Trading Infinite Memory for
Uniform Randomness in Timed Games[*]

Krishnendu Chatterjee[1], Thomas A. Henzinger[1,2], and Vinayak S. Prabhu[1]

[1]EECS, UC Berkeley; [2]CCS, EPFL;
{c_krish,vinayak}@eecs.berkeley.edu, tah@epfl.ch

Abstract. We consider concurrent two-player timed automaton games with ω-regular objectives specified as parity conditions. These games offer an appropriate model for the synthesis of real-time controllers. Earlier works on timed games focused on pure strategies for each player. We study, for the first time, the use of *randomized* strategies in such games. While pure (i.e., nonrandomized) strategies in timed games require infinite memory for winning even with respect to reachability objectives, we show that randomized strategies can win with finite memory with respect to all parity objectives. Also, the synthesized randomized real-time controllers are much simpler in structure than the corresponding pure controllers, and therefore easier to implement. For safety objectives we prove the existence of pure finite-memory winning strategies. Finally, while randomization helps in simplifying the strategies required for winning timed parity games, we prove that randomization does not help in winning at more states.

1 Introduction

Timed automata [2] are models of real-time systems in which states consist of discrete locations and values for real-time clocks. The transitions between locations are dependent on the clock values. *Timed automaton games* [9, 1, 7, 13, 12] are used to distinguish between the actions of several players (typically a "controller" and a "plant"). We shall consider two-player timed automaton games with ω-regular objectives specified as *parity conditions*. The class of ω-regular objectives can express all safety and liveness specifications that arise in the synthesis and verification of reactive systems, and parity conditions are a canonical form to express ω-regular objectives [19]. The construction of a winning strategy for player 1 in such games corresponds to the *controller synthesis problem for real-time systems* [11, 16, 17, 20] with respect to achieving a desired ω-regular objective.

The issue of *time divergence* is crucial in timed games, as a naive control strategy might simply block time, leading to "zeno" runs. Such invalid solutions

have often been avoided by putting strong syntactic constraints on the cycles of timed automaton games [17, 4, 13, 5], or by semantic conditions that discretize time [14]. Other works [16, 11, 6, 7] have required that time divergence be ensured by the controller —a one-sided, unfair view in settings where the player modeling the plant is allowed to block time. We use the more general, semantic and fully symmetric formalism of [9, 15] for dealing with the issue of time divergence. This setting places no syntactic restriction on the game structure, and gives both players equally powerful options for advancing time, but for a player to win, she must not be *responsible* for causing time to converge. It has been shown in [15] that this is equivalent to requiring that the players are restricted to the use of *receptive* strategies [3, 18], which, while being required to not prevent time from diverging, are not required to ensure time divergence. More formally, our timed games proceed in an infinite sequence of rounds. In each round, both players simultaneously propose moves, with each move consisting of an action and a time delay after which the player wants the proposed action to take place. Of the two proposed moves, the move with the shorter time delay "wins" the round and determines the next state of the game. Let a set Φ of runs be the desired objective for player 1. Then player 1 has a *winning* strategy for Φ if she has a strategy to ensure that, no matter what player 2 does, one of the following two conditions hold: (1) time diverges and the resulting run belongs to Φ, or (2) time does not diverge but player-1's moves are chosen only finitely often (and thus she is not to be blamed for the convergence of time).

The winning strategies constructed in [9] for such timed automaton games assume the presence of an infinitely precise global clock to measure the progress of time, and the strategies crucially depend on the value of this global clock. Since the value of this clock needs to be kept in memory, the constructed strategies require *infinite memory*. In fact, the following example (Example 2) shows that infinite memory is necessary for winning with respect to reachability objectives. Besides the infinite-memory requirement, the strategies constructed in [9] are structurally complicated, and it would be difficult to implement the synthesized controllers in practice. Before offering a novel solution to this problem, we illustrate the problem with an example of a simple timed game whose solution requires infinite memory.

Example 1 (Signaling hub). Consider a signaling hub that both sends and receives signals at the same port. At any time the port can either receive or send a signal, but it cannot do both. Moreover, the hub must accept all signals sent to it. If both the input and the output signals arrive at the same time, then the output signal of the hub is discarded. The input signals are generated by other processes, and infinitely many signals cannot be generated in a finite amount of time. The time between input signals is not known a priori. The system may be modeled by the timed automaton game shown in Figure 1. The actions b_1 and b_2 correspond to input signals, and a_1 and a_2 to output signals. The actions b_i are controlled by the environment and denote input signals; the actions a_i are controlled by the hub and denote signals sent by the hub. The clock x models the time delay between signals: all signals reset this clock, and signals can arrive
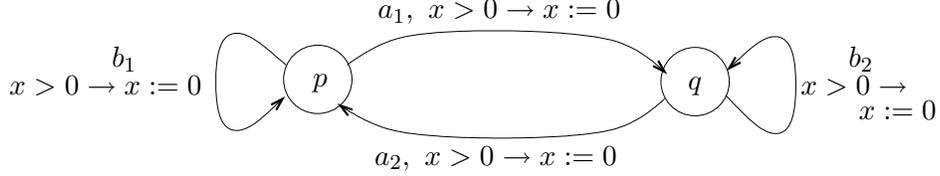
Fig. 1. A timed automaton game.

or be sent provided the value of x is greater than 0, ensuring that there is a positive delay between signals. The objective of the hub controller is to keep sending its own signals, which can be modeled as the generalized Büchi condition of switching infinitely often between the locations p and q (ie., the LTL objective $\Box(\Diamond p \wedge \Diamond q)$). □

Example 2 (Winning requires infinite memory). Consider the timed game of Figure 1. We let κ denote the valuation of the clock x. We let the special "action" $\bot$ denote a time move (representing time passage without an action). The objective of player 1 is to reach q starting from $s_0 = \langle p, x = 0 \rangle$ (and similarly, to reach p from q). We let π_1 denote the strategy of player 1 which prescribes moves based on the history $r[0..k]$ of the game at stage k. Suppose player 1 uses only finite memory. Then player 1 can propose only moves from a finite set when at s_0. Since a zero time move keeps the game at p, we may assume that player 1 does not choose such moves. Let $\Delta > 0$ be the least time delay of these finitely many moves of player 1. Then player 2 can always propose a move $\langle \Delta/2, b \rangle$ when at s_0. This strategy will prevent player 1 from reaching q, and yet time diverges. Hence player 1 cannot win with finite memory; that is, there is no hub controller that uses only finite memory. However, player 1 has a winning strategy with infinite memory. For example, consider the player 1 strategy π_2 such that $\pi_2(r[0..k]) = \langle 1/2^{k+2}, a_1 \rangle$ if $r[k] = \langle p, \kappa \rangle$. and $\pi_2(r[0..k]) = \langle 1, \bot \rangle$ otherwise. □

In this paper we observe that the infinite-memory requirement of Example 1 is due to the determinism of the permissible strategies: a strategy is *deterministic* (or *pure*) if in each round of the game, it proposes a unique move (i.e., action and time delay). A more general class of strategies are the randomized strategies: a *randomized* strategy may propose, in each round, a probability distribution of moves. We now show that in the game of Example 2 finite-memory randomized winning strategies do exist. Indeed, the needed randomization has a particularly simple form: player 1 proposes a unique action together with a time interval from which the time delay is chosen uniformly at random. Such a strategy can be implemented as a controller that has the ability to wait for a randomly chosen amount of time.

