

A Decidable Dense Branching-time Temporal Logic

Salvatore La Torre

University of Pennsylvania and Università degli Studi di Salerno

latorre@seas.upenn.edu

Margherita Napoli

Università degli Studi di Salerno

napoli@unisa.it

Abstract

Timed computation tree logic (TCTL) extends CTL by allowing timing constraints on the temporal operators. The semantics of TCTL is defined on a dense tree. The satisfiability of TCTL-formulae is undecidable even if the structures are restricted to dense trees obtained from timed graphs. According to the known results there are two possible causes of such undecidability: the denseness of the underlying structure and the equality in the timing constraints. We prove that the second one is the only source of undecidability when the structures are defined by timed graphs. In fact, if the equality is not allowed in the timing constraints of TCTL-formulae then the finite satisfiability in TCTL is decidable. We show this result by reducing this problem to the emptiness problem of timed tree automata, so strengthening the already well-founded connections between finite automata and temporal logics.

1 Introduction

In 1977 Pnueli proposed Temporal Logic as a formalism to specify and verify computer programs [Pnu77]. This formalism turned out to be greatly useful for reactive systems [HP85], that is systems maintaining some interaction with their environment, such as operating systems and network communication protocols. Several temporal logics have been introduced and studied in literature, and now this formalism is widely accepted as specification language for reactive systems (see [Eme90] for a survey).

Temporal logic formulae allow to express temporal requirements on the occurrence of events. Typical temporal operators are “until”, “next”, “sometimes”, “always”, and a typical assertion is “p is true until q is true”. These operators allow us only to express qualitative requirements, that is constraints on the temporal ordering of the events, but we cannot place bounds on the time a certain property must be true. As a consequence traditional temporal logics have been augmented by adding timing constraints to temporal operators, so that assertions such as “p is true until q is true within time 5” can be expressed. These logics, which are often referred to as real-time or quantitative temporal logics, are suitable when it is necessary to explicitly refer to time delays between events and then we want to check that some hard real-time constraints are satisfied.

Besides the usual classification in linear and branching-time logics, real-time logics are classified

according to the nature of the time model they use. Temporal logics based on discrete time models are presented in [EMSS90, JM86, Koy90, PH88]. An alternative approach is to model time as a dense domain. Temporal logics with this time model are MITL [AFH96], TCTL [ACD93], STCTL [LN97], and GCTL [PH88]. For more about real-time logics, see [AH93, Hen98].

In this paper we are interested in branching-time temporal logics which use a dense time domain and in particular we will consider the satisfiability problem in TCTL. Given a formula φ we want to determine if there exists a structure M satisfying it. The syntax of TCTL is given by augmenting the temporal operators of CTL [CE81] (except for the “next” which is discarded since it does not have any meaning in a dense time domain) with a timing constraint of type $\approx c$, where \approx is one among $<$, \leq , $>$, \geq , and $=$, and c is a rational number. The semantics of TCTL is given on a dense (or continuous) tree. It turns out that the satisfiability problem in TCTL is undecidable even if the semantics is restricted to dense trees obtained from timed graphs (*finite satisfiability* [ACD93]), that is, timed transition systems where the transitions depend also on the current value of a finite number of clock variables.

Another real-time branching-time temporal logic is STCTL [LN97] which is obtained by restricting both the semantics and the syntax of TCTL. Instead of a dense tree, a timed ω -tree is used to define the semantics of formulae, and the equality is not allowed in the timing constraints. With these restrictions the STCTL-satisfiability problem turns out to be decidable. This result is obtained by reducing the STCTL-satisfiability problem to the emptiness problem of finite automata on timed ω -trees, which is shown to be decidable in [LN97]. Introducing the equality in the timing constraints causes the loss of the decidability. A similar phenomenon was observed in MITL, where the decidability is lost when the restriction to non-singular intervals is relaxed [AFH96].

In this paper we prove that this indeed holds also for the finite satisfiability in TCTL. In particular, we reduce the finite satisfiability problem of TCTL-formulae without equality in the timing constraints, to the emptiness problem of timed tree automata, via translation to the satisfiability problem of TCTL-formulae with respect to a proper subclass of STCTL-structures. Restricting the class of STCTL-structures is necessary since along any path of a timed graph the truth assignments of the atomic propositions vary according to a sequence of left-closed right-opened intervals, while in general in STCTL-structures the truth assignments change according to sequences of time intervals which are alternatively singular and opened. Defined the language of a logic as the set of formulae which are satisfiable, as a consequence of the previous result we have that TCTL interpreted on timed graphs is language equivalent to a proper restriction of STCTL. Moreover, in this paper we also introduce a concept of highly-deterministic timed tree automaton with the aim of matching the concept of regular tree in ω -tree languages.

The use of the theory of timed tree automata to achieve the decidability of the TCTL finite satisfiability, strengthens the relationship between finite automata and temporal logics, also in the case of real-time logics. In a recent paper [DW99] an automata-theoretic approach to TCTL-model checking has been presented. There the authors introduced timed alternating tree automata and rephrased the model-checking problem as a particular word problem for these automata. For timed alternating tree automata, this decision problem is decidable while the emptiness problem is not decidable.

The rest of the paper is organized as follows. In section 2 we recall the main definitions and results from the theory of timed tree automata, and we introduce a concept of highly-deterministic timed tree automaton. In section 3 we recall the temporal logics TCTL and STCTL with the related decidability

results. The main result of this paper is presented in section 4, where the finite satisfiability in TCTL is shown to be decidable via reduction to the emptiness problem of timed tree automata. Finally, we give our conclusions in section 5.

