The Element of Surprise in Timed Games*

Luca de Alfaro[1], Marco Faella[1,2], Thomas A. Henzinger[3], Rupak Majumdar[3], and Mariëlle Stoelinga[1]

[1] Department of Computer Engineering, UC Santa Cruz, USA
[2] Dipartimento di Informatica ed Applicazioni, Università di Salerno, Italy
[3] Department of Electrical Engineering and Computer Sciences, UC Berkeley, USA

Abstract. We consider concurrent two-person games played in real time, in which the players decide both which action to play, and when to play it. Such timed games differ from untimed games in two essential ways. First, players can take each other by surprise, because actions are played with delays that cannot be anticipated by the opponent. Second, a player should not be able to win the game by preventing time from diverging. We present a model of timed games that preserves the element of surprise and accounts for time divergence in a way that treats both players symmetrically and applies to all ω-regular winning conditions. We prove that the ability to take each other by surprise adds extra power to the players. For the case that the games are specified in the style of timed automata, we provide symbolic algorithms for their solution with respect to all ω-regular winning conditions. We also show that for these timed games, memory strategies are more powerful than memoryless strategies already in the case of reachability objectives.

1 Introduction

Games have become a central modeling paradigm in computer science. In synthesis and control, it is natural to view a system and its environment as players of a game that pursue different objectives [Chu63,RW89,PR89]. Similarly, in modular specification and verification it is often appropriate to model the components of a system as individual players that may or may not cooperate, depending on the application [AHK02,AdAHM99]. Such games are played on a state space and proceed in an infinite sequence of rounds. In each round, the players choose actions to play, and the chosen actions determine the successor state. For the synthesis and modular analysis of *real-time* systems, we need to use games where time elapses between actions [MPS95]. In such *timed* games, each player chooses both which action to play, and *when* to play it. Timed games differ from their untimed counterparts in two essential ways. First, players can take each other by surprise, because actions are played with delays that cannot be anticipated by

* Supported in part by the AFOSR MURI grant F49620-00-1-0327, the DARPA grant F33615-C-98-3614, the MARCO grant 98-DT-660, the ONR grant N00014-02-1-0671, the NSF grants CCR-9988172, CCR-0225610, and CCR-0234690, the NSF CAREER award CCR-0132780, and the MIUR grant MEFISTO.

the opponent. Second, a player should not be able to win the game by preventing time from diverging [SGSAL98,AH97]. We present a model of timed games that preserves the element of surprise and accounts for the need of time divergence. We study both the properties of the winning strategies and the algorithms for their construction.

We consider two-player timed games that are played over a possibly infinite state space. In each state, each player chooses, simultaneously and independently of the other player, a move $\langle \Delta, a \rangle$, indicating that the player wants to play the action a after a delay of $\Delta \in \mathbb{R}_{\geq 0}$ time units. A special action, $\perp$, signifies the player's intention to remain idle for the specified time delay. Of the moves chosen by the two players, the one with the smaller delay is carried out and determines the successor state; if the delays are equal, then one of the chosen moves occurs nondeterministically (this models the fact that, in real-time interaction, true contemporaneity cannot be achieved). This process, repeated for infinitely many rounds, gives rise to a *run* of the game. Our definition of moves preserves the element of surprise: a player cannot anticipate when the opponent's action will occur in the current round. This contrasts with many previous definitions of timed games (e.g., [AH97,HHM99,dAHM01b,MPS95,AMPS98]), where players can only either play immediately an action a, or wait for a delay Δ. Such formulations may be simpler and more elegant for timed transition systems (i.e., one-player games), but in the case of two-player formulations, the element of surprise is lost, because after each delay both players have the opportunity to propose a new move. This allows a player to intercept the opponent's move $\langle \Delta, a \rangle$ just before the action a is carried out. We show that the element of surprise gives a distinct advantage to a player. In particular, we prove that there are simple reachability games that can be won under our formulation of moves, but not under the previous "no-surprise" versions.