Example 3 (Randomization instead of infinite memory). Recall the game in Figure 1. Player 1 can play a randomized memoryless strategy π_3 such that $\pi_3(\langle p, \kappa \rangle) = \langle \mathsf{Uniform}((0, 1 - \kappa(x))), a_i \rangle$; that is, the action a_i is proposed to take

place at a time chosen uniformly at random in the interval $(0, 1 - \kappa(x))$. Suppose player 2 always proposes the action b_i with varying time delays Δ_j at round j. Then the probability of player-1's move being never chosen is $\prod_{j=1}^{\infty}(1 - \Delta_j)$, which is 0 if $\sum_{j=1}^{\infty} \Delta_j = \infty$ (see [8] for the proof). Interrupting moves with pure time moves does not help player 2, as $1 - \frac{\Delta_j}{1 - \kappa(x)} < 1 - \Delta_j$. Thus the simple randomized strategy π_3 is winning for player 1 with probability 1. $\qquad\square$

Previously, only deterministic strategies were studied for timed games; here, for the first time, we study randomized strategies. We show that randomized strategies are not more powerful than deterministic strategies in the sense that if player 1 can win with a randomized strategy, then she can also win with a deterministic strategy. However, as the example illustrated, randomization can lead to a reduction in the memory required for winning, and to a significant simplification in the structure of winning strategies. Randomization is therefore not only of theoretical interest, but can improve the implementability of synthesized controllers. It is for this reason that we set out, in this paper, to systematically analyze the trade-off between randomization requirements (no randomization; uniform randomization; general randomization), memory requirements (finite memory and infinite memory) and the presence of extra "controller clocks" for various classes of ω-regular objectives (safety; reachability; parity objectives).

Our results are as follows. First, we show that for safety objectives pure (no randomization) finite-memory winning strategies exist. Next, for reachability objectives, we show that pure (no randomization) strategies require infinite memory for winning, whereas uniform randomized finite-memory winning strategies exist. We then use the results for reachability and safety objectives in an inductive argument to show that uniform randomized finite-memory strategies suffice for all parity objectives, for which pure strategies require infinite memory (because reachability is a special case of parity). In all our uses of randomization, we only use uniform randomization over time, and more general forms of randomization (nonuniform distributions; randomized actions) are not required. This shows that in timed games, infinite memory can be traded against uniform randomness. Finally, we show that while randomization helps in simplifying winning strategies, and thus allows the construction of simpler controllers, randomization does not help a player in winning at more states, and thus does not allow the construction of more powerful controllers. In other words, the case for randomness rests in the simplicity of the synthesized real-time controllers, not in their expressiveness.

We note that in our setting, player 1 (i.e., the controller) can trade infinite memory also against finite memory together with an extra clock. We assume that the values of all clocks of the plant are observable. For an ω-regular objective Φ, we define the following winning sets depending on the power given to player 1: let $[\![\Phi]\!]_1$ be the set of states from which player 1 can win using any strategy (finite or infinite memory; pure or randomized) and any number of infinitely precise clocks; in $[\![\Phi]\!]_2$ player 1 can win using a pure finite-memory strategy and only one extra clock; in $[\![\Phi]\!]_3$ player 1 can win using a pure finite-memory strategy and no extra clock; and in $[\![\Phi]\!]_4$ player 1 can win using a randomized finite-memory strategy and no extra clock. Then, for every timed automaton

game, we have $\llbracket \Phi \rrbracket_1 = \llbracket \Phi \rrbracket_2 = \llbracket \Phi \rrbracket_4$. We also have $\llbracket \Phi \rrbracket_3 \subseteq \llbracket \Phi \rrbracket_1$, with the subset inclusion being in general strict. It can be shown that at least one bit of memory is required for winning of reachability objectives despite player 1 being allowed randomized strategies. We do not know whether memory is required for winning safety objectives (even in the case of pure strategies).

2 Timed Games

In this section we present the definitions of timed game structures, runs, objectives, strategies and the notions of sure and almost-sure winning in timed game structures.

Timed game structures. A *timed game structure* is a tuple $\mathcal{G} = \langle S, A_1, A_2, \Gamma_1, \Gamma_2, \delta \rangle$ with the following components.

- S is a set of states.
- A_1 and A_2 are two disjoint sets of actions for players 1 and 2, respectively. We assume that $\perp \notin A_i$, and write $A_i^\perp$ for $A_i \cup \{\perp\}$. The set of *moves* for player i is $M_i = \mathbb{R}_{\geq 0} \times A_i^\perp$. Intuitively, a move $\langle \Delta, a_i \rangle$ by player i indicates a waiting period of Δ time units followed by a discrete transition labeled with action a_i.
- $\Gamma_i : S \mapsto 2^{M_i} \setminus \emptyset$ are two move assignments. At every state s, the set $\Gamma_i(s)$ contains the moves that are available to player i. We require that $\langle 0, \perp \rangle \in \Gamma_i(s)$ for all states $s \in S$ and $i \in \{1, 2\}$. Intuitively, $\langle 0, \perp \rangle$ is a time-blocking stutter move.
- $\delta : S \times (M_1 \cup M_2) \mapsto S$ is the transition function. We require that for all time delays $\Delta, \Delta' \in \mathbb{R}_{\geq 0}$ with $\Delta' \leq \Delta$, and all actions $a_i \in A_i^\perp$, we have (1) $\langle \Delta, a_i \rangle \in \Gamma_i(s)$ iff both $\langle \Delta', \perp \rangle \in \Gamma_i(s)$ and $\langle \Delta - \Delta', a_i \rangle \in \Gamma_i(\delta(s, \langle \Delta', \perp \rangle))$; and (2) if $\delta(s, \langle \Delta', \perp \rangle) = s'$ and $\delta(s', \langle \Delta - \Delta', a_i \rangle) = s''$, then $\delta(s, \langle \Delta, a_i \rangle) = s''$.

