Dense Real-time Games*

Marco Faella
Università degli Studi di Salerno
mfaella@dia.unisa.it

Salvatore La Torre
University of Pennsylvania &
Università degli Studi di Salerno
latorre@seas.upenn.edu

Aniello Murano
Rice University &
Università degli Studi di Salerno
murano@dia.unisa.it

Abstract

The rapid development of complex and safety-critical systems requires the use of reliable verification methods and tools for system design (synthesis). Many systems of interest are reactive, in the sense that their behavior depends on the interaction with the environment. A natural framework to model them is a two-player game: the system versus the environment. In this context, the central problem is to determine the existence of a winning strategy according to a given winning condition. We focus on real-time systems, and choose to model the related game as a nondeterministic timed automaton. We express winning conditions by formulas of the branching-time temporal logic TCTL. *While timed games have been studied in the literature, timed games with dense-time winning conditions constitute a new research topic. The main result of this paper is an exponential-time algorithm to check for the existence of a winning strategy for* TCTL *games where equality is not allowed in the timing constraints. Our approach consists on translating to timed tree automata both the game graph and the winning condition, thus reducing the considered decision problem to the emptiness problem for this class of automata. The proposed algorithm matches the known lower bound on timed games. Moreover, if we relax the limitation we have placed on the timing constraints, the problem becomes undecidable.*

*This work was partially supported by the MURST in the framework of the project "Metodi Formali per la Sicurezza" (MEFISTO). The second author was also supported in part by the NSF award CCR99-70925, the NSF ITR award, by the SRC award 99-TJ-688, and by the DARPA ITO Mobies award F33615-00-C-1707. The third author was also supported in part by the NSF award CCR99-88322.

1. Introduction

The theory of games, traditionally related to the economic-theoretic environment (see for instance [19]), has been attracting the interest of many researchers in computer science. The notion of a game naturally arises in the verification of reactive systems and program synthesis [1, 20]. In the compositional approach, a reactive system is seen as a set of interacting components, each of them is modeled as an *open system* (that is, a system whose behavior depends on the current state as well as the behavior of the environment in which it is embedded). The interaction between a component and the rest of the system, or the interaction between a controller and the related plant, both can be modeled as a two-player game. For verifying *closed systems*, where a system behavior is completely determined by the current state, model-checking is a very successful technique [7, 8, 9]. The equivalent of model-checking for open systems is checking for the existence of a winning strategy in a two-player game [15].

In this paper we focus on the verification of open real-time systems, modeled as nondeterministic *timed automata* [3]: finite automata augmented with a finite set of real-valued *clocks*. The transitions of a timed automaton are enabled according to the current state, that is, the current *location* and the current clock values. In a transition, clocks can be instantaneously reset. The value of a clock is exactly the time elapsed since the last time it was reset. A clock constraint (*guard*) is associated to each transition with the meaning that a transition can be taken only if the associated guard is enabled. When interpreting a nondeterministic timed automaton as a game graph, we capture the choices of the *protagonist* by the symbols associated with the transitions and nondeterminism is used to model the possible choices of the *antagonist*[1]. To model the case that the protagonist stays *idle* and the antagonist is moving, we use a special symbol denoted by ε. The case that both players stay idle is captured by letting time elapse in a location. A *play* of a timed game is thus constructed in the following way. At each time, a player declares the time it will wait idling, along with its next choice. At the time one of the players or both move, both players are allowed to redeclare their next move and the time they will issue it. Technically, a play is a run of the automaton modeling the game. The winning condition for the protagonist is expressed as a predicate over system behaviors (runs).

Discrete-time games with branching-time specifications are considered in [15]. In this paper we solve TCTL games, that is, timed games with a winning condition given by a formula of the branching-time temporal logic TCTL. The decision problem that we wish to solve is thus establishing whether the protagonist has a strategy to ensure that any resulting computation will satisfy the given TCTL specification. The logic TCTL was introduced to verify real-time systems using model-checking [2]. The syntax of TCTL is given by augmenting the temporal operators of CTL [7] (except for the "next", which is discarded since it does not have a clear 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 natural number. As in CTL, TCTL temporal modalities are universally or existentially quantified.

A typical property that can be asserted in TCTL is the time-bounded response property. Suppose there are several resources $q_1, \ldots, q_n$ in our system, and several users that can request them. Let $p_1, \ldots, p_n$ be the atomic propositions denoting the requests for each resource. The system, along with all the constraints related to the uses and the availability of the resources, can be modeled as a timed automaton. We might be interested in synthesizing a scheduler which assigns the system resources fulfilling the requirement that each request is always granted within a certain delay. Such requirement can be expressed in TCTL by the formula $\varphi = \forall\Box \wedge_{i=1}^{n} (p_i \rightarrow \forall\Diamond_{\leq c_i} q_i)$. Since the sequence of requests received by the scheduler is determined by the users, and thus is not controllable by the scheduler, a suitable model for this scenario is a TCTL game of the scheduler versus the users. Our decision problem is thus to establish whether there exists a scheduler that can ensure the property φ in the open system modeled by the given timed automaton. Clearly, time-bounded response properties cannot be captured by untimed specifications.

Model-checking of TCTL-formulas is PSPACE-complete while checking for their satisfiability is undecidable [2]. In a recent paper, the satisfiability of TCTL is proved to be EXPTIME-complete if equality is not allowed in timing constraints of the formulas [16]. Here we obtain the same complexity/decidability result for games. We prove that TCTL games are EXPTIME-complete, if equality is not allowed in timing constraints, and undecidable, otherwise. Our approach to solving TCTL games relies on a reduction to the emptiness problem of timed Büchi tree automata. Given a timed automaton Γ and a TCTL–formula φ, we solve the game (Γ, φ) by constructing two timed Büchi tree automata: A_Γ and A_φ. The automaton A_Γ accepts, for each strategy $\mathcal{F}$ of the protagonist, a timed ω-tree which is "embedded" into $\mathcal{F}$. The automaton A_φ accepts instead all the models of φ with arity bounded above by the arity of Γ. By the known results on timed Büchi tree automata [17], the intersection of A_Γ and A_φ is still a timed Büchi tree automaton, and accepts all the trees that satisfy φ and are embedded into a strategy. Thus, a timed game Γ admits a winning strategy with respect to a formula φ if and only if the language accepted by the intersection automaton is not empty. Denoting the product of the sizes of φ and Γ by K, we have that the number of states of the intersection automaton is exponential in K while the size of its clock constraints is linear in K. Since checking the emptiness for a timed Büchi tree automaton is polynomial in the number of states and exponential in clock size [17], we obtain an algorithm that takes exponential time. The result we obtain is tight, in the sense that TCTL games are EXPTIME-complete (hardness is a consequence of the EXPTIME-hardness of the reachability timed games [14]). Moreover, we show that allowing equality in the timing constraints makes the problem undecidable. To prove this we reduce the satisfiability problem of TCTL–formulas, which is known to be undecidable [2].

Timed games have been considered in several papers. Reachability problems in timed games were solved in [6, 5], and the EXPTIME-hardness of reachability timed games was proved in [14]. Rectangular hybrid games with winning conditions expressed by LTL formulas were studied in [13]. An automata-theoretic approach to solve LTL and CTL timed games is discussed in [11]. In all these papers, while the game graph is a dense-time model, the winning condition either does not take into account time or considers time only from a qualitative point of view (i.e., no bounds on the time of occurrence of the events can be asserted and thus properties as the time-bounded response property described above cannot be expressed).

