho \,].$$

Proof. Immediate from Lemma 1. □

Hence, we can follow the usual approach of mapping the universal quantifier over timed words, in the definition above, to an universal quantifier over a function variable, in the analytical defintion.

Let $\Sigma = \{a_0, a_1, \ldots, a_{k-1}\}$ be an alphabet and let $\mathcal{F}$ be the set of all functions over the naturals. A timed word in Σ^{rt} can be effectively encoded as a function in $\mathcal{F}$. Take $d_\rho : \mathcal{F} \to \Sigma^{\text{rt}}$ as the onto function defined as follows. Given $f \in \mathcal{F}$, let $d_\rho(f) = (\overline{\sigma}, \overline{\tau})$, where for all $i \geq 1$

- $\sigma_i = a_{f(3(i-1)) \pmod{k}}$, and
- $\tau_i = \tau_{i-1} + \{[(f(3(i-1)+1)) + 1]/[(f(3(i-1)+2)) + 1]\}$, and $\tau_0 = 0$.

Given a TBA $\mathcal{B}$, we can, in a similar manner, obtain an effective onto function $d_r : \mathcal{F} \to \mathcal{R}_{\mathcal{B}}$, where $\mathcal{R}_{\mathcal{B}}$ is the set of all rational runs of $\mathcal{B}$. Then, $U_{\text{TBA}} \subset \mathbb{N}$ is the set

$$U_{\text{TBA}} = \{ z \mid \forall f \, \exists g \, [\, d_\rho(f) \text{ is progressive } \Rightarrow \\ d_r(g) \text{ is an accepting run of } \mathcal{B} \text{ over } d_\rho(f) \,]\}$$

where f and g range over $\mathcal{F}$. The predicate inside the square brackets can be defined arithmetically using quantifiers over the natural numbers and recursive relations with oracles for f and g. For the approach of mapping the universal quantifier over f to timed words, there seems to be no way of getting rid of the existential quantifier over the function g. This intuition is supported by the language $L(\mathcal{A}_2)$ in Section 3, for which Theorem 1 asserts that a TBA must make infinitely many nondeterministic choices in order to recognize that a given timed word is in $L(\mathcal{A}_2)$.

Now let $\mathcal{A}_0, \mathcal{A}_1, \ldots$ be a recursive indexing of all ADTBAs. As was the case for TBAs, we have an effective onto encoding $d_r : \mathcal{F} \to \mathcal{R}_{\mathcal{A}}$, when $\mathcal{R}_{\mathcal{A}}$ is the set of all rational runs of an ADTBA $\mathcal{A}$. For an ADTBA, however, if $d_r(g)$ is an accepting run over $d_\rho(f)$, then it must be deterministic from a certain point on. Thus, if we have a finite prefix $(q_0, \nu_0), (q_1, \nu_1), \ldots, (q_k, \nu_k)$ of $d_r(g)$, such that q_k is an accepting location of $\mathcal{A}$, then the remainder of $d_r(g)$ is uniquely determined by $d_\rho(f)$. Hence, it can be retrieved by consulting the oracle f. It turns out that finite prefixes of runs of $\mathcal{A}$ can be effectively encoded as natural numbers using standard techniques. Consider an effective onto function $d_p : \mathbb{N} \to \mathcal{R}^{\text{fin}}_{\mathcal{A}}$, where $\mathcal{R}^{\text{fin}}_{\mathcal{A}}$ is the set of all finite prefixes of rational runs of $\mathcal{A}$. We have:

Theorem 3 $U_{\text{ADTBA}} \in \Pi^1_1$.