The objective for a player is given by a set Φ of desired game outcomes. A player achieves this goal if all game outcomes belong to Φ. For a timed game to be physically meaningful, a player should not be able to achieve a goal by stopping the progress of time. For instance, if Φ consists of the set of runs that stay forever in a certain set U of states, and if player 2 has an action to leave U only after a delay of 4, then player 1 should not be able to win by always playing $\langle 0, \perp \rangle$. Therefore, several conditions $WC_i(\Phi)$ have been proposed in the literature to express when player $i \in \{1, 2\}$ wins a timed game with goal Φ.

In [SGSAL98,AH97] the winning condition $WC_1(\Phi)$ is defined to be $\Phi \cap (td \cup \mathit{Blameless}_1)$, where td is the set of runs along which time diverges, and $\mathit{Blameless}_1$ is the set of runs along which player 1 proposes the shorter delay only finitely often. Clearly, player 1 is not responsible if time converges along a run in $\mathit{Blameless}_1$. Informally, the condition states that player 1 must achieve the goal Φ, and moreover, either time diverges or player 1 is blameless for its convergence. This definition works if the goal Φ is a safety property, but not if it is a reachability or, more general, a ω-regular property. To see this, observe that player 1 must achieve the goal even if player 2 stops the progress of time. Consider a game where the goal consists of reaching a set U of states, and where

player 1 has an action leading to U which is always available once time advances beyond 1. Then, player 1 cannot win: player 2 can stop time, preventing the action from ever becoming enabled, and ensuring that no run is in Φ.

In [MPS95], the winning condition $\Phi \cap td$ is proposed. This condition requires player 1 to guarantee time divergence, which is not possible in models where player 2 can block the progress of time. In [dAHS02], this condition is modified to $WC_i^*(\Phi) = (\Phi \cap td) \cup Blameless_i$ for player $i \in \{1, 2\}$. While this is appropriate in the asymmetric setting considered there, the problem in our setting, where both players are treated completely symmetrically, is that the two conditions $WC_1^*(\Phi)$ and $WC_2^*(\neg\Phi)$ are not disjoint (here $\neg\Phi$ is the complementary language of Φ). This means that there are games in which both players can win: for instance, player 1 can ensure $\Phi \cap td$, and player 2 can ensure $Blameless_2$. Other works on timed games (e.g., [AMPS98,FLM02]) have avoided the issue of time divergence altogether by putting syntactic constraints on the game structures.

We define timed games and their winning conditions in a completely symmetric fashion, and in a way that works for all goals (in particular for all ω-regular goals) and ensures that players can win only by playing in a physically meaningful way. The winning conditions we propose are $WC_i(\Phi) = (\Phi \cap td) \cup (Blameless_i \setminus td)$, for $i \in \{1, 2\}$. These winning conditions imply that $WC_1(\Phi) \cap WC_2(\neg\Phi)$ is empty, ensuring that at most one player can win. Note that there are runs that belong neither to $WC_1(\Phi)$ nor to $WC_2(\neg\Phi)$: this contrasts with the traditional formulation of untimed games, where runs are either winning for a player with respect to a goal, or winning for the opponent with respect to the complementary goal. We argue that the lack of run-level determinacy is unavoidable in timed games. To see this, consider a run $\bar{r}$ along which both players take turns in proposing moves with delay 0, thus stopping the progress of time. If we somehow assign this run to be winning for a player, say player 1, then it would be possible to construct games in which the moves with delay 0 are the *only* moves available, and in which player 1 could nevertheless win. This would go against our intention that a player can win only in a physically meaningful way. The lack of run-level determinacy also implies that there are states from which neither player can win.

The form of the winning conditions for timed games have other important implications. We show that to win with respect to a reachability goal, in contrast to the untimed case, strategies with memory may be required. For safety goals, however, memoryless strategies suffice also in the timed case. We prove several additional structural properties of the winning strategies for timed games. For instance, we define a class of *persistent* strategies, in which players do not change their mind about the time of future moves when interrupted by a $\langle \Delta, \bot \rangle$ move of the opponent. We show that persistent strategies always suffice to win games, for all possible goals.