Here we allow dense real-time in both the game graph and the winning condition. Recently, problems closely related to solving timed games with real-time specifications, have been formulated in [10, 18]. In [18], the real-time *module-checking* of TCTL–formulas is introduced and it is shown that this problem is undecidable when equality is allowed in the timing constraints. The proof is obtained by

[1]Games are usually defined in a symmetric way with respect to each of the players. The results discussed in this paper also hold in this formulation.

reducing the problem of deciding whether a nondeterministic two-counter machine has a recurring computation (this problem is known to be Σ_1^1-*hard* [12]). Here, our undecidability result for TCTL games is inherited from the undecidability of the satisfiability problem for TCTL with equality. In [10], the controller synthesis of timed automata with linear-time specifications given also as timed automata is addressed, while we study here branching-time specifications.

The rest of the paper is organized as follows. In Section 2 we recall the main definitions and some results from the theory of timed automata and the logic TCTL. In Section 3 we introduce the TCTL games, and discuss a discrete-branching representation of strategies when equality is not allowed in the timing constraints. The main result of this paper is presented in Section 4, where we prove that the problem of the existence of a winning strategy in a TCTL game without equality in the timing constraints of the winning condition is EXPTIME-complete. In Section 5 we prove that if we relax the restriction we placed on the timing constraints, checking for the existence of a winning strategy in TCTL games becomes undecidable. Finally, we give our conclusions in Section 6.

2. Preliminaries

In this section, we recall some definitions and results concerning timed automata. Since we deal with trees and branching-time logic, we give the definition of timed automata on timed ω–trees [17] that includes as a particular case (trees with arity 1) the original model given on ω-sequences [3]. We also recall the real-time temporal logic TCTL [2] and some results concerning the finite satisfiability of TCTL–formulas.

2.1 Timed Automata

Let Σ be an alphabet and $dom(t)$ be a subset of $\{1,\ldots,k\}^*$, for an integer $k > 0$, such that (i) the empty word belongs to $dom(t)$, and (ii) if $v \in dom(t)$, then for some $j \in \{1,\ldots,k\}$, $vi \in dom(t)$ 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 $t : dom(t) \longrightarrow \Sigma$. In each Σ-valued ω-tree t, the root corresponds to the empty word and we will denote it by $root$. For each node u of t, we denote with $pre(u)$ the set of prefixes of u (including u itself) and with $deg(u)$ the arity of u. A *path* to a node u is clearly the set $pre(u)$, and a *full path* π in t is a maximal subset of $dom(t)$ linearly ordered by the prefix relation. In the following, when no confusion can arise we will use the name path to refer also to a full path, and we denote a path also as the sequence of its nodes in the order given by the prefix relation. Let $\mathbb{R}_+$ be the set of the nonnegative real numbers, we define a *timed* Σ-*valued* ω-*tree* as a pair (t, τ) where t is a Σ-valued ω-tree and τ, called *time tree*, is a mapping from $dom(t)$ into $\mathbb{R}_+$ such that *(i)* $\tau(u) > 0$ for each $u \in dom(t) - \{\varepsilon\}$ (*positiveness*), and *(ii)* for each path π and for each $x \in \mathbb{R}_+$ there exists $u \in \pi$ such that $\sum_{v \in pre(u)} \tau(v) \geq x$ (*progress*). According to this definition, at each node we associate a symbol and a real-time value. For the root the real number is the absolute time of occurrence. The root, differently from the other nodes, can be labeled with 0 since it can correspond the event starting the computations represented by the tree. For all the other nodes ui, where $1 \leq i \leq deg(u)$, the associated number is the time which has elapsed since the node u was read. Positiveness implies that a positive delay occurs between any two consecutive nodes along a path. Progress guarantees that infinitely many events (i.e. nodes appearing at input) cannot occur in a finite slice of time (*nonzenoness*). In the rest of the paper, we will use the term *tree* to refer to a timed Σ-valued ω-tree for some alphabet Σ, and a tree language is a set of trees.

Now, we introduce the notion of timed Büchi tree automaton. In timed automata, we use a central (real-valued) clock to scan time, and a finite set of *clock variables* (also simply named *clocks*) along with timing constraints to check the satisfaction of time requirements. Each clock can be seen as a chronograph synchronized with the central clock, and it can be read or set to zero (*reset*): after a reset, a clock restarts automatically. Clocks are used for testing *timing constraints* on which state transitions depend. Timing constraints are expressed by *clock constraints*. Let C be a set of clocks, the set of clock constraints $\Phi(C)$ 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 $x, y \in C$. A *clock valuation* is a mapping $\nu : C \longrightarrow \mathbb{R}_+$. If ν is a clock valuation, λ is a set of clocks and d is a real number, we denote with $[\lambda \leftarrow 0](\nu + d)$ the clock valuation that gives 0 for each clock $x \in \lambda$ (i.e., *resets* each clock $x \in \lambda$) and $\nu(x) + d$ for each clock $x \notin \lambda$. With $\bar{0}$ we will denote the clock assignment $\bar{0}(x) = 0$, for all $x \in C$.

A *timed Büchi tree automaton* (TBTA) is a tuple $A = (S, \Sigma, s_0, C, \Delta, F)$, 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 $\bigcup_{k \geq 0}(S \times \Sigma \times S^k \times (2^C)^k \times \Phi(C))$, and $F \subseteq S$ is a set of final locations.

A state of the system is completely determined by a location and a clock valuation, thus it is denoted by a pair (s, ν). The state $(s_0, \bar{0})$ is called the *initial state*. A transition rule $e = (s, \sigma, s_1, \ldots, s_k, \lambda_1, \ldots, \lambda_k, \delta) \in \Delta$ can be described as follows. Let us suppose that when the system entered in the location s the clock values were given by ν and after a time τ a symbol σ is read. Then the system can take the transition e 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 locations $s_1, \ldots, s_k$ with clock values given respectively by $\nu_1 = [\lambda_1 \leftarrow 0](\nu + \tau), \ldots, \nu_k = [\lambda_k \leftarrow 0](\nu + \tau)$. 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. Let $A = (S, \Sigma, S_0, C, \Delta, F)$ be a TBTA, a *run* of A on a timed tree (t, τ) is a pair (r, θ), where *(i)* $r : dom(t) \longrightarrow S$ and $\theta : dom(t) \longrightarrow \mathbb{R}_+^{|C|}$; *(ii)* $r(root) = s_0$ and $\theta(root) = \nu_0$; *(iii)* for $u \in dom(t)$, $k = deg(u)$: $(r(u), t(u), r(u1), \ldots, r(uk), \lambda_1, \ldots, \lambda_k, \delta) \in \Delta$, $\theta(u) + \tau(u)$ fulfills δ and $\theta(ui) = [\lambda_i \rightarrow 0](\theta(u) + \tau(u))$, for all $i \in \{1, \ldots, k\}$. A timed tree (t, τ) is accepted by a TBTA A if and only if there exists a run (r, θ) of A on (t, τ) such that for every path π in r, there are infinitely many nodes $u \in \pi$, such that $r(u) \in F$. The language accepted by a TBTA A, denoted by $L(A)$, is the set of all trees accepted by A. The following remark summarizes some important results on TBTAs.