The game proceeds as follows. If the current state of the game is s, then both players simultaneously propose moves $\langle \Delta_1, a_1 \rangle \in \Gamma_1(s)$ and $\langle \Delta_2, a_2 \rangle \in \Gamma_2(s)$. The move with the shorter duration "wins" in determining the next state of the game. If both moves have the same duration, then the move of player 2 determines the next state.[1] We use this setting as our goal is to compute the winning set for player 1 against all possible strategies of player 2. Formally, we define the *joint destination function* $\delta_{\mathsf{jd}} : S \times M_1 \times M_2 \mapsto S$ by

$$\delta_{\mathsf{jd}}(s, \langle \Delta_1, a_1 \rangle, \langle \Delta_2, a_2 \rangle) = \begin{cases} \delta(s, \langle \Delta_1, a_1 \rangle) & \text{if } \Delta_1 < \Delta_2; \\ \delta(s, \langle \Delta_2, a_2 \rangle) & \text{if } \Delta_2 \leq \Delta_1. \end{cases}$$

The time elapsed when the moves $m_1 = \langle \Delta_1, a_1 \rangle$ and $m_2 = \langle \Delta_2, a_2 \rangle$ are proposed is given by $\mathsf{delay}(m_1, m_2) = \min(\Delta_1, \Delta_2)$. The boolean predicate

---

[1] Alternatively, we can define the next state to be determined nondeterministically in the case of ties, without changing our results. We give the player-2 move priority only to simplify the presentation.

$\mathsf{blame}_i(s, m_1, m_2, s')$ indicates whether player i is "responsible" for the state change from s to s' when the moves m_1 and m_2 are proposed. Denoting the opponent of player i by $\sim i = 3 - i$, for $i \in \{1, 2\}$, we define

$$\mathsf{blame}_i(s, \langle \Delta_1, a_1 \rangle, \langle \Delta_2, a_2 \rangle, s') \;=\; \big(\Delta_i \leq \Delta_{\sim i} \;\wedge\; \delta(s, \langle \Delta_i, a_i \rangle) = s' \big).$$

Runs. A *run* of the timed game structure $\mathcal{G}$ is an infinite sequence $r = s_0, \langle m_1^0, m_2^0 \rangle, s_1, \langle m_1^1, m_2^1 \rangle, \ldots$ such that $s_k \in S$ and $m_i^k \in \Gamma_i(s_k)$ and $s_{k+1} = \delta_{\mathsf{jd}}(s_k, m_1^k, m_2^k)$ for all $k \geq 0$ and $i \in \{1, 2\}$. For $k \geq 0$, let $\mathsf{time}(r, k)$ denote the "time" at position k of the run, namely, $\mathsf{time}(r, k) = \sum_{j=0}^{k-1} \mathsf{delay}(m_1^j, m_2^j)$ (we let $\mathsf{time}(r, 0) = 0$). By $r[k]$ we denote the $(k+1)$-th state s_k of r. The run prefix $r[0..k]$ is the finite prefix of the run r that ends in the state s_k. Let Runs be the set of all runs of $\mathcal{G}$, and let $\mathsf{FinRuns}$ be the set of run prefixes.

Objectives. An *objective* for the timed game structure $\mathcal{G}$ is a set $\Phi \subseteq \mathsf{Runs}$ of runs. We will be interested in the classical reachability, safety and parity objectives. Parity objectives are canonical forms for ω-regular properties that can express all commonly used specifications that arise in verification.

- Given a set of states Y, the *reachability* objective $\mathsf{Reach}(Y)$ is defined as the set of runs that visit Y, formally, $\mathsf{Reach}(Y) = \{r \mid$ there exists i such that $r[i] \in Y\}$.
- Given a set of states Y, the *safety* objective consists of the set of runs that stay within Y, formally, $\mathsf{Safe}(Y) = \{r \mid$ for all i we have $r[i] \in Y\}$.
- Let $\Omega : S \mapsto \{0, \ldots, k-1\}$ be a parity index function. The parity objective for Ω requires that the maximal index visited infinitely often is even. Formally, let $\mathsf{InfOften}(\Omega(r))$ denote the set of indices visited infinitely often along a run r. Then the parity objective defines the following set of runs: $\mathsf{Parity}(\Omega) = \{r \mid \max(\mathsf{InfOften}(\Omega(r)))$ is even $\}$.

A timed game structure $\mathcal{G}$ together with the index function Ω constitute a *parity timed game* (of index k) in which the objective of player 1 is $\mathsf{Parity}(\Omega)$. We use similar notations for reachability and safety timed games.

Strategies. A *strategy* for a player is a recipe that specifies how to extend a run. Formally, a *probabilistic strategy* π_i for player $i \in \{1, 2\}$ is a function π_i that assigns to every run prefix $r[0..k]$ a probability distribution $\mathcal{D}_i(r[k])$ over $\Gamma_i(r[k])$, the set of moves available to player i at the state $r[k]$. *Pure strategies* are strategies for which the state space of the probability distribution of $\mathcal{D}_i(r[k])$ is a singleton set for every run r and all k. We let Π_i^{pure} denote the set of pure strategies for player i, with $i \in \{1, 2\}$. For $i \in \{1, 2\}$, let Π_i be the set of strategies for player i. Given two strategies $\pi_1 \in \Pi_1$ and $\pi_2 \in \Pi_2$, the set of possible *outcomes* of the game starting from a state $s \in S$ is denoted $\mathsf{Outcomes}(s, \pi_1, \pi_2)$. Given strategies π_1 and π_2, for player 1 and player 2, respectively, and a starting state s we denote by $\mathrm{Pr}_s^{\pi_1, \pi_2}(\cdot)$ the probability space given the strategies and the initial state s.

Receptive strategies. We will be interested in strategies that are meaningful (in the sense that they do not block time). To define them formally we first present the following two sets of runs.

- A run r is *time-divergent* if $\lim_{k \to \infty} \mathsf{time}(r,k) = \infty$. We denote by Timediv the set of all time-divergent runs.
- The set $\mathsf{Blameless}_i \subseteq \mathsf{Runs}$ consists of the set of runs in which player i is responsible only for finitely many transitions. A run $s_0, \langle m_1^0, m_2^0 \rangle, s_1, \langle m_1^1, m_2^1 \rangle, \ldots$ belongs to the set $\mathsf{Blameless}_i$, for $i = \{1,2\}$, if there exists a $k \geq 0$ such that for all $j \geq k$, we have $\neg\, \mathsf{blame}_i(s_j, m_1^j, m_2^j, s_{j+1})$.

A strategy π_i is *receptive* if for all strategies $\pi_{\sim i}$, all states $s \in S$, and all runs $r \in \mathsf{Outcomes}(s, \pi_1, \pi_2)$, either $r \in \mathsf{Timediv}$ or $r \in \mathsf{Blameless}_i$. Thus, no what matter what the opponent does, a receptive strategy of player i cannot be responsible for blocking time. Strategies that are not receptive are not physically meaningful. A timed game structure $\mathcal{G}$ is *well-formed* if both players have receptive strategies. We restrict our attention to well-formed timed game structures. We denote Π_i^R to be the set of receptive strategies for player i. Note that for $\pi_1 \in \Pi_1^R, \pi_2 \in \Pi_2^R$, we have $\mathsf{Outcomes}(s, \pi_1, \pi_2) \subseteq \mathsf{Timediv}$.

Sure and almost-sure winning modes. Let $\mathsf{Sure}_1^{\mathcal{G}}(\varPhi)$ (resp. $\mathsf{AlmostSure}_1^{\mathcal{G}}(\varPhi)$) be the set of states s in $\mathcal{G}$ such that player 1 has a receptive strategy $\pi_1 \in \Pi_1^R$ such that for all receptive strategies $\pi_2 \in \Pi_2^R$, we have $\mathsf{Outcomes}(s, \pi_1, \pi_2) \subseteq \varPhi$ (resp. $\mathrm{Pr}_s^{\pi_1, \pi_2}(\varPhi) = 1$). Such a winning strategy is said to be a sure (resp. almost sure) winning receptive strategy. In computing the winning sets, we shall quantify over *all* strategies, but modify the objective to take care of time divergence. Given an objective $\varPhi$, let $\mathsf{TimeDivBl}_1(\varPhi) = (\mathsf{Timediv} \cap \varPhi) \cup (\mathsf{Blameless}_1 \setminus \mathsf{Timediv})$, i.e., $\mathsf{TimeDivBl}_1(\varPhi)$ denotes the set of paths such that either time diverges and $\varPhi$ holds, or else time converges and player 1 is not responsible for time to converge. Let $\underline{\mathsf{Sure}}_1^{\mathcal{G}}(\varPhi)$ (resp. $\underline{\mathsf{AlmostSure}}_1^{\mathcal{G}}(\varPhi)$) be the set of states in $\mathcal{G}$ such that for all $s \in \underline{\mathsf{Sure}}_1^{\mathcal{G}}(\varPhi)$ (resp. $\underline{\mathsf{AlmostSure}}_1^{\mathcal{G}}(\varPhi)$), player 1 has a strategy $\pi_1 \in \Pi_1$ such that for all strategies $\pi_2 \in \Pi_2$, we have $\mathsf{Outcomes}(s, \pi_1, \pi_2) \subseteq \varPhi$ (resp. $\mathrm{Pr}_s^{\pi_1, \pi_2}(\varPhi) = 1$). Such a winning strategy is said to be a sure (resp. almost sure) winning for the non-receptive game. The following result establishes the connection between Sure and $\underline{\mathsf{Sure}}$ sets.

Theorem 1. [15] *For all well-formed timed game structures $\mathcal{G}$, and for all ω-regular objectives $\varPhi$, we have $\underline{\mathsf{Sure}}_1^{\mathcal{G}}(\mathsf{TimeDivBl}_1(\varPhi)) = \mathsf{Sure}_1^{\mathcal{G}}(\varPhi)$.*

We now define a special class of timed game structures, namely, timed automaton games.

Timed automaton games. Timed automata [2] suggest a finite syntax for specifying infinite-state timed game structures. A *timed automaton game* is a tuple $\mathcal{T} = \langle L, C, A_1, A_2, E, \gamma \rangle$ with the following components:

- L is a finite set of locations.
- C is a finite set of clocks.
- A_1 and A_2 are two disjoint sets of actions for players 1 and 2, respectively.
- $E \subseteq L \times (A_1 \cup A_2) \times \mathsf{Constr}(C) \times L \times 2^C$ is the edge relation, where the set $\mathsf{Constr}(C)$ of *clock constraints* is generated by the grammar: $\theta ::= x \leq d \mid d \leq x \mid \neg\theta \mid \theta_1 \wedge \theta_2$, for clock variables $x \in C$ and nonnegative integer

constants d. For an edge $e = \langle l, a_i, \theta, l', \lambda \rangle$, the clock constraint θ acts as a guard on the clock values which specifies when the edge e can be taken, and by taking the edge e, the clocks in the set $\lambda \subseteq C$ are reset to 0. We require that for all edges $\langle l, a_i, \theta', l', \lambda' \rangle, \langle l, a_i, \theta'', l'', \lambda'' \rangle \in E$ with $l' \neq l''$, the conjunction $\theta' \wedge \theta''$ is unsatisfiable. This requirement ensures that a state and a move together uniquely determine a successor state.

- $\gamma : L \mapsto \mathsf{Constr}(C)$ is a function that assigns to every location an invariant for both players. All clocks increase uniformly at the same rate. When at location l, each player i must propose a move out of l before the invariant $\gamma(l)$ expires. Thus, the game can stay at a location only as long as the invariant is satisfied by the clock values.

A *clock valuation* is a function $\kappa : C \mapsto \mathbb{R}_{\geq 0}$ that maps every clock to a non-negative real. The set of all clock valuations for C is denoted by $K(C)$. A *state* $s = \langle l, \kappa \rangle$ of the timed automaton game $\mathcal{T}$ is a location $l \in L$ together with a clock valuation $\kappa \in K(C)$ such that the invariant at the location is satisfied, that is, $\kappa \models \gamma(l)$. We let S be the set of all states of $\mathcal{T}$. Given a timed automaton game $\mathcal{T}$, the definition of an associated timed game structure $[\![\mathcal{T}]\!]$ is standard [9]. We shall restrict our attention to randomization over time — a random move of a player will consist of a distribution over time over some interval I, denoted $\mathcal{D}^I$, together with a discrete action a_i.

Clock region equivalence. We use the standard *clock-region equivalence* relation from the theory of timed automata [2] (the formal definition is being ommited here as it is fairly standard). We denote two clock-region equivalent clock valuations κ_1, κ_2 by $\kappa_1 \cong \kappa_2$. Two states $\langle l_1, \kappa_1 \rangle, \langle l_2, \kappa_2 \rangle \in S$ are *clock-region equivalent*, denoted $\langle l_1, \kappa_1 \rangle \cong \langle l_2, \kappa_2 \rangle$, iff $l_1 = l_2$ and $\kappa_1 \cong \kappa_2$. A *clock region* is an equivalence class of states with respect to $\cong$. There are finitely many clock regions; more precisely, the number of clock regions is bounded by $|L| \cdot \prod_{x \in C} (c_x + 1) \cdot |C|! \cdot 2^{|C|}$.

For a state $s \in S$, we write $\mathsf{Reg}(s) \subseteq S$ for the clock region containing s. For a run r, we let the *region sequence* $\mathsf{Reg}(r) = \mathsf{Reg}(r[0]), \mathsf{Reg}(r[1]), \cdots$. Two runs r, r' are region equivalent if their region sequences are the same. Given a distribution $\mathcal{D}_{\mathsf{states}}$ over states, we obtain a corresponding distribution $\mathcal{D}_{\mathsf{reg}} = \mathsf{Reg}_d(\mathcal{D}_{\mathsf{states}})$ over regions as follows: for a region R we have $\mathcal{D}_{\mathsf{reg}}(R) = \mathcal{D}_{\mathsf{states}}(\{s \mid s \in R\})$. An ω-regular objective Φ is a region objective if for all region-equivalent runs r, r', we have $r \in \Phi$ iff $r' \in \Phi$. A strategy π_1 is a *region strategy*, if for all prefixes r_1 and r_2 such that $\mathsf{Reg}(r_1) = \mathsf{Reg}(r_2)$, we have $\mathsf{Reg}_d(\pi_1(r_1)) = \mathsf{Reg}_d(\pi_1(r_2))$. The definition for player 2 strategies is analogous. Two region strategies π_1 and π'_1 are region-equivalent if for all prefixes r we have $\mathsf{Reg}_d(\pi_1(r)) = \mathsf{Reg}_d(\pi'_1(r))$. A parity index function Ω is a region parity index function if $\Omega(s_1) = \Omega(s_2)$ whenever $s_1 \cong s_2$. Henceforth, we shall restrict our attention to region objectives.

Encoding time divergence by enlarging the game structure. Given a timed automaton game $\mathcal{T}$, consider the enlarged game structure $\widehat{\mathcal{T}}$ with the state space $\widehat{S} \subseteq S \times \mathbb{R}_{[0,1)} \times \{\text{TRUE}, \text{FALSE}\}^2$, and an augmented transition relation $\widehat{\delta} : \widehat{S} \times (M_1 \cup M_2) \mapsto \widehat{S}$. In an augmented state $\langle s, \mathfrak{z}, tick, bl_1 \rangle \in \widehat{S}$, the component

$s \in S$ is a state of the original game structure $[\![\mathcal{T}]\!]$, $\mathfrak{z}$ is value of a fictitious clock z which gets reset to 0 every time it hits 1, or if a move of player 1 is chosen, *tick* is true iff z hit 1 at last transition and bl_1 is true if player 1 is to blame for the last transition. Note that any strategy π_i in $[\![\mathcal{T}]\!]$, can be considered a strategy in $\widehat{\mathcal{T}}$. The values of the clock z, *tick* and bl_1 correspond to the values each player keeps in memory in constructing his strategy. Any run r in $\mathcal{T}$ has a corresponding unique run $\widehat{r}$ in $\widehat{\mathcal{T}}$ with $\widehat{r}[0] = \langle r[0], 0, \text{FALSE}, \text{FALSE} \rangle$ such that r is a projection of $\widehat{r}$ onto $\mathcal{T}$. For an objective Φ, we can now encode time-divergence as: $\mathsf{TimeDivBl}(\Phi) = (\Box\Diamond \, tick \to \Phi) \wedge (\neg\Box\Diamond \, tick \to \Diamond\Box\neg \, bl_1)$. Let $\widehat{\kappa}$ be a valuation for the clocks in $C \cup \{z\}$. A state of $\widehat{\mathcal{T}}$ can then be considered as $\langle\langle l, \widehat{\kappa}\rangle, tick, bl_1\rangle$. We extend the clock equivalence relation to these expanded states: $\langle\langle l, \widehat{\kappa}\rangle \, tick, bl_1\rangle \cong \langle\langle l', \widehat{\kappa}'\rangle, tick', bl_1'\rangle$ iff $l = l'$, $tick = tick'$, $bl_1 = bl_1'$ and $\widehat{\kappa} \cong \widehat{\kappa}'$. For every ω-regular region objective Φ of $\mathcal{T}$, we have $\mathsf{TimeDivBl}(\Phi)$ to be an ω-regular region objective of $\widehat{\mathcal{T}}$.

We now present a lemma that states for region ω-regular objectives region winning strategies exist, and all strategies region-equivalent to a region winning strategy are also winning.

Lemma 1. *Let $\mathcal{T}$ be a timed automaton game and $\widehat{\mathcal{T}}$ be the corresponding enlarged game structure. Let $\widehat{\Phi}$ be an ω-regular region objective of $\widehat{\mathcal{T}}$. Then the following assertions hold.*

- *There is a pure finite-memory region strategy π_1 that is sure winning for $\widehat{\Phi}$ from the states in $\underline{\mathsf{Sure}}_1^{\widehat{\mathcal{T}}}(\widehat{\Phi})$.*
- *If π_1 is a pure region strategy that is sure winning for $\widehat{\Phi}$ from $\underline{\mathsf{Sure}}_1^{\widehat{\mathcal{T}}}(\widehat{\Phi})$ and π_1' is a pure strategy that is region-equivalent to π_1, then π_1' is a sure winning strategy for $\widehat{\Phi}$ from $\underline{\mathsf{Sure}}_1^{\widehat{\mathcal{T}}}(\widehat{\Phi})$.*

Note that there is an infinitely precise global clock z in the enlarged game structure $\widehat{\mathcal{T}}$. If $\mathcal{T}$ does not have such a global clock, then strategies in $\widehat{\mathcal{T}}$ correspond to strategies in $\mathcal{T}$ where player 1 (and player 2) maintain the value of the infinitely precise global clock in memory (requiring infinite memory).

3   Safety: Pure Finite-Memory Strategies

In this section we show the existence of pure finite-memory sure winning strategies for safety objectives in timed automaton games. Given a timed automaton game $\mathcal{T}$, we define two functions $P_{>0} : C \mapsto \{\text{TRUE}, \text{FALSE}\}$ and $P_{\geq 1} : C \mapsto \{\text{TRUE}, \text{FALSE}\}$. For a clock x, the values of $P_{>0}(x)$ and $P_{\geq 1}(x)$ indicate if the clock x was greater than 0 or greater than or equal to 1 respectively, during the last transition (excluding the originating state). Consider the enlarged game structure $\widetilde{\mathcal{T}}$ with the state space $\widetilde{S} = S \times \{\text{TRUE}, \text{FALSE}\} \times \{\text{TRUE}, \text{FALSE}\}^C \times \{\text{TRUE}, \text{FALSE}\}^C$ and an augmented transition relation $\widetilde{\delta}$. A state of $\widetilde{\mathcal{T}}$ is a tuple $\langle s, bl_1, P_{>0}, P_{\geq 1}\rangle$, where s is a state of $\mathcal{T}$, the component bl_1 is TRUE iff player 1 is to be blamed for the last transition, and $P_{>0}, P_{\geq 1}$

are as defined earlier. The clock equivalence relation can be lifted to states of $\widetilde{\mathcal{T}}$: $\langle s, bl_1, P_{>0}, P_{\geq 1} \rangle \cong_{\widetilde{A}} \langle s', bl_1', P_{>0}', P_{\geq 1}' \rangle$ iff $s \cong_{\mathcal{T}} s'$, $bl_1 = bl_1'$, $P_{>0} = P_{>0}'$ and $P_{\geq 1} = P_{\geq 1}'$.

Lemma 2. *Let $\mathcal{T}$ be a timed automaton game, and $\widetilde{\mathcal{T}}$ be the corresponding enlarged game. Then player 1 has a receptive strategy from a state s iff $\langle s, \cdot \rangle \in$ $\underline{\mathsf{Sure}}_1^{\widetilde{\mathcal{T}}}(\Phi^*)$, where $\Phi^* = \Box\Diamond(bl_1 = \text{TRUE}) \to \bigvee_{X \subseteq C} \left(\bigwedge_{x \in X} \Diamond\Box(x > c_x) \wedge \phi_X \right)$, and*

$$\phi_X = \left(\bigwedge_{x \in C \backslash X} \Box\Diamond(x = 0) \right) \wedge \left(\begin{array}{c} \left(\bigvee_{x \in C \backslash X} \Box\Diamond((P_{>0}(x) = \text{TRUE}) \wedge (bl_1 = \text{TRUE})) \right) \\ \vee \\ \left(\bigvee_{x \in C \backslash X} \Box\Diamond((P_{\geq 1}(x) = \text{TRUE}) \wedge (bl_1 = \text{FALSE})) \right) \end{array} \right).$$

The clause $\bigwedge_{x \in X} \Diamond\Box(x > c_x)$ specifies which clocks escape beyond their maximum values (and are never reset). The rest of the clocks hence thus be reset infinitely often in a time diverging run, this is specified by the clause $\bigwedge_{x \in C \backslash X} \Box\Diamond(x = 0)$. Suppose we have $(\bigvee_{x \in C \backslash X} \Box\Diamond((P_{\geq 1}(x) = \text{TRUE}) \wedge (bl_1 = \text{FALSE}))$. Then clearly time diverges as some clock x goes from 0 to 1 infinitely often. Suppose we have $(\bigvee_{x \in C \backslash X} \Box\Diamond((P_{>0}(x) = \text{TRUE}) \wedge (bl_1 = \text{TRUE}))$. We note that if a player can take a move from $\widetilde{s}$ to a region $\widetilde{R}$ where the value of some clock x is such that $0 < x \leq c_x$, then there is some clock $y \in C$ (possibly different from x) such that the player can take a move from $\widetilde{s}$ to the region $\widetilde{R}$ such that the value of y will be more than $1/2$ (and less than or equal to c_y). Thus, in this case, some clock goes from 0 to $1/2$ infinitely often and time diverges. It turns out that these conditions are also required. The full proof can be found in [8].

Theorem 2. *Let $\mathcal{T}$ be a timed automaton game and $\widetilde{\mathcal{T}}$ be the corresponding enlarged game. Let Y be a union of regions of $\mathcal{T}$. Then the following assertions hold.*

1. *$\mathsf{Sure}_1^{\widetilde{\mathcal{T}}}(\Box Y) = \underline{\mathsf{Sure}}_1^{\widetilde{\mathcal{T}}}((\Box Y) \wedge \Phi^*)$, where Φ^* is as defined in Lemma 2.*
2. *Player 1 has a pure, finite-memory, receptive, region strategy that is sure winning for the safety objective $\mathsf{Safe}(Y)$ at every state in $\mathsf{Sure}_1^{\widetilde{\mathcal{T}}}(\Box Y)$.*

4 Reachability: Randomized Finite-Memory Strategies

We have seen in Example 2 that pure sure winning strategies require infinite memory in general for reachability objectives. In this section, we shall show that uniform randomized almost-sure winning strategies with finite memory exist. This shows that we can trade-off infinite memory with uniform randomness.

Let S_R be the destination set of states that player 1 wants to reach. We only consider S_R such that S_R is a union of regions of $\mathcal{T}$. For the timed automaton $\mathcal{T}$, consider the enlarged game structure of $\mathcal{T}$. We let $\widehat{S_R} = S_R \times$

$\mathbb{R}_{[0,1]} \times \{\text{TRUE}, \text{FALSE}\}^2$. From the reachability objective (denoted $\mathsf{Reach}(S_R)$) we obtain the reachability parity objective with index function Ω_R as follows: $\Omega_R(\langle s, \mathfrak{z}, tick, bl_1 \rangle) = 1$ if $tick \lor bl_1 = \text{TRUE}$ and $s \notin S_R$ (0 otherwise). We assume the states in S_R are absorbing. We let $\widehat{S}_R = S_R \times \mathbb{R}_{[0,1]} \times \{\text{TRUE}, \text{FALSE}\}^2$. We now present a μ-calculus characterization for the sure winning set (using only pure strategies) for player 1 for reachability objectives. We first translate the reachability objective to a parity objective of index 2 (the μ-calculus expression will use the parity index function).

Lemma 3. *For a timed automaton game $\mathfrak{T}$, with the reachability objective S_R, consider the enlarged game structure $\widehat{\mathfrak{T}}$, and the corresponding reachability parity function Ω_R. Then we have that $\underline{\mathsf{Sure}}_1(\mathsf{TimeDivBl}(\mathsf{Reach}(S_R))) = \underline{\mathsf{Sure}}_1(\mathsf{Parity}(\Omega_R))$.*

The *controllable predecessor* operator for player 1, $\mathsf{CPre}_1 : 2^{\widehat{S}} \mapsto 2^{\widehat{S}}$, defined formally by $\widetilde{s} \in \mathsf{CPre}_1(Z)$ iff $\exists m_1 \in \widehat{\Gamma}_1(\widehat{s})\ \forall m_2 \in \widehat{\Gamma}_2(\widehat{s}) . \widehat{\delta}_{\mathsf{jd}}(\widehat{s}, m_1, m_2) \subseteq Z$. Informally, $\mathsf{CPre}_1(Z)$ consists of the set of states from which player 1 can ensure that the next state will be in Z, no matter what player 2 does. From Lemma 3 it follows that the sure winning set can be described as the μ-calculus formula: $\mu Y \nu X \left[(\Omega^{-1}(1) \cap \mathsf{CPre}_1(Y)) \cup (\Omega^{-1}(0) \cap \mathsf{CPre}_1(X)) \right]$. The winning set can then be computed as a fixpoint iteration on regions of $\widehat{\mathfrak{T}}$. We can also obtain a pure winning strategy π_{pure} of player 1 as in [10]. Note that this strategy π_{pure} corresponds to an infinite-memory strategy of player 1 in the timed automaton game $\mathfrak{T}$, as she needs to maintain the value of the clock z in memory.

To compute randomized finite-memory almost-sure winning strategies, we will use the structure of the μ-calculus formula. Let $Y^* = \mu Y \nu X \left[(\Omega^{-1}(1) \cap \mathsf{CPre}_1(Y)) \cup (\Omega^{-1}(0) \cap \mathsf{CPre}_1(X)) \right]$. The iterative fixpoint procedure computes $Y_0 = \emptyset \subseteq Y_1 \subseteq \cdots \subseteq Y_n = Y^*$, where $Y_{i+1} = \nu X \left[(\Omega^{-1}(1) \cap \mathsf{CPre}_1(Y_i)) \cup (\Omega^{-1}(0) \cap \mathsf{CPre}_1(X)) \right]$. We can consider the states in $Y_i \setminus Y_{i-1}$ as being added in two steps, T_{2i-1} and $T_{2i}(= Y_i)$ as follows:

1. $T_{2i-1} = \Omega^{-1}(1) \cap \mathsf{CPre}_1(Y_{i-1})$. T_{2i-1} is clearly a subset of Y_i.
2. $T_{2i} = \nu X \left[T_{2i-1} \cup (\Omega^{-1}(0) \cap \mathsf{CPre}_1(X)) \right]$. Note $(T_{2i} \setminus T_{2i-1}) \cap \Omega^{-1}(1) = \emptyset$.

Thus, in odd stages we add states with index 1, and in even stages we add states with index 0. The *rank* of a state $\widehat{s} \in Y^*$ is j if $\widehat{s} \in T_j \setminus \cup_{k=0}^{j-1} T_k$. The set S_R is a union of regions of $\mathfrak{T}$. $T_0 = T_1 = \emptyset$, and T_2 contains the states in $\widehat{S}_R$ together with the states where $tick = bl_1 = \text{FALSE}$, and from where player 1 can ensure that the next state is either in $\widehat{S}_R$, or the next state continues to have $tick = bl_1 = \text{FALSE}$; formally $T_2 = \nu X(\Omega^{-1}(0) \cap \mathsf{CPre}_1(X))$. Henceforth, when we refer to a region R of $\mathfrak{T}$, we shall mean the states $R \times \mathbb{R}_{[0,1]} \times \{\text{TRUE}, \text{FALSE}\}^2$ of $\widehat{\mathfrak{T}}$.

Lemma 4. *Let $T_2 = \nu X(\Omega^{-1}(0) \cap \mathsf{CPre}_1(X))$. Then player 1 has a (randomized) memoryless strategy π_{rand} such that she can ensure reaching $\widehat{S}_R \subseteq \Omega^{-1}(0)$ with probability 1 against all receptive strategies of player 2 from all states $\widehat{s}$ of a region R such that $R \cap T_2 \neq \emptyset$. Moreover, π_{rand} is independent of the values of the global clock, $tick$ and bl_1.*

We now present the main proof ideas for Lemma 4. For a set T of states, we shall denote by $\mathsf{Reg}(T)$ the set of states that are region equivalent in $\mathcal{T}$ to some state in T. The proof first shows that from every state $\widehat{s}$ in $\mathsf{Reg}(T_2 \setminus \widehat{S}_R)$, either a) player 1 has a move to $\widehat{S}_R$, or b) the invariant of the location of $\widehat{s}$ is closed, such that time can progress from $\widehat{s}$ to the endpoint without the clock z crossing 1, and with player 2 having no moves outside of $\mathsf{Reg}(T_2 \setminus \widehat{S}_R)$. From every state in $\mathsf{Reg}(T_2 \setminus \widehat{S}_R)$, the strategy π_{pure} prescribes a move to either $\widehat{S}_R$, or to the right endpoint of the invariant of the location if it is right closed (effectively relinquishing the move to player 2). If player 1 relinquishes the move in such a manner, then so does π_{rand}. If π_{pure} prescribes a move to $\widehat{S}_R$, there is an earliest destination region $\widehat{R}'$ for player 1 to come to $\widehat{S}_R$, and a corresponding interval I of times which takes her to $\widehat{R}'$. If this interval is left closed, then π_{rand} chooses this leftmost endpoint. If I is not left closed, then π_{rand} proposes a randomized move, with the time being distributed uniformly at random over $(\alpha_l, \alpha_l + 1/2]$ where α_l is the leftmost endpoint of I. We then show that if player 2 plays with a receptive strategy, then if player 1 plays a randomized move infinitely often, then a randomized move with $\alpha_l = 0$ must occur infinitely often. Finally, we demonstrate that for this randomized move (of $\alpha_l = 0$) that player 1 plays infinitely often, the probability of it not being chosen against a receptive strategy of player 2 is 0. The full proof can be found in [8].

The following lemma states that if for some state $s \in \mathcal{T}$, we have $(s, \mathfrak{z}, tick, bl_1) \in T_{2i+1}$, for some i, then for some $\mathfrak{z}', tick', bl_1'$ we have $(s, \mathfrak{z}', tick', bl_1') \in T_{2i}$. Then in Lemma 6 we present the inductive case of Lemma 4. The proof of Lemma 6 is similar to the base case i.e., Lemma 4.

Lemma 5. *Let R be a region of $\mathcal{T}$ such that $R \cap T_{2i+1} \neq \emptyset$. Then $R \cap T_{2i} \neq \emptyset$.*

Lemma 6. *Let R be a region of $\mathcal{T}$ such that $R \cap T_{2i} \neq \emptyset$, and $R \cap T_j = \emptyset$ for all $2 \leq j < 2i$. Then player 1 has a (randomized) memoryless strategy π_{rand} to go from R to some R' such that $R' \cap T_j \neq \emptyset$ for some $j < 2i$ with probability 1 against all receptive strategies of player 2. Moreover, π_{rand} is independent of the values of the global clock, tick and bl_1.*

Once player 1 reaches the target set, she can switch over to the finite-memory receptive strategy of Lemma 2. Thus, using Lemmas 2, 4, 5, and 6 we have the following theorem.

Theorem 3. *Let $\mathcal{T}$ be a timed automaton game, and let S_R be a union of regions of $\mathcal{T}$. Player 1 has a randomized, finite-memory, receptive, region strategy π_1 such that for all states $s \in \mathsf{Sure}_1(\mathsf{Reach}(S_R))$, the following assertions hold: (a) for all receptive strategies π_2 of player 2 we have $\mathrm{Pr}_s^{\pi_1, \pi_2}(\mathsf{Reach}(S_R)) = 1$; and (b) for all strategies π_2 of player 2 we have $\mathrm{Pr}_s^{\pi_1, \pi_2}(\mathsf{TimeDivBl}_1(\mathsf{Reach}(S_R))) = 1$.*

5 Parity: Randomized Finite-Memory Strategies

In this section we show that randomized finite-memory almost-sure strategies exist for parity objectives. Let $\Omega : S \mapsto \{0, \ldots, k\}$ be the parity index function.

We consider the case when $k = 2d$ for some d, and the case when $k = 2d - 1$, for some d can be proved using similar arguments. Given a timed game structure $\mathfrak{T}$, a set $X \subsetneq S$, and a parity function $\Omega : S \mapsto \{0, \ldots, 2d\}$, with $d > 0$, let $\langle \mathfrak{T}', \Omega' \rangle = \mathsf{ModifyEven}(\mathfrak{T}, \Omega, X)$ be defined as follows: (a) the state space S' of $\mathfrak{T}'$ is $\{s^\perp\} \cup S \setminus X$, where $s^\perp \notin S$; (b) $\Omega'(s^\perp) = 2d - 2$, and $\Omega' = \Omega$ otherwise; (c) $\Gamma_i'(s) = \Gamma_i(s)$ for $s \in S \setminus X$, and $\Gamma_i'(s^\perp) = \Gamma_i(s^\perp) = \mathbb{R}_{\geq 0} \times \perp$; and (d) $\delta'(s, m) = \delta(s, m)$ if $\delta(s, m) \notin S \setminus X$, and $\delta'(s, m) = s^\perp$ otherwise. We will use the function $\mathsf{ModifyEven}$ to play timed games on a subset of the original structure. The extra state, and the modified transition function are to ensure well-formedness of the reduced structure. We will now obtain receptive strategies for player 1 for the objective $\mathsf{Parity}(\Omega)$ using winning strategies for reachability and safety objectives. We consider the following procedure.

1. $\mathfrak{T}_0 = \mathfrak{T}$, and $i := 0$;
2. Compute $X_i = \mathsf{Sure}_1^{\mathfrak{T}_i}(\Diamond(\Omega^{-1}(2d)))$.
3. Let $\langle \mathfrak{T}_i', \Omega' \rangle = \mathsf{ModifyEven}(\mathfrak{T}_i, \Omega, X_i)$; and let $Y_i = \mathsf{Sure}_1^{\mathfrak{T}_i'}(\mathsf{Parity}(\Omega'))$. Let $L_i = S_i \setminus Y_i$, where S_i is the set of states of $\mathfrak{T}_i$.
4. Compute $Z_i = \mathsf{Sure}_1^{\mathfrak{T}_i}(\Box(S_i \setminus L_i))$.
5. Let $(\mathfrak{T}_{i+1}, \Omega) = \mathsf{ModifyEven}(\mathfrak{T}, \Omega, S \setminus Z_i)$ and $i := i + 1$.
6. Go to step 2, unless $Z_{i-1} = S_i$.

It can be shown that (a) when the above procedure terminates, then we have $(S \setminus Z_i) \cap \mathsf{Sure}_1^{\mathfrak{T}}(\mathsf{Parity}(\Omega)) = \emptyset$, and (b) there is a randomized, finite-memory, receptive, region almost-sure winning strategy for every state in Z_i, provided that player 1 has access to imprecise clock events such that between any two events, some time more than Δ passes for a fixed real $\Delta > 0$. Player 1 only uses the clock events to observe the passage of some amount of time greater than Δ (it does not have access to the exact amount of time that passes). The details of the winning strategy can be found in [8].

Theorem 4. *Let $\mathfrak{T}$ be a timed automaton game, and let Ω be a region parity index function. Suppose that player 1 has access to imprecise clock events such that between any two events, some time more than Δ passes for a fixed real $\Delta > 0$. Then, player 1 has a randomized, finite-memory, receptive, region strategy π_1 such that for all states $s \in \mathsf{Sure}_1(\mathsf{Parity}(\Omega))$, the following assertions hold: (a) for all receptive strategies π_2 of player 2 we have $\mathrm{Pr}_s^{\pi_1, \pi_2}(\mathsf{Parity}(\Omega)) = 1$; and (b) for all strategies π_2 of player 2 we have $\mathrm{Pr}_s^{\pi_1, \pi_2}(\mathsf{TimeDivBl}_1(\mathsf{Parity}(\Omega))) = 1$.*

Finally, the following theorem shows that though randomization can get rid of infinite memory with respect to almost-sure winning, it does not help to win in more states, and hence that the set of sure and almost-sure winning states coincide for ω-regular objectives.

Theorem 5. *Consider a timed automaton game and an ω-regular objective Φ. For every state $s \notin \mathsf{Sure}_1^{\mathfrak{T}}(\Phi)$, every real $\varepsilon > 0$, and every randomized strategy $\pi_1 \in \Pi_1$ for player 1, there is a pure strategy $\pi_2^\varepsilon \in \Pi_2^{\mathsf{pure}}$ for player 2 such that $\mathrm{Pr}_s^{\pi_1, \pi_2^\varepsilon}(\mathsf{TimeDivBl}_1(\Phi)) \leq \varepsilon$.*

References

1. B. Adler, L. de Alfaro, and M. Faella. Average reward timed games. In *FORMATS 05*, LNCS 3829, pages 65–80. Springer, 2005.
2. R. Alur and D.L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126(2):183–235, 1994.
3. R. Alur and T.A. Henzinger. Modularity for timed and hybrid systems. In *CONCUR 97*, LNCS 1243, pages 74–88. Springer, 1997.
4. E. Asarin and O. Maler. As soon as possible: Time optimal control for timed automata. In *HSCC 99*, LNCS 1569, pages 19–30. Springer, 1999.
5. P. Bouyer, F. Cassez, E. Fleury, and K.G. Larsen. Optimal strategies in priced timed game automata. In *FSTTCS 04*, LNCS 3328, pages 148–160. Springer, 2004.
6. P. Bouyer, D. D'Souza, P. Madhusudan, and A. Petit. Timed control with partial observability. In *CAV 03*, LNCS 2725, pages 180–192. Springer, 2003.
7. F. Cassez, A. David, E. Fleury, K.G. Larsen, and D. Lime. Efficient on-the-fly algorithms for the analysis of timed games. In *CONCUR 05*, LNCS 3653, pages 66–80. Springer, 2005.
8. K. Chatterjee, T.A. Henzinger, and V.S. Prabhu. Trading infinite memory for uniform randomness in timed games. Technical Report UCB/EECS-2008-4, EECS Department, University of California, Berkeley, Jan 2008.
9. L. de Alfaro, M. Faella, T.A. Henzinger, R. Majumdar, and M. Stoelinga. The element of surprise in timed games. In *CONCUR 03*, LNCS 2761, pages 144–158. Springer, 2003.
10. L. de Alfaro, T.A. Henzinger, and R. Majumdar. Symbolic algorithms for infinite-state games. In *CONCUR 01*, LNCS 2154, pages 536–550. Springer, 2001.
11. D. D'Souza and P. Madhusudan. Timed control synthesis for external specifications. In *STACS 02*, LNCS 2285, pages 571–582. Springer, 2002.
12. M. Faella, S. La Torre, and A. Murano. Automata-theoretic decision of timed games. In *VMCAI 02*, LNCS 2294, pages 94–108. Springer, 2002.
13. M. Faella, S. La Torre, and A. Murano. Dense real-time games. In *LICS 02*, pages 167–176. IEEE Computer Society, 2002.
14. T.A. Henzinger and P.W. Kopke. Discrete-time control for rectangular hybrid automata. *Theoretical Computer Science*, 221:369–392, 1999.
15. T.A. Henzinger and V.S. Prabhu. Timed alternating-time temporal logic. In *FORMATS 06*, LNCS 4202, pages 1–17. Springer, 2006.
16. O. Maler, A. Pnueli, and J. Sifakis. On the synthesis of discrete controllers for timed systems (an extended abstract). In *STACS 95*, pages 229–242, 1995.
17. A. Pnueli, E. Asarin, O. Maler, and J. Sifakis. Controller synthesis for timed automata. In *Proc. System Structure and Control*. Elsevier, 1998.
18. R. Segala, R. Gawlick, J.F. Søgaard-Andersen, and N.A. Lynch. Liveness in timed and untimed systems. *Inf. Comput.*, 141(2):119–171, 1998.
19. W. Thomas. Languages, automata, and logic. In *Handbook of Formal Languages*, volume 3, Beyond Words, chapter 7, pages 389–455. Springer, 1997.
20. H. Wong-Toi and G. Hoffmann. The control of dense real-time discrete event systems. In *Proc. of 30th Conf. Decision and Control*, pages 1527–1528, 1991.