While we define timed games at first semantically, we also offer a timed-automaton-style [AD94] syntax for a specific class of timed games. We show that for these *timed automaton games* the winning states with respect to any ω-regular goal can be computed by a symbolic algorithm that iterates a control-

lable predecessor operator on clock regions. In particular, we prove that timed automaton games can be won using *region strategies*, where the players need only remember the history of the game as a sequence of regions, rather than more precisely, as a sequence of states. Furthermore, the problem of solving these games is shown to be, as expected [AH97], complete for EXPTIME.

2 Timed Games

2.1 Timed Game Structures

A *timed game structure* is a tuple $\mathcal{G} = (S, Acts_1, Acts_2, \Gamma_1, \Gamma_2, \delta)$, where

- S is a set of states.
- $Acts_1$ and $Acts_2$ are two disjoint sets of actions for player 1 and player 2, respectively. We assume that $\bot \notin Acts_i$ and write $Acts_i^\bot = Acts_i \cup \{\bot\}$. The set of moves of player i is given by $M_i = \mathbb{R}_{\geq 0} \times Acts_i^\bot$.
- For $i = 1, 2$, the function $\Gamma_i : S \mapsto 2^{M_i} \setminus \emptyset$ is an enabling condition, which assigns to each state s a set $\Gamma_i(s)$ of moves available to player i in that state.
- $\delta : S \times (M_1 \cup M_2) \mapsto S$ is a destination function that, given a state and a move of either player, determines the next state in the game.

We require that the move $\langle 0, \bot \rangle$ is always enabled and does not leave the state: $\langle 0, \bot \rangle \in \Gamma_i(s)$ and $\delta(s, \langle 0, \bot \rangle) = s$ for all $s \in S$. Similarly to [Yi90], we require for all $0 \leq \Delta' \leq \Delta$ and $a \in Acts_i^\bot$, that (1) $\langle \Delta, a \rangle \in \Gamma_i(s)$ if and only if $\langle \Delta', \bot \rangle \in \Gamma_i(s)$ and $\langle \Delta - \Delta', a \rangle \in \Gamma_i(\delta(s, \langle \Delta', \bot \rangle))$, and (2) if $\delta(s, \langle \Delta', \bot \rangle) = s'$, and $\delta(s', \langle \Delta - \Delta', a \rangle) = s''$, then $\delta(s, \langle \Delta, a \rangle) = s''$.

Intuitively, at each state $s \in S$, player 1 chooses a move $\langle \Delta_1, a_1 \rangle \in \Gamma_1(s)$, and simultaneously and independently, player 2 chooses a move $\langle \Delta_2, a_2 \rangle \in \Gamma_2(s)$. If $\Delta_1 < \Delta_2$, then the move $\langle \Delta_1, a_1 \rangle$ is taken; if $\Delta_2 < \Delta_1$, then the move $\langle \Delta_2, a_2 \rangle$ is taken. If $\Delta_1 = \Delta_2$, then the game takes nondeterministically one of the two moves $\langle \Delta_1, a_1 \rangle$ or $\langle \Delta_2, a_2 \rangle$. Formally, we define the *joint destination function* $\widetilde{\delta} : S \times M_1 \times M_2 \mapsto 2^S$ by

$$\widetilde{\delta}(s, \langle \Delta_1, a_1 \rangle, \langle \Delta_2, a_2 \rangle) = \begin{cases} \{\delta(s, \langle \Delta_1, a_1 \rangle)\} & \text{if } \Delta_1 < \Delta_2, \\ \{\delta(s, \langle \Delta_2, a_2 \rangle)\} & \text{if } \Delta_1 > \Delta_2, \\ \{\delta(s, \langle \Delta_1, a_1 \rangle), \delta(s, (\Delta_2, a_2))\} & \text{if } \Delta_1 = \Delta_2. \end{cases}$$