Remark 1 *[17]*

- *The Emptiness problem for TBTAs is decidable in time exponential in the number of clocks and the size of constants in the timing constraints, and polynomial in the number of locations;*
- *TBTAs are closed under union and intersection.*

A word is a tree whose nodes have all arity 1, and a timed Büchi automaton on words is simply a timed Büchi tree automaton where the transition rules are a finite subset of $(S \times \Sigma \times S \times 2^C \times \Phi(C))$. We will use for the words and the related timed automaton, a notation simpler than that introduced for the trees, and we will refer also to a prefix of a run. In particular, a word is denoted as a $(\omega-)$sequence (σ, τ) of pairs of symbols and nonnegative reals. A run r to (s_k, v_k) of a timed automaton A on (σ, τ), where $\sigma = \sigma_1 \ldots \sigma_k$ and $\tau = \tau_1 \ldots \tau_k$, is a finite sequence $(s_0, \nu_0) \xrightarrow{\tau_1, \sigma_1} (s_1, \nu_1) \xrightarrow{\tau_2, \sigma_2} \ldots \xrightarrow{\tau_k, \sigma_k} (s_k, \nu_k)$ such that $\nu_0 = \bar{0}$, for $i = 1, \ldots, k$, $(s_{i-1}, \sigma_i, s_i, \lambda_i, \delta_i)$ is a transition, $\nu_{i-1} + \tau_i$ satisfies δ_i, and $\nu_i = [\lambda_i \leftarrow 0](\nu_{i-1} + \tau_i)$. Also, for all $i \leqslant k$ we denote by r_i the run up to (s_i, ν_i) defined as $(s_0, \nu_0) \xrightarrow{\tau_1, \sigma_1} (s_1, \nu_1) \xrightarrow{\tau_2, \sigma_2} \ldots \xrightarrow{\tau_i, \sigma_i} (s_i, \nu_i)$, that is the prefix of r up to the i-th step. Finally, with Run(A) we denote the set of all finite runs of A.

In the following, we will use the term *game graph* to refer to a timed automaton on words with no set of final locations. Game graphs will be used to define timed games by associating with them a condition on runs.

2.2 Timed computation tree logic

In this section we recall the branching-time temporal logic TCTL that was introduced in [2].

The syntax of TCTL is given by augmenting the temporal operators of CTL (except for the "next", which is discarded since it does not have a clear meaning in a dense time domain) with a timing constraint of the type $\approx c$, where $\approx$ is one among $<$, $\leq$, $>$, and $\geq$, and c is a natural number[2]. As in CTL, TCTL temporal modalities are either universally or existentially quantified. Formally, let AP be a set of atomic propositions, the syntax of TCTL–formulas is given by the following grammar:

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

where $p \in AP$, $\approx$ belongs to $\{<, \leq, >, \geq\}$, and c is a natural number.

Before giving the semantics of TCTL, we introduce some common notation. In the rest of the paper with AP we denote the set of atomic propositions of the considered TCTL–formulas. If it is not differently stated, with $\approx$ we refer to a relational operator in $\{<, \leq, >, \geq\}$, and with c to a natural number. We define a *dense path* through a set of nodes Q as a function $\rho : \mathbb{R}_+ \longrightarrow Q$. 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 (Q, μ, f) where:

- Q is a set of nodes;
- $\mu : Q \longrightarrow \Sigma$ is a labeling function;
- f is a function assigning to each $q \in Q$ a set of dense paths through Q, starting at q, and satisfying the *tree constraints*: (i) $\forall \rho \in f(q)$ and $\forall t \in \mathbb{R}_+$, $\rho_{[0,t)} \cdot f(\rho(t)) \subseteq f(q)$, and (ii) $\forall \rho \in f(q)$ and $\forall t \in \mathbb{R}_+, \rho_{[t,\infty)} \in f(\rho(t))$.

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

- for $p \in AP$, $M, q \models p$ if and only if $p \in \mu(q)$;
- $M, q \models \neg p$ if and only if $not(M, q \models p)$;
- $M, q \models \varphi_1 \wedge \varphi_2$ if and only if $M, q \models \varphi_1$ and $M, q \models \varphi_2$;
- $M, q \models \varphi_1 \vee \varphi_2$ if and only if $M, q \models \varphi_1$ or $M, q \models \varphi_2$;
- $M, q \models \forall[\varphi_1 U_{\approx c} \varphi_2]$ if and only if $\forall \rho \in f(q)$, $\exists\, d \approx c$ such that $M, \rho(d) \models \varphi_2$ and $M, \rho(d') \models \varphi_1$ for all d' such that $0 \leq d' < d$;

[2] Notice that in the original definition of TCTL also equality is allowed in the timing constraints. Here to obtain the decidability of TCTL games we do not allow it.

- $M, q \models \exists[\varphi_1 U_{\approx c} \varphi_2]$ if and only if $\exists \rho \in f(q)$, $\exists\, d \approx c$ such that $M, \rho(d) \models \varphi_2$ and $M, \rho(d') \models \varphi_1$ for all d' such that $0 \leq d' < d$.

We say that M is a model of φ if and only if $M, q \models \varphi$ for some $q \in Q$. Some abbreviations which are commonly used are $\diamond_{\approx c}\varphi$ for $\mathrm{TRUE}\,\mathcal{U}_{\approx c}\varphi$ and $\square_{\approx c}\varphi$ for $\neg\diamond_{\approx c}\neg\varphi$, where TRUE is the constant that holds true on all nodes (i.e., it is equivalent to $\varphi \vee \neg\varphi$ for any formula φ).

Satisfiability in TCTL is studied with respect to the class of dense trees defined by the runs of a *timed graph*, i.e., a timed automaton with locations labeled by subsets of atomic propositions (*finite satisfiability*) [2]. Therefore, we say that a TCTL–formula φ is *satisfiable* if and only if there exists a model of φ which is obtained from a timed graph. We do not recall in more detail the finite satisfiability here, but we discuss an equivalent concept of satisfiability for TCTL–formulas, the *discrete-branching satisfiability* [16].

Given a timed tree (t, τ) and a node u, let $\gamma_u = \sum_{v \in pre(u)} \tau(v)$, i.e., γ_u is the time of occurrence of u. For a path $\pi = v_0, v_1, \ldots$ in (t, τ), and a real number $d \geq 0$, the corresponding dense path is defined as $\rho^\pi_d : \mathbb{R}_+ \to Q$ such that $\rho^\pi_d(d') = t(v_i)$, if $d + d' = \gamma_{v_i}$, and $\rho^\pi_d(d') = t(v_i)$, if $\gamma_{v_i} < d + d' < \gamma_{v_{i+1}}$. The ω–tree (t, τ) defines a dense tree $M^{t,\tau} = (Q, \mu, f)$, where *(i)* $Q = \{(vi, d) \,|\, v \in dom(t), 1 \leq i \leq deg(v), \text{ and } 0 < d \leq \tau(vi)\} \cup \{(root, 0)\}$, *(ii)* $\mu(root, 0) = t(root)$, $\mu(vi, d) = t(v)$ if $d < \tau(vi)$, and $\mu(vi, d) = t(vi)$ otherwise, and *(iii)* $f(root, 0)$ is the set of all dense paths ρ^π_0 of (t, τ) and $f(vi, d)$ is the set of all the dense paths $\rho^\pi_{\gamma_v + d}$ of (t, τ), for all $\pi \in dom(t)$, $v \in \pi$ and $d \in \mathbb{R}_+$. We say that a TCTL–formula φ is satisfiable with a discrete branching if there exists a timed tree (t, τ) such that $M^{t,\tau} \models \varphi$. We also consistently write $(t, \tau) \models \varphi$, meaning that $M^{t,\tau} \models \varphi$.

In [17] the problem of the discrete-branching satisfiability of TCTL–formulas is proved to be EXPTIME-complete, and in [16], it is proved to be equivalent to the TCTL-satisfiability, i.e., given a TCTL–formula φ, φ is discrete-branching satisfiable if and only if φ is TCTL-satisfiable. The proofs given in [17] and [16], can be slightly modified in order to prove the following lemma.

Lemma 1 *Given a* TCTL*–formula φ and a positive integer k, there exists a timed Büchi tree automaton A_φ with $O(2^{|\varphi|})$ states and timing constraints of total size $O(|\varphi|)$ such that $(t, \tau) \in L(A_\varphi)$ if and only if the following three conditions hold:*

1. *$(t, \tau) \models \varphi$;*
2. *(t, τ) has a branching degree bounded above by k;*
3. *given a node $vi \in dom(t) - \{root\}$, for any subformula ψ of φ either $M^{t,\tau}, (vi, d) \models \psi$ holds for all d such that $0 < d < \tau(vi)$, or $M^{t,\tau}, (vi, d) \models \psi$ does not hold for all d such that $0 < d < \tau(vi)$.*

In the following, we say that a 2^{AP}-valued timed tree (t, τ) is TCTL-*consistent* if it satisfies the above property 3.

3. Timed Games

Infinite two-player games offer algorithmic tools for studying reactive systems, that is systems whose behavior is determined not only by their internal state, but also by the interaction with the environment. In this section we define TCTL games and discuss a construction that will be used in the next section to obtain an exponential-time algorithm to decide these games.

We give an asymmetric definition of timed two-player games. We call the two players respectively the protagonist and the antagonist. A timed game is modeled as a game graph and a winning condition. The transitions of the game graph represent joint moves, where the protagonist chooses a symbol (action) and the antagonist chooses a transition among those which are enabled and which correspond to the selected symbol. The antagonist can also move on his own, by taking a transition labeled with the special symbol ε. After a transition, the protagonist should declare his next move, that is the time he will wait and the symbol he will choose. The antagonist can take either an ε-move, before the next move of the protagonist occurs, or a joint move at the time declared by the protagonist. A *play* of the game is a sequence of moves taken by the two players and is formally represented by a run of the game graph. Clearly, in the time interval between two consecutive moves of a play both players stay idle.

Given a game graph Γ, a *strategy* is a function $\mathcal{F}$: *Plays*$(\mathcal{F}) \longrightarrow (\mathbb{R}_+ - \{0\}) \times \Sigma$, where *Plays*$(\mathcal{F}) \subseteq$ Run(Γ), $(s_0, \bar{0}) \in$ *Plays*$(\mathcal{F})$ and for any $r = (s_0, \nu_0) \xrightarrow{\tau_1, \sigma_1} (s_1, \nu_1) \xrightarrow{\tau_2, \sigma_2} \ldots \xrightarrow{\tau_k, \sigma_k} (s_k, \nu_k)$ belonging to *Plays*$(\mathcal{F})$, it holds that for $i = 0, \ldots, k-1$, either $\mathcal{F}(r_i) = (\tau_{i+1}, \sigma_{i+1})$ or $\mathcal{F}(r_i) = (d, \sigma)$, $\tau_{i+1} < d$ and $\sigma_{i+1} = \varepsilon$. In other words, a strategy gives the moves of the protagonist on each play which is "consistent" with the strategy itself and the case $\sigma_{i+1} = \varepsilon$ corresponds to a move of the antagonist taken before the next declared move of the protagonist. The set of infinite runs r such that each prefix of r is a play consistent with $\mathcal{F}$, is called the set of maximal plays of $\mathcal{F}$. Each strategy $\mathcal{F}$ has an associated *dense tree* $M_\mathcal{F} = (Q, \mu, f)$, which is defined as follows:

- $Q = S \times \mathbb{R}_+^{|C|}$, that is the state space of Γ;
- $\mu(s, \nu) = AP(s)$;
- $f(s, \nu)$ is the set of all the suffixes of maximal plays of $\mathcal{F}$ starting at (s, ν).

A path in $M_\mathcal{F}$ thus can be mapped to a maximal play of $\mathcal{F}$.

We can now give a formal definition of a timed game. A *timed game* is a pair (Γ, φ) where $\Gamma = (S, \Sigma^\varepsilon, s_0, C, \Delta)$ is

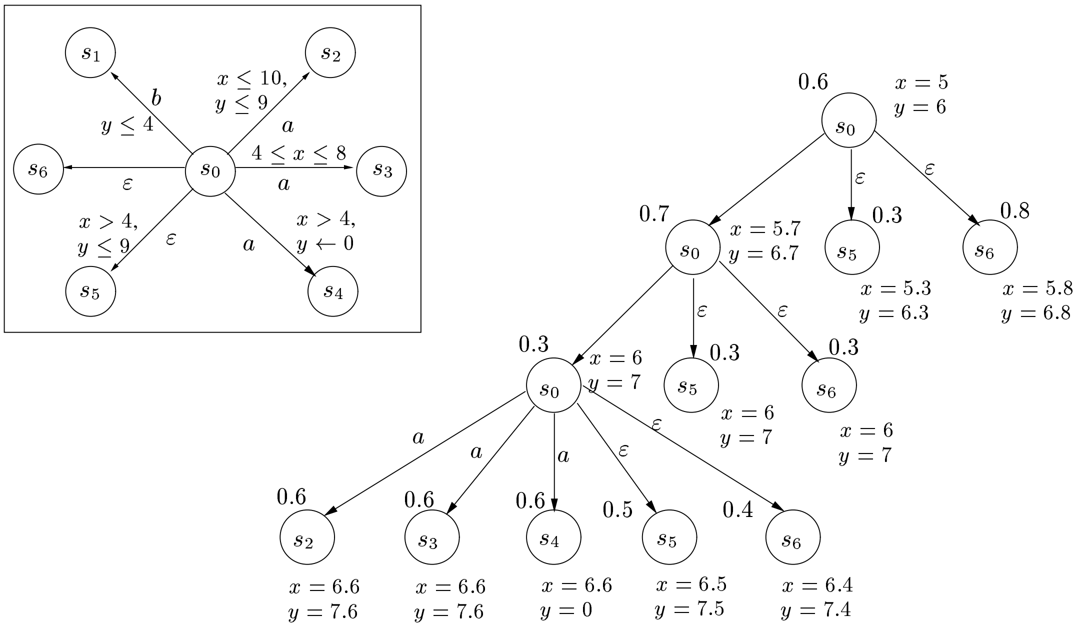

Figure 1. Fragments of a timed game and of a sampling of a strategy.

a game graph, Σ^ε is an alphabet containing also a special symbol ε and φ is a winning condition. A winning condition asserts a property of the plays and depending on the winning condition several kinds of timed games can be defined. In this paper we consider TCTL games, that is timed games whose winning condition is a TCTL–formula. Given a TCTL game (Γ, φ), a strategy $\mathcal{F}$ is *winning* if $M_\mathcal{F} \models \varphi$. We are interested to solving the following decision problem: "given a TCTL game (Γ, φ), is there a winning strategy satisfying the condition expressed by the formula φ?" In the next section we will give a decision algorithm to solve TCTL games.