2 Timed tree automata

In this section we recall some definitions and results concerning to timed automata [AD94, LN97], and introduce the concept of highly-deterministic timed tree automaton.

Let Σ be an alphabet and $dom(t)$ be a subset of $\{1, \dots, k\}^*$, for an integer $k > 0$, such that (i) $\varepsilon \in dom(t)$, and (ii) if $v \in dom(t)$, then for some $j \in \{1, \dots, k\}$, $vi \in dom(t)$ for any i such that $1 \leq i \leq j$ and $vi \notin dom(t)$ for any $i > j$. A Σ -valued ω -tree is a mapping $t : dom(t) \rightarrow \Sigma$. Given a Σ -valued ω -tree t a subtree t_u of t rooted at $u \in dom(t)$ is the Σ -valued ω -tree defined as $dom(t_u) = \{v \mid uv \in dom(t)\}$ and $t_u(v) = t(uv)$ for each $uv \in dom(t)$. For $v \in dom(t)$, we denote with $pre(v)$ the set of prefixes and with $deg(v)$ the arity of v . A *path* in t is a maximal subset of $dom(t)$ linearly ordered by the prefix relation. Often we will denote a path π with the ordered sequence of its nodes v_0, v_1, v_2, \dots where v_0 is ε . With $In(t|\pi)$ we denote the set of symbols labelling infinitely many nodes on a path π in t . With \mathfrak{R}_+ we denote the set of the nonnegative real numbers. A *timed Σ -valued ω -tree* is a pair (t, τ) where t is a Σ -valued ω -tree and τ , called *time tree*, is a mapping from $dom(t)$ into \mathfrak{R}_+ such that (i) $\tau(v) > 0$, for each $v \in dom(t) - \{\varepsilon\}$, and $\tau(\varepsilon) \geq 0$ (*positiveness*), and (ii) for each path π and for each $x \in \mathfrak{R}_+$ there exists $v \in \pi$ such that $\sum_{u \in pre(v)} \tau(u) \geq x$ (*progress property*). Nodes of a timed ω -tree become available as the time elapses, that is, at a given time only a finite portion of the tree is available. Each node of a timed ω -tree is labelled by a pair (symbol, real number): for the root the real number is the absolute time of occurrence, while for the other nodes is the time which has elapsed since their parent node was read. Positiveness implies that a positive delay occurs between any two consecutive nodes along a path. Progress property guarantees that infinitely many events (i.e. nodes appearing at input) cannot occur in a finite slice of time (*nonzenoness*). Given a Σ -valued timed ω -tree (t, τ) and a node v , we denote with γ_v the time at which v is available, that is $\gamma_v = \sum_{u \in pre(v)} \tau(u)$. In the rest of the paper, we will consistently use γ to denote absolute time, i.e. time elapsed from the beginning of a computation, and τ to denote delays between events. Moreover, we will use the term *tree* to refer to a Σ -valued ω -tree for some alphabet Σ and the term *timed tree* to refer to a timed Σ -valued ω -tree. A (timed) tree language is any set of (timed) trees. Now we recall the definition of timed Büchi tree automaton. It is possible to extend this paradigm by considering other acceptance conditions such as Muller, Rabin, or Streett [Tho90]. Timed Muller tree automata as well as timed Büchi tree automata were introduced and studied in [LN97]. To define timed automata we introduce the notion of clock, timing constraint, and clock valuation. A finite set of *clock variables* (or simply *clocks*) is used to test timing constraints. Each clock can be seen as a chronograph which is synchronized to a unique system clock. Clocks can be read or set to zero (reset): after a reset, a clock automatically restarts. Timing constraints are expressed by clock constraints. Let C be a set of clocks, the set of clock constraints $\Xi(C)$ contains boolean combinations of simple clock constraints of type $x \leq y + c$, $x \geq y + c$, $x \leq c$, and $x \geq c$, where $x, y \in C$ and c is a rational number. A *clock valuation* is a mapping $\nu : C \rightarrow \mathfrak{R}_+$. If ν is a clock valuation, λ is a set of clocks and d is a real number, we denote with $[\lambda \rightarrow 0](\nu + d)$ the clock valuation that gives 0 for each clock $x \in \lambda$ and $\nu(x) + d$ for each

clock $x \notin \lambda$.

A *Büchi timed tree automaton* is a 6-tuple $A = (\Sigma, S, S_0, C, \Delta, F)$, where:

- Σ is an alphabet;
- S is a finite set of locations;
- $S_0 \subseteq S$ is the set of starting locations;
- C is a finite set of clocks;
- Δ is a finite subset of $\bigcup_{k \geq 0} (S \times \Sigma \times S^k \times (2^C)^k \times \Xi(C))$;
- $F \subseteq S$ is the set of accepting locations.

A timed Büchi tree automaton A is *deterministic* if $|S_0| = 1$ and for each pair of different tuples $(s, \sigma, s_1, \dots, s_k, \lambda_1, \dots, \lambda_k, \delta)$ and $(s, \sigma, s'_1, \dots, s'_k, \lambda'_1, \dots, \lambda'_k, \delta')$ in Δ , δ and δ' are inconsistent (i.e., $\delta \wedge \delta' = \text{false}$ for all clock valuations).

A state system is completely determined by a location and a clock valuation, thus it is denoted by a pair (s, ν) . A transition rule $(s, \sigma, s_1, \dots, s_k, \lambda_1, \dots, \lambda_k, \delta) \in \Delta$ can be described as follows. Suppose that the system is in the state (s, ν) , and after a time τ the symbol σ is read. The system can take the transition $(s, \sigma, s_1, \dots, s_k, \lambda_1, \dots, \lambda_k, \delta)$ if the current clock valuation (i.e. $\nu + \tau$) satisfies the clock constraint δ . As a consequence of the transition, the system will enter the states $(s_1, \nu_1), \dots, (s_k, \nu_k)$ where $\nu_1 = [\lambda_1 \rightarrow 0](\nu + \tau), \dots, \nu_k = [\lambda_k \rightarrow 0](\nu + \tau)$. Each node of a timed tree has thus assigned a location and a clock valuation, according to the transition rules in Δ . Formally, this is captured by the concept of run. A *run* of A on a timed tree (t, τ) is a pair (r, ν) , where:

- $r : \text{dom}(t) \rightarrow S$ and $\nu : \text{dom}(t) \rightarrow \mathfrak{R}_+^C$;
- $r(\varepsilon) \in S_0$ and $\nu(\varepsilon) = \nu_0$, where $\nu_0(x) = 0$ for any $x \in C$;
- for $v \in \text{dom}(t)$, $k = \text{deg}(v)$: $(r(v), t(v), r(v_1), \dots, r(v_k), \lambda_1, \dots, \lambda_k, \delta) \in \Delta$, $\nu(v) + \tau(v)$ fulfils δ and $\nu(v_i) = [\lambda_i \rightarrow 0](\nu(v) + \tau(v)) \forall i \in \{1, \dots, k\}$.

Clearly, deterministic timed automata have at most one run for each timed tree. A timed tree (t, τ) is accepted by A if and only if there is a run (r, ν) of A on (t, τ) such that $\text{In}(r|\pi) \cap F \neq \emptyset$ for any path π in r . The language accepted by A , denoted by $T(A)$, is the set of all timed trees accepted by A . In the following we refer to (timed) Büchi tree automata simply as (timed) tree automata. No confusion will arise since we do not consider other acceptance conditions.

For a timed tree automaton the set of states is infinite. However, they can be finitely partitioned according to a finite-index equivalence relation over the clock valuations. Each equivalence class, called *clock region*, is defined in such a way that all the clock valuations in an equivalence class satisfy the same set of clock constraints from a given timed automaton (see [AD94] for a precise definition). Given a clock valuation ν , $[\nu]$ denotes the clock region containing ν . A clock region α' is said to be a *time-successor* of a clock region α if and only if for any $\nu \in \alpha$ there is a $d \in \mathfrak{R}_+$ such that $\nu + d \in \alpha'$. The *region automaton* of a timed tree automaton A is a transition system 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 $R(S_0) = \{\langle s_0, \alpha_0 \rangle \mid s_0 \in S_0 \text{ and } \alpha_0 \text{ satisfies } x = 0 \text{ for all } x \in C\}$;
- the transition rules $R(\Delta)$ such that: $(\langle s, \alpha \rangle, \sigma, \langle s_1, \alpha_1 \rangle, \dots, \langle s_k, \alpha_k \rangle) \in R(\Delta)$ if and only if

$(s, \sigma, s_1, \dots, s_k, \lambda_1, \dots, \lambda_k, \delta) \in \Delta$ and there is a time-successor α' of α such that α' satisfies δ and $\alpha_i = [\lambda_i \rightarrow 0]\alpha'$ for all $i \in \{1, \dots, k\}$.

The region automaton is the key to reduce the emptiness problem of timed tree automata to the emptiness problem of tree automata. Given a timed tree language T , $\text{Untime}(T(A))$ is the tree language $\{t \mid (t, \tau) \in T\}$. We will denote by $R(A)$ the timed tree automaton accepting $\text{Untime}(T(A))$ and obtained by the region automaton (see [LN97] for more details).

Remark 1 [LN97]

1. *The emptiness problem of timed Büchi tree automata is decidable in time exponential in the length of timing constraints and polynomial in the number of locations.*
2. *The class of timed Büchi tree automata is closed under union and intersection.*

We end this section by introducing a new definition. The aim is to define for timed tree automata a concept which captures some of the properties that regular trees have in the context of tree languages. We will use this notion to relate timed tree automata to timed graphs. A timed tree automaton $A = (\Sigma, S, S_0, \Delta, C, F)$ is said to be *highly deterministic* if $\text{Untime}(T(A))$ contains a unique tree, and for $s \in S$, $e = (s, \sigma, s_1, \dots, s_k, \lambda_1, \dots, \lambda_k, \delta) \in \Delta$ and $e' = (s, \sigma', s'_1, \dots, s'_h, \lambda'_1, \dots, \lambda'_h, \delta') \in \Delta$ 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 $G = (V, E, \text{lab})$ such that there is a mapping $\theta : \text{dom}(t) \rightarrow V$ such that:

- for any $u, u' \in \text{dom}(r)$, $\theta(u) = \theta(u')$ implies that $\text{deg}(u) = \text{deg}(u')$, and for each $i = 1, \dots, \text{deg}(u)$, $\theta(ui) = \theta(u'i)$;
- $E = \{(\theta(u), \theta(ui), i) \mid u \in \text{dom}(r) \text{ and } i \leq \text{deg}(u)\}$, and $(v, v', i) \in E$ is an edge from v to v' labelled by i ;
- for $v \in V$, $\text{lab}(v) = (r(u), t(u))$ 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 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 $\text{Untime}(T(A))$ is also not empty. Thus there exist an accepting regular run r of $R(A)$ and a corresponding regular tree $t \in \text{Untime}(T(A))$ [Rab72]. Let $G = (V, E, \text{lab})$ be a shrink of r and t , where ε corresponds to v_0 and $\text{lab}(v_0) = (r(\varepsilon), t(\varepsilon))$. We define A_{det} as the timed tree automaton $(\Sigma, S_{\text{det}}, \{s'_0\}, \Delta_{\text{det}}, C, S_{\text{det}})$ where:

- $\langle s'_0, [\nu_0] \rangle = r(\varepsilon)$ with $\nu_0(x) = 0$ for any $x \in C$;
- $S_{\text{det}} = \{s \mid \exists v \in V \text{ such that } \text{lab}(v) = (\langle s, \alpha \rangle, \sigma)\}$;
- a transition rule $(s, \sigma, s_1, \dots, s_k, \lambda_1, \dots, \lambda_k, \delta) \in \Delta$ belongs to Δ_{det} if and only if the sequence $(v, v_1, 1), \dots, (v, v_h, h) \in E$ of all the edges from v is such that (1) $h = k$, (2) $\text{lab}(v) = (\langle s, \alpha \rangle, \sigma)$,

and $lab(v_i) = (\langle s_i, \alpha_i \rangle, \sigma_i)$ for $i = 1, \dots, k$, and (3) there exists a $d > 0$ such that $(\alpha + d)$ satisfies δ and $\alpha_i = [\lambda_i \rightarrow 0](\alpha + d)$ for $i = 1, \dots, k$.

Directly from the above definition we have that $\text{Uptime}(T(A_{det})) = \{t\}$, 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 . ■

Later in the paper we will use the following property. Given a highly-deterministic timed tree automaton A , there is a highly-deterministic timed automaton A' such that $T(A) = T(A')$ and for any transition rule $(s, \sigma, s_1, \dots, s_k, \lambda_1, \dots, \lambda_k, \delta)$ 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.

3 Timed Computation Tree Logic

In this section we recall the real-time branching-time temporal logics TCTL [ACD93] and STCTL [LN97].

Let AP be a set of atomic propositions, the syntax of TCTL-formulae is given by the following grammar:

$$\varphi := p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \exists[\varphi U_{\approx c} \varphi] \mid \forall[\varphi U_{\approx c} \varphi]$$

where $p \in AP$, $\approx \in \{<, \leq, >, \geq\}$, and c is a rational number. Notice that the TCTL-syntax given in [ACD93] allows the use of equality in the timing constraints. Here we restrict the syntax to obtain the decidability of the finite satisfiability which is, in general, undecidable.

Before giving the semantics of TCTL, we introduce some common notation. The constant FALSE is equivalent to $\varphi \wedge \neg\varphi$, the constant TRUE is equivalent to $\neg \text{FALSE}$, $\diamond_{\approx c} \varphi$ and $\square_{\approx c} \varphi$ are equivalent to $\text{TRUE} U_{\approx c} \varphi$ and $\neg \diamond_{\approx c} \neg \varphi$, respectively. In the rest of the paper with AP we denote the set of atomic propositions of the considered TCTL-formulae. If it is not differently stated, with \approx we refer to a relational operator in $\{<, \leq, >, \geq\}$, and with c to a rational number. We define a dense path through a set of nodes S as a function $\rho : \mathfrak{R}_+ \longrightarrow S$. With ρ_I we denote the restriction of ρ to an interval I and with $\rho_{[0,b)} \cdot \rho'$ the dense path defined as $(\rho_{[0,b)} \cdot \rho')(d) = \rho(d)$, if $d < b$, and $(\rho_{[0,b)} \cdot \rho')(d) = \rho'(d)$, otherwise. The semantics of TCTL is given with respect to a *dense tree*. A Σ -valued dense tree M is a triple (S, μ, f) where:

- S is a set of nodes;
- $\mu : S \longrightarrow \Sigma$ is a labelling function;
- f is a function assigning to each $s \in S$ a set of dense paths through S , starting at s , and satisfying the *tree constraint*: $\forall \rho \in f(s)$ and $\forall t \in \mathfrak{R}_+$, $\rho_{[0,t)} \cdot f(\rho(t)) \subseteq f(s)$.

Given a 2^{AP} -valued dense tree $M = (S, \mu, f)$, a state s , and a formula φ , φ is satisfied at s in M if and only if $M, s \models \varphi$, where the relation \models is defined as follows:

- for $\psi \in AP$, $M, s \models \psi$ if and only if $\psi \in \mu(s)$;
- $M, s \models \neg\psi$ if and only if $\text{not}(M, s \models \psi)$;
- $M, s \models \psi_1 \wedge \psi_2$ if and only if $M, s \models \psi_1$ and $M, s \models \psi_2$;
- $M, s \models \exists[\psi_1 U_{\approx c} \psi_2]$ if and only if $\exists \rho \in f(s)$ and $\exists d \approx c$ such that $M, \rho(d) \models \psi_2$ and for each d' such that $0 \leq d' < d$, $M, \rho(d') \models \psi_1$;
- $M, s \models \forall[\psi_1 U_{\approx c} \psi_2]$ if and only if $\forall \rho \in f(s)$, $\exists d \approx c$ such that $M, \rho(d) \models \psi_2$ and $M, \rho(d') \models \psi_1$ for each d' such that $0 \leq d' < d$.

We say that M is a TCTL-model of φ if and only if $M, s \models \varphi$ for some $s \in S$. Moreover, a TCTL-formula φ is said to be *satisfiable* if and only if there exists a TCTL-model of φ .

Let φ be a TCTL-formula. We define the *closure* of φ , denoted by $cl(\varphi)$, as the set of all the subformulae of φ and the *extended closure*, denoted by $ecl(\varphi)$, as the set $cl(\varphi) \cup \{\neg p \mid p \in cl(\varphi)\}$. Moreover, we define $S_\varphi \subseteq 2^{ecl(\varphi)}$ as the collection of sets s with the following properties:

- $\psi \in s \implies \neg\psi \notin s$;
- $\psi_1 \wedge \psi_2 \in s \implies \psi_1 \in s$ and $\psi_2 \in s$;
- $\alpha[\psi_1 U_{\approx c} \psi_2] \in s$, $\alpha \in \{\forall, \exists\} \implies \psi_1 \in s$ or $(\psi_2 \in s$ and $(0 \approx c))$;
- s is maximal, that is for each $\psi \in ecl(\varphi)$: either $\psi \in s$ or $\neg\psi \in s$.

Note that S_φ contains the maximal sets of formulae in $ecl(\varphi)$ which are consistent, in the sense that given an $s \in S_\varphi$ and a dense tree (S, μ, f) , the fulfilment at a given $r \in S$ of a formula in s does not prevent all the other formulae in s from being satisfied at r . From now on we only consider TCTL-formulae, thus we will refer to them simply as formulae since no confusion can arise. In the rest of this section we recall two semantic restrictions to TCTL that have been considered in literature.

3.1 Finite satisfiability

A first restriction of TCTL-semantics consists of considering only dense trees defined by runs of a timed graph [ACD93]. A *timed graph* is a tuple $G = (V, \mu, s_0, E, C, \Lambda, \xi)$, where:

- V is a finite set of vertices;
- $\mu : V \longrightarrow 2^{AP}$ is a labelling function;
- s_0 is the start vertex;
- $E \subseteq V \times V$ is the set of edges;
- C is a finite set of clocks;
- $\Lambda : E \longrightarrow 2^C$ maps each edge to a set of clocks to reset;
- $\xi : E \longrightarrow \Xi(C)$ maps each edge to a clock constraint.

A timed graph is a timed transition system, where vertices correspond to locations and edges to transitions. A state is given by the current location and the array of all clock values. When a clock constraint is satisfied by the clock valuation of the current state, the corresponding transition can be

taken. A transition e forces the system to move, instantaneously, to a new state which is described by the target location of e , and the clock values obtained by resetting the clocks in the reset set of e . Any computation of the system maps reals to states. This concept is captured by the notion of run. Given a state (s, ν) of a timed graph G , an (s, ν) -run of G is an infinite sequence of triples $(s_1, \nu_1, \tau_1), (s_2, \nu_2, \tau_2), \dots$ where:

- $s_1 = s$, $\nu_1 = \nu$, and $\tau_1 = 0$;
- for $i > 1$ $s_i \in S$, $\tau_i \in \mathfrak{R}_+$, and ν_i is a clock valuation;
- $e_i = (s_i, s_{i+1}) \in E$, $\nu_{i+1} = [\Lambda(e_i) \rightarrow 0](\nu_i + \tau_{i+1})$, $(\nu_i + \tau_{i+1})$ satisfies the enabling condition $\xi(e_i)$, and the series of reals τ_i is divergent (*progress condition*).

An (s, ν) -run can be also seen as a real-valued mapping $\rho(d)$ defined as $\rho(d) = (s_i, \nu_i + d - \gamma_i)$ for $d \in \mathfrak{R}_+$ such that $\gamma_i \leq d < \gamma_{i+1}$ (ρ is also said to be a dense path of G). Notice that a dense path ρ gives for each time a truth assignment of the atomic propositions. Moreover, the truth values stay unchanged in intervals of type $[\gamma_i, \gamma_{i+1})$. The dense tree M defined by a timed graph G is a tuple $(S \times \mathfrak{R}^n, \mu', f)$ where $\mu'(s, \nu) = \mu(s)$ and $f(s, \nu)$ is the set of all the paths corresponding to (s, ν) -runs of G . For a formula φ , we say that $G \models \varphi$ if and only if $M, (s_0, \nu_0) \models \varphi$ where $\nu_0(x) = 0$ for any clock $x \in C$. Thus a formula φ is *finitely satisfiable* if and only if there exists a timed graph G such that $G \models \varphi$.

3.2 Restricting the semantics to timed trees

In this section we recall the temporal logic STCTL which is obtained restricting the TCTL-semantics to dense trees obtained from $2^{AP} \times 2^{AP}$ -valued ω -trees. An STCTL-structure is a timed $2^{AP} \times 2^{AP}$ -valued ω -tree (t, τ) with $\tau(\varepsilon) = 0$. Given an STCTL-structure (t, τ) we denote by t_{open} and t_{sing} the functions defined as $(t_{open}(v), t_{sing}(v)) = t(v)$ for each $v \in dom(t)$. An open and a singular interval along the paths in (t, τ) correspond to each node v : $t_{open}(v)$ and $t_{sing}(v)$ are the sets of the atomic propositions which are true in these two intervals. Given a path $\pi = v_0, v_1, v_2, \dots$ in an STCTL-structure (t, τ) , a dense path in (t, τ) corresponding to π and shifted by d is a function $\rho_d^\pi : \mathfrak{R}_+ \rightarrow 2^{AP}$ such that for any natural number i :

$$\rho_d^\pi(d') = \begin{cases} t_{sing}(v_i) & \text{if } d + d' = \gamma_{v_i} \\ t_{open}(v_{i+1}) & \text{if } \gamma_{v_i} < d + d' < \gamma_{v_{i+1}}. \end{cases}$$

Thus any dense path in (t, τ) corresponds to a sequence of alternatively open and singular intervals where the truth values stay unchanged. Clearly, an STCTL-structure has a dense time semantics on paths and a discrete branching-time structure. In particular, an STCTL-structure (t, τ) defines the dense tree $M^{t, \tau} = (S, \mu, f)$ where (1) $S = \{(vi, d) \mid v \in dom(t) \text{ and } 0 < d \leq \tau(vi)\} \cup \{(\varepsilon, 0)\}$, (2) $\mu(\varepsilon, 0) = t_{sing}(\varepsilon)$, $\mu(vi, d) = t_{open}(vi)$ if $d < \tau(vi)$, and $\mu(vi, d) = t_{sing}(vi)$ otherwise, and (3) $f(\varepsilon, 0)$ is the set of all dense paths ρ_0^π of (t, τ) and $f(vi, d)$ is the set of all the dense paths $\rho_{\gamma_v + d}^\pi$ of (t, τ) . For a formula φ , we say that $(t, \tau) \models \varphi$, i.e. (t, τ) is an STCTL-model of φ , if and only if $M^{t, \tau}, (\varepsilon, 0) \models \varphi$. Thus a formula φ is STCTL-satisfiable if and only there exists an STCTL-model of φ .

In [LN97] the problem of STCTL-satisfiability is reduced to the emptiness problem of timed tree automata. In particular, given a formula φ it is possible to construct a timed tree automaton accepting STCTL-models of φ if and only if φ is STCTL-satisfiable. The corresponding construction leads to the following results.

Remark 2 Given a formula φ , if φ is STCTL-satisfiable then there exists an STCTL-model (t, τ) of φ such that:

- for each $v \in \text{dom}(t)$, $\text{deg}(v) \leq 2 \max_{s \in S_\varphi} |\{\exists \psi \mid \exists \psi \in s\}| + 1$, and
- there exists a mapping $\eta : \text{dom}(t) \rightarrow S_\varphi \times S_\varphi$ such that $M^{t, \tau}, (v, d) \models \psi$ for each $\psi \in \mu(v, d)$, where $M^{\eta, \tau} = (S, \mu, f)$.

Moreover, there exists a timed ω -tree automaton A_φ with $O(2^{|\varphi|})$ states and timing constraints of total size $O(|\varphi|)$ such that (t, τ) is an STCTL-model of φ satisfying the above properties if and only if $(t, \tau) \in T(A_\varphi)$.

By the above Remarks 1 and 2 the satisfiability problem in STCTL is decidable in exponential time.

4 Decidability of finite satisfiability

In this section we prove the main result of this paper. We show that the finite satisfiability of formulae is decidable. This result is obtained by proving that a formula is finitely satisfiable if and only if it is satisfiable over a particular class of STCTL-structures, the left-closed right-opened STCTL-structures. Then we show that the satisfiability of formulae on these structures is decidable via a reduction to the emptiness problem of timed tree automata. Finally, we prove that the set of formulae which are STCTL-satisfiable strictly contains the set of finitely-satisfiable formulae.

Let L_{Fin} be the language of formulae that are finitely-satisfiable. We start providing a characterization of L_{Fin} based on a subclass of STCTL-structures. Let (t, τ) be an STCTL-structure, (t, τ) is said to be a *left-closed right-opened* STCTL-structure if $t_{sing}(v) = t_{open}(vi)$ for any $v \in \text{dom}(t)$ and $i \in \{1, \dots, k\}$. Before to show that the set of formulae which are finitely-satisfiable is exactly the set of formulae which are satisfiable over left-closed right-opened STCTL-structures, we prove that the existence of a left-closed right-opened STCTL-model of a formula is decidable. The decision procedure we give is obtained, as for the STCTL-satisfiability, via a reduction to the emptiness problem of timed tree automata.

Lemma 1 Given a formula φ , there exists a timed tree automaton A such that (1) $T(A)$ is not empty if and only if there is a left-closed right-opened STCTL-model of φ , and (2) for each $(t, \tau) \in T(A)$ there exists a function $\eta : \text{dom}(t) \rightarrow S_\varphi \times S_\varphi$ such that $M^{t, \tau}, (v, d) \models \psi$ for each $\psi \in \mu(v, d)$, where $M^{\eta, \tau} = (S, \mu, f)$. Moreover, the existence of a left-closed right-opened STCTL-model of φ can be checked in exponential time.

Proof. Let A_φ be a timed tree automaton accepting STCTL-structures of φ if and only if φ is STCTL-satisfiable. Let A' be a timed tree automaton accepting all the left-closed right-opened STCTL-structures (this automaton should just to check that $t_{sing}(v) = t_{open}(vi)$ for any $v \in \text{dom}(t)$ and $i = 1, \dots, \text{deg}(v)$). By Remarks 2 and 1, we have that there is an automaton accepting $T(A') \cap T(A_\varphi)$ with $O(2^{|\varphi|})$ states and timing constraints of total size $O(|\varphi|)$. Clearly, any timed tree in $T(A') \cap T(A_\varphi)$ is a left-closed right-opened STCTL-model of φ . Now, let (t, τ) be a left-closed right-opened STCTL-model of φ . By chopping out all the paths in (t, τ) that are not necessary to the fulfilment of φ , we obtain a left-closed right-opened STCTL-model (t', τ') of φ such that for each node $u \in \text{dom}(t')$, $\text{deg}(u) \leq 2 \max_{s \in S_\varphi} |\{\exists \psi \mid \exists \psi \in s\}| + 1$. Thus by Remark 2 and since any left-closed right-opened