The time elapsed when moves $m_1 = \langle \Delta_1, a_1 \rangle$ and $m_2 = \langle \Delta_2, a_2 \rangle$ are played is given by $delay(m_1, m_2) = \min(\Delta_1, \Delta_2)$. For $i \in \{1, 2\}$, the boolean predicate $bl_i(s, m_1, m_2, s')$ holds if player i is responsible for the state change from s to s'. Formally, denoting with $\sim i = 3 - i$ the opponent of player i, we define $bl_i(s, m_1, m_2, s')$ iff both $\Delta_i \leq \Delta_{\sim i}$ and $s' = \delta(s, m_i)$. Note that both $bl_1(s, m_1, m_2, s')$ and $bl_2(s, m_1, m_2, s')$ may hold at the same time.

An *infinite run* (or simply a *run*) of the timed game structure $\mathcal{G}$ is a sequence $s_0, \langle m_1^1, m_1^2 \rangle, s_1, \langle m_2^1, m_2^2 \rangle, s_2, \ldots$ such that $s_k \in S$, $m_{k+1}^1 \in \Gamma_1(s_k)$, $m_{k+1}^2 \in \Gamma_2(s_k)$, and $s_{k+1} \in \widetilde{\delta}(s_k, m_{k+1}^1, m_{k+1}^2)$ for all $k \geq 0$. A *finite run* $\bar{r}$

is a finite prefix of a run that terminates at a state s; we then set $last(\bar{r}) = s$. We denote by *FRuns* the set of all finite runs of the game structure, and by *Runs* the set of its infinite runs. A finite or infinite run $\bar{r} = s_0, \langle m_1^1, m_1^2\rangle, s_1, \ldots$ induces a *trace* $states(\bar{r}) = s_0, s_1, \ldots$ of states occurring in $\bar{r}$. A state s' is *reachable* from another state s if there exist a finite run $s_0, \langle m_1^1, m_1^2\rangle, s_1, \ldots, s_n$ such that $s_0 = s$ and $s_n = s'$.

A *strategy* π_i for player $i \in \{1, 2\}$ is a mapping $\pi_i : FRuns \mapsto M_i$ that associates with each finite run $s_0, \langle m_1^1, m_1^2\rangle, s_1, \ldots, s_k$ the move $\pi_i(s_0, \langle m_1^1, m_1^2\rangle, s_1, \ldots, s_k)$ to be played at s_k. We require that the strategy only selects enabled moves, that is, $\pi_i(\bar{r}) \in \Gamma_i(last(\bar{r}))$ for all $\bar{r} \in FRuns$. For $i \in \{1, 2\}$, let Π_i denote the set of all player i strategies, and $\Pi = \Pi_1 \cup \Pi_2$ the set of all strategies. For all states $s \in S$ and strategies $\pi_1 \in \Pi_1$ and $\pi_2 \in \Pi_2$, we define the set of *outcomes* $Outcomes(s, \pi_1, \pi_2)$ as the set of all runs $s_0, \langle m_1^1, m_1^2\rangle, s_1, \ldots$ such that $s_0 = s$, and for all $k \geq 0$ and $i = 1, 2$, we have $\pi_i(s_0, \langle m_1^1, m_1^2\rangle, s_1, \ldots, s_k) = m_{k+1}^i$. Note that in our timed games, two strategies and a start state yield a *set* of outcomes, because if the players propose moves with the same delay, a nondeterministic choice between the two moves is made. According to this definition, strategies can base their choices on the entire history of the game, consisting of both past states and moves. In Proposition 1 we show that, to win the game, strategies need only consider past states.