We recall the well-known equivalence relation over clock valuations which leads to a partition of $\mathbb{R}_+^{|C|}$ into a finite set of *clock regions*, denoted *Reg*(Γ). For a game graph Γ we denote by c_x the largest constant in the clock constraints involving the clock variable x. A clock region is described by: *(i)* a constraint of type $c - 1 < x < c$, $x > c_x$, or $x = c$ for each clock variable x and a natural number $c \leq c_x$, and *(ii)* the ordering of the fractional parts of the clock variables x such that $x < c_x$. In the following, for a clock region α we denote by $\delta_\alpha \in \Phi(C)$ the conjunction of all the constraints of part *(i)* of its description. A clock region is a convex set of clock valuations, and given a clock valuation ν, with $[\nu]$ we denote the clock region containing ν. A clock region α is *open* if for all clocks x and for all naturals $c \leq c_x$, $x = c$ does not hold in α. Otherwise, α is said to be a *boundary* region. A *region* is a pair (s, α) where s is a location and α is a clock region. Regions have the property that no TCTL–formula can distinguish among the valuations of a same region [2]. We can define a partial order on *Reg*(Γ) in the following way: for all $\alpha, \beta \in$ *Reg*(Γ), $\alpha \leqslant \beta$ if and only if for all $\nu \in \alpha$ there exists a $d \in \mathbb{R}_+$ such that $\nu + d \in \beta$. The top element of this ordering is the clock region where each clock x assumes a value greater than c_x. The immediate time-successor of a clock region α is a clock region $\beta \neq \alpha$ such that $\alpha \leqslant \beta$ and for any other clock region γ such that $\alpha \leqslant \gamma$, also $\beta \leqslant \gamma$ holds.

Let $M = (Q, \mu, f)$ be a dense tree, we say that the ω-tree (t, τ) is contained into M if $M^{t,\tau} = (Q', \mu', f')$ is such that $Q' \subseteq Q$, and for any $q \in Q'$ we have $\mu'(q) = \mu(q)$ and $f'(q) \subseteq f(q)$. We now define a way to associate ω–trees with strategies. Each of these ω–trees is contained into the dense tree $M_\mathcal{F}$ and is TCTL-consistent (see section 2.2). Also, these trees have a limited branching degree, which is known beforehand.

Before giving the formal definition, we discuss an example. Consider the fragment of a game graph represented by the fragment of timed automaton shown in the frame of Figure 1. The automaton uses clocks x and y, thus a state is represented by a location and the values of these clocks. Suppose that the game is in location s_0 with clock values $x = 5, y = 6$ and that the current move of a given strategy $\mathcal{F}$ is $(1.6, a)$. The corresponding fragment of a TCTL-consistent ω-tree, which is contained into $M_\mathcal{F}$, is shown also in Figure 1. The three nodes labeled with s_0 represent, starting from the root, respectively the regions in location s_0 with clock regions $[x = 5, y = 6]$, $[5 < x < 6, 6 < y < 7, x + 1 = y]$, and $[x = 6, y = 7]$.

Letting time elapse in the region corresponding to the root, there are three possible things that can happen: the antagonist stays idle (corresponding to the first child of the root) or chooses to take one of the two ε-moves enabled in $[5 < x < 6, 6 < y < 7, x+1 = y]$ (represented with the two children of the root labeled respectively with s_5 and s_6). The time delays labeling these three nodes correspond to a valuation in the region $(s_0, [5 < x < 6, 6 < y < 7, x+1 = y])$, for the first node, and to a possible choice of the delay at which the corresponding ε-move will occur, in the other two cases. We repeat the same construction for the first child of the root. Notice that in this case since the immediate time-successor is a boundary region, the delay associated with its children is the same. Finally, from the last node labeled with s_0, the protagonist takes an a-move (according to the strategy given above). Since there are three a-moves enabled in this region and two ε-moves, the antagonist can decide to pick a joint move with the protagonist (one of the three a-moves) or anticipate its move taking one of the two ε-moves within time 0.6. Notice that the leaves labeled with s_2, s_3, and s_4 are available exactly 1.6 time units from the root. Moreover, each node of the tree is labeled with an actual state of the game graph, and each path can be mapped to a corresponding play of the game. The formal definition follows.

Given a timed game (Γ, φ), where $\Gamma = (S, \Sigma^\varepsilon, s_0, C, \Delta)$, and a strategy $\mathcal{F}$, in the following we inductively define the set of 2^{AP}-labeled ω-trees $\mathit{Sample}(M_\mathcal{F})$. For this purpose, we use a labeling function g that associates, with each node of a tree $(t, \tau) \in \mathit{Sample}(M_\mathcal{F})$, the corresponding state of the game graph Γ. For the root we have $t(\mathit{root}) = AP(s_0)$, $\tau(\mathit{root}) = 0$ and $g(\mathit{root}) = (s_0, \bar{0})$. Assume now that we have completed the construction of the portion of (t, τ) corresponding to a play r of $\mathcal{F}$ from $(s_0, \bar{0})$ to (s, ν_1) and that node v_1 is such that $g(v_1) = (s, \nu_1)$. Suppose also that $\mathcal{F}(r) = (d, \sigma)$ (i.e., the next move of the protagonist will be σ and it will be taken after a delay d). Denote as $\nu_2, \ldots, \nu_{k+1}$ some clock valuations such that for $i = 1, \ldots, k$, clock region $[\nu_{i+1}]$ is the immediate time-successor of $[\nu_i]$, $\nu_{i+1} = \nu_i + \tau_i$, $\tau_i > 0$, and $\tau_1 + \ldots + \tau_k = d$ (here we give the detailed description only in the case $k \neq 0$, that is $\nu_1 + d \notin [\nu_1]$; the construction for the other case can be easily obtained using similar ideas). For $i = 2, \ldots, k+1$, let $V_i^\varepsilon = \{ (s, \varepsilon, s^\varepsilon_{i,j}, \lambda^\varepsilon_{i,j}, \delta^\varepsilon_{i,j}) \in \Delta \mid j = 2, \ldots, m_i \}$ be the set of ε-labeled moves enabled at (s, ν_i). Assuming that $[\nu_1]$ is a boundary region[3], for $i = 1, \ldots, k-1$, we add the nodes $v_i 1, \ldots, v_i m_{i+1}$ to (t, τ) such that:

- $v_{i+1} = v_i 1$, $t(v_{i+1}) = t(v_1)$, $\tau(v_{i+1}) = \tau_i$, and $g(v_{i+1}) = (s, \nu_{i+1})$;
- for $j = 2, \ldots, m_{i+1}$, $t(v_i j) = AP(s^\varepsilon_{i+1,j})$, $\tau(v_i j)$ is such that $\nu_i + \tau(v_i j) \in [\nu_{i+1}]$, and $g(v_i j) = (s^\varepsilon_{i+1,j}, [\lambda^\varepsilon_{i+1,j} \leftarrow 0](\nu_i + \tau(v_i j)))$.

It is worth noticing that the path from root to $v_i j$ corresponds to a play of $\mathcal{F}$ ending at $g(v_i j)$. Moreover, the nodes $v_i 2, \ldots, v_i m_i$ correspond to states reached by taking the related ε-move respectively after delays $\tau(v_i 2), \ldots, \tau(v_i m_i)$ starting from (s_i, ν_i). We will refer to such nodes as nodes corresponding to an ε-move.

To complete the construction, we need to define the children of v_k. We distinguish two cases, depending on whether $[\nu_{k+1}]$ is a boundary region or not. Denote as $V^\sigma = \{ (s, \sigma, s^\sigma_j, \lambda^\sigma_j, \delta^\sigma_j) \in \Delta \mid j = 1, \ldots, n \}$ the set of σ-labeled moves that are enabled at (s, ν_{k+1}). If $[\nu_{k+1}]$ is a boundary region, then we add nodes $v_k j$ for $j = 1, \ldots, n$, such that $t(v_k j) = AP(s^\sigma_j)$, $\tau(v_k j) = \tau_k$, $g(v_k j) = (s^\sigma_j, [\lambda^\sigma_j \leftarrow 0]\nu_{k+1})$. If $[\nu_{k+1}]$ is an open region, we add, to the construction given in the other case, the nodes $v_k j$ for $j = (n+1), \ldots, (n + m_{k+1} - 1)$, such that $t(v_k j) = AP(s^\varepsilon_{k+1,j-n})$, $0 < \tau(v_k j) < \tau_k$, and $g(v_k j) = (s^\varepsilon_{k+1,j-n}, [\lambda^\varepsilon_{k+1,j-n} \leftarrow 0](\nu_k + \tau(v_k j)))$. In Figure 1, we have determined the children of the third node labeled with s_0 using the rule described in this second case. In the following, we will refer to such nodes $v_k 1, \ldots, v_k n$ as nodes corresponding to a σ-move.

We observe that, according to the above definition, each 2^{AP}-labeled ω-tree $(t, \tau) \in \mathit{Sample}(M_\mathcal{F})$ is a "discretization" of the strategy tree $M_\mathcal{F}$ based on the region graph of the considered timed game. Moreover, each dense path of a $(t, \tau) \in \mathit{Sample}(M_\mathcal{F})$ is also a path of $M_\mathcal{F}$ and each path of $M_\mathcal{F}$ is a dense path of some $(t, \tau) \in \mathit{Sample}(M_\mathcal{F})$. Thus, (t, τ) is contained into $M_\mathcal{F}$. As we show in the following lemma, each tree in $\mathit{Sample}(M_\mathcal{F})$ contains enough information to establish the fulfillment of any TCTL–formula on the dense tree $M_\mathcal{F}$.

Lemma 2 *Given a* TCTL*–formula φ, if $M_\mathcal{F} \models \varphi$ then for all $(t, \tau) \in \mathit{Sample}(M_\mathcal{F})$ it holds that $(t, \tau) \models \varphi$. Moreover, if there exists a $(t, \tau) \in \mathit{Sample}(M_\mathcal{F})$ such that $(t, \tau) \models \varphi$, then $M_\mathcal{F} \models \varphi$.*

Proof : Let $(t, \tau) \in \mathit{Sample}(M_\mathcal{F})$, and $\pi = v_0, v_1, \ldots$ be a path of (t, τ). We denote by ρ_π the function from $\mathbb{R}_+$ to the state space of Γ defined as $\rho_\pi(d - \gamma_{v_0}) = (s_i, \nu_i + d')$ if: $g(v_i) = (s_i, \nu_i)$, $\gamma_{v_i} \leq d < \gamma_{v_{i+1}}$, and $d' = d - \gamma_{v_i}$ (we recall that, for a timed tree, γ_v is the time of occurrence of node v). Given a function ρ from $\mathbb{R}_+$ to the state space of Γ, we denote by $[\rho]$ the function defined as $[\rho](d) = [\rho(d)]$ for all $d \in \mathbb{R}_+$, i.e., $[r](d)$ is labeled by the region $[r(d)]$. We recall that the nodes of $M_\mathcal{F}$ are states of Γ and the dense paths of $M_\mathcal{F}$ branching from a node (s, ν) are suffixes of maximal plays of $\mathcal{F}$ starting at (s, ν). By the definitions of $M_\mathcal{F}$ and $\mathit{Sample}(M_\mathcal{F})$, we have that: for each node (s, ν)

[3]This is actually not a limitation, since we could make ν_1 a boundary region simply by adding an extra clock to Γ, that is reset on each move.

of $M_{\mathcal{F}}$ and for each dense path ρ in $M_{\mathcal{F}}$ starting at (s,ν), there exist a node v and a path π of $M^{t,\tau}$ starting at v such that $[\rho]=[\rho_\pi]$. The vice-versa also holds.

It is known that there is no TCTL–formula that can distinguish between two states (s,ν) and (s,ν') of timed graph belonging to the same region (see [2]). Thus, using the above results it is possible to complete the proof of the lemma by a simple induction on the structure of TCTL-formulas. We omit further details. $\square$

4. A solution to TCTL games

In this section we describe a solution to TCTL games. Given a TCTL game (Γ,φ), we first show how to construct a timed Büchi tree automaton A_Γ that accepts all the timed trees which are discretizations (in the sense discussed in Section 3) of strategies of the protagonist. Then, we combine this result with some known results on TCTL and timed games, and obtain a solution to the stated game problem.

In the previous section, we give a definition of the trees contained in $Sample(M_{\mathcal{F}})$ requiring the property that sibling nodes corresponding to σ-moves have associated a same delay d. Also, if there are siblings of such nodes which correspond to ε-moves, the delay associated with them is less than d (see Figure 1). Since no timed tree automaton can check either one of these properties, we are forced to relax them. We design a timed tree automaton accepting trees such that sibling nodes corresponding to ε and σ-moves represent states that can be reached by simply a transition from states of a region which is an immediate time-successor of the state corresponding to the parent node. We will prove that this is sufficient to prove our results.

Formally, given a game (Γ,φ), where $\Gamma=(S,\Sigma^\varepsilon,s_0,\Delta,C)$, we introduce a timed tree automaton $A_\Gamma=(X,\Sigma^\varepsilon,x_0,\Delta_\Gamma,C,F)$ whose transitions are obtained by grouping in a non-trivial way those of Γ. Each location $x\in X$ is a triple $(s,\lambda,\alpha)\in S\times 2^C\times Reg(\Gamma)$. The initial state is $x_0=\langle s_0,\emptyset,[\bar{0}]\rangle$. We need to store this additional information since, in a timed tree, the time value associated to a node represents the amount of time elapsed between the occurrence of the parent node and the occurrence of the current node. So that, if the automaton has to check that a branch of the tree corresponds to a certain move $(s,\sigma,s',\lambda,\delta)$ of the game, then it must first read the time associated to the corresponding child and then verify that the current clock valuation satisfies δ. In other terms, the information on the delays is shifted one level downward with respect to the information on the truth values of the atomic propositions.

