Electronic Notes in Theoretical Computer Science 42 (2001)
URL: http://www.elsevier.nl/locate/entcs/volume42.html 16 pages

# A Model of Finite Automata on Timed $\omega$-Trees

Salvatore La Torre [a,b] and Margherita Napoli [b]

[a] *University of Pennsylvania*

[b] *Università degli Studi di Salerno*

sallat,napoli@unisa.it

**Abstract**

In the last decade Alur and Dill introduced a model of automata on timed $\omega$-sequences which extends the traditional models of finite automata. In this paper we present a theory of timed $\omega$-trees which extends both the theory of timed $\omega$-sequences and the theory of $\omega$-trees. Main motivation is to introduce a new way of specifying real-time systems and provide tools for studying decidability problems in the field of quantitative temporal logics. We study properties and decision problems of the obtained classes of timed $\omega$-tree languages. Our main result is the decidability of the emptiness problem for all the classes of timed $\omega$-tree automata we consider.

## 1 Introduction

In the last decades, a significant amount of literature has been devoted to the theory of finite automata on $\omega$-sequences and $\omega$-trees. In the sixties with their pioneering works Büchi [3], McNaughton [11], and Rabin [12] introduced this theory, which, more recently, turned out to be an important source of tools for synthesis and verification of nonterminating computer programs. A recent survey on automata on infinite objects is [15]. Connections with Temporal Logic have been particularly successful [4,7,16,17]. The main results in this perspective consist of reducing temporal logic problems to decidable problems in the automata theory field. To model and verify real-time systems (that is, systems that interact with physical processes and whose correct running crucially depends upon real-time considerations) finite automata, as well as other specification tools, have been powered with clocks in order to explicitly consider time. In [2] a model of finite automata on timed $\omega$-sequences is proposed and a theory of timed languages is developed. In this model time grows continuously in the range of the positive real numbers. Here we present an extension to timed $\omega$-trees of previously existing automata models and, as in [2], we assume that time is dense. Our aim is the introduction of a new formalism to specify real-time systems and provide a robust theory for

studying decidability problems of branching-time temporal logics with timing constraints. Branching-time logics are suitable for reasoning about nondeterminism which, in turn, is useful to model concurrent programs (nondeterministic interleaving of atomic processes). Moreover, they provide an existential path quantifier, which allows to express lower bounds on nondeterminism and concurrency, particularly helpful in applications such as program synthesis. Furthermore, inevitability of a predicate and possibility of a predicate can be distinguished and the closure under semantic negation of the logic is ensured. See also [4,6]. We consider timed $\omega$-trees, i.e. $\omega$-trees in which a real-valued time of occurrence is associated with each node, and introduce finite automata on timed $\omega$-trees. We obtain various models by considering both deterministic and nondeterministic paradigms and both Muller and Büchi acceptance conditions. We prove that, differently from the timed $\omega$-sequences, for timed $\omega$-trees the Muller acceptance condition turns out to be strictly stronger than the Büchi one, and the nondeterministic Büchi and the deterministic Muller acceptance conditions result to be not comparable. Moreover, we prove that all the classes are closed under both union and intersection, but they are not closed under complementation. The nondeterministic classes turn out to be closed also under concatenation and $\omega$-iteration. As regards decision problems, we prove the decidability of the emptiness problem for these models: this result is important in connection to the application to temporal logic [10]. We also show that the equivalence problem is undecidable for nondeterministic timed tree automata while it is decidable for the deterministic ones. Finally, we introduce the concept if highly-deterministic timed tree automaton and prove that a language accepted by a timed tree automaton is not empty if and only if this automaton contains a highly-deterministic timed tree automaton. Since our proof is not constructive this result cannot be used to obtain an alternative proof of the decidability of the emptiness problem. Anyway, it is still interesting since it can be used to relate timed tree automata to timed graphs so reducing the finite satisfiability of TCTL [1] to the emptiness problem of Büchi tree automata (see [9]).

## 2 The model

In this section we introduce the concept of timed $\omega$-tree and timed $\omega$-tree automaton and define various models of automata, by considering both deterministic and nondeterministic paradigms and different conditions on the acceptance of a timed $\omega$-tree.

Let $\Sigma$ be an alphabet and $dom(t)$ be a subset of $\{1,\dots,k\}^*$, for a positive integer $k$, with the properties:

- if $wj \in dom(t)$, then $wi \in dom(t)$ for all $i$ such that $1 \leq i < j$;
- if $w \in dom(t)$, then there exists $i \in \{1,\dots,k\}$ such that $wi \in dom(t)$;
- if $w \in dom(t)$ and $w = ui$, with $i \in \{1,\dots,k\}$, then $u \in dom(t)$.

A $\Sigma$-*valued* $\omega$-*tree* is a mapping $t : dom(t) \longrightarrow \Sigma$. Each $w \in dom(t)$ is called *a node of* $t$ , or simply *a node*. Given a node $w$, we denote with $deg(w)$ the arity of $w$, that is $deg(w) = max\{j \mid wj \in dom(t)\}$, and with $pre(w)$ the set of the prefixes of $w$. Moreover, a *path* in $t$ is a maximal subset of $dom(t)$ linearly ordered by the prefix relation. Often, we will denote a path with the ordered sequence of its nodes, that is, given a path $\pi$ we denote it as $\pi = v_0, v_1, v_2, \ldots$ where $v_0$ is $\varepsilon$. With $In(t|\pi)$ we denote the set of the symbols labelling infinitely many nodes on the path $\pi$ in $t$. Let $\Re_+$ be the set of the non negative real numbers. A *timed* $\Sigma$-*valued* $\omega$-*tree* is a pair $(t, \tau)$ where $t$ is a $\Sigma$-valued $\omega$-tree and $\tau$, called *time tree*, is a mapping from $dom(t)$ into $\Re_+$ with the properties:

- positiveness: $\tau(w) > 0 \ \forall w \in dom(t) - \{\varepsilon\}$ and $\tau(\varepsilon) \geq 0$;
- progress: $\forall$ *path* $\pi$ and $\forall\ x \in \Re_+ \ \exists\ w \in \pi$ such that $\sum_{v \in pre(w)} \tau(v) \geq x$.

We consider timed $\omega$-trees as objects accepted by a finite-state automaton. We assume that the nodes of an $\omega$-tree becomes available as the time elapses, that is at a given time only a finite portion of the $\omega$-tree is available for reading. Then, we capture this situation by labelling each node of a timed $\omega$-tree by a pair (symbol, real number). The real number is (except for the root $\varepsilon$, where this number is assumed to be the absolute time of occurrence) the time which has elapsed since the parent node has been scanned at input. The positiveness property of a time tree implies that a positive delay exists between any two consecutive nodes. Progress requirement guarantees that infinitely many events (i.e. nodes appearing at input) cannot occur in a finite slice of time (*nonzenoness*). Given a timed $\omega$-tree $(t, \tau)$ and a node $w$, we denote with $\gamma_w$ the time at which $w$ is available at input, that is $\gamma_w = \sum_{v \in pre(w)} \tau(v)$. Furthermore, we denote with $T_\Sigma^k$ the set of the $\Sigma$-valued timed $\omega$-trees $(t, \tau)$ with $dom(t) \subseteq \{1, \ldots, k\}^*$. From now on, in this paper we use the term *tree* to refer to a $\Sigma$-valued $\omega$-tree for some alphabet $\Sigma$ and the term *timed tree* to refer to a timed $\Sigma$-valued $\omega$-tree. Moreover, a (timed) tree language is any set of (timed) trees.

Now, we introduce an automaton recognizing timed tree languages. It is similar to the one defined in [2] for timed $\omega$-sequences and it models a system with only one (real-valued) clock that scans the time for the whole system. A finite set of *clock variables* (also said simply *clocks*) are used for testing timing constraints on which state transitions depend. Each clock can be seen as a chronograph synchronized with the system clock. Their values can be read or set to zero (reset): after a reset, a clock restarts automatically. In the automaton the timing constraints are expressed by the clock constraints. Let $C$ be a set of clocks, the set of clock constraints $\Phi$(C) contains:

- $x \leq y + c$, $x \geq y + c$, $x \leq c$ and $x \geq c$ where $x, y \in C$ and $c$ is a rational number;
- $\neg\delta$ and $\delta_1 \wedge \delta_2$ where $\delta, \delta_1, \delta_2 \in \Phi(C)$.

Furthermore, a *clock interpretation* is a mapping $\nu : C \longrightarrow \Re_+$. If $\nu$ is a clock

interpretation, $\lambda$ is a set of clocks and $d$ is a real number, we denote with $[\lambda \to 0](\nu + d)$ the clock interpretation that for each clock $x \in \lambda$ gives 0 and for each clock $x \notin \lambda$ gives the value $\nu(x) + d$. A *nondeterministic timed tree transition table* is the 5-tuple ($\Sigma$,S,$S_0$,$\Delta$,C), where:

- $\Sigma$ is an alphabet;
- S is a finite set of states;
- $S_0 \subseteq$S is the set of starting states;
- $C$ is a finite set of clocks;
- $\Delta$ is a finite subset of $\bigcup_{k \geq 0}(S \times \Sigma \times S^k \times (2^C)^k \times \Phi(C))$.

A timed tree transition table is *deterministic* if $|S_0| = 1$ and for each pair of different tuples $(s, \sigma, s_1, \ldots, s_k, \lambda_1, \ldots, \lambda_k, \delta)$ and $(s, \sigma, s'_1, \ldots, s'_k, \lambda'_1, \ldots, \lambda'_k, \delta')$ in $\Delta$, $\delta$ and $\delta'$ are inconsistent (i.e., $\delta \wedge \delta' = false$ for all clock interpretations). Informally, a transition rule $(s, \sigma, s_1, \ldots, s_k, \lambda_1, \ldots, \lambda_k, \delta) \in \Delta$ can be described as follows. Suppose that when the system entered the state $s$ the clock values were given by $\nu$ and after a delay $d$ the symbol $\sigma$ is ready at input. The system can actually take the transition $(s, \sigma, s_1, \ldots, s_k, \lambda_1, \ldots, \lambda_k, \delta)$ if the current clock evaluation (i.e. $\nu + d$) satisfies the clock constraint $\delta$. As a consequence of the transition, the system will enter the states $s_1, \ldots, s_k$ with clock values given respectively by $[\lambda_1 \to 0](\nu + d), \ldots, [\lambda_k \to 0](\nu + d)$. Then, a timed tree transition table $(\Sigma, S, S_0, \Delta, C)$ associates to each node of a timed tree a state belonging to $S$ and a clock interpretation, according to the transition rules in $\Delta$. Formally, the behaviour of a timed tree transition table is captured by the following definition. Let A=($\Sigma$, S, $S_0$, $\Delta$, C) be a (deterministic or nondeterministic) timed tree transition table and (t,$\tau$) be a timed tree. A *run* of A on (t,$\tau$) is a pair (r,$\nu$), where:

- r:dom(t)$\longrightarrow$S and $\nu$:dom(t)$\longrightarrow \Re_+^C$;
- $r(\varepsilon) \in S_0$ and $\nu(\varepsilon) = \nu_0$ where $\nu_0$(x)=0 $\forall x \in C$;
- $\forall w \in dom(t)$, $k = deg(w)$: (r(w),t(w),r(w1),...,r(wk),$\lambda_1, \ldots, \lambda_k$,$\delta$)$\in\Delta$, $\nu$(w)+$\tau$(w) fulfils $\delta$ and $\nu$(wi)= $[\lambda_i \to 0]$ ($\nu$(w)+$\tau$(w)) $\forall i \in \{1, \ldots, k\}$.

Clearly, deterministic transition tables have at most one run for each timed tree. Given a transition table we define a timed tree automaton by specifying the acceptance conditions: obviously, the runs of an automaton are those of the corresponding transition table. A *nondeterministic (resp. deterministic) Büchi timed tree automaton* is a 6-tuple A=($\Sigma$,S,$S_0$,$\Delta$,C,F), where ($\Sigma$,S,$S_0$,$\Delta$,C) is a nondeterministic (resp. deterministic) timed tree transition table and F$\subseteq$S is the set of the final states. A timed tree (t,$\tau$) is accepted by a Büchi timed tree automaton A if an only if there is a run (r,$\nu$) of A on (t,$\tau$) such that $In(r|\pi) \cap F \neq \emptyset$ for each path $\pi$ in r. The language accepted by $A$, denoted by $T(A)$, is defined as the set $\{(t, \tau) \mid (t, \tau)$ is accepted by $A\}$. We define a nondeterministic (resp. deterministic) Muller timed tree automaton

analogously. The only changes needed are: an accepting family $\Phi \subseteq 2^S$ for the set of final states $F$ and the new acceptance condition "$In(r|\pi) \in \Phi$" for "$In(r|\pi) \cap F \neq \emptyset$". Hence, we have four classes of timed tree automata and we denote them (and the corresponding classes of languages) with the abbreviations: TMTA (nondeterministics Muller timed tree automata), TBTA (nondeterministic Büchi timed tree automata), DTMTA (deterministic Muller timed tree automata) and DTBTA (deterministic Büchi timed tree automata). Moreover, we denote the classes of (deterministic) Muller tree automata (and the corresponding classes of languages) with (D)MTA and (deterministic) Büchi tree automata (and the corresponding classes of languages) with (D)BTA. The classes MTA and BTA are treated in [15].

## 3 From timed to untimed languages

In this section we give two theorems that relate timed tree languages and tree languages. These results are crucial, but their proofs are similar to those given in [2] for $\omega$-sequence languages, so some details will be omitted. We start with the definition of the so-called *Untime* operator. Let $T$ be a timed tree language, *Untime(T)* is $\{t \mid (t,\tau) \in T \text{ for some time tree } \tau\}$. We say that a clock $x_0$ of a timed tree automaton $A$ is *nondivergent* if and only if for all $(t,\tau) \in T(A)$, for all runs (r,$\nu$) of $A$ on $(t,\tau)$, and for all paths $\pi$ in $t$ there are $u_1 < v_1 < u_2 < v_2 <\ldots$ in $\pi$ such that $\nu(u_h)(x_0) = 0$ and $\nu(v_h)(x_0) \geq 1$, for all $h \geq 1$.

The following result holds.

**Lemma 3.1** *Let $A$ be a timed tree automaton in TMTA (resp. TBTA, DTMTA and DTBTA), then there is a timed tree automaton $A'$ in TMTA (resp. TBTA, DTMTA and DTBTA) with a nondivergent clock such that $T(A) = T(A')$.*

**Proof :** Let $A = (\Sigma, S, S_0, \Delta, C, \Phi)$ be a nondeterministic Muller timed tree automaton. We define the non deterministic Muller tree automaton $A' = (\Sigma, S \times \{0,1\}, S_0 \times \{0\}, \Delta', C', \Phi')$, where:

- $C' = C \cup \{x_0\}$ with $x_0 \notin C$;
- $\Delta'$ contains $((s,0), \sigma, (s_1,0), \ldots, (s_k,0), \lambda_1, \ldots, \lambda_k, \delta \wedge (x_0 < 1))$, $((s,0), \sigma, (s_1,1), \ldots, (s_k,1), \lambda_1, \ldots, \lambda_k, \delta \wedge (x_0 \geq 1))$ and $((s,1), \sigma, (s_1,0), \ldots, (s_k,0), \lambda_1 \cup \{x_0\}, \ldots, \lambda_k \cup \{x_0\}, \delta)$, for all rules $(s, \sigma, s_1, \ldots, s_k, \lambda_1, \ldots, \lambda_k, \delta) \in \Delta$;
- $\Phi' = \{Y \subseteq S \times \{0,1\} \mid \rho_1(Y) \in \Phi\}$, where $\rho_1$ is the projection of the first component of a pair.

Note that the constraints on the clock $x_0$ affect only the second component of the states of $A'$. The first component, which is relevant for the acceptance, follows the transition rules of $A$. Thus, we have that $T(A) = T(A')$ and, due to the progress property of timed trees, the clock $x_0$ is obviously nondivergent. Furthermore, if $A$ is in TBTA, the Büchi automaton $A'$ is obtained by considering as set of final states the set $\{s \in S \times \{0,1\} \mid \rho_1(s) \in F\}$, where $F$ is the

set of final states of $A$. The above constructions preserve the determinism, hence it also holds for $A$ in DTMTA or DTBTA. □

For traditional automata the transitions are determined by the current state and the current symbol at input. With the introduction of the time, the transitions are also influenced by clock values. Thus, the concept of state is now replaced by the concept of *extended state* $\langle s, \nu \rangle$, i.e. a state $s$ of the automaton together with the values of the clocks given by the clock interpretation $\nu$. For a timed tree automaton the number of clock interpretations is infinite. However, they can be partitioned in a finite number of equivalence classes, called *clock regions*, so that all the clock interpretations in an equivalence class satisfy the same set of clock constraints of the considered transition table. To formalize this notion, we introduce first some notations. For $h \in \Re_+$ and a natural $k$, we denote with $\lfloor h \rfloor_k$ the integer $n$ and with $fract_k(h)$ the real $m$ such that $h = n\frac{1}{k} + m$ and $0 \leq m < k$. Given a timed transition table $A = (\Sigma, S, S_0, \Delta, C)$, let $u$ be the least common denominator among all the rational numbers in the clock constraints of $A$ and for all clocks $x \in C$, let $c_x$ be the largest integer $c$ such that $(x \leq cu)$ or $(x \geq cu)$ is a subformula of some clock constraint in $\Delta$. The *region equivalence*, denoted by $\simeq$, is defined as the equivalence relation over the pairs of clock interpretations such that $\nu \simeq \nu'$ if and only if the following conditions hold:

- for $x \in C$, either $\lfloor \nu(x) \rfloor_u = \lfloor \nu'(x) \rfloor_u$ or both $\nu(x) > c_x u$ and $\nu'(x) > c_x u$;
- for $x, y \in C$ with $\nu(x) \leq c_x u$ and $\nu(y) \leq c_y u$, $fract_u(\nu(x)) \leq fract_u(\nu(y))$ if and only if $fract_u(\nu'(x)) \leq fract_u(\nu'(y))$;
- for $x \in C$ with $\nu(x) \leq c_x u$, $fract_u(\nu(x)) = 0$ if and only if $fract_u(\nu'(x)) = 0$.

Then a *clock region* is an equivalence class of clock interpretations induced by $\simeq$. Note that the definition of clock region we introduced is equivalent to the one introduced in [2] for timed automata on $\omega$-sequences. In [2], the authors proved that the number of the clock regions is upper bounded by $|C|!2^{|C|}\Pi_{x \in C}(2c_x + 2)$. Obviously, this upper bound holds here, too. Given a clock interpretation $\nu$, $[\nu]$ denotes the clock region containing $\nu$. From the definition of region equivalence, it holds that if $\nu$ satisfies a clock constraint $\delta$ then it is so for all $\nu' \in [\nu]$. Then, we consistently say that $[\nu]$ satisfies a clock constraint $\delta$ if $\nu$ satisfies $\delta$. Moreover, a clock region $\alpha'$ is said to be a *time-successor* of a clock region $\alpha$ if and only if for all $\nu \in \alpha$ there is a positive $h \in \Re_+$ such that $\nu + h \in \alpha'$. Let $A = (\Sigma, S, S_0, \Delta, C)$ be a timed tree transition table, the corresponding *Region Automaton* $R(A)$ is a transition table defined by:

- the set of states $R(S) = \{\langle s, \alpha \rangle \mid s \in S \text{ and } \alpha \text{ is a clock region for } A\}$;
- the set of starting states $R(S_0) = \{\langle s_0, \alpha_0 \rangle \mid s_0 \in S_0 \text{ and for } x \in C, \alpha_0 \text{ satisfies } x = 0\}$;
- the transition rules $R(\Delta)$ defined as: $(\langle s, \alpha \rangle, \sigma, \langle s_1, \alpha_1 \rangle, \ldots, \langle s_k, \alpha_k \rangle) \in R(\Delta)$ if and only if $(s, \sigma, s_1, \ldots, s_k, \lambda_1, \ldots, \lambda_k, \delta) \in \Delta$ and there is a time-successor

$\alpha'$ of $\alpha$ such that $\alpha'$ satisfies $\delta$ and $\alpha_i = [\lambda_i \to 0]\alpha'$ for all $i \in \{1, \ldots, k\}$.

The Region Automaton is the transition table of a tree automaton accepting the Untime of the language accepted by a given timed tree automaton. Let us consider a nondeterministic Muller timed tree automaton $A = (\Sigma, S, S_0, \Delta, C, \Phi)$ accepting $T$ and having a nondivergent clock $x_0$. We define the Muller tree automaton $A^U = (\Sigma, S^U, S_0^U, \Delta^U, \Phi^U)$, where:

(i) $S^U$ contains $\langle s, \alpha, h\rangle$ where $s \in S$, $\alpha$ is a clock region and $h \in \{0, 1, 2\}$;

(ii) $S_0^U$ contains $\langle s, \alpha, 2\rangle$ where $\langle s, \alpha\rangle \in R(S_0)$;

(iii) $(\langle s, \alpha, h\rangle, \sigma, \langle s_1, \alpha_1, h_1\rangle, \ldots, \langle s_k, \alpha_k, h_k\rangle) \in \Delta^U$ if and only if $(\langle s, \alpha\rangle, \sigma, \langle s_1, \alpha_1\rangle, \ldots, \langle s_k, \alpha_k\rangle) \in R(\Delta)$ and $\forall i \in \{1, \ldots, k\}$:

$$h_i = \begin{cases} 1 \; if \; \alpha_i \; satisfies \; x_0 \geq 1 \\ 2 \; if \; \alpha_i \; satisfies \; x_0 = 0 \; and \; h = 1 \\ 0 \text{ otherwise.} \end{cases}$$

(iv) $\Phi^U = \{Y \subseteq S^U \mid \rho_1(Y) \in \Phi \text{ and } \rho_3(Y) \cap \{2\} \neq \emptyset\}$ with $\rho_i$ the projection of the $i$-th component of a triple.

If the automaton $A$ is a TBTA, then a BTA analogous to $A^U$ can be defined. In particular the unique difference is that the set of the final states becomes $F^U = \{s \in S^U \mid \rho_1(s) \in S \text{ and } \rho_3(s) = 2\}$. We call again this automaton $A^U$.

The following theorem states the relationship between a timed tree language $T$ and $Untime(T)$.

**Theorem 3.2** *If $T$ is a timed tree language in TMTA (resp. in TBTA) then Untime(T) is in MTA (resp. in BTA).*

**Proof :** Let $A$ be a TMTA and $A^U$ be defined as above. We show that $T(A^U) = Untime(T(A))$. Let $(r, \nu)$ be an accepting run of $A$ on timed tree $(t, \tau)$, we define $r'$ as $r'(w) = \langle r(w), [\nu(w)]\rangle$ for every $w \in dom(t)$. By the definition of $A^U$ it is easy to verify that $r'$ is an accepting run of $A^U$ on $t$. Vice-versa let $r'$ be an accepting run of $A^U$ on a tree $t$, with $r'(w) = \langle s_w, \alpha_w\rangle$ for all $w \in dom(t)$. We observe that, since $r'$ is an accepting run, for all paths $\pi$ there are $u_1 < v_1 < u_2 < v_2 < \ldots$ in $\pi$ such that $\alpha_{u_h}$ satisfies $x_0 = 0$ and $\alpha_{v_h}$ satisfies $x_0 \geq 1$ for all $h \in \{1, 2, 3, \ldots\}$. For all $w \in dom(t)$, let $\tau(w) = d$ where $d$ is such that $k = deg(w)$, $(s_w, t(w), s_{w1}, \ldots, s_{wk}, \lambda_{w1}, \ldots, \lambda_{wk}, \delta) \in \Delta$, $\alpha_{wi} = [\lambda_i \to 0](\alpha_w + d)$ for $i = 1, \ldots, k$ and $\alpha_w + d$ satisfies $\delta$. By the definition of $R(\Delta)$, $\tau(w)$ is always defined and positive. Moreover, the above property for the paths of an accepting run of $A^U$ guarantees that $\tau$ is a time tree. An accepting run $(r, \nu)$ of $A$ on $(t, \tau)$ can be obtained in this way: $r(w) = s_w$ and $\nu(w) \in \alpha_w$. So, there exists an accepting run of $A$ on $t$ if and only if there exist $\tau$ and an accepting run of $A^U$ on $(t, \tau)$. Hence, $T(A^U) = Untime(T(A))$. The result for Büchi automata can be proved in a similar way. □

Given a symbol $c \in \Sigma$, we define a set $Q_c$ of the states $\langle s, \alpha\rangle$ of $R(A)$

such that $A^U$, starting from $\langle s, \alpha, 2\rangle$, accepts some timed tree whose root is labelled by $c$. We denote this set as the set of the $c$-starting states of $A$. The following theorem gives a strong result concerning the relation between a particular class of timed languages and their corresponding Untime.

**Theorem 3.3** *Let $T$ be a tree language, then:*

- $\{(t,\tau) \mid t \in T$ *and* $\tau$ *is a time tree*$\}$ *is in (D)TBTA if and only if $T$ is in (D)BTA;*
- $\{(t,\tau) \mid t \in T$ *and* $\tau$ *is a time tree*$\}$ *is in (D)TMTA if and only if $T$ is in (D)MTA.*

**Proof :** The "if" part of all the assertions is immediate since the automaton accepting $\{(t,\tau) \,|\, t \in T$ and $\tau$ is a time tree$\}$ is obtained directly from the one accepting $T$ by simply adding the constant true as clock constraint in all the transitions. The "only if" part for the nondeterministic classes comes from Theorem 3.2. For the deterministic classes, we consider first the language $T' = \{(t,\tau) \,|\, t \in T,\ \tau(w) = 1 \,\forall w \in dom(t)\}$ which is in the same class of $\{(t,\tau) \,|\, t \in T$ and $\tau$ is a time tree$\}$. In fact, a timed tree automata for $T'$ can be designed starting from the one accepting $\{(t,\tau) \,|\, t \in T$ and $\tau$ is a time tree$\}$ by simply considering a new clock variable, say $x$, and adding both the constraint $x = 1$ ad the reset of $x$ in all the transitions. Then, by the same construction used in Theorem 3.2, we obtain an automaton accepting $T$. Since in $T'$ the delay associated to all nodes is fixed, the above construction preserves the determinism and, then, the theorem is proved. □

Note that in general, for a timed tree language $T' \in (D)TBTA$ (resp. $T' \in (D)TMTA$), it is not true that $Untime(T') \in (D)BTA$ (resp. $Untime(T') \in (D)MTA$). The following counter-example is due to Alur and Dill [2].

**Example 3.4** *Let $T$ be the language $\{(t,\tau) \in T^k_{\{a\}} \mid$ for all paths $\pi$ in $t$ and all $v \in \pi$ there exists $w \in \pi$ such that $\sum_{u \in pre(w) - pre(v)} \tau(u) = 1\}$. $Untime(T) \in DBTA$ while $T$ does not belong to any of the timed tree language classes we have introduced.*

## 4 Properties and decision problems

In this section we compare the different classes of timed tree languages. Then we state the closure of all the considered classes with respect to union and intersection and the nonclosure under complementation. Next, we prove the closure of the nondeterministic classes and the nonclosure of the deterministic ones under concatenation and $\omega$-iteration. Moreover, the decidability of the emptiness problem for all the classes under consideration is shown. Finally, we prove that the equivalence problem is decidable for the classes of timed tree languages recognized by deterministic automata while it is undecidable for the nondeterministic models; nevertheless, one can introduce a weaker notion of equivalence, called untimed equivalence, whose problem turns out to be

decidable in all classes.

**Language comparisons.** The relationships between the considered classes of languages are inherited from those between tree languages, by means of Theorem 3.3. It is known that BTA is strictly included in MTA [13]. The proof of this result can be easily modified to obtain DBTA $\subset$ DMTA. The following theorem complete the relationships among all the classes of tree languages.

**Theorem 4.1** *It holds that:*

(i) $DBTA \subset BTA \cap DMTA$;

(ii) *BTA and DMTA are not comparable;*

(iii) $BTA \cup DMTA \subset MTA$.

**Proof :**

"1." Let $T_1$ be the language $\{t \in T^1_{\{a,b\}} \mid \pi = \varepsilon, 1, 11, \ldots, 1^i, \ldots \text{ and } a \notin In(t|\pi)\}$. Note that $T_1$ is a set of $\omega$-sequences which is not accepted by a deterministic Büchi automaton and is accepted by a nondeterministic Büchi automaton (see also [15]). Moreover, the nondeterministic Büchi automata and the deterministic Muller automata on $\omega$-sequences are equivalent [11]. Thus, we have DBTA $\subset$ BTA $\cap$ DMTA.

"2." Let $k > 1$ and $T_2$ be the language $\{t \in T^k_{\{a,b,c\}} \mid \forall \text{ paths } \pi \text{ in } t,\ a \notin In(t|\pi)\}$. Rabin in [13] showed that $T_2$ is accepted by a deterministic Muller tree automaton but it is not accepted by a Büchi tree automaton. Thus, $T_2 \in$ DMTA $-$ BTA. Let $k > 1$ and $T_3$ be the language $\{t \in T^k_{\{a,b,c\}} \mid$ there exists a path $\pi$ in $t$ such that $b \in In(t|\pi)\}$. The language $T_3$ is accepted by a nondeterministic Büchi tree automaton that guesses a path $\pi$ on which it checks the infiniteness of the occurrences of the symbol "$b$" and accepts unconditionally on the other paths. Furthermore, by using a standard argument, similar for example to the one used to show the weakness of the deterministic frontier-to-root tree automata over finite trees compared to the nondeterministic ones (see [8]), we can prove that $T_3$ is not in DMTA. Thus, $T_3 \in$ BTA $-$ DMTA.

"3." Let $k > 1$ and $T_4$ be the tree language $\{t \in T^k_{\{a,b,c\}} \mid \forall \text{ paths } \pi \text{ in } t,$ $a \notin In(t|\pi)$ and there exists a path $\pi'$ in $t$ such that $b \in In(t|\pi')\}$. Note that $T_4 = T_2 \cap T_3$ and, then, from the closure under intersection of MTA we have $T_4 \in$ MTA. In order to show that $T_4 \notin$ BTA, let us suppose that there is a Büchi tree automaton $A$ accepting $T_4$ and let $A'$ be the automaton obtained from $A$ by replacing transition rules on symbol "$b$" with identical, except for the input symbol, transition rules on symbol "$c$". Thus, $A'$ would accept the language $\{t \in T^k_{\{a,c\}} \mid \forall \text{ paths } \pi \text{ in } t,\ a \notin In(t|\pi)\}$ which is not in BTA, hence $T_4 \notin$ BTA. By using a similar argument as that used for $T_3$, it can be shown that $T_4$ is not in DMTA and, then, BTA $\cup$ DMTA $\subset$ MTA. □

The next corollary extends the previous results to timed tree languages.

**Corollary 4.2** $DTBTA \subset TBTA \cap DTMTA$, $TBTA \cup DTMTA \subset TMTA$,

*TBTA and DTMTA are not comparable.*

**Proof :** By the usual constructions we have DTBTA $\subseteq$ DTMTA and TBTA $\subseteq$ TMTA. The proof is completed by Theorems 3.3 and 4.1. □

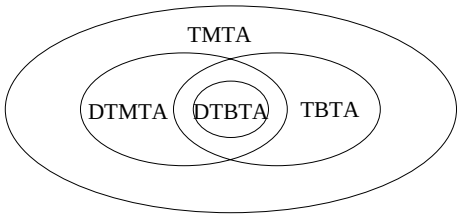


Fig. 1. Relationships between the classes TMTA, TBTA, DTMTA and DTBTA.

**Closure properties.**

**Theorem 4.3** *The classes DTBTA, DTMTA, TBTA and TMTA are closed under intersection and union, but they are not closed under complementation.*

**Proof :** The positive results are proved with the usual constructions having care of the timing features (see for example the constructions in [2]). The nonclosure under complementation is proved with counter-examples which are similar to the counter-example given in [2]. Let $T = \{(t,\tau) \in T^k_{\{a\}} \mid$ there exists a path $\pi$ in $t$ and $v, w \in \pi$ such that $\sum_{u \in pre(w)-pre(v)} \tau(u) = 1\}$. It can be seen that $T \in$TBTA, but its complement with respect to $T^k_{\{a\}}$ is not in TMTA. Finally, the language $\{(t,\tau) \in T^k_{\{a,b\}} \mid$ for all paths $\pi$ in $t$, a $\in In(t|\pi)\} \in$ DTBTA and its complement with respect to $T^k_{\{a,b\}}$ is not in DTMTA. □

In spite of the nonclosure under complementation of the considered classes we can prove that the complement of a language belonging to DTMTA is in TMTA. This result will be useful to prove the decidability of the equivalence problem for DTMTA. Let $\overline{\text{DTMTA}}$ be the set of languages obtained by complementation of the languages in DTMTA. It holds the following result.

**Theorem 4.4** *DTMTA*$\cup\overline{DTMTA} \subset$ *TMTA.*

**Proof :** Let us consider a DTMTA $A$ having exactly one run for each $(t,\tau) \in T^k_\Sigma$. The timed tree automaton $A'$, accepting the complement of $T(A)$, nondeterministically guesses a path on which it verifies that the accepting conditions of $A$ do not hold and accepts unconditionally on the other paths. Hence we have the containment. From the nonclosure of TMTA under complementation (Theorem 4.3), it follows that the containment is strict. □

The concatenation of two timed tree languages $T_1$ and $T_2$, denoted with $T_1 \cdot_c T_2$, is defined as the $\omega$-tree language obtained from timed trees in $T_1$ by replacing a timed tree of $T_2$ for each subtree rooted in a node labelled with the "first" occurrence of $c$ along a path. Note that different timed trees of $T_2$ can be substituted for different subtrees of a given $t \in T_1$. The $\omega$-iteration of a timed tree language $T$, denoted with $T^{\omega_c}$, is defined as the infinite iteration of the concatenation.

**Theorem 4.5** *The classes TBTA and TMTA are closed under concatenation and $\omega$-iteration.*

**Proof :** We start with the closure under concatenation for TBTA. Let $A_i = (\Sigma, S_i, S_i^0, \Delta_i, C_i, F_i)$ for $i = 1, 2$ be two nondeterministic Büchi timed tree automata with $S_1 \cap S_2 = \emptyset$ and $C_1 \cap C_2 = \emptyset$. Let $Q_c$ be the set of the $c$-starting states of $A_1$. Given a clock region $\alpha$ and a set of clocks $\lambda$, let $\delta(\alpha, \lambda)$ be the clock constraint $\delta_1 \vee \ldots \vee \delta_r$ where, given the clock regions $\alpha'_1, \ldots, \alpha'_r$ such that $\alpha = [\lambda \to 0]\alpha'_i + d$ for a $d \in \Re_+$, $\delta_i$ is the clock constraint equivalent to $\alpha'_i$ (in the sense that a clock interpretation $\nu$ satisfies $\delta_i$ if and only if $\nu$ is a clock interpretation in $\alpha'_i$). A nondeterministic Büchi timed tree automaton accepting $T(A_1)\cdot_c T(A_2)$ is $A = (\Sigma, S_1 \cup S_2, S^0, \Delta, C_1 \cup C_2, F_1 \cup F_2)$ where $S^0 = S_1^0 \cup S_2^0$ if $\exists s \in S_1^0$ and $\langle s, \alpha_0, 2\rangle \in Q_c$ (where $\alpha_0$ is the clock region containing the clock interpretation $\nu(x) = 0,\ \forall x \in C_1$) else $S^0 = S_1^0$, and $\Delta$ contains $\Delta_2$ and the set of the rules $(s, \sigma, s_1, \ldots, s_k, \lambda_1, \ldots, \lambda_k, \delta)$ such that:

- $\sigma \neq c$;
- there exist a $(s, \sigma, s'_1, \ldots, s'_k, \lambda'_1, \ldots, \lambda'_k, \delta') \in \Delta_1$, $0 \leq i_1, \ldots, i_h \leq k$, and regions $\alpha_{i_1}, \ldots, \alpha_{i_h}$, such that (1) $\langle s'_{i_j}, \alpha_{i_j}\rangle \in Q_c$, $s_{i_j} \in S_2^0$ and $\lambda_{i_j} = C_2$ $\forall j \in \{1, \ldots, h\}$, (2) $s_j = s'_j$ and $\lambda_j = \lambda'_j$ $\forall j \notin \{i_1, \ldots, i_h\}$ and (3) $\delta = \delta' \wedge \delta(\alpha_{i_1}, \lambda'_{i_1}) \wedge \ldots \wedge \delta(\alpha_{i_h}, \lambda'_{i_h})$.

When the automaton $A$ starts, it behaves as $A_1$. When $A_1$ enters an extended state $\langle s, \nu\rangle$ and $\langle s, [\nu]\rangle$ is a $c$-starting state, a possible root of a tree in $T(A_2)$ is processed, and then $A$ switches to $A_2$. Since the nodes on which the trees in $T_2$ are pasted are a priori unknown, a nondeterministic choice occurs on every node (including the root). The same construction holds for $A_1$ and $A_2$ in TMTA with the unique difference that the accepting family of $A$ is $\Phi = \Phi_1 \cup \Phi_2$, where $\Phi_i$ is the accepting family of $A_i$, $i = 1, 2$.

For the closure under $\omega$-iteration, we suppose that the automaton $A_1$ has the property that in its runs the starting states appear only at the root. A timed tree automaton accepting $(T(A_1))^{\omega_c}$ is $A' = (\Sigma - \{c\}, S_1, S_1^0, \Delta, C_1, F_1 \cup S_1^0)$ where $\Delta$ contains $(s, \sigma, s_1, \ldots, s_k, \lambda_1, \ldots, \lambda_k, \delta)$ such that $(s, \sigma, s'_1, \ldots, s'_k, \lambda'_1, \ldots, \lambda'_k, \delta') \in \Delta_1$, $\{\langle s'_{i_1}, \alpha_{i_1}\rangle, \ldots, \langle s'_{i_h}, \alpha_{i_h}\rangle\} \subseteq Q_c$ and: (1) $s_{i_j} \in S_1^0$ and $\lambda_{i_j} = C_1$ $\forall j \in \{1, \ldots, h\}$, (2) $s_j = s'_j$ and $\lambda_j = \lambda'_j$ $\forall j \notin \{i_1, \ldots, i_h\}$, and (3) $\delta = \delta' \wedge \delta(\alpha_{i_1}, \lambda'_{i_1}) \wedge \ldots \wedge \delta(\alpha_{i_h}, \lambda'_{i_h})$. If we consider Muller timed tree automata, in the previous construction some changes are needed. First, the set of states is $S_1 \times \{0, 1\}$. The second component of each state implements a binary counter that is incremented every time a starting state of $A_1$ is entered. As a consequence, the transition rules are suitably modified. Last, the accepting family is $\{X \subseteq S_1 \times \{0, 1\} \mid \rho_1(X) \in \Phi_1 \text{ or } \rho_2(X) = \{0, 1\}\}$ where $\rho_i$ projects the $i$-th component of a pair and $\Phi_1$ is the accepting family of $A_1$. □

The above results do not hold for the deterministic classes. A counterexample for the concatenation is given by the languages $T_1 = \{(t, \tau) \in T^k_{\{a,c\}} \mid \forall \text{ paths } \pi = \varepsilon, v_1, v_2, \ldots \exists i \text{ such that } t(v_i) = c\}$ and $T_2 = \{(t, \tau) \in T^k_{\{a\}} \mid \forall$

paths $\pi = \varepsilon, v_1, v_2, \ldots$ $\exists i$ such that $\sum_{j=1}^{i} \tau(v_j) = 1\}$. Thus, the language $T = T_1 \cdot_c T_2$ contains the timed $\{a\}$-valued trees having two cuts such that the delay between the nodes of the first cut and those of the second one is 1. A deterministic timed tree automaton accepting $T$ would have an unbounded number of clocks, thus $T$ is not in DTMTA, while the languages $T_1$ and $T_2$ are both in DTBTA. The language $\{(t, \tau) \in T^k_{\{a,c\}} \mid t(\varepsilon) = a$ and $\forall$ paths $\pi = \varepsilon, v_1, v_2, \ldots$ there are $i, j$ such that $i < j$, $t(v_h) = a$ for every $1 \leq h \leq i$, $t(v_j) = c$, and $\sum_{h=1}^{i} \tau(v_h) = 1\}$ gives the counter-example for the $\omega$-iteration.

| | DTBTA | DTMTA | TBTA | TMTA |
|---|---|---|---|---|
| Union | Yes | Yes | Yes | Yes |
| Intersection | Yes | Yes | Yes | Yes |
| Complementation | No | No | No | No |
| Concatenation | No | No | Yes | Yes |
| $\omega-$iteration | No | No | Yes | Yes |

Fig. 2. Summary of results on closure properties.

**Decision problems.**

**Theorem 4.6** *The emptiness problem is decidable for TMTA.*

**Proof :** Let T$\in$TMTA, we have that T is empty if and only if Untime(T) is empty. From the Theorem 3.2, Untime(T) is accepted by Muller tree automata. Since the emptiness problem is decidable for the tree languages accepted by Muller automata (see [14]), we have the decidability for TMTA. □

By Corollary 4.2 and the above theorem we have that the the emptiness problem is decidable also for DTBTA, DTMTA and TBTA.

**Theorem 4.7** *The equivalence problem is undecidable for TMTA and TBTA while it is decidable for DTMTA and DTBTA.*

**Proof :** The undecidability is inherited from the undecidability of the equivalence problem for the corresponding classes of timed $\omega$-sequence languages (see [2]). The decidability for the deterministic classes follows from Theorems 4.4 and 4.6. □

Let $A$ and $A'$ be two timed tree automata, we say that $A$ is *untimed equivalent* to $A'$ if and only if $Untime(T(A)) = Untime(T(A'))$. From the decidability of the equivalence problem for MTA and Theorem 3.2, we can state the following:

**Theorem 4.8** *The untimed equivalence problem is decidable for TMTA.*

| | DTBTA | DTMTA | TBTA | TMTA |
|---|---|---|---|---|
| Emptiness | Yes | Yes | Yes | Yes |
| Equivalence | Yes | Yes | No | No |
| Untimed equivalence | Yes | Yes | Yes | Yes |

Fig. 3. Summary of results on decision problems.

## 5 Highly-deterministic timed tree automata

In this section we define for timed tree automata a concept which captures some of the properties that regular trees have in the context of tree languages.

A timed tree automaton $A = (\Sigma, S, S_0, \Delta, C, F)$ is said to be *highly deterministic* if Untime$(T(A))$ contains a unique tree, and for $s \in S$, $e = (s, \sigma, s_1, \ldots, s_k, \lambda_1, \ldots, \lambda_k, \delta) \in \Delta$ and $e' = (s, \sigma', s'_1, \ldots, s'_h, \lambda'_1, \ldots, \lambda'_h, \delta') \in \Delta$ imply that $e = e'$. The second property of highly-deterministic timed tree automata simply states that there is at most one transition rule that can be executed in each location $s \in S$. Given a timed tree automaton $A = (\Sigma, S, S_0, \Delta, C, F)$, we say that a timed tree automaton $A' = (\Sigma, S', S'_0, \Delta', C, F')$ is *contained* in $A$ if $S' \subseteq S$, $S'_0 \subseteq S_0$, $\Delta' \subseteq \Delta$, and $F' \subseteq F$. Clearly, $T(A') \subseteq T(A)$ holds. We recall that a regular tree contains a finite number of subtrees. Given a timed tree automaton $A = (\Sigma, S, S_0, \Delta, C, F)$, and a regular run $r$ of $R(A)$ on a regular tree $t \in T(R(A))$, we define a *shrink* of $r$ and $t$ as the labelled directed finite graph $G = (V, E, lab)$ such that there is a mapping $\theta : dom(t) \longrightarrow V$ such that:

- for any $u, u' \in dom(r)$, $\theta(u) = \theta(u')$ implies that $deg(u) = deg(u')$, and for each $i = 1, \ldots, deg(u)$, $\theta(ui) = \theta(u'i)$;
- $E = \{(\theta(u), \theta(ui), i) \mid u \in dom(r) \text{ and } i \leq deg(u)\}$, and $(v, v', i) \in E$ is an edge from $v$ to $v'$ labelled by $i$;
- for $v \in V$, $lab(v) = (r(u), t(u))$ for any $u$ such that $v = \theta(u)$.

From the definition of regular tree, such a graph $G$ always exists. Thus, the following theorem holds.

**Theorem 5.1** *Given a timed tree automaton $A$, $T(A)$ is not empty if and only if there exists a highly-deterministic timed tree automaton contained in $A$.*

**Proof :** We consider first the forward direction. By hypothesis $T(A)$ is not empty, then Untime$(T(A))$ is also not empty. Thus there exist an accepting regular run $r$ of $R(A)$ and a corresponding regular tree $t \in$ Untime$(T(A))$ [14]. Let $G = (V, E, lab)$ be a shrink of $r$ and $t$, where $\varepsilon$ corresponds to $v_0$ and $lab(v_0) = (r(\varepsilon), t(\varepsilon))$. We define $A_{det}$ as the timed tree automaton $(\Sigma, S_{det}, \{s'_0\}, \Delta_{det}, C, S_{det})$ where:

- $\langle s'_0, [\nu_0] \rangle = r(\varepsilon)$ with $\nu_0(x) = 0$ for any $x \in C$;

- $S_{det} = \{s \mid \exists v \in V$ such that $lab(v) = (\langle s, \alpha \rangle, \sigma)\}$;
- a transition rule $(s, \sigma, s_1, \ldots, s_k, \lambda_1, \ldots, \lambda_k, \delta) \in \Delta$ belongs to $\Delta_{det}$ if and only if the sequence $(v, v_1, 1), \ldots, (v, v_h, h) \in E$ of all the edges from $v$ is such that (1) $h = k$, (2) $lab(v) = (\langle s, \alpha \rangle, \sigma)$, and $lab(v_i) = (\langle s_i, \alpha_i \rangle, \sigma_i)$ for $i = 1, \ldots, k$, and (3) there exists a $d > 0$ such that $(\alpha + d)$ satisfies $\delta$ and $\alpha_i = [\lambda_i \to 0](\alpha + d)$ for $i = 1, \ldots, k$.

Directly from the above definition we have that $\text{Untime}(T(A_{det})) = \{t\}$, and for each $s \in S_{det}$ there is only a transition rule that can be executed from $s$. Thus $A_{det}$ is a highly-deterministic timed tree automaton. Moreover, $A_{det}$ is contained in $A$, and thus we have proved that if $T(A)$ is not empty then there exists a highly-deterministic timed tree automaton contained in $A$. The converse direction is a direct consequence of the facts that any highly-deterministic timed tree automaton $A'$ accepts a non-empty language and $T(A') \subseteq T(A)$ since $A'$ is contained in $A$. □

For any highly-deterministic timed tree automaton $A$, it is easy to prove that there exists a highly-deterministic timed automaton $A'$ such that $T(A) = T(A')$ and for any transition rule $(s, \sigma, s_1, \ldots, s_k, \lambda_1, \ldots, \lambda_k, \delta)$ of $A'$ we have that $s_i \neq s_j$ for $i \neq j$. We call such an automaton a *graph-representable* timed tree automaton, since it corresponds to a labelled directed graph such that for any ordered pair of locations $(s, s')$ there is exactly an edge connecting $s$ to $s'$ in the graph. This does not hold in general for a highly-deterministic timed tree automaton. We can easily obtain $A'$ from $A$ by simply adding multiple copies of the $A$ locations that break the graph-representability property.

## 6 Conclusions

In this paper we have introduced a theory of finite automata on timed $\omega$-trees. We have considered both deterministic and nondeterministic paradigms and both Muller and Büchi acceptance conditions. We have studied the relationships among the various classes of languages, some closure properties, and decision problems. Concerning to this theory the main result is the decidability of the emptiness problem, which turned out to be extremely useful in obtaining decidability results in the field of dense-time temporal logics. In particular, in [10] the satisfiability problem of Stctl (a real-time extension of Ctl [5]) is reduced to the emptiness problem of Büchi automata on timed $\omega$-trees. Moreover, the result on highly-deterministic timed tree automata presented in Section 5 is used to relate timed tree automata to timed graphs and to reduce the finite satisfiability of Tctl [1] to the emptiness problem of Büchi automata on timed $\omega$-trees (see [9]).

## Acknowledgments

This research was partially supported by the M.U.R.S.T. in the framework of project Tosca, NSF Career award CCR97-34115 and SRC award 99-TJ-688.

## References

[1] R. Alur, C. Courcoubetis, and D.L. Dill. Model-checking in dense real-time. *Information and Computation*, 104(1):2–34, 1993.

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

[3] J.R. Büchi. On a decision method in restricted second-order arithmetic. In *Proceedings of the International Congress on Logic, Methodology, and Philosophy of Science 1960*, pages 1–12. Stanford University Press, 1962.

[4] O. Bernholtz, M.Y. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model checking. In *Proceedings of the 6th International Conference on Computer Aided Verification*, LNCS 818, pages 142–155. Springer-Verlag, 1994.

[5] E.A. Emerson and E.M. Clarke. Characterizing correctness properties of parallel programs as fixpoints. In *Proceedings of the 7-th International Colloquium on Automata, Languages and Programming*, LNCS 85, pages 169–181. Springer-Verlag, 1981.

[6] E.A. Emerson and J.Y. Halpern. Sometimes and not never revisited: On branching versus linear time. *Journal of the ACM*, 33(1):151–178, 1986.

[7] E.A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 995–1072. Elsevier Science Publishers, 1990.

[8] F. Gécseg and M. Steinby. *Tree Automata*. Akademíai Kiado, Budapest, 1984.

[9] S. LaTorre and M. Napoli. A decidable dense branching-time temporal logic. In *Proceedings of the 20th Conference on the Foundations of Software Technology and Theoretical Computer Science, FSTTCS'00*, LNCS 1974. Springer-Verlag, 2000.

[10] S. LaTorre and M. Napoli. Timed tree automata with an application to temporal logic. *To appear on Acta Informatica*, 2001.

[11] R. McNaughton. Testing and generating infinite sequences by a finite automaton. *Information and Control*, 9:521–530, 1966.

[12] M.O. Rabin. Decidability of second-order theories and automata on infinite trees. *Trans. Amer. Math. Soc.*, 141:1–35, 1969.

[13] M.O. Rabin. Weakly definable relations and special automata. *Mathematical Logic and Foundations of Set theory*, 1970.

[14] M.O. Rabin. Automata on infinite objects and Church's problem. *Trans. Amer. Math. Soc.*, 1972.

[15] W. Thomas. Automata on infinite objects. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 133–191. Elsevier Science Publishers, 1990.

[16] M.Y. Vardi. Nontraditional applications of automata theory. In *Proceedings of the International Symposium TACS'94*, LNCS 789, pages 575–597. Springer-Verlag, 1994.

[17] M.Y. Vardi and P. Wolper. Automata-theoretic techniques for modal logics of programs. *Journal of Computer and System Sciences*, 32:182–211, 1986.