STCTL-model of φ is also an STCTL-model of φ , we have that $(t', \tau') \in T(A_\varphi)$. Moreover, each left-closed right-opened STCTL-structure is in $T(A')$, and thus $T(A') \cap T(A_\varphi) \neq \emptyset$. Hence the automaton accepting $T(A') \cap T(A_\varphi)$ satisfies property (1). We observe also that any $(t, \tau) \in T(A') \cap T(A_\varphi)$ satisfies property (2) since all timed trees accepted by A_φ satisfy this property. Finally, to check that φ has a left-closed right-opened STCTL-model is equivalent to check the emptiness of $T(A') \cap T(A_\varphi)$, and by Remark 1 this can be done in exponential time. ■

The next two lemmata show that the finitely-satisfiable formulae are exactly the formulae which are satisfiable over left-closed right-opened STCTL-structures.

Lemma 2 *Given a formula φ , if φ is finitely satisfiable then φ is satisfiable on a left-closed right-opened STCTL-structure.*

Proof. Let G be a timed graph such that $G \models \varphi$. For each subformula $\psi = \exists \psi'$ of φ such that $G \models \psi$, we denote by ρ_ψ a dense path in G such that ψ is satisfied on the path ρ_ψ . Let Π be the set of all these paths. If Π is empty, then we add an arbitrary dense path of G . Now, consider the dense tree obtained deleting all the paths from G but the paths in Π . Since there are only a finite number of these paths, this tree can be mapped into an STCTL-structure (t, τ) such that: (1) for each dense path ρ in (t, τ) there exists a unique $\rho' \in \Pi$ such that $\rho = \rho'$, and (2) for each $\rho' \in \Pi$ there exists a unique dense path ρ in (t, τ) such that $\rho' = \rho$. It is easy to verify that $(t, \tau) \models \varphi$ and (t, τ) is a left-closed right-opened STCTL-structure. ■

Lemma 3 *Given a formula φ , if φ has a left-closed right-opened STCTL-model then φ is finitely satisfiable.*

Proof. From Lemma 1 we have that there exists a timed tree automaton A_φ accepting left-closed right-opened STCTL-models of φ , if there are any. We can consider a new timed tree automaton A'_φ accepting 2^{AP} -valued ω -trees obtained from the timed trees $(t, \tau) \in T(A_\varphi)$ by ignoring $t_{open}(v)$ for each $v \in dom(t)$, since for left-closed right-opened STCTL-structures it holds that $t_{sing}(v) = t_{open}(vi)$. Clearly, $T(A'_\varphi)$ is not empty, and hence by Theorem 1, there exists a highly-deterministic timed tree automaton contained in A'_φ and, as a consequence, there exists a graph-representable timed tree automaton $A' = (2^{AP}, S', s_0, \Delta', S')$ such that $T(A') \subseteq T(A'_\varphi)$. Let G be a timed graph $(S', \mu, s_0, E, C, \Lambda, \Delta)$ such that $\mu(s) = \sigma$, $\Delta(e) = \delta$ for any $e = (s', s) \in E$, $e_i = (s, s_i) \in E$ and $\Lambda(e_i) = \lambda_i$ for $i = 1, \dots, k$ if and only if $(s, \sigma, s_1, \dots, s_k, \lambda_1, \dots, \lambda_k, \delta) \in \Delta'$. Notice that, due to the properties of A' , G is well defined. Denoted as ν_0 the clock valuation mapping each clock to 0, by the above construction we have that each $\langle s_0, \nu_0 \rangle$ -run ρ of G is a continuous path of a timed tree $(t, \tau) \in T(A')$, and on the other hand, for each $(t, \tau) \in T(A')$ any continuous path ρ' in (t, τ) is also an $\langle s_0, \nu_0 \rangle$ -run of G . Moreover, by Lemma 1 since $T(A') \subseteq T(A'_\varphi)$, for $(t, \tau) \in T(A')$ there is a timed tree (η, τ) such that $\eta : dom(t) \rightarrow S_\varphi \times S_\varphi$ and $M^{t, \tau}(v, d) \models \psi$ for each $\psi \in \mu(v, d)$, where $M^{\eta, \tau} = (S_\eta, \mu, f)$ is the dense tree corresponding to (η, τ) . Notice that η is independent by the choice of $(t, \tau) \in T(A')$, since A' is highly deterministic. Thus, since A' is graph-representable, η defines in an obvious way a labelling function η' of the G vertices such that $G \models \psi$ for each $\psi \in \eta'(s_0)$. Since $(t, \tau) \models \varphi$, it holds that $M^{\eta, \tau}(v, d) \models \varphi$ and thus $\varphi \in \eta'(s_0)$. Hence $G \models \varphi$, and φ is finitely satisfiable. ■

Directly from the last two lemmata we have the following theorem.

Theorem 2 *A TCTL-formula φ is finitely satisfiable if and only if φ has a left-closed right-opened*

STCTL-model.

As a consequence of the results we have just proved, membership in L_{Fin} is decidable in exponential time and can be reduced to the emptiness problem of timed tree automata.

Theorem 3 *The finite satisfiability of TCTL-formulae is decidable in exponential time.*

Proof. By Theorem 2, we have that φ is finitely satisfiable in TCTL if and only if φ has a left-closed right-opened STCTL-model. Thus by Lemma 1, the finite satisfiability of TCTL-formulae is decidable in exponential time. ■

We end this section by proving that the set L_{Fin} is a proper subset of L_{STCTL} , where L_{STCTL} is the language of the STCTL-satisfiable formulae. By Theorem 2 and since left-closed right-opened STCTL-structures are also STCTL-structures, we have that $L_{Fin} \subseteq L_{STCTL}$. The strict containment can be proved by showing that there exists a Formula φ such that φ is STCTL-satisfiable but is not finitely satisfiable.

Example 1 *Consider the formula $\varphi = \forall \square_{\leq c} p \wedge \forall \square_{> c} \neg p$. Let (t, τ) be an STCTL-structure such that (1) for any $i \leq \text{deg}(\varepsilon)$, $t_{\text{sing}}(\varepsilon) = t_{\text{open}}(i) = t_{\text{sing}}(i) = p$ and $\tau(i) = c$, and (2) $t_{\text{sing}}(v) = t_{\text{open}}(v) = \neg p$ for any other $v \in \text{dom}(t)$. Clearly φ is an STCTL-model of φ , and thus we have that $\varphi \in L_{STCTL}$. Moreover $\varphi \notin L_{Fin}$ since truth assignments of a dense path in a timed graph vary on left-closed right-opened intervals.*

Thus we have the following lemma.

Lemma 4 *L_{Fin} is strictly contained in L_{STCTL} .*

5 Conclusions

In this paper we have proved the decidability of the finite satisfiability of the TCTL-formulae that do not contain the equality in the timing constraints. The result is obtained by reducing this problem to the emptiness problem for timed tree automata. The presented construction uses as intermediate step the decidability of formulae on left-closed right-opened STCTL-structures. According to the previously known results there were two possible causes of the undecidability of TCTL-finite satisfiability: the denseness of the underlying structure and the equality in the timing constraints. Our results prove that the only source of undecidability when the structures are defined by timed graphs is the presence of the equality in the timing constraints. We have also compared TCTL to STCTL, via the language of the formulae which are satisfiable in each of them. The interesting result we obtained is that the satisfiability problem in TCTL is decidable on a set of structures more general than those obtained from timed graphs. As a consequence there exists a more general formulation of dense trees with dense branching time that matches the language of formulae which are satisfiable in STCTL. Finally, we prove our results by relating to the theory of timed tree automata, so strengthening the already well-founded connections between the field of logics and the field of finite automata.

Acknowledgements

We would like to thank Rajeev Alur for the helpful discussions and suggestions.

References

- [ACD93] R. Alur, C. Courcoubetis, and D.L. Dill. Model-checking in dense real-time. *Information and Computation*, 104(1):2 – 34, 1993.
- [AD94] R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183 – 235, 1994.
- [AFH96] R. Alur, T. Feder, and T.A. Henzinger. The benefits of relaxing punctuality. *Journal of the ACM*, 43(1):116 – 146, 1996.
- [AH93] R. Alur and T.A. Henzinger. Real-time logics: complexity and expressiveness. *Information and Computation*, 104(1):35 – 77, 1993.
- [CE81] E.M. Clarke and E.A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Proceedings of Workshop on Logic of Programs*, LNCS 131, pages 52 – 71. Springer-Verlag, 1981.
- [DW99] M. Dickhofer and T. Wilke. Timed alternating tree automata: the automata-theoretic solution to the TCTL model checking problem. In *Proceedings of the 26th International Colloquium on Automata, Languages and Programming*, LNCS 1644, pages 281 – 290. Springer-Verlag, 1999.
- [Eme90] 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.
- [EMSS90] E.A. Emerson, A.K. Mok, A.P. Sistla, and J. Srinivasan. Quantitative temporal reasoning. In *Proceedings of the 2nd International Conference on Computer Aided Verification*, LNCS 531, pages 136 – 145. Springer-Verlag, 1990.
- [Hen98] T.A. Henzinger. It’s about time: Real-time logics reviewed. In *Proceedings of the 9th International Conference on Concurrency Theory, CONCUR’98*, LNCS 1466, pages 439 – 454. Springer-Verlag, 1998.
- [HP85] D. Harel and A. Pnueli. On the development of reactive systems. In *Logics and Models of Concurrent Systems*, volume F-13 of *NATO Advanced Summer Institutes*, pages 477 – 498. Springer-Verlag, 1985.
- [JM86] F. Jahanian and A.K. Mok. Safety analysis of timing properties in real-time systems. *IEEE Transactions on Software Engineering*, SE - 12(9):890 – 904, 1986.
- [Koy90] R. Koymans. Specifying real-time properties with metric temporal logic. *Journal of Real-Time Systems*, 2:255 – 299, 1990.
- [LN97] S. LaTorre and M. Napoli. Timed tree automata with an application to temporal logic. Technical report, Dipartimento di Informatica ed Applicazioni, Università degli Studi di Salerno, Italy, 1997. URL: “<http://www.cis.upenn.edu/~latorre/Papers/stctl.ps.gz>”.
- [PH88] A. Pnueli and E. Harel. Applications of temporal logic to the specification of real-time systems. In *Formal Techniques in Real-time and Fault-tolerant Systems*, LNCS 331, pages 84 – 98. Springer-Verlag, 1988.
- [Pnu77] A. Pnueli. The temporal logic of programs. In *Proceedings of the 18th IEEE Symposium on Foundations of Computer Science*, pages 46 – 77, 1977.
- [Rab72] M.O. Rabin. Automata on infinite objects and Church’s problem. *Trans. Amer. Math. Soc.*, 1972.
- [Tho90] 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.