In the following construction, given a location $(s,\lambda,\alpha)\in X$, we denote by β the immediate time-successor of the region $[\lambda\leftarrow 0]\alpha$ (notice that β is the immediate time-successor of α if $\lambda=\emptyset$). For all $\sigma\in\Sigma$, define $V^\sigma=\{\,(s,\sigma,s',\lambda',\delta')\in\Delta\mid\beta\text{ satisfies }\delta'\,\}$ and $V^\varepsilon=\{\,(s,\varepsilon,s',\lambda',\delta')\in\Delta\mid\beta\text{ satisfies }\delta'\,\}$, that are the sets of all moves of the game starting from location s, enabled in the clock region α (resp. β) and labeled with σ (resp. ε). We assume that $V^\sigma=\{(s,\sigma,s_1^\sigma,\lambda_1^\sigma,\delta_1^\sigma),\ldots,(s,\sigma,s_n^\sigma,\lambda_n^\sigma,\delta_n^\sigma)\}$ and $V^\varepsilon=\{(s,\varepsilon,s_1^\varepsilon,\lambda_1^\varepsilon,\delta_1^\varepsilon),\ldots,(s,\varepsilon,s_m^\varepsilon,\lambda_m^\varepsilon,\delta_m^\varepsilon)\}$.

The transition function Δ_Γ is the smallest set satisfying the following rules:

- $(\langle s,\lambda,\alpha\rangle, AP(s), \langle s,\emptyset,\beta\rangle, \langle s_1^\varepsilon,\lambda_1^\varepsilon,\beta\rangle,\ldots,\langle s_m^\varepsilon,\lambda_m^\varepsilon,\beta\rangle,\underbrace{\lambda,\ldots,\lambda}_{m+1},\delta_\alpha)\in\Delta_\Gamma$;
- $(\langle s,\lambda,\alpha\rangle, AP(s), \langle s_1^\sigma,\lambda_1^\sigma,\beta\rangle,\ldots,\langle s_n^\sigma,\lambda_n^\sigma,\beta\rangle,\underbrace{\lambda,\ldots,\lambda}_{n},\delta_\alpha)\in\Delta_\Gamma$, if β is a boundary region;
- $(\langle s,\lambda,\alpha\rangle, AP(s), \langle s_1^\sigma,\lambda_1^\sigma,\beta\rangle,\ldots,\langle s_n^\sigma,\lambda_n^\sigma,\beta\rangle,\langle s_1^\varepsilon,\lambda_1^\varepsilon,\beta\rangle,\ldots,\langle s_m^\varepsilon,\lambda_m^\varepsilon,\beta\rangle,\underbrace{\lambda,\ldots,\lambda}_{n+m},\delta_\alpha)\in\Delta_\Gamma$, if β is an open region.

Finally, we set $F=X$.

Define $Samples$ as the union of $Sample(M_{\mathcal{F}})$ over all strategies $\mathcal{F}$. In the following lemma, we show that A_Γ accepts all samplings of strategy trees.

Lemma 3 *Given a game* (Γ,φ)*,*

$$L(A_\Gamma)\supseteq Samples.$$

Proof : Given a strategy $\mathcal{F}$, and a tree $(t,\tau)\in Sample(M_{\mathcal{F}})$, we claim that from the function g, as defined on page 7, we can easily build an accepting run (r,θ) of A_Γ on (t,τ). Our candidate run is the following:

- $r(root)=\langle s_0,\emptyset,[\bar{0}]\rangle$ and $\theta(root)=\bar{0}$;
- for all $ui\in dom(t)$, where i is a natural number, if $g(u)=(s,\nu)$ and $g(ui)=(s',\nu')$, we set $r(ui)=(s',\lambda,\beta)$, where $\lambda=\{\,x\in C\mid\nu'(x)=0\,\}$ and β is the immediate time-successor of $[v]$, and $\theta(ui)=\nu'$.

Directly from the definition of function g, it is possible to prove by induction that (r,θ) is a run of A_Γ on (t,τ), and since all the locations of A_Γ are accepting, we have also that (r,θ) is an accepting run. $\square$

Lemma 4 *For all* $(t,\tau)\in L(A_\Gamma)$*, there exists a* τ' *such that* $(t,\tau')\in Samples$*. Moreover, given a* TCTL*–formula* φ*, it holds that:*

$$(t,\tau)\models\varphi \textit{ if and only if } (t,\tau')\models\varphi.$$

Proof : Let $(t, \tau) \in L(A_\Gamma)$. If $(t, \tau) \notin Samples$, define τ' as the time tree obtained from τ by labeling all sibling nodes corresponding to σ-moves with a same time delay d, which is selected among the delays with which these nodes are associated by τ. If such nodes have also siblings which correspond to ε-moves, τ' assigns to each of these last nodes an arbitrary time delay less than d. By the definition of $Sample(M_\mathcal{F})$, it is possible to prove by induction that there exists a strategy $\mathcal{F}$ such that $(t, \tau') \in Sample(M_\mathcal{F})$. To complete the proof, we observe that by the definition of τ' and the property that TCTL–formulas cannot distinguish among states of the same region (see [2]), we have that $(t, \tau) \models \varphi$ if and only if $(t, \tau') \models \varphi$, for any TCTL–formula φ. $\square$

Denote as A_φ the timed Büchi tree automaton such that the properties stated in Lemma 1 are satisfied with respect to an integer k not smaller than the maximum arity among all the trees in $Samples$.

Theorem 1 *Given a game* (Γ, φ)*, there exists a winning strategy of the protagonist in* (Γ, φ) *if and only if* $L(A_\Gamma) \cap L(A_\varphi) \neq \emptyset$.

Proof : First, suppose that there exists a timed tree $(t, \tau) \in L(A_\Gamma) \cap L(A_\varphi)$. By Lemma 1, from $(t, \tau) \in L(A_\varphi)$ we have $(t, \tau) \models \varphi$. Since $(t, \tau) \in L(A_\Gamma)$ and $(t, \tau) \models \varphi$, from Lemma 4 we have that there exists a τ' such that $(t, \tau') \models \varphi$ and $(t, \tau') \in Samples$. Thus by Lemma 2, there exists a winning strategy of the protagonist. Vice-versa, suppose that there exists a winning strategy of the protagonist, by Lemma 2, there exists a timed tree $(t, \tau) \in Samples$ such that $(t, \tau) \models \varphi$, and by Lemma 3, $(t, \tau) \in L(A_\Gamma)$. By the definition of A_φ and Lemma 1, we have that (t, τ) also belongs to $L(A_\varphi)$. $\square$

Now, we prove the main result of this paper.

Theorem 2 *Deciding* TCTL *games is* EXPTIME*-complete.*

Proof : In Theorem 1, we reduced the problem of deciding TCTL games to the emptiness of $L(A_\Gamma) \cap L(A_\varphi)$. Since A_Γ and A_φ are TBTAs and this class of automata is closed under intersection (Remark 1), we can build the automaton $A_{\Gamma\cap\varphi}$, accepting the intersection $L(A_\Gamma) \cap L(A_\varphi)$, and check for its emptiness. By Lemma 1 the automaton A_φ has $O(2^{|\varphi|})$ locations and clock size is $O(|\varphi|)$. Denote by k the number of clocks of Γ, since A_Γ has $O(|\Gamma| \cdot c_{max}^k \cdot 2^k)$ locations, that is $O(2^{|\Gamma|})$ locations, and the size of its constants and the number of clocks are both $O(|\Gamma|)$, we have that the intersection automaton has a number of locations which is exponential in both the size of the formula and the size of the game graph. Since, by Lemma 1, checking the emptiness of TBTAs takes polynomial time in the number of locations and exponential time in the number of clocks and the size of constants in the clock constraints, it follows that deciding TCTL games is in EXPTIME. To prove EXPTIME-hardness, we observe that the winning condition of reachability timed games can be expressed by a simple TCTL-formula $\forall\Diamond\varphi$ where φ is a boolean combination over atomic propositions representing the winning locations. Since reachability timed games are EXPTIME-hard [14], we thus have that TCTL games are EXPTIME-complete. $\square$

5. TCTL games with equality

In the definition of the logic TCTL given in Section 2.2, we allow to use only timing constraints of the form $\approx c$ for $\approx$ being one among $\leq$, $<$, $>$, and $\geq$. In this section, we denote by TCTL$^=$ the logic TCTL augmented with equality in the timing constraints[4], and show that timed games with winning conditions expressed by TCTL$^=$ formulas are undecidable. This result is quite expected since it is already known that equality makes the satisfiability problem for TCTL undecidable [2]. The same was also observed for the linear-time temporal logic MITL [4]. Moreover, our undecidability result is closely related to the one recently proved for the TCTL module checking [18]. There, the authors give a direct reduction from the problem of deciding if a nondeterministic two-counter machine has a recurring computation (i.e., the starting state is visited infinitely often).

The proof we give in this paper is quite simple and reduces instead the satisfiability problem for the universally quantified fragment of TCTL$^=$, denoted $\forall$TCTL$^=$. Formally, the syntax of $\forall$TCTL$^=$ is defined by the grammar:

$$\varphi := p \,|\, \neg p \,|\, \varphi \wedge \varphi \,|\, \varphi \vee \varphi \,|\, \forall[\varphi U_{\approx c} \varphi]$$

where $p \in AP$, $\approx$ belongs to $\{<, \leq, =, >, \geq\}$, and c is a natural number. The satisfiability problem for this fragment of TCTL$^=$ is also undecidable [2].

Theorem 3 *Checking for the existence of a winning strategy in a* TCTL$^=$ *game is not decidable.*

Proof : Given a $\forall$TCTL$^=$ formula φ, we claim that φ is satisfiable if and only if it is satisfiable on a timed ω-word. To see this, observe that from a model M of φ we can obtain a timed ω-word on which φ is fulfilled by simply selecting a dense path. The vice-versa is trivially true since a word is a tree of arity 1. Let AP be the set of atomic propositions and Γ be the game graph with only one clock x, the set 2^{AP} as both the set of locations and the alphabet, the empty set as the starting location, and transition rules $(s, s', s', \emptyset, true)$ (that is, the clock constraints are always satisfied and x never gets reset). It is easy to verify that for any timed ω-word over the alphabet 2^{AP} there is exactly one corresponding run of Γ. Moreover, since there

[4] We recall that this was the definition of TCTL given in [2].

are no ε-moves, any strategy of the protagonist selects exactly a timed ω-word. Thus, for any timed ω-word w over the alphabet 2^{AP}, there exists a corresponding strategy of Γ which selects w and vice-versa. Hence, we have that for a given $\forall$TCTL$^{=}$ formula φ, φ is satisfiable if and only if there exists a winning strategy of the protagonist in the timed game (Γ, φ). Since the $\forall$TCTL$^{=}$ satisfiability problem is undecidable [2], we are done. $\square$

6. Conclusion

In this paper we have investigated the verification of open real-time systems modeled as TCTL games, that is, timed games with winning conditions given by TCTL-formulas. Our approach relies on a reduction to the emptiness problem for timed Büchi tree automata. We have also shown that including equality in the timing constraints, TCTL games become undecidable. This result was also proved for a closely related problem on TCTL-formulas, the so-called TCTL module-checking [18].

An interesting aspect of this work is that theory of automata can be successfully used to solving timed games. This was already observed for timed games with discrete-time specifications [11], and here we extend it to dense-time branching-time specifications.

The controller synthesis for timed automata with specification given by a nondeterministic timed automaton turns out to be decidable if the resources of the controller are a priori bounded [10]. For TCTL$^{=}$ games, constraining the resources of the controller may also lead to decidability and it would be interesting to establish this. Moreover, the expressiveness of TCTL can be increased by allowing intervals in timing constraints or removing the syntactic restriction of pairing path quantifiers with temporal operators (TCTL*). Decision problems for these extensions of TCTL are worth investigating.

Acknowledgments

We would like to thank Margherita Napoli for supporting and encouraging us in all the stages of this work. We are also grateful to P. Madhusudan for helpful discussions.

References

[1] M. Abadi, L. Lamport, and P. Wolper. Realizable and unrealizable specifications of reactive systems. In *Proc. of the 16th Intern. Colloquium on Automata, Languages and Programming, ICALP'89*, LNCS 372, pages 1–17, 1989.

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

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

[4] R. Alur, T. Feder, and T. Henzinger. The benefits of relaxing punctuality. *Journal of the ACM*, 43(1):116 – 146, 1996.

[5] E. Asarin and O. Maler. As soon as possible: Time optimal control for timed automata. In *Proc. of the 2nd International Workshop on Hybrid Systems: Computation and Control*, LNCS 1569, pages 19 – 30. Springer-Verlag, 1999.

[6] E. Asarin, O. Maler, A. Pnueli, and J. Sifakis. Controller synthesis for timed automata. In *Proc. IFAC Symposium on System Structure and Control*, pages 469 – 474. Elsevier, 1998.

[7] E. Clarke and E. Emerson. Using branching time temporal logic to synthesize synchronization skeletons. *Science of Computer Programming*, 2(3):241 – 266, 1982.

[8] E. Clarke, E. Emerson, and A. Sistla. Automatic verification of finite-state concurrent systems using temporal-logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244 – 263, 1986.

[9] E. Clarke and R. Kurshan. Computer-aided verification. *IEEE Spectrum*, 33(6):61 – 67, 1996.

[10] D. D'Souza and P. Madhusudan. Timed control synthesis for external specifications. In *Proc. of the 19th Annual Symposium on Theoretical Aspects of Computer Science, STACS'02*, LNCS 2285, pages 571 – 582. Springer, 2002.

[11] M. Faella, S. La Torre, and A. Murano. Automata-theoretic decision of timed games. In *Proc. of the 3rd International Workshop on Verification, Model Checking, and Abstract Interpretation, VMCAI'02*, LNCS 2294, pages 94 – 108. Springer, 2002.

[12] D. Harel, A. Pnueli, and J. Stavi. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, 26:222 – 243, 1983.

[13] T. Henzinger, B. Horowitz, and R. Majumdar. Rectangular hybrid games. In *Proc. of the 10th International Conference on Concurrency Theory, CONCUR'99*, LNCS 1664, pages 320 – 335, 1999.

[14] T. Henzinger and P. Kopke. Discrete-time control for rectangular hybrid automata. *Theoretical Computer Science*, 221(1–2):369–392, 1999.

[15] O. Kupferman, M. Y. Vardi, and P. Wolper. Module checking. *Information and Computation*, 164(2):322– 344, 2001.

[16] S. La Torre and M. Napoli. A decidable dense branching-time temporal logic. In *Proc. of the 20th Conference on the Foundations of Software Technology and Theoretical Computer Science, FSTTCS'00*, LNCS 1974, pages 139 – 150. Springer-Verlag, 2000.

[17] S. La Torre and M. Napoli. Timed tree automata with an application to temporal logic. *Acta Informatica*, 38(2):89–116, 2001.

[18] A. Murano and M. Vardi. *Real-Time Module Checking*. 2002. Rice University.

[19] J. V. Neumann and O. Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1944.

[20] A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proc. of the 16th ACM Symposium on Principles of Programming Languages*, pages 179 – 190, 1989.