Proof. Consider a Turing machine M_{H_1} with one oracle. M_{H_1} accepts a given pair $(i, j) \in \mathbb{N}^2$ iff $\tau_j > i$, where τ_j is obtained by consulting the oracle according to d_ρ. Let $H_1 \subseteq \mathcal{F} \times \mathbb{N}^2$ be relation such that $(f, i, j) \in H_1$ iff M_{H_1} with oracle f accepts the pair (i, j). Then H_1 is a recursive relation. We write $H_1(f, i, j)$ for $(f, i, j) \in H_1$. Consider another Turing machine M_{H_2} with one oracle which, given a tuple $(p, i, j, z) \in \mathbb{N}^4$, behaves as follows:

1. If $j \leq i$, then M_{H_2} rejects.
2. M_{H_2} decodes z and p, thus obtaining $\mathcal{A}_z = \langle \Sigma, Q, Q_0, X, T, F \rangle$ and $d_p(p) = (q_0, \nu_0), (q_1, \nu_1), \ldots, (q_k, \nu_k)$.
3. If $q_k \notin F$, then M_{H_2} rejects.
4. M_{H_2} consults the oracle according to d_ρ, and obtains a finite timed word ρ_a, with k symbols. If $d_p(p)$ is not a run of $\mathcal{A}_z$ over ρ_a, then M_{H_2} rejects.
5. M_{H_2} consults the oracle according to d_ρ and obtains a finite timed word ρ_b, with $k+j$ symbols. Observe that ρ_b has ρ_a as a prefix.
6. M_{H_2} constructs the run $(q_0, \nu_0), \ldots, (q_k, \nu_k), \ldots, (q_{k+j}, \nu_{k+j})$ of $\mathcal{A}_z$ over ρ_b. Note that there is only one such run.
7. If $q_{k+j} \notin F$, then M_{H_2} rejects, otherwise it accepts.

Take $H_2 \subseteq \mathcal{F} \times \mathbb{N}^4$ as the recursive relation such that $H_2(f, p, i, j, z)$ iff M_{H_2} accepts the tuple (p, i, j, z) with oracle f. Finally, we have that $U_{\text{ADTBA}} = \{z \mid \forall f \, [\, (\forall i \, \exists j \, H_1(f, i, j)) \Rightarrow (\exists p \, \forall i \, \exists j \, H_2(f, p, i, j, z)) \,]\}$. □

Now, we turn to the Π^1_1-hardness of U_{ADTBA}. In [12] it is shown that the problem of deciding whether a nondeterministic Turing machine has an infinite computation over the empty tape that visits its start state infinitely often is Σ^1_1-complete. In [3], the complement of this problem is reduced to U_{TBA}, establishing that U_{TBA} is Π^1_1-hard. A similar reduction can be used to show that U_{ADTBA} is Π^1_1-hard.

A nondeterministic 2-counter machine M consists of a sequence of n instructions and two counters, C and D. There are 6 types of instructions: (a) increment C and jump nondeterministically to instruction x or y; (b) decrement C and jump nondeterministically to instruction x or y; (c) if $C = 0$ jump to instruction x, otherwise jump to instruction y; (d), (e) and (f) are the same as above, exchanging D and C. A configuration of M is a tuple (i, c, d), where c and d are the counter values, and i is the instruction to be executed. A computation of M is a sequence of related configurations beginning with $(1, 0, 0)$. A computation is recurring iff instruction 1 is executed infinitely often.

Define the timed language L over the alphabet $\{b_1, b_2, \ldots, b_n, a_1, a_2\}$ in such a way that $(\overline{\sigma}, \overline{\tau}) \in L$ iff:

1. $\sigma = b_{i_1} a_1^{c_1} a_2^{d_1} b_{i_2} a_1^{c_2} a_2^{d_2} \ldots$, where $(i_1, c_1, d_1)(i_2, c_2, d_2) \ldots$ is a recurring computation of M;
2. for all $j \geq 1$, b_{i_j} occurs at time j;
3. for all $j \geq 1$:
   (a) if $c_{j+1} = c_j$, then for all a_1 at time $t \in (j, j+1)$ there is an a_1 at time $t+1$;
   (b) if $c_{j+1} = c_j + 1$, then for all a_1 at time $t \in (j+1, j+2)$ there is an a_1 at time $t-1$, except for the last one;
   (c) if $c_{j+1} = c_j - 1$, then for all a_1 at time $t \in (j, j+1)$ there is an a_1 at time $t+1$, except for the last one;
4. is the same as 3. exchanging a_2 and a_1, and exchanging d and c.

It is clear that $\overline{L}$, the complement of L, can be defined as a finite disjunction of several simple timed languages, which can all be accepted by ADTBAs. The disjoint union of all these ADTBAs is universal iff M *does not* have a recurring computation. The next section describes a series of figures giving the details of the ADTBAs needed to accept $\overline{L}$.

4.1 The ADTBAs needed to show the Π_1^1-hardness of Universality

The timed language $\overline{L}$ can be accepted by the disjoint union of the following ADTBAs [3]: $\mathcal{A}_0, \mathcal{A}_{\text{init}}, \mathcal{A}_{\text{recur}}$ for the boundary conditions; and $\mathcal{A}_i$, $1 \leq i \leq n$, one automaton for each instruction of M. In the following descriptions, consider a timed word $(\overline{\sigma}, \overline{\tau})$ in $\overline{L}$.

Figure 4 gives the automata for the boundary conditions:

- $\mathcal{A}_0^1$ states that there is no symbol b_i at time j, for some $j \geq 1$;
- $\mathcal{A}_0^2$ states that the interval $(j, j+1)$ is not of the form $a_1^* a_2^*$, for some $j \geq 1$;
- $\mathcal{A}_{\text{init}}$ states that either $\sigma_1 \neq b_1$ or $\tau_1 \neq 1$ or $\tau_2 < 2$;
- $\mathcal{A}_{\text{recur}}$ states that $\sigma_j = b_1$ for finitely many $j \geq 1$.

From the 6 types of instructions, we consider only three, since the other ones follow by similar techniques. Assume that: instruction 6 is of type (d), with $x = 11$ and $y = 5$; instruction 10 is of type (b), with $x = 9$ and $y = 2$; and instruction 31 is of type (f), with $x = 8$ and $y = 22$.

Figure 5 and Fig. 6 give the automaton $\mathcal{A}_6$:

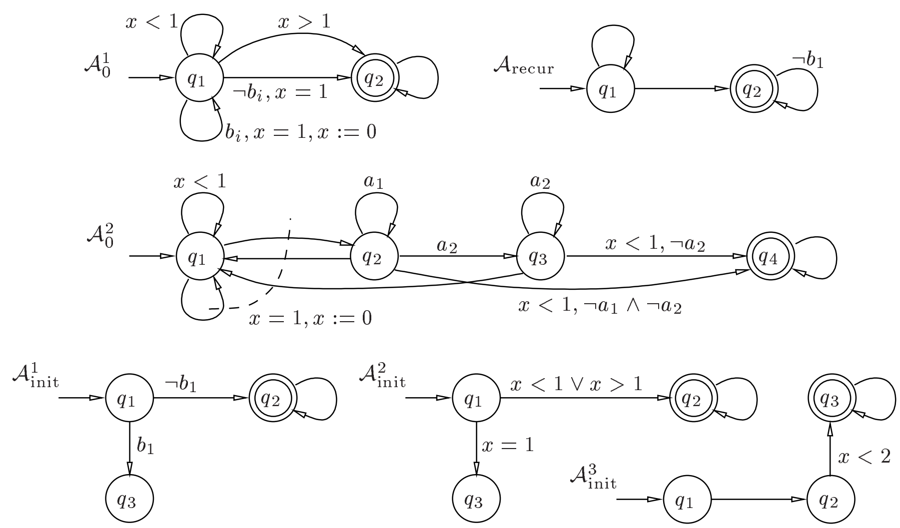

Figure 4: The automata needed for the boundary conditions

- $\mathcal{A}_6^1$ states that, for some $j \geq 1$, $\sigma_j = b_6$ and there is neither b_{11} nor b_5 at time $j+1$;
- Ensuring that the a_1's do not match:
  - $\mathcal{A}_6^2$ states that there is a b_6, then a a_1 at time t, and *no* a_1 at time $t+1$;
  - $\mathcal{A}_6^3$ and $\mathcal{A}_6^4$ state that there is a b_6 at time t, a a_1 at time $t' \in (t+1, t+2)$, and *no* a_1 at time $t'-1$;
- Ensuring that the a_2's do not reflect an increment:
  - $\mathcal{A}_6^5$, omitted in Fig. 6, is analogous to $\mathcal{A}_6^2$ exchanging a_2 and a_1;
  - $\mathcal{A}_6^6$ states that there is a b_6 at time t, and the last a_2 at time $t' \in (t+1, t+2)$ has a matching a_2 at time $t'-1$;
  - $\mathcal{A}_6^7$ states that there is a b_6 at time t, a a_2 at time $t' \in (t+1, t+2)$ which is not the last one in $(t+1, t+2)$, and there is no a_2 at time $t'-1$.

For the automaton $\mathcal{A}_{10}$, we have:

- $\mathcal{A}_{10}^1$ is analogous to $\mathcal{A}_6^1$ exchanging b_{10} and b_6, b_9 and b_{11}, b_2 and b_5;
- Ensuring that the a_2's do not match:
  - $\mathcal{A}_{10}^2$, $\mathcal{A}_{10}^3$ and $\mathcal{A}_{10}^4$ are analogous to $\mathcal{A}_6^2$, $\mathcal{A}_6^3$ and $\mathcal{A}_6^4$, respectively, exchanging b_{10} and b_6, a_2 and a_1;
- Ensuring that the a_1's do not reflect a decrement:
  - $\mathcal{A}_{10}^5$ and $\mathcal{A}_{10}^6$ are analogous to $\mathcal{A}_6^3$ and $\mathcal{A}_6^4$ exchanging b_{10} and b_6;

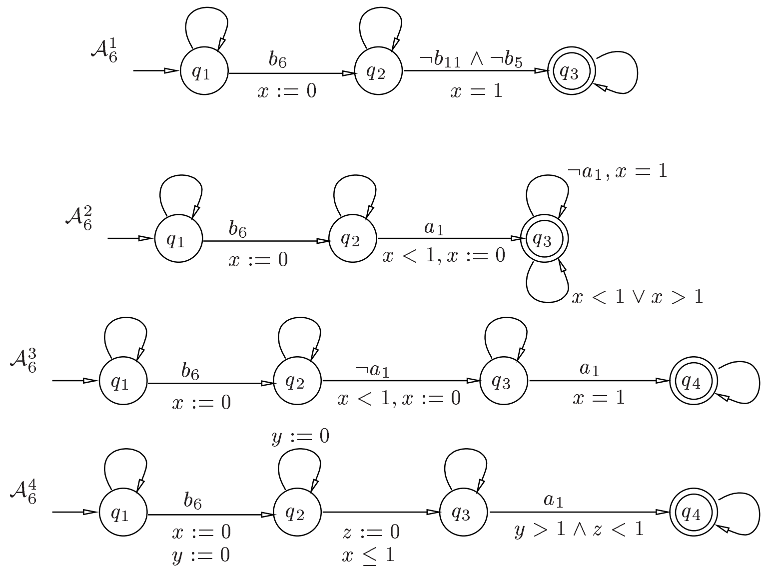

Figure 5: Accepting when the a_1's do not match

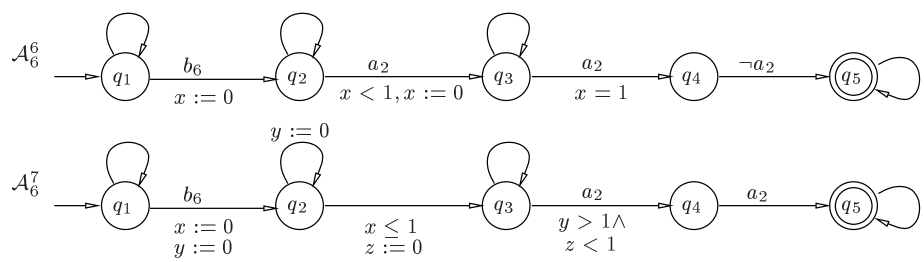

Figure 6: Accepting when the a_2's do not reflect an increment

- $\mathcal{A}_{10}^7$, in Fig. 7, states that there is a b_{10} at time t, and the last a_1 at time $t' \in (t, t+1)$ has a matching a_1 at time $t'+1$;
- $\mathcal{A}_{10}^8$, in Fig. 7, states that there is a b_{10} at time t, a a_1 at time $t' \in (t, t+1)$ which is not the last one in $(t, t+1)$, and there is no a_1 at time $t'+1$.

For the automaton $\mathcal{A}_{31}$, we have:

- Ensuring that the a_1's or the a_2's do not match:
  - Automata $\mathcal{A}_{31}^1, \mathcal{A}_{31}^2, \ldots, \mathcal{A}_{31}^6$, are analogous, respectively, to $\mathcal{A}_6^2$, $\mathcal{A}_6^3$, $\mathcal{A}_6^4$, $\mathcal{A}_{10}^2$, $\mathcal{A}_{10}^3$ and $\mathcal{A}_{10}^4$ exchanging b_6 and b_{31}, b_{10} and b_{31};
- $\mathcal{A}_{31}^7$, in Fig. 8, states that there is a b_{31} at time t, then *no* a_2 in $(t, t+1)$, and there is not b_8 at time $t+1$;

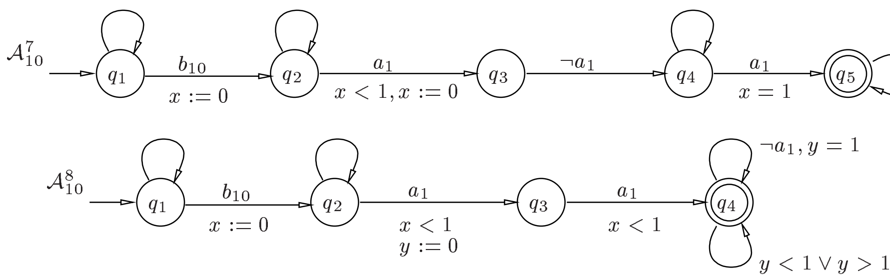

Figure 7: Accepting when the a_1's do not reflect a decrement

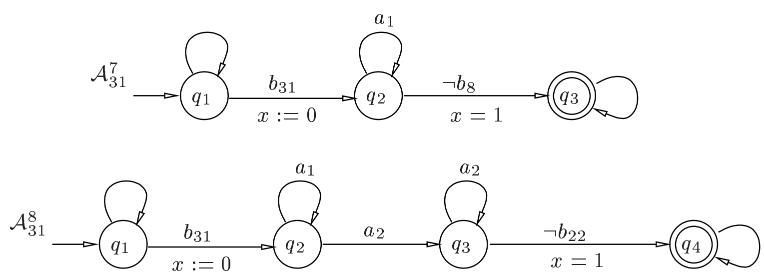

Figure 8: Accepting when the jump is *not* correct

- $\mathcal{A}_{31}^{8}$, in Fig. 8, states that there is a b_{31} at time t, then at least one a_2 in $(t, t+1)$, and there is not b_{22} at time $t+1$.

5 Varying the Acceptance Condition

In [3] only Büchi (B) and Muller (M) acceptance conditions were considered. These are generally regarded as the least expressive and most expressive conditions, respectively. In a preceding paper [2], the same authors suggested the investigation of the other two commonly used conditions, Rabin (R) and Streett (S), noting that they could possibly define intermediate classes of languages. In this section, we propose definitions for almost deterministic Rabin and Streett TAs. We also argue that the Rabin and the Streett conditions do not define new classes besides the ones already defined by the Büchi and the Muller acceptance conditions.

A *timed (Muller|Rabin|Streett|Büchi) automaton* $\mathcal{A}$ is a timed table $\langle \Sigma, Q, Q_0, X, T \rangle$ together with an acceptance condition, given by Table 1 below, where $\mathbf{d}(S)$ indicates that the set $S \subseteq Q$ is deterministic.

A Büchi condition can be viewed as a special case of both the Rabin and the Streett conditions, by simply taking $\{(F, \emptyset)\}$ and $\{(Q, F)\}$, respectively. Büchi, Rabin or Streett conditions can be phrased as a Muller condition by selecting exactly the subsets satisfying the given condition. These observations readily show that $\mathcal{TMA} = \mathcal{TRA} = \mathcal{TSA} = \mathcal{TBA}$, since $\mathcal{TMA} = \mathcal{TBA}$ [3].

	syntax	semantics	almost deterministic if
M	$\mathbf{F} \subseteq 2^Q$	$\mathit{inf}(r) \in \mathbf{F}$	$\forall C \in \mathbf{F}\,[\,\mathbf{d}(\mathrm{Reach}(C))\,]$
R	$\{(L_i, U_i)\}_i$ $L_i, U_i \subseteq Q$	$\exists i\,[\,\mathit{inf}(r) \cap L_i \neq \emptyset\ \wedge$ $\mathit{inf}(r) \cap U_i = \emptyset\,]$	$\forall i\,[\,\mathbf{d}(\mathrm{Reach}(L_i))\,]$
S	$\{(L_i, U_i)\}_i$ $L_i, U_i \subseteq Q$	$\forall i\,[\,\mathit{inf}(r) \cap L_i = \emptyset\ \vee$ $\mathit{inf}(r) \cap U_i \neq \emptyset\,]$	$\exists i\,[\,\mathbf{d}(\mathrm{Reach}(U_i))\ \wedge$ $L_i = Q\,]$
B	$F \subseteq Q$	$\mathit{inf}(r) \cap F \neq \emptyset$	$\mathbf{d}(\mathrm{Reach}(F))$

Table 1: Defining almost deterministic timed automata

The last column of Table 1 gives the proposed definitions for ADTMA, ADTRA and ADTSA. Note that these purely syntactical restrictions enforce the intended property that every accepting run of an almost deterministic automaton makes finitely many nondeterministic choices.

The traditional translation from a Muller automaton to a Büchi automaton seems to "introduce nondeterminism". Actually, it introduces almost determinism. This can be seen in the proof of the following theorem:

Theorem 4 *$\mathcal{DTMA} \subseteq \mathcal{ADTBA}$.*

Proof. Let $A = \langle \Sigma, Q, Q_0, X, T, \mathbf{F} \rangle$ be a DTMA. Since $\mathcal{ADTBA}$ is closed under union, we can assume that $|\mathbf{F}| = 1$. We construct an equivalent ADTBA $A' = \langle \Sigma, Q', Q'_0, X', T', F \rangle$. Let $C = \{c_1, c_2, \ldots, c_k\}$ be the only set in $\mathbf{F}$. Then, A' consists of $k+1$ copies of A numbered from 0 to k: $Q' = Q \times \{0, 1, \ldots, k\}$, $Q'_0 = \{(q_0, 0)\}$, where q_0 is the only location in Q_0, and $X' = X$. Informally, the copy numbered 0 is equal to A, with the addition of appropriate nondeterministic transitions to the copy 1. Then it is required to cycle deterministically through the remaining copies, inside C, guaranteeing satisfaction of the Muller condition. The transition set is defined as follows: $\langle q, q', a, \lambda, \delta \rangle \in T'$ iff there is $\langle s, s', a, \lambda, \delta \rangle \in T$, $q = (s, i)$, $q' = (s', j)$, *and*:

1. $i = j = 0$; or
2. $s \in C$, $s' \in C$, $i = 0$ and $j = 1$; or
3. $s \in C$, $s' \in C$, $s \neq c_i$ and $i = j$; or
4. $s = c_i$, $s' \in C$, $i = k$ and $j = 1$; or
5. $s = c_i$, $s' \in C$, $1 \leq i \leq k-1$ and $j = i + 1$.

The acceptance condition is $F = \{(c_k, k)\}$. □

Corollary 2 $\mathcal{ADTMA} = \mathcal{ADTRA} = \mathcal{ADTSA} = \mathcal{ADTBA}$.

Proof. Using a similar construction as in Theorem 4 to show $\mathcal{ADTMA} \subseteq \mathcal{ADTBA}$, and noting that Büchi, Rabin or Streett conditions can be phrased as a Muller condition preserving almost determinism. □

Corollary 3 $\mathcal{DTMA} \subset \mathcal{ADTBA}$.

Proof. $\mathcal{DTMA} \subseteq \mathcal{ADTBA}$ from Theorem 4, but $\mathcal{DTMA}$ is closed under complementation [3] and $\mathcal{ADTBA}$ is not. The proper containment follows. □

It remains to consider deterministic TA. In what follows, we assume complete TA. $\mathcal{DTBA}$ is a proper subclass of $\mathcal{DTMA}$ [3] and it would be interesting to see whether $\mathcal{DTRA}$ and $\mathcal{DTSA}$ lay properly between them. The next theorem shows that this is not the case.

Theorem 5 $\mathcal{DTMA} = \mathcal{DTRA} = \mathcal{DTSA}$.

Proof. First note that the Rabin and the Streett conditions are complementary for deterministic automata. Thus, $\mathcal{DTRA} = \overline{\mathcal{DTSA}}$. Since $\mathcal{DTMA}$ is closed under complementation, it suffices to show that given a DTMA $A = \langle \Sigma, Q, Q_0, X, T, \mathbf{F} \rangle$, we can obtain an equivalent DTRA $A' = \langle \Sigma, Q', Q'_0, X', T', \mathbf{P} \rangle$. Since it can be shown that $\mathcal{DTRA}$ is closed by union, we can further assume that $|\mathbf{F}| = 1$. Let $C = \{c_0, c_1, \ldots, c_{k-1}\}$ be the only set in $\mathbf{F}$. Then, A' consists, loosely, of k copies of A, as follows. $Q' = Q \times \{0, 1, \ldots, k-1\}$, $Q'_0 = \{(q_0, 0)\}$, where q_0 is the only location in Q_0 and $X' = X$. The transition set is defined in such a way that any run jumps from copy i to copy $(i+1) \pmod{k}$ whenever it reaches location c_i. That is, we define $\langle q, q', a, \lambda, \delta \rangle$ to be in T' iff there is $\langle s, s', a, \lambda, \delta \rangle \in T$, $q = (s, i)$, $q' = (s', j)$, *and*:

1. $i = j$ and $s \neq c_i$; or
2. $j = (i+1) \pmod{k}$ and $s = c_i$.

Let $D = \{(s, i) \mid s \in C\}$. The desired acceptance condition is $\mathbf{P} = \{(\{(c_0, 0)\}, Q' \setminus D)\}$. □

Table 2 summarizes some of the results in the theory of timed automata. It is interesting to note that in a analogue table for ω-automata all classes would collapse into $\mathcal{DMA}$, with the exception of $\mathcal{DBA}$. The results concerning the inclusion problem are all corollaries of the results for the universality problem. The symbol $\bigcup$ stands for proper containment.

6 Conclusions

We discussed the concept of almost determinism, which has no especial importance in the theory of ω-automata, and showed that it plays an interesting role in the theory of timed automata. From Section 3, almost deterministic timed automata is a proper subclass of nondeterministic timed automata, and seems to be *much* less expressive. The main

class	union	inters.	compl.	empt.	universality	inclusion
$\mathcal{T}(\mathcal{M}\|\mathcal{R}\|\mathcal{S}\|\mathcal{B})\mathcal{A}$	closed	closed	not cl.	decid.	Π^1_2, Π^1_1-hard	Π^1_2, Π^1_1-hard
$\bigcup$						
$\mathcal{ADT}(\mathcal{M}\|\mathcal{R}\|\mathcal{S}\|\mathcal{B})\mathcal{A}$	closed	closed	not cl.	decid.	Π^1_1-complete	Π^1_1-complete
$\bigcup$						
$\mathcal{DT}(\mathcal{M}\|\mathcal{R}\|\mathcal{S})\mathcal{A}$	closed	closed	closed	decid.	decidable	decidable
$\bigcup$						
$\mathcal{DTBA}$	closed	closed	not cl.	decid.	decidable	decidable

Table 2: A summary of results in the theory of timed automata

significance of this result is related to the open question about the analytical complexity of testing universality of nondeterministic timed automata. In [3], the authors have left this problem open. In light of the results of Sections 3 and 4, it would be surprising if that problem comes out to be Π^1_1-complete. For then it would be recursively isomorphic [15] to the problem of testing universality of almost deterministic timed automata.

On the other hand, it would be even more surprising if that problem were shown to be Π^1_2-complete. In this case, it would be isomorphic to the universality problem for two other known formalisms which are a lot more expressive: nondeterministic ω-Turing machines [9]; and recursive infinite-state ω-automata [17]. There remains the possibility of the problem being in $\Pi^1_2 \setminus \Sigma^1_2$ while not complete for Π^1_2, or else, being in $\Delta^1_2 \setminus \{\Pi^1_1 \cup \Sigma^1_1\}$. This would be, perhaps, the most surprising situation because it would contradict the generally observed phenomenon [15, p. 330] that naturally defined problems, when inside the hierarchies of the undecidable, happen to be complete for some of their levels.

References

[1] R. Alur. Timed automata. In *CAV'99*, 1999. LNCS 1633.

[2] R. Alur and D. Dill. Automata for modeling real-time systems. In *ICALP'90*, pages 322–355. Springer-Verlag, 1990. LNCS 443.

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

[4] E. Asarin and O. Maler. Achilles and the tortoise climbing up the arithmetical hierarchy. *J. of Comp. and Sys. Sci.*, 57:389–398, 1998.

[5] Daniele Beauquier. Pumping lemmas for timed automata. In *FoSSaCS'1998*, pages 81–94. Springer-Verlag, 1998. LNCS 1378.

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

[7] O. Bournez. Achilles and the tortoise climbing up the hyper-arithmetical hierarchy. *Theoretical Computer Science*, 210:21–71, 1999.

[8] P. Bouyer and A. Petit. Decomposition and composition of timed automata. In *ICALP'99*, pages 210–219, 1999. LNCS 1466.

[9] J. Castro and F. Cucker. Nondeterministic ω-computations and the analytical hierarchy. *Zeitschr. f. math. Logik und Grundlagen d. Math.*, 35:333–342, 1989.

[10] Y. Choueka. Theories of automata on ω-tapes: A simplified approach. *Journal of Computer and System Sciences*, 8:117–141, 1974.

[11] C. Courcoubetis and M. Yannakakis. The complexity of probabilistic verification. *Journal of the ACM*, 42(4):857–907, 1995.

[12] D. Harel, A. Pnueli, and J. Stavi. Propositional dynamic logic of nonregular programs. *J. Comput. System Sci.*, 26:222–243, 1983.

[13] T. Henzinger, J. Raskin, and P. Schobbens. The regular real-time languages. In *ICALP'98*, pages 580–591, 1998. LNCS 1443.

[14] J. Miller. Decidability and complexity results for timed automata and semi-linear hybrid automata. In *Hybrid Systems'2000*, pages 296–309. Springer-Verlag, 2000. LNCS 1790.

[15] H. Rogers. *Theory of Recursive Functions and Effective Computability*. The MIT Press, 1987.

[16] S. Safra. On the complexity of ω-automata. In *Proc. of the 29th IEEE Foundations of Computer Science*, pages 319–327, October 1988.

[17] A. P. Sistla. On verifying that a concurrent program satisfies a nondeterministic specification. *Information Proc. Letters*, 32:17–23, 1989.