2.2 Timed Goals and Timed Winning Conditions

We consider winning conditions given by sets of infinite traces. A *goal* Φ is a subset of S^ω; we write $[\Phi]_r = \{\bar{r} \in Runs \mid states(\bar{r}) \in \Phi\}$. We write $\neg\Phi$ for the set $S^\omega \setminus \Phi$. We often use linear-time temporal logic formulas to specify goals; the propositional symbols of the formula consist of sets of states of the timed game [MP91]. We distinguish between the goal of a player and the corresponding *winning condition*. The goal represents the control objective that the player must attain; for instance, staying forever in a region of "safe" states. To win the game, however, a player must not only attain this goal, but also make sure that this is done in a physically meaningful way: this is encoded by the winning condition. To this end, we define the set of *time divergent runs* td as the set of all runs $s_0, \langle m_1^1, m_1^2\rangle, s_1, \langle m_2^1, m_2^2\rangle, s_2, \ldots$ such that $\sum_{k=1}^{\infty} delay(m_k^1, m_k^2) = \infty$. For $i \in \{1, 2\}$, we define the set of *player* i *blameless runs* $Blameless_i$ as the set of all runs in which player i plays first (proposes a shorter delay) only finitely many times. Formally, $Blameless_i$ consists of all runs $s_0, \langle m_1^1, m_1^2\rangle, s_1, \ldots$ such that there exists an $n \in \mathbb{N}$ with $\neg bl_i(s_k, m_{k+1}^1, m_{k+1}^2, s_{k+1})$ for all $k \geq n$. Corresponding to the goal Φ, we define the following winning condition:

$$WC_i(\Phi) : \ (td \cap [\Phi]_r) \cup (Blameless_i \setminus td).$$

Informally, this condition states that if time diverges, the goal must be met, and if time does not diverge, the player must be blameless.

Given a goal Φ and a state $s \in S$, we say that player i *wins* from s the game with goal Φ, or equivalently, wins from s the game with winning condition

$WC_i(\Phi)$, if there exists a player i strategy $\pi_i \in \Pi_i$ such that for all opposing strategies $\pi_{\sim i} \in \Pi_{\sim i}$, we have $Outcomes(s, \pi_1, \pi_2) \subseteq WC_i(\Phi)$. In that case, $\pi_i \in \Pi_i$ is called a *winning strategy*. Given a goal Φ, we let $\langle i \rangle \Phi$ be the states from which player i can win the game with goal Φ. A state s is *well-formed* if for every state s' reachable from s, and each player $i \in \{1, 2\}$, we have $s' \in \langle i \rangle S^\omega$. States that are not well-formed are "pathological": if a player cannot win the goal S^ω, then he cannot ensure that the game outcomes are physically meaningful.

3 Timed Automaton Games

In this section, we introduce *timed automaton games*, a syntax derived from timed automata [AD94] for representing timed games. As in timed automata, a finitely specified timed automaton game usually represents a timed game with infinitely many states. A *clock condition* over a set C of clocks is a boolean combination of formulas of the form $x \preceq c$ or $x - y \preceq c$, where c is an integer, $x, y \in C$, and $\preceq$ is either $<$ or $\leq$. We denote the set of all clock conditions over C by $ClkConds(C)$. A *clock valuation* is a function $\kappa : C \mapsto \mathbb{R}_{\geq 0}$, and we denote by $K(C)$ the set of all clock valuations for C.

A *timed automaton game* is a tuple $\mathcal{A} = (Q, C, Acts_1, Acts_2, E, \theta, \rho, Inv_1, Inv_2)$, where:

- Q is a finite set of locations.
- C is a finite set of clocks which includes the unresettable clock z, which measures the time since the start of the game.
- $Acts_1$ and $Acts_2$ are two disjoint, finite sets of actions for player 1 and player 2, respectively.
- $E \subseteq Q \times (Acts_1 \cup Acts_2) \times Q$ is an edge relation.
- $\theta : E \mapsto ClkConds(C)$ is a mapping that associates with each edge a clock condition that specifies when the edge can be traversed. We require that for all $(q, a, q_1), (q, a, q_2) \in E$ with $q_1 \neq q_2$, the conjunction $\theta(q, a, q_1) \wedge \theta(q, a, q_2)$ is unsatisfiable. In other words, the game move and clock values determine uniquely the successor location.
- $\rho : E \mapsto 2^{C \setminus \{z\}}$ is a mapping that associates with each edge the set of clocks to be reset when the edge is traversed.
- $Inv_1, Inv_2 : Q \to ClkConds(C)$ are two functions that associate with each location an invariant for player 1 and 2, respectively.

Given a clock valuation $\kappa : C \mapsto \mathbb{R}_{\geq 0}$ and $\Delta \in \mathbb{R}_{\geq 0}$, we denote by $\kappa + \Delta$ the valuation defined by $(\kappa + \Delta)(x) = \kappa(x) + \Delta$ for all clocks $x \in C$. The clock valuation $\kappa : C \mapsto \mathbb{R}_{\geq 0}$ *satisfies* the clock constraint $\alpha \in ClkConds(C)$, written $\kappa \models \alpha$, if the condition α holds when the clocks have the values specified by κ. For a subset $D \subseteq C$ of clocks, $\kappa[D := 0]$ denotes the valuation defined by $\kappa[D := 0](x) = 0$ if $x \in D$, and by $\kappa[D := 0](x) = \kappa(x)$ otherwise.

The timed automaton game $\mathcal{A}$ induces a timed game structure $[\![\mathcal{A}]\!]$, whose states consist of a location of $\mathcal{A}$ and a clock valuation over C. The idea is the following. A player i move $\langle \Delta, \perp \rangle$ is enabled in state $\langle q, \kappa \rangle$ if either $\Delta = 0$ or

the invariant $Inv_i(q)$ holds continuously when we let Δ time units pass, that is, $\kappa + \Delta' \models Inv_i(q)$ for all $\Delta' \leq \Delta$. Taking the move $\langle \Delta, \bot \rangle$ leads to the state $\langle q, \kappa + \Delta \rangle$. For $a \in Acts_i$, the move $\langle \Delta, a \rangle$ is enabled in $\langle q, \kappa \rangle$ if (1) the invariant $Inv_i(q)$ holds continuously when we let Δ time units pass, (2) there is a transition (q, a, q') in E which is enabled in the state $\langle q, \kappa + \Delta \rangle$, and (3) the invariant $Inv_i(q')$ holds when the game enters location q'. The move $\langle \Delta, a \rangle$ leads to the state $\langle q', \kappa' \rangle$, where κ' is obtained from $\kappa + \Delta$ by resetting all clocks in $\rho(q, a, q')$.

Formally, the timed automaton game $\mathcal{A} = (Q, C, Acts_1, Acts_2, E, \theta, \rho, Inv_1, Inv_2)$ induces the timed game structure $[\![\mathcal{A}]\!] = (S, Acts_1, Acts_2, \Gamma_1, \Gamma_2, \delta)$. Here, $S = Q \times K(C)$ and for each state $\langle q, \kappa \rangle \in S$, the set $\Gamma_i(\langle q, \kappa \rangle)$ is given by:

$$\begin{aligned}\Gamma_i(\langle q, \kappa \rangle) = \{\langle \Delta, a \rangle \in M_i \mid \forall \Delta' \in [0, \Delta] \,.\, \kappa + \Delta' \models Inv_i(q) \wedge \\ (a \neq \bot \Rightarrow \exists q' \in Q \,.\, ((q, a, q') \in E \wedge (\kappa + \Delta) \models \theta(q, a, q') \wedge \\ (\kappa + \Delta)[\rho(q, a, q') := 0] \models Inv_i(q')))\} \quad \cup \quad \{\langle 0, \bot \rangle\}.\end{aligned}$$

The destination function δ is defined by $\delta(\langle q, \kappa \rangle, \langle \Delta, \bot \rangle) = \langle q, \kappa + \Delta \rangle$, and for $a \in Acts_1 \cup Acts_2$, by $\delta(\langle q, \kappa \rangle, \langle \Delta, a \rangle) