Acta Informatica 38, 89–116 (2001)

# Timed tree automata with an application to temporal logic

**Salvatore La Torre**[1,2], **Margherita Napoli**[2]

[1] University of Pennsylvania, Department of Computer and Information Science, Philadelphia, PA 19104, USA

[2] Università degli Studi di Salerno, Dipartimento di Informatica ed Applicazioni, 84081 Baronissi, Italy (e-mail: {sallat,napoli}@unisa.it)

Received: 8 September 1997 / 27 June 2001

**Abstract.** Finite automata on $\omega$-sequences and $\omega$-trees were introduced in the sixties by Büchi, McNaughton and Rabin. Finite automata on timed $\omega$-sequences were introduced by Alur and Dill. In this paper we extend the theory of timed $\omega$-sequences to $\omega$-trees. The main motivation is the introduction of a new way to specify real-time systems and to study, using automata-theoretic techniques, branching-time temporal logics with timing constraints. We study closure properties and decision problems for the obtained classes of timed $\omega$-tree languages. In particular, we show the decidability of the emptiness problem. As an application of the introduced theory, we give a new decidable branching time temporal logic (STCTL) whose semantics is based upon timed $\omega$-trees.

## 1 Introduction

Finite automata on $\omega$-sequences and $\omega$-trees were introduced in the sixties by Büchi [7], McNaughton [18] and Rabin [20]. Starting from these first works, a great amount of literature has been devoted to this theory which turned out to be fundamental for those areas where nonterminating computations are studied (see [23] for a survey). Over the last decade the theory of finite automata on $\omega$-sequences and $\omega$-trees has been an important source of tools for the specification and verification of computer programs. In par-

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.

ticular, the connections between the theory of automata on $\omega$-objects and Temporal Logic have been extensively and profitably investigated ([6,11, 24–26]). These connections rely on the consideration that $\omega$-sequences and $\omega$-trees whose symbols are truth values of a finite set of atomic propositions can represent nonterminating computations and sets of nonterminating computations. On the other hand, given a formula in a linear-time or branching-time temporal logic, one can construct a finite automaton accepting the $\omega$-sequences or the $\omega$-trees which satisfy the formula. The main results in this perspective consist in the reduction of temporal logic problems to decidable problems in the automata theory field. This approach has the advantage of providing a very simple and neat theory which is easily extendible. In order to correctly capture the behaviour of real-time systems that interact with physical processes and whose correct running crucially depends upon real-time considerations, the time has been explicitly introduced in tools for system specification and verification.

Finite automata on timed $\omega$-sequences were introduced by Alur and Dill [2]. They developed a theory of languages accepted by timed finite automata similar to the theory of $\omega$-regular languages. Their goal was that of specifying and verifying models of real-time systems as easily as qualitative models (that consider the time only from a qualitative point of view).

In this paper we extend the theory of timed $\omega$-sequences to $\omega$-trees. The main motivation is the introduction of a new way to specify real-time systems and to study, using automata-theoretic techniques, branching-time temporal logics with timing constraints. Branching time logics are more suitable than linear ones for reasoning about nondeterminism which, in turn, is useful to model concurrent programs (nondeterministic interleaving of atomic processes). In fact, many concurrent programs are intrinsically nondeterministic. Moreover, branching time logics provide an existential path quantifier, which allows to express lower bounds on the nondeterminism and concurrency. These lower bounds are helpful in applications such as program synthesis. Furthermore, to distinguish between inevitability of a predicate and possibility of a predicate becomes possible and, then, the closure under semantic negation of the logic is ensured. See also [6,8,9, 12].

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 of timed $\omega$-trees. We prove that, differently from the timed $\omega$-sequences where Muller and Büchi acceptance conditions result equivalent in the nondeterministic models, 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. Finally, we show that the equivalence problem is undecidable for nondeterministic timed tree automata while it is decidable for the deterministic ones.

As an application of the introduced theory, we give a new branching time temporal logic (STCTL) whose semantics is based upon timed $\omega$-trees. Temporal logics which consider explicitly the time constitute a new research area of increasing interest. See [13,15,19] for temporal logics based on either discrete time or fictitious clock models. A timed temporal logic that uses a dense time model is TCTL (see [1]). TCTL presents an undecidable satisfiability problem even if the structures are restricted to dense trees obtained from timed graphs (finite satisfiability). STCTL mainly differs from TCTL for a simpler structure by means of which the semantics is defined and for the lack of the equality in the timing constraints. A decidable linear time logic with dense time model is MITL in [3]. For specifying real-time systems, the Global Clock Temporal Logic (GCTL), with a dense time domain, is also introduced in [19]. For more about real-time logics, see also [4,14]. Here the main result is the decidability of the satisfiability problem which follows from the decidability of the emptiness problem for finite automata on timed $\omega$-trees. Introducing the equalities in the timing constraints causes the lost of the decidability.

The introduction of STCTL as an application of the theory of timed tree automata emphasizes that the theoretic approach can be profitable also for real-time logics. We argue that STCTL has interesting features to model real-time behaviours. First, it uses a dense domain of time and is a branching time logic; further, the idea of coupling a real number to a system state, so identifying the instant in which the system enters that state, is quite intuitive and allows to capture a change of the system state exactly when it occurs.

## 2 Timed tree automata

In this section we introduce the timed $\omega$-trees and the timed $\omega$-tree automata. Similarly to the case of $\omega$-tree languages, we obtain various models of timed tree automata, by considering both deterministic and nondeterministic paradigms and different conditions on the acceptance of a timed $\omega$-tree. In the Sect. 2.1 we give the basic definitions. In the Sect. 2.2 we state a relationship between timed $\omega$-tree languages and $\omega$-tree languages.

## *2.1 Basic definitions*

Let $\Sigma$ be an alphabet and $dom(t)$ be a subset of $\{1,\ldots,k\}^*$, 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 $i \in \{1,\ldots,k\}$ such that $wi \in dom(t)$;
- if $w \in dom(t)$ and $w = ui$, with $i \in \{1,\ldots,k\}$, then $u \in dom(t)$.

A *$\Sigma$-valued $\omega$-tree* is a mapping $t : dom(t) \longrightarrow \Sigma$. We denote with $deg(w)$ the arity of the node $w \in dom(t)$ and with $pre(v)$ the set of the prefixes of $v$. 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 $\epsilon$. With $In(t|\pi)$ we denote the set of the symbols labeling 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: $\tau(w) > 0 \; \forall w \in dom(t) - \{\epsilon\}$ and $\tau(\epsilon) \geq 0$;
- 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 are interested in studying the timed $\omega$-trees as objects accepted by an automaton (a device provided with a finite control and scanning an $\omega$-tree at input). We assume that the nodes of an $\omega$-tree become 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 considering a timed $\omega$-tree. Each node of a timed $\omega$-tree is labeled by a pair (symbol, real number). The real number is (except for the root $\epsilon$, 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 $T^k_\Sigma$ 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 $[\lambda \rightarrow 0](\nu + d)$ the clock interpretation that for each clock $x \in \lambda$ gives 0 and for each clock $x \notin \lambda$ gives the value $\nu(x) + d$.

Now, we can formally introduce our models starting from the definition of the timed tree transition table.

**Definition 1** *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 time $\tau$ the the symbol $\sigma$ is ready at input. Then, 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. $\nu + \tau$) 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 $[\lambda_1 \rightarrow 0](\nu + \tau), \ldots, [\lambda_k \rightarrow 0](\nu + \tau)$. 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.

**Definition 2** *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)$\longrightarrow$ $\Re_{+}^{C}$;*
- *$r(\epsilon) \in S_0$ and $\nu(\epsilon) = \nu_0$ 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),$\lambda_1$,..., $\lambda_k$,$\delta$)$\in\Delta$, $\nu$(w)+$\tau$(w) fulfils $\delta$ and $\nu$(wi)= $[\lambda_i \rightarrow 0]$ ($\nu$(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.

**Definition 3** *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 $\{(t, \tau) \,|\, t$ is accepted by $A\}$.*

We define a nondeterministic (resp. deterministic) Muller timed tree automaton analogously. The only changes needed are: an accepting family $\mathcal{F} \subseteq 2^S$ for the set of final states $F$ and the new acceptance condition "$In(r|\pi) \in \mathcal{F}$" 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 [23].

## *2.2 Basic results*

In this subsection 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:

**Definition 4** *Let T be a timed tree language,* Untime$(T)$ *is* $\{t \mid (t, \tau) \in T$ *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 $\nu(v_h)(x_0) \geq 1$ with $h = 1, 2, 3, \ldots$. The following result holds.

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

- $C' = C \cup \{x_0\}$ with $x_0 \notin C$;
- for all rules $(s, \sigma, s_1, \ldots, s_k, \lambda_1, \ldots, \lambda_k, \delta) \in \Delta$, $\Delta'$ 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
  $((s,1), \sigma, (s_1,0), \ldots, (s_k,0), \lambda_1 \cup \{x_0\}, \ldots, \lambda_k \cup \{x_0\}, \delta)$;
- $\mathcal{F}' = \{Y \subseteq S \times \{0,1\} \mid \phi_1(Y) \in \mathcal{F}\}$, where $\phi_1$ 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 $\{s \in s \times \{0,1\} \mid \phi_1(s) \in F\}$, 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 $0 \leq m < \frac{1}{k}$. 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 $\frac{c}{u}$ is a constant appearing in some clock constraint of $A$ involving $x$. 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 all $x \in C$, either $\lfloor \nu(x) \rfloor_u = \lfloor \nu'(x) \rfloor_u$ or both $\nu(x) > \frac{c_x}{u}$ and $\nu'(x) > \frac{c_x}{u}$;
- for all $x, y \in C$ with $\nu(x) \leq \frac{c_x}{u}$ and $\nu(y) \leq \frac{c_y}{u}$, $fract_u(\nu(x)) \leq fract_u(\nu(y))$ if and only if $fract_u(\nu'(x)) \leq fract_u(\nu'(y))$;
- for all $x \in C$ with $\nu(x) \leq \frac{c_x}{u}$, $fract_u(\nu(x)) = 0$ 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 bounded above 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 $\nu + h \in \alpha'$.

Thus, the *Region Automaton* is defined as:

**Definition 5** *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 $R(S) = \{\langle s, \alpha \rangle \mid s \in S$ and $\alpha$ is a clock region for $A\}$;*
- *the set of starting states $R(S_0) = \{\langle s_0, \alpha_0 \rangle \mid s_0 \in S_0$ and for all $x \in C \alpha_0$ 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 $\alpha_i = [\lambda_i \rightarrow 0]\alpha'$ for all $i \in \{1, \ldots, k\}$.*

It is possible to prove that 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 $A = (\Sigma, S, S_0, \Delta, C, \mathcal{F})$ accepting $T$ and having a nondivergent clock $x_0$. We define the Muller tree automaton $A^U = (\Sigma, S^U, S_0^U, \Delta^U, \mathcal{F}^U)$, where:

1. $S^U$ contains $\langle s, \alpha, h \rangle$ where $s \in S$, $\alpha$ is a clock region and $h \in \{0, 1, 2\}$;
2. $S_0^U$ contains $\langle s, \alpha, 2 \rangle$ where $\langle s, \alpha \rangle \in R(S_0)$;

3. $(\langle s, \alpha, h \rangle, \sigma, \langle s_1, \alpha_1, h_1 \rangle, \ldots, \langle s_k, \alpha_k, h_k \rangle) \in \Delta^U$ 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 $\forall i \in \{1, \ldots, k\}$:
$$h_i = \begin{cases} 1 \; if \; \alpha_i \; satisfies \; x_0 \geq 1 \\ 2 \; if \; \alpha_i \; satisfies \; x_0 = 0 \; and \; h = 1 \\ 0 \text{ otherwise.} \end{cases}$$
4. $\mathcal{F}^U = \{Y \subseteq S^U \mid \phi_1(Y) \in \mathcal{F}$ and $\phi_3(Y) \cap \{2\} \neq \emptyset\}$ with $\phi_i$ 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 $F^U = \{s \in S^U \mid \phi_1(s) \in S$ and $\phi_3(s) = 2\}$. We call again this automaton $A^U$.

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

**Theorem 1** *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 $k = deg(w)$, $(s_w, t(w), s_{w1}, \ldots, s_{wk}, \lambda_{w1}, \ldots, \lambda_{wk}, \delta) \in \Delta$, $\alpha_{wi} = [\lambda_i \rightarrow 0](\alpha_w + d)$ 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 $c \in \Sigma$, it is possible to define a set $Q_c$ containing the states $\langle s, \alpha, 2 \rangle$ of $A^U$ such that $A^U$, starting from $\langle s, \alpha, 2 \rangle$, accepts some timed tree whose root is labeled by $c$. We denote this set as the set of the $c$-starting states of $A^U$.

The following theorem gives a strong result concerning the relation between a particular class of timed languages and their corresponding Untime.

**Theorem 2** *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 1. 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 1, 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, for a timed tree language $T'$, it is not true in general that $Untime(T') \in (D)BTA$ (resp. $Untime(T') \in (D)MTA$) implies $T' \in (D)TBTA$ (resp. $T' \in (D)TMTA$). The following counter-example is due to Alur and Dill [2].

*Example 1* 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.

## 3 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.

The relationships between the considered classes of languages are inherited from those between tree languages, by means of Theorem 2. It is known that BTA is strictly included in MTA [21]. 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 3** *It holds that:*

1. *DBTA$\subset$BTA$\cap$DMTA;*
2. *BTA and DMTA are not comparable;*
3. *BTA$\cup$DMTA$\subset$MTA.*

*Proof.* "1." Let $T_1$ be the language $\{t \in T^1_{\{a,b\}} \mid \pi = \epsilon, 1, 11, \ldots, 1^i, \ldots$ 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 [23]). Moreover, the nondeterministic Büchi automata and the deterministic Muller automata on $\omega$-sequences are equivalent [18]. Thus, we have $DBTA \subset BTA \cap DMTA$.

"2." Let $k > 1$ and $T_2$ be the language $\{t \in T^k_{\{a,b,c\}} \mid \forall$ paths $\pi$ in $t,\ a \notin In(t|\pi)\}$. Rabin in [21] 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 [10]), we can prove that $T_3$ is not in DMTA. Thus, $T_3 \in BTA - DMTA$ and we are done with point 2.

"3." Let $k > 1$ and $T_4$ be the tree language $\{t \in T^k_{\{a,b,c\}} \mid \forall$ paths $\pi$ 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 deleting all the transition rules on the symbol "b". Trivially, $A'$ would accept the language $\{t \in T^k_{\{a,c\}} \mid \forall$ paths $\pi$ 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 1** *DTBTA$\subset$TBTA$\cap$DTMTA, TBTA$\cup$DTMTA$\subset$TMTA, TBTA and DTMTA are not comparable.*

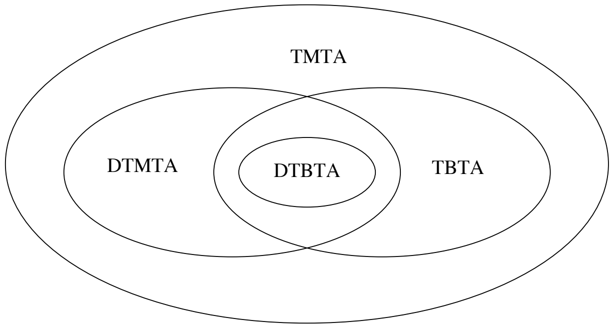

**Fig. 1.** Relationships between the classes TMTA, TBTA, DTMTA and DTBTA

*Proof.* By the usual constructions we have $DTBTA \subseteq DTMTA$ and $TBTA \subseteq TMTA$. The proof is completed by Theorems 2 and 3.

The results of the above corollary are summarized in Fig. 1.

Now we give some closure properties for the defined classes of timed tree languages.

**Theorem 4** *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 exist 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 for complementation of the languages in DTMTA. The following result holds.

**Theorem 5** *DTMTA*$\cup\overline{\textit{DTMTA}}$ $\subset$*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), 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 labeled 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 6** *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 $A^U$ be the tree automaton accepting $Untime(T(A_1))$ and $Q_c$ the set of the $c$-starting states of $A^U$. 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 $\alpha'_1, \ldots, \alpha'_r$ the clock regions 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:

- $\sigma \neq c$;
- $(s, \sigma, s'_1, \ldots, s'_k, \lambda'_1, \ldots, \lambda'_k, \delta') \in \Delta_1$, $\{\langle s'_{i_1}, \alpha_{i_1}, 2\rangle, \ldots, \langle s'_{i_h}, \alpha_{i_h}, 2\rangle\} \subseteq Q_c$ and (1) $s_{i_j} \in S_2^0$ 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 $A_1$. If a possible root of a tree in $T(A_2)$ is processed, $A$ 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 an unique difference that the accepting family of $A$ is $\mathcal{F} = \mathcal{F}_1 \cup \mathcal{F}_2$, where $\mathcal{F}_i$ 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 on the root. Then, the 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 the rules $(s, \sigma, s_1, \ldots, s_k, \lambda_1, \ldots, \lambda_k, \delta)$ such that:

$(s, \sigma, s'_1, \ldots, s'_k, \lambda'_1, \ldots, \lambda'_k, \delta') \in \Delta_1$, $\{\langle s'_{i_1}, \alpha_{i_1}, 2\rangle, \ldots, \langle s'_{i_h}, \alpha_{i_h}, 2\rangle\} \subseteq Q_c$ 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 $\{X \subseteq S_1 \times \{0,1\} \mid \phi_1(X) \in \mathcal{F}_1$ or $\phi_2(X) = \{0,1\}\}$ where $\phi_i$ projects the $i$-th component of a pair and $\mathcal{F}_1$ is the accepting family of $A_1$.

The above results do not hold for the deterministic classes. A counter-example for the concatenation is given by the languages $T_1 = \{(t, \tau) \in T^k_{\{a,c\}} \mid \forall$ paths $\pi = \epsilon, v_1, v_2, \ldots\ \exists i$ such that $t(v_i) = c\}$ and $T_2 = \{(t, \tau) \in T^k_{\{a\}} \mid \forall$ paths $\pi = \epsilon, v_1, v_2, \ldots\ \exists i$ 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 from the nodes of the first cut to 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 $\{(t, \tau) \in T^k_{\{a,c\}} \mid t(\epsilon) = a$ and $\forall$ paths $\pi = \epsilon, v_1, v_2, \ldots$ 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.

Closure properties are summarized in Fig. 2.

| | 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.** Closure properties

Now, we deal with decision problems.

**Theorem 7** *The emptiness problem is decidable for TMTA.*

*Proof.* Let T∈TMTA, we have that T is empty if and only if Untime(T) is empty. From the Theorem 1, Untime(T) is accepted by Muller tree automata. Since the emptiness problem is decidable for the tree languages accepted by Muller automata (see [22]), we have the decidability for TMTA.

**Corollary 2** *The emptiness problem is decidable for DTBTA, DTMTA and TBTA.*

**Theorem 8** *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 5 and 7.

**Definition 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 1, we can state the following:

**Theorem 9** *The untimed equivalence problem is decidable for TMTA.*

Figure 3 summarizes the results obtained on decision problems for the classes (D)TBTA and (D)TMTA.

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

**Fig. 3.** Decision problems

## 4 A branching time temporal logic: STCTL

In this section we present a real-time temporal logic, with decidable satisfiability problem, which is obtained from CTL [8] by introducing timing constraints on the linear operators. We call this logic STCTL, which is the acronym for Simplified Timed Computational Tree Logic. A well-known timed temporal logic that uses a dense time model is TCTL [1], for which the satisfiability problem is undecidable. In comparison to TCTL, STCTL does not allow the equality in the timing constraints and its semantics is defined by means of a simpler structure. A decidable linear real-time logic is MITL [3]. An MITL-structure is obtained by coupling a sequence of contiguous intervals with a sequence of states and MITL-syntax allows, as timing constraints, nonsingular intervals with rational end-points. The restriction to nonsingular intervals has the same effects of excluding the equality in timing constraints.

This section is divided in two subsections. In the first one we give the syntax and the semantics of STCTL. Then, in the second one we prove the decidability of both the satisfiability and the validity problems.

## *4.1 Syntax and semantics of STCTL*

The choice of the STCTL-operators is the same as in CTL, except for the next operator that is not considered here. We have the path quantifiers E ("for some future") and A ("for all futures"), the usual linear operator U ("until") and the boolean connectives $\wedge$ ("and") and $\neg$ ("not"). With linear operators, we can specify timing constraints.

Given a set of atomic propositions $AP$, the syntax of the STCTL-formulae is:

- for each $p \in AP$, $p$ is an STCTL-formula;
- if $p$ and $q$ are STCTL-formulae, then $p \wedge q$ and $\neg p$ are STCTL-formulae;
- if $p$ and $q$ are STCTL-formulae, then $pU_{\approx c}q$ is a path formula (where $\approx \in \{<, \leq, >, \geq\}$ and $c$ is a rational number);
- if $p$ is a path formula, then $Ep$ and $Ap$ are STCTL-formulae.

Remark that CTL-formulae, constructed without the "next" operator, are also STCTL-formulae. In fact, in CTL there are not bounds on the time at which an assertion has to be fulfilled, then, as will be clear from the semantics, the "until" corresponds to the STCTL "until" with the timing constraint "$\geq 0$".

In the rest of this section with $AP$ we will denote the set of atomic propositions of the considered STCTL-formulae. If not differently stated, we refer with $\approx$ to a relational operator in $\{<, \leq, >, \geq\}$, with $c$ to a rational number, and to a formula of type $E[pU_{\approx c}q]$ or $A[pU_{\approx c}q]$ as an until formula.

A state of a system can be represented by the set of the atomic propositions which are true in that state. A run of a system can be modeled by a function $\pi_0$ mapping each time instant to the subset of $AP$ representing the state of the system in that time instant. Such mapping induces a split of the time domain into a sequence of contiguous time intervals, with the property that the system does not change its state during each time interval. More precisely, the function $\pi_0$ splits the time domain in a (possibly finite) sequence of pairwise disjoint intervals $I_0, I_1, \ldots$ such that:

- $\pi(d) = \pi(d')$ for all $d, d' \in I_i$,
- $d < d'$ for all $d \in I_i$ and $d' \in I_{i+1}$, and
- $\bigcup_{i\geq 0} I_i = \Re_+ \cup \{0\}$.

We can assume that in this sequence open intervals alternate with singular ones. To describe the behaviour of a system usually requires to consider many runs that can be arranged in a tree-like structure. This is the case, for example, when we consider concurrent programs with nondeterministic interleaving of processes. In these trees the instants at which a run can continue in many different ways are represented by nodes. If for all paths there are only a countable number of nodes we can express them by a timed $\omega$-tree.

An *STCTL-structure* is a timed $(2^{AP} \times 2^{AP})$-valued $\omega$-tree $(t, \tau)$ with $\tau(\epsilon) = 0$. Given an STCTL-structure $(t, \tau)$ we denote with $t_{open}$ and $t_{sing}$ the functions defined as $(t_{open}(v), t_{sing}(v)) = t(v)\ \forall v \in dom(t)$. An open and a singular interval correspond to each node $v$: $t_{open}$ and $t_{sing}$ are the sets of the atomic propositions holding in these two time intervals. Thus each path of an STCTL-structure defines a function mapping each time instant to a subset of $AP$ as follows. Given an STCTL-structure $(t, \tau)$, a path $\pi = v_0, v_1, v_2, \ldots$ in $t$, and $d \in \Re_+ \cup \{0\}$, a timed path shifted by $d$ on $\pi$, denoted by $\pi_d$, is a mapping defined as:

$$\pi_d(d') = \begin{cases} t_{open}(v) \ \gamma_w < d + d' < \gamma_v \\ t_{sing}(v) \ d + d' = \gamma_v \end{cases}$$

where $v, w \in \pi$ and $v = wj$ for some $1 \leq j \leq deg(w)$.

We remark that, according to the definition of timed paths and since $\tau(\epsilon) = 0$, $t_{open}(\epsilon)$ has no actual meaning and we will not consider it in the rest of the paper. An STCTL-structure has thus a dense-time semantics. It can be seen as a set of timed paths arranged on a tree-like structure (discrete nondeterminism): time is continuous along all paths but there is a discrete branching-time structure. This means that we discretize the points of nondeterminism in our models, but we still admit continuous behaviours. The truth of atomic propositions along a path changes according to a sequence of alternating singular and open time intervals. This way we are able to consider a state of a system to stay unchanged for close, left open, right open, or open time interval. We do not allow a full dense branching-time semantics (as in TCTL) and this allows us to construct a generator of STCTL-models.

A time instant in a computation is uniquely determined by a path and a time occurrence in the STCTL-structure. Then, we will define the models of STCTL-formulae with respect to the pairs $(\pi, d)$, where $\pi$ is a path and $d$ is a non-negative real number. The concept of satisfiability for STCTL-formulae is established by the following definition:

**Definition 7** *Let $p$ be an STCTL-formula (resp. a path formula), $(t, \tau)$ be an STCTL-structure, $\pi = v_0, v_1, v_2, \ldots$ be a path in $t$ and $d \geq 0$, then $p$ is satisfied in $(t, \tau)$ on the path $\pi$ at the time $d$ (resp. on the timed path $\pi_d$) if and only if $(t, \tau), (\pi, d) \models p$ (resp. $(t, \tau), \pi_d \models p$). The relation $\models$ is defined as follows:*

- *if* $p \in AP$*, then* $(t,\tau),(\pi,d) \models p$ *if and only if* $p \in \pi_d(0)$*;*
- $(t,\tau),(\pi,d) \models p \wedge q$ *if and only if* $(t,\tau),(\pi,d) \models p$ *and* $(t,\tau),(\pi,d) \models q$*;*
- $(t,\tau),(\pi,d) \models \neg p$ *if and only if* $not((t,\tau),(\pi,d) \models p)$*;*
- $(t,\tau),(\pi,d) \models Ep$ *if and only if* $\exists$ *a path* $\pi' = v'_0, v'_1, v'_2, \ldots$ *in* $t$ *such that* $\pi_0(d') = \pi'_0(d')$ *for all* $d' \leq d$ *and* $(t,\tau),\pi'_d \models p$*;*
- $(t,\tau),(\pi,d) \models Ap$ *if and only if* $\forall$ *paths* $\pi' = v'_0, v'_1, v'_2, \ldots$ *in* $t$ *such that* $\pi_0(d') = \pi'_0(d')$ *for all* $d' \leq d$*, it results that:* $(t,\tau),\pi'_d \models p$*;*
- $(t,\tau),\pi_d \models pU_{\approx c}q$ *if and only if* $\exists d' \geq d : (t,\tau),(\pi,d'') \models p$ *for all* $d \leq d'' < d'$, $(d'-d) \approx c$ *and* $(t,\tau),(\pi,d') \models q$*.*

Then, the definition of an STCTL-model is:

**Definition 8** *Let* $(t,\tau)$ *be an STCTL-structure and* $p$ *an STCTL-formula,* $(t,\tau)$ *is an* STCTL-model *of* $p$ *if and only if* $(t,\tau),(\pi,0) \models p$ *for a path* $\pi$ *in* $t$*. Moreover, an STCTL-formula* $p$ *is said* satisfiable *if and only if there is an STCTL-model of* $p$*.*

Note that we do not have a *next* operator as in CTL: it is quite standard to have this operation in logics without timing constraints, but when a dense time model is dealt with it is not meaningful, since the concept of next time is lost. Anyway, a next operator could be added without affecting the decidability of STCTL. The meaning of this new operator would be to check a truth value in a "next" interval.

In the next subsection we will use STCTL-formulae in *positive normal form* (PNF), i.e. formulae with negations only on the atomic propositions. To this aim, we enlarge the syntax by introducing the logical connective $\vee$ and a new operator, the *before* operator, denoted by $B$.

The syntax for the before operator is:

- let $p$ and $q$ be two STCTL-formulae, then $pB_{\approx c}q$ with $\approx \in \{<,\leq,>,\geq\}$ is a path formula.
  Given an STCTL-structure $(t,\tau)$, a time $d \geq 0$ and a path $\pi = v_0, v_1, v_2, \ldots$ in $t$, the semantic rule for $B$ is:
- $(t,\tau),\pi_d \models (pB_{\approx c}q)$ if and only if
  $\forall d' \geq d : (((t,\tau),(\pi,d') \models q$ and $(d'-d) \approx c) \implies \exists d \leq d'' < d'$ such that $(t,\tau),(\pi,d'') \models p)$.

In the same way as for an until formula, with before formula we refer to a formula of type $E[pB_{\approx c}q]$ or $A[pB_{\approx c}q]$. Obviously, the new operators do not change the expressiveness of the logic and the following relation holds:

- $(\neg pB_{\approx c}q) \equiv \neg(pU_{\approx c}q)$.

It is obvious that all STCTL-formulae can be converted in PNF.

## 4.2 Decidability of STCTL

Let $p_0$ be an STCTL-formula in PNF. We define the *closure* of $p_0$, denoted with $cl(p_0)$, as the set of all the subformulae of $p_0$ and the *extended closure*, denoted with $ecl(p_0)$, as the set: $cl(p_0) \cup \{\sim p \mid p \in cl(p_0)\}$, where $\sim p$ is the STCTL-formula in PNF equivalent to $\neg p$. Moreover, we define $S_{p_0} \subseteq 2^{ecl(p_0)}$ as the family containing the sets $s$ with the following properties:

- $p \in s \Longrightarrow \sim p \notin s$;
- $p \wedge q \in s \implies p \in s$ and $q \in s$;
- $p \vee q \in s \implies p \in s$ or $q \in s$;
- $\alpha[pU_{\approx c}q] \in s, \alpha \in \{A, E\} \implies p \in s$ or $(q \in s$ and $(0 \approx c))$;
- $\alpha[pB_{\approx c}q] \in s, \alpha \in \{A, E\} \implies \sim q \in s$ or $not(0 \approx c)$;
- $s$ is maximal, that is $\forall p \in ecl(p_0) : p \in s$ or $\sim p \in s$.

Note that $S_{p_0}$ contains the maximal sets of formulae in $ecl(p_0)$ which are consistent, in the sense that given an $s \in S_{p_0}$ the fulfillment at a given time $d$ along a given path $\pi$ of a formula in $s$ does not prevent all the other formulae in $s$ from being satisfied at $(\pi, d)$. Another concept we will use is the *local satisfiability*. We say that the STCTL-formulae $\alpha[pU_{\approx c}q]$ and $\alpha[\sim pB_{\approx c}q]$ belonging to $s$, for some $s \in S_{p_0}$, are not *locally satisfied* in $s$ if $p \in s$ and, if $\approx$ is either $<$ or $\leq$, then $\sim q \in s$. The fact that an until or a before formula is not locally satisfied in $s$ means that it is not satisfied yet but its fulfillment is still possible starting from $s$, that is, it is necessary to look forward in the paths to verify the possible fulfillment of this formula.

The following two lemmata allow us to restrict to a particular class of STCTL-structures. Informally speaking, the first lemma states that there is a class of structures satisfying the property that we can label the time intervals induced by the timed paths with formulae that are true at all the time instants of those intervals. In other words, we can label a node $v$ of an STCTL-structure with a pair $(s_{open}, s_{sing}) \in S_{p_0} \times S_{p_0}$, with the meaning that:

- the formulae in $s_{open}$ are satisfied at every time of the open interval associated to $v$[1] along all paths going through $v$ and
- the formulae in $s_{sing}$ are true in the singular interval corresponding to $v$ on the paths going through $v$.

The second lemma shows that if the formula is satisfiable then there are STCTL-models such that the arity of their nodes is bounded above by a constant depending on the formula. Both these results are important to use theory of automata for the STCTL-satisfiability problem.

[1] Note that this interval is the same for all paths going through $v$ because of the tree structure.

**Lemma 2** *Let $p_0$ be an STCTL-formula in PNF. If $p_0$ is satisfiable then there is an STCTL-model $(t, \tau)$ of $p_0$ such that for all paths $\pi$ in $t$ and for all $w, v \in \pi$, $v = wj$, and $j \leq deg(w)$:*

*a) there exists $s \in S_{p_0}$ such that $\forall p \in s$: $(t, \tau), (\pi, 0) \models p$;*
*b) there exists $s \in S_{p_0}$ such that $\forall p \in s$ and $\forall d$ such that $\gamma_w < d < \gamma_v$, $(t, \tau), (\pi, d) \models p$;*
*c) there exists $s' \in S_{p_0}$ such that $\forall p \in s'$: $(t, \tau), (\pi, \gamma_v) \models p$.*

*Proof.* Properties a) and c) are trivially true for all STCTL-structures. Thus we only prove the property b).

Consider the following operation on STCTL-structures. Let $(t', \tau')$ be an STCTL-structure, $v \in dom(t')$ and $e_0, \ldots, e_n \in \Re_+$ such that $e_0 + \ldots + e_n = \tau(v)$, $n \geq 0$, then define $(t'', \tau'')$ as the STCTL-structure obtained from $(t', \tau')$ by replacing the subtree rooted at $v$ with the subtree described as:
1) $t''(v1^i) = (t'_{open}(v), t'_{open}(v))$, $\tau''(v1^i) = e_i$ for $i = 0, \ldots, n-1$, $t''(v1^n) = (t'_{open}(v), t'_{sing}(v))$ and $\tau''(v1^n) = e_n$;
2) the subtree of $(t'', \tau'')$ rooted at $v1^n$ is equal to the subtree of $(t', \tau')$ rooted at $v$ except for $\tau'(v)$, since $\tau''(v1^n)$ is equal to $e_n$.

We say that $(t'', \tau'')$ is obtained from $(t', \tau')$ by the substitution of the node $v$ with a chain of $n+1$ nodes. It is easy to see that if $(t', \tau')$ is an STCTL-model of $p_0$ then the STCTL-structure $(t'', \tau'')$ is still an STCTL-model of $p_0$. Given an STCTL-structure $(t', \tau')$ and an STCTL-formula $p_0$, we can obtain an STCTL-structure $(t, \tau)$ with the property b) by the substitution of each node of $(t', \tau')$ with a chain of at most $2^h$ nodes, where $h$ is the number of the until formulae in $ecl(p_0)$. This assertion will be proved by induction on $h$. For $h = 0$, that is $p_0$ does not contain until formulae, the assertion is trivially true. In fact, $p_0$ is a boolean combination of atomic propositions and, directly from the semantics, the property b) is true for all the STCTL-structures of $p_0$. Now, suppose that the assertion is true for $i \leq h$. Let the number of until formulae in $ecl(p_0)$ be equal to $(h+1)$. Then, $p_0$ is the conjunction or the disjunction of a formula $p_1$ with either a formula $\alpha[p_2 U_{\approx c} p_3]$ or $\alpha[p_2 B_{\approx c} p_3]$, where $\alpha$ is $A$ or $E$. Let $h_1$, $h_2$ and $h_3$ be the number of until formulae in $ecl(p_1)$, $ecl(p_2)$ and $ecl(p_3)$, respectively. Thus, $h_1 + h_2 + h_3 = h$. By applying sequentially the inductive hypothesis to $(t', \tau')$ and $p_1, p_2$, and $p_3$, we obtain a structure $(t'', \tau'')$, by substitution of each node of $(t', \tau')$ with a chain of at most $2^{h_1} \cdot 2^{h_2} \cdot 2^{h_3} = 2^h$ nodes and such that property b) is true for all the formulae in $ecl(p_1) \cup ecl(p_2) \cup ecl(p_3)$. To complete the inductive step we have to consider the until formula in $ecl(p_0) - \cup_{i=1}^{3} ecl(p_i)$. Let $p$ be this formula. Since by the inductive hypothesis we have obtained a structure such that property b) holds for all the formulae in $\cup_{i=1}^{3} ecl(p_i)$, the truth of $p$ can only vary because of the timing constraint on its outermost until. Then, it holds that, for all paths $\pi$ in $t''$ and for all $w, v \in \pi$, $v = wj$, and $j \leq deg(w)$, there is at most a $d'$ such that either: i) $(t'', \tau''), (\pi, d) \models p$ for all $d$ such that $\gamma_w < d < d'$ and $(t'', \tau''), (\pi, d) \models \sim p$ for all $d$ such that $d' < d < \gamma_v$ or ii) the dual case holds. Thus, each node of $(t'', \tau'')$ is substituted with a chain of at most two nodes in order to obtain an STCTL-structure with the property b) and we are done.

**Lemma 3** *Let $p_0$ be an STCTL-formula in PNF. If $p_0$ is satisfiable then there is an STCTL-model of $p_0$ such that, for each node $w$, $deg(w) \leq 2 \max_{s \in S_{p_0}} \mid \{Ep \in s \mid Ep \text{ is not locally satisfied in } s\} \mid +1$.*

*Proof.* Let $(t, \tau)$ be an STCTL-model of $p_0$ satisfying properties a), b), and c) of Lemma 2. Then, there exists a function $\phi_{open}$ (resp. $\phi_{sing}$) such that, for all $w, v \in \pi$ where $v = wj$ and $j \leq deg(w)$, $\phi_{open}(v)$ (resp. $\phi_{sing}(w)$) is the set of formulae in $ecl(p_0)$ which are satisfied at a time $d$ on $\pi$ for $\gamma_w < d < \gamma_v$ (resp. $d = \gamma_w$). Furthermore, for each $v \in dom(t) - \{\epsilon\}$ (resp. $v \in dom(t)$) let $\psi_{open}(v)$ (resp. $\psi_{sing}(v)$) be the set of formulae in $\phi_{open}(v)$ (resp. $\phi_{sing}(v)$) which are not locally satisfied in $\phi_{open}(v)$ (resp. $\phi_{sing}(v)$). Given $w, v \in \pi$ where $v = wj$ and $j \leq deg(w)$, for all $Ep \in \psi_{open}(v) \cup \psi_{sing}(v)$ we pick a path, denoted as $\pi^{p,v}$, such that $v \in \pi$ and $p$ is satisfied on $\pi^{p,v}_d$ for $\gamma_w < d \leq \gamma_v$. For all $Ep \in \psi_{sing}(\epsilon)$, we choose a path, denoted as $\pi^{p,\epsilon}$ such that $p$ is satisfied on $\pi^{p,\epsilon}_0$. We can suppose that for different formulae in $\psi_{sing}(\epsilon)$ we choose different paths, and analogously, for $v \in dom(t) - \{\epsilon\}$, for different formulae in $\psi_{open}(v) \cup \psi_{sing}(v)$ we choose paths without common nodes except for those belonging to $pre(v)$. In fact, if it is not the case, we can duplicate some subtrees of $(t, \tau)$ so obtaining again an STCTL-model for $p_0$ with that property. We can now prune $(t, \tau)$ as follows:

– for every $Ep \in \psi_{sing}(\epsilon)$ mark all the nodes in $\pi^{p,\epsilon}$ and then cut all the subtrees rooted in unmarked children of $\epsilon$;
– for every marked node $v$ and for every $Ep \in \psi_{open}(v) \cup \psi_{node}(v)$ mark the nodes belonging to $\pi^{p,v}$ and then cut the subtrees rooted in unmarked children of $v$.

This procedure considers each node just once. When a node $v$ is considered, just one of its children has already been marked and exactly $k_{open} + k_{sing}$ children are further marked, where $k_{open}$ is the cardinality of $\{Ep \in \psi_{open}(v) \mid Ep \text{ is not locally satisfied in } \psi_{open}(v)\}$ and $k_{sing}$ is the cardinality of $\{Ep \in \psi_{sing}(v) \mid Ep \text{ is not locally satisfied in } \psi_{sing}(v)\}$. Thus, after pruning the subtrees rooted in unmarked children of $v$, we obtain $deg(v) \leq k_{open} + k_{sing} + 1$ and, trivially, the obtained timed tree is still an STCTL-model for $p_0$.

To prove the decidability of the STCTL-satisfiability problem, we reduce it to the emptiness problem for TBTA. Thus, we discuss how to construct a Büchi timed tree automaton $\mathcal{A}_{p_0}$ accepting models, if there are any, of an STCTL-formula. More precisely, $\mathcal{A}_{p_0}$ accepts the models of $p_0$ such that the properties stated in the above two lemmata hold.

Let $p_0$ be an STCTL-formula in PNF, we denote with $\alpha_1[p_1 U_{\approx_1 c_1} q_1], \ldots, \alpha_n[p_n U_{\approx_n c_n} q_n]$ all the until formulae in $ecl(p_0)$. Furthermore, let $\Sigma$ be equal to $2^{AP} \times 2^{AP}$. Then, we define a TBTA $\mathcal{A}_{p_0} = (\Sigma, S', S'_0, \Delta, C, F)$ where:

- $S' = S \times S \times \{0,1,2\}^{2n} \times \{0,1,\ldots,n\}$;
- $S'_0 = S \times S_0 \times \{0\}^{2n} \times \{0\}, S_0 = \{s \in S \mid p_0 \in s\}$;
- $C = \{x_1, \ldots, x_{2n}\}$;
- $F = S \times S \times \{0,1,2\}^{2n} \times \{0\}$;
- $\Delta \subseteq \bigcup_{i=1}^{h} [S' \times \Sigma \times (S')^i \times (2^C)^i \times \Phi(C)]$ and $h = 2\ \max_{s \in S} \mid \{Ep \in s \mid Ep$ is not locally satisfied in $s\} \mid +1$.

We give only an informal description of the behaviour of $\mathcal{A}_{p_0}$. The automaton $\mathcal{A}_{p_0}$ uses $2n$ clocks for checking the timing constraints in the formulae in $ecl(p_0)$:

- the clock $x_i$, with $i \in \{1, \ldots, n\}$, is used for the $i$-th until formula;
- the clock $x_{n+i}$, with $i \in \{1, \ldots, n\}$, is used for the before formula which is the negation of to the $i$-th until formula.

Given an STCTL-model $(t, \tau)$ of $p_0$, an accepting run of $\mathcal{A}_{p_0}$ on $(t, \tau)$ associates to a node $v \in \pi$ a state $\beta = (s, s', b_1, \ldots, b_{2n}, l) \in S'$ such that the first and second component of $\beta$ (that we call the $s_{open}$-component and the $s_{sing}$-component of $\beta$, respectively) satisfy the following statements:
if $v = \epsilon$ then $\forall p \in s' :\ (t, \tau), (\pi, 0) \models p$, otherwise
1) $\forall p \in s :\ (t, \tau), (\pi, d) \models p$ for all $d$ such that $\gamma_w < d < \gamma_v,\ v = wj$ and $1 \leq j \leq deg(w)$;
2) $\forall p \in s' :\ (t, \tau), (\pi, \gamma_v) \models p$.

Values of the component $b_i$, for $i = 1, \ldots, 2n$, indicate whether the clock $x_i$ is in use (values 1 and 2) or not (value 0). Then, they are used along with the clocks to check the fulfillment of the timing constraints. The component $l$ of $\beta$ is used to implement a counter. We observe that $\mathcal{A}_{p_0}$ associates to the root of the structure at the input a state whose $s_{sing}$-component is a member of $S_0$. As a consequence, if there exists a run of $\mathcal{A}_{p_0}$ on an STCTL-structure $(t, \tau)$, then $(t, \tau)$ is a model of $p_0$. A detailed description of $\mathcal{A}_{p_0}$ is quite complex, then we will simply sketch the behaviour of $\mathcal{A}_{p_0}$ for different formulae in $ecl(p_0)$.

Atomic propositions are locally checked by $\mathcal{A}_{p_0}$ that moves from a state $\beta$ only if the pair of subsets of $AP$ which are contained in the first two components of $\beta$ matches the pair of subsets of $AP$ labeling the node currently at input. The fulfillment of the STCTL-formulae of type $\sim p$, $p \wedge q$ and $p \vee q$ is checked by the fulfillment of $p$ and $q$ in an obvious way.

More attention is needed for the until and the before formulae. First, we observe that $\mathcal{A}_{p_0}$ checks the fulfillment of a formula $Ep$ at a time $d$ on a path $\pi$ by nondeterministically guessing a path $\pi'$ such that $\pi_0(d') = \pi'_0(d')$ for all $d' \le d$ and, then, checking the fulfillment of $p$ on $\pi_d$. Analogously, the fulfillment of a formula $Ap$ at a time $d$ on a path $\pi$ concerns all paths $\pi'$ such that $\pi_0(d') = \pi'_0(d')$ for all $d' \le d$.

Another nondeterministic choice, which is needed, consists of guessing for each interval of the input structure if either an until formula or its negation is satisfied. Depending on the formula under consideration, $\mathcal{A}_{p_0}$ will update the corresponding clock following in a different way. In fact, consider a formula $p_i\, U_{\approx_i} q_i$ and its negation $\sim p_i\, B_{\approx_i} q_i$ along a path $\pi$. By Lemma 2, $\pi$ can be split in a sequence of intervals (whose end-points coincide with the nodes of a structure) where $p_i\, U_{\approx_i} q_i$ and $\sim p_i\, B_{\approx_i} q_i$ are alternately true on a path $\pi$. If $\approx_i$ is either $<$ or $\le$ then:
1) clock $x_i$ is reset at the left end-points of the intervals where $p_i\, U_{\approx_i} q_i$ is true and, within these intervals, at nodes $v = wj, j \le deg(w)$, such that $q_i$ is true at $(\pi, d)$ for $d \in ]\gamma_w, \gamma_v[$ or $d = \gamma_v$;
2) clock $x_{n+i}$ is reset at all nodes within the intervals where $\sim p_i\, B_{\approx_i} q_i$ is true.

If $\approx_i$ is either $>$ or $\ge$, then the clocks for $\sim p_i\, B_{\approx_i} q_i$ are reset similarly to point 1) and those for $p_i\, U_{\approx_i} q_i$ as in point 2).

Since time intervals in which until and before formulae are satisfiable may be close, left open and/or right open, the automaton $\mathcal{A}_{p_0}$ uses the component $b_i$ of the state to distinguish between:
1) a left open ($b_i = 1$) and a left close ($b_i = 2$) time interval, if the clock constraints is "$x < c$" or "$x \le c$", and
2) a right open ($b_i = 1$) and a right close ($b_i = 2$) time interval, otherwise.

To better understand the necessity of this distinction, let us consider a simple example. Let $p_0 = E[pU_{<c}q]$ and $\pi$ be a model of $p_0$ whose nodes have arity 1. We identify this model with its unique path. Furthermore, let us suppose that:
1) $p$ is true at every time of the time interval $[d, d+c[$ on $\pi$ and $q$ is true only at the time $d+c$ on $\pi$;
2) on $\pi$ the time $d$ corresponds to the node $w$ and $d+c$ to the node $v$.
Then, $p_0$ is satisfied in $]d, d+c[$ on the path $\pi$ and $\sim p_0$ is satisfied at $d$ on $\pi$. In an accepting run, $\mathcal{A}_{p_0}$ resets the clock corresponding to $p_0$ at the node $w$ and, then, at the node $v$ this clock has value $c$. If $\mathcal{A}_{p_0}$ did not distinguish between left open and left close interval, it either would not accept this model of $p_0$ (if "$x < c$" is the clock constraint) or it could accept also structure that are not models of $p_0$ (otherwise).

The description of how $\mathcal{A}_{p_0}$ certifies the fulfillment of formulae $\alpha_i[p_i U_{\approx c_i} q_i]$ for $i = 1, \ldots, n$ is summarized in Fig. 4. In this figure with

$s, s'$ we denote, respectively, the $s_{open}$-component and the $s_{sing}$-component associated to the node currently at the input and with $s_j, s'_j$ those of its children on the considered path (or on all paths when $\alpha_i$ is $A$). Each row entry corresponds to a decision of the automaton. The first column, with label "$\approx_i$", contains the relational operator of the formula $\alpha_i[p_i U_{\approx c_i} q_i]$. The second column, with label "$b_i$", contains either 1 or 2. The case $b_i = 0$ is not considered: if a clock is not used the automaton $\mathcal{A}_{p_0}$ just ignores its value. The third column, with label "Clock Constraints", contains the clock constraint that is true for the clock variable $x_i$, at the time of decision. The fourth and last column with label, "Fulfillment", contains conditions of type "$q \in s$", with the meaning that the formula $\alpha_i[p_i U_{\approx c_i} q_i]$ is satisfied if $q$ belongs to $s$.

The case considered in the above example corresponds to the entry 2 of the table.

| | $\approx_i$ | $b_i$ | Clock Constraints | Fulfillment |
|---|---|---|---|---|
| 1. | $<$ | 1 | $x_i < c_i$ | $q_i \in s',\ q_i \in s_j$ |
| 2. | | | $x_i = c_i$ | $q_i \in s'$ |
| 3. | | 2 | $x_i < c_i$ | $q_i \in s',\ q_i \in s_j$ |
| 4. | $\leq$ | 1 | $x_i \leq c_i$ | $q_i \in s',\ q_i \in s_j$ |
| 5. | | 2 | $x_i < c_i$ | $q_i \in s',\ q_i \in s_j$ |
| 6. | | | $x_i = c_i$ | $q_i \in s'$ |
| 7. | $>$ | 1 | $x_i > c_i$ | $q_i \in s',\ p_i \wedge q_i \in s$ |
| 8. | | | $x_i = c_i$ | $q_i \in s'$ |
| 9. | | 2 | $x_i > c_i$ | $p_i \wedge q_i \in s,\ q_i \in s'$ |
| 10. | $\geq$ | 1 | $x_i \geq c_i$ | $p_i \wedge q_i \in s,\ q_i \in s'$ |
| 11. | | 2 | $x_i = c_i$ | $q_i \in s'$ |
| 12. | | | $x_i > c_i$ | $p_i \wedge q_i \in s,\ q_i \in s'$ |

**Fig. 4.** The description of the fulfillment of the until formulae

When a case which is not considered in the Fig. 4 occurs there are two possibilities:
a) the formula is pending: $\mathcal{A}_{p_0}$ checks that $p_i \in s \cup s'$ and if $(x_i \approx_i c_i)$ then $\sim q_i \in s \cap s'$, no decision about the fulfillment of the formula is possible at this time;
b) the formula is false: one of the remaining cases occurs and $\mathcal{A}_{p_0}$ halts.

To handle the before formulae $\mathcal{A}_{p_0}$ behaves in a symmetric way. The treatment of the before formulae differs from that of teh until formulae since the fulfillment of until formulae is certified on a finite fragment of the model, while before formulae can be fulfilled on infinite fragments (notice that this happens when the corresponding until formulae are pending forever). The automaton $\mathcal{A}_{p_0}$ uses a counter and the acceptance condition to detect this situation. The counter is implemented in the last component of the states of $\mathcal{A}_{p_0}$, is incremented at each transition step, and halts when its value is $i$ and the $i$-th until formula is pending. Thus, the last component of the states in a path changes infinitely often if and only if there is not a formula pending forever and this can be checked by a Büchi automaton. We observe that the automaton we have just described can be modified in order to use only $n$ clocks instead of $2n$, since we can use the same clock for an until formula and the corresponding before formula (along a path they are satisfied in disjoint time intervals).

By Lemmata 2 and 3 and the above arguments the following lemmata hold:

**Lemma 4** *Let $p_0$ be an STCTL-formula in PNF. If $p_0$ is satisfiable then there is an STCTL-model of $p_0$ that is accepted by $\mathcal{A}_{p_0}$.*

**Lemma 5** *Let $p_0$ be an STCTL-formula in PNF. Each timed tree accepted by $\mathcal{A}_{p_0}$ is an STCTL-model for $p_0$.*

We now give the main result of this section.

**Theorem 10** *The STCTL-satisfiability problem is decidable.*

*Proof.* This problem can be reduced to the emptiness problem for TBTA. The reduction is obtained by the construction of the timed tree automaton $\mathcal{A}_{p_0}$, given an STCTL-formula $p_0$. By Lemmata 4 and 5, we have that an STCTL-formula $p_0$ is satisfiable if and only if $T(A_{p_0})$ is not empty. By Corollary 2 we have the decidability.

The satisfiability problem is dual to the validity problem, that is $p$ is satisfiable if and only if $\neg p$ is not valid. Hence, the following corollary holds.

**Corollary 3** *The validity problem is decidable for STCTL-formulae.*

Finally, we observe that the validity and the satisfiability problems for STCTL are in EXPTIME. This follows from the the fact that the cardinality of the extended closure of an STCTL-formula in PNF is linear in the length of the formula, the automaton construction is exponential in the number of the formulae in the extended closure and the emptiness problem for TBTA is exponential in the length of the timing constraints and polynomial in the number of states and transition rules [21,25].

## 5 Conclusions

In this paper we have introduced the 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. This result turns out to be important to show the decidability of STCTL, which is introduced in the second part of the paper. Our logic is a real-time extension of CTL.

The main motivation for this logic is to introduce a decidable real-time branching-time logic. In order to achieve this, two main limitations have been placed on it.

The first limitation concerns the syntax: we do not allow equality in the timing constraints. Denote by $\mathrm{STCTL}^{=}$ the temporal logic obtained by STCTL by adding the "=" in the timing constrains. It can be proved by a reduction from the problem of determining the existence of a recurring computation of a nondeterministic 2-counters machine[2] that the satisfiability problem for $\mathrm{STCTL}^{=}$ is undecidable. A detailed proof can be found in [16].

The second limitation concerns the semantics: our time structure has a discrete branching, that is branching is not allowed at any time of an interval along a path. In spite of this limitation, in comparison with traditional temporal logics, it is still possible to consider in STCTL continuous behaviours. In fact, the fulfillment of the formulae is interpreted on a dense time structure, the timed path. Nevertheless, this second limitation seems to be less tight than the previous one. At one hand, it is known that by allowing a full dense branching-time structure along with the equality (TCTL) the satisfiability problem is not decidable [1]. On the other hand, we have the decidability of STCTL. Besides the lack of the equality in the timing constraints (which is indeed necessary), the only gap in comparison to TCTL is the fact that we discretized the time instants at which a state can have many "immediate" futures. It is not clear if TCTL without equality has a decidable satisfiability problem. In [17] the finite satisfiability of TCTL without equality has been proved to be decidable by reducing this problem to the satisfiability problem in a proper fragment of STCTL.

*Acknowledgements.* We thank Emanuela Fachini for the constructive discussions and for supporting us in the early stages of this research. We also thank Rajeev Alur for the helpful discussions and suggestions. Further, we greatly appreciate the useful criticism provided by the anonymous referee which allowed us to improve this paper.

## References

1. R. Alur, C. Courcoubetis, D. Dill: Model-checking in Dense Real-time. Inform. Comput. 104: 2–34 (1993)

[2] This problem has been proved to be $\Sigma_1^1$-hard in [5].

2. R. Alur, D. Dill: A theory of timed automata. Theor. Comput. Sci. 126: 183–235 (1994)
3. R. Alur, T. Feder, T.A. Henzinger: The Benefits of Relaxing Punctuality. JACM 43: 116–146 (1996)
4. R. Alur, T.A. Henzinger: Real-Time Logics: Complexity and Expressiveness. Inform. Comput. 104: 35–77 (1993)
5. R. Alur, T.A. Henzinger: A really temporal logic. JACM 41: 181–204 (1994)
6. O. Bernholtz, M.Y. Vardi, P. Wolper: An automata theoretic approach to branching-time model-checking. Proc. of the 6th International Conference on Computer Aided Verification (CAV'94), Lecture Notes in Computer Science, Vol. 818. pp. 142–155, Berlin Heidelberg New York: Springer 1994
7. J.R. Büchi: On a decision method in restricted second order arithmetic. In: E. Nagel et al. (eds.) Proc. Internat. Congr. on Logic, Methodology and Philosophy of Science. pp. 1–11. Stanford, CA: Stanford Univ. Press 1960
8. E.M. Clarke, E.A. Emerson: Design and synthesis of synchronization skeletons using Branching Time Temporal Logic. Proc. of Workshop on Logics of Programs, Lecture Notes in Computer Science, Vol. 131, pp. 52–71. Berlin Heidelberg New York: Springer 1981
9. C. Courcoubetis, M.Y. Vardi, P. Wolper: Reasoning about Fair Concurrent Programs. Proc. of the 18th ACM Symposium on Theory of Computing, pp. 283–294. Berkeley, 1986
10. F. Gécseg, M. Steinby: Tree Automata. Budapest: Akademiai Kiado 1984
11. E.A. Emerson: Temporal and Modal logic. In: J. van Leeuwen, (ed.) Handbook of Theoretical Computer Science, Vol. B, pp. 997–1072. Amsterdam: North-Holland 1990
12. E.A. Emerson, J.Y. Halpern: Sometimes and Not Never Revisited: On Branching versus Linear Time Temporal Logic. JACM 33: 151–178 (1986)
13. E.A. Emerson, A.K. Mok, A.P. Sistla, J. Srinivasan: Quantitative Temporal Reasoning. Proc. of the 2nd International Conference on Computer-Aided Verification (CAV'90), Lecture Notes in Computer Science, Vol. 531, pp. 136–145. Berlin Heidelberg New York: Springer 1991
14. T.A. Henzinger: It's about time: Real-time logics reviewed. Proc. of the 9th International Conference on Concurrency Theory (CONCUR'98), Lecture Notes in Computer Science, Vol. 1466, pp. 439–454. Berlin Heidelberg Newe York: Springer 1998
15. F. Jahanian, A.K. Mok: Safety analysis of timing properties in real-time systems. IEEE Trans. Softw. Eng. 12(9): 890–904 (1986)
16. S. La Torre: Advances in Finite Automata and Temporal Logic for System Verification. Doctorate Thesis, University of Napoli, 1999
17. S. La Torre, M. Napoli: A decidable dense branching-time temporal logic. Proc. of the 20th Conference on the Foundations of Software Technology and Theoretical Computer Science, (FSTTCS'00), Lecture Notes in Computer Science, Vol. 1974, pp. 139–150. Berlin Heidelberg New York: Springer 2000
18. R. McNaughton, Testing and generating infinite sequences by a finite automaton. Inform. Control 9: 521–530 (1966)
19. A. Pnueli, E. Harel: Applications of Temporal Logic to the Specification of Real Time Systems. Formal Techniques in Real-time and Fault-tolerant Systems, Lecture Notes in Computer Science, Vol. 331, pp. 84–98. Berlin Heidelberg New York: Springer 1988
20. M.O. Rabin: Decidability of second-order theories and automata on infinite trees. Trans. Amer. Math. Soc. 141: 1–35 (1969)
21. M.O. Rabin: Weakly definable relations and special automata. In: Y. Bar-Hillel (ed.) Mathematical Logic and Foundations of Set theory. Amsterdam: North-Holland 1970
22. M.O. Rabin: Automata on Infinite Objects and Church's Problem. Amer. Math. Soc. Providence, **RI**, 1972

23. W. Thomas: Automata on Infinite Objects. In: J. van Leeuwen (ed.) Handbook of Theoretical Computer Science Vol. B, pp. 133-191. Amsterdam: North-Holland 1990
24. M.Y. Vardi: Nontraditional Applications of Automata Theory. Proc. of the International Symposium TACS'94, Lecture Notes in Computer Science, Vol. 789, pp. 575–597. Berlin Heidelberg New York: Springer 1994
25. M.Y. Vardi, P. Wolper: Automata-theoretic techniques for modal logics of programs. J. Comput. Syst. Sci. 32: 183–211 (1986)
26. M.Y. Vardi, P. Wolper: Reasoning about infinite computations. Inform. Comput. 115: 1–37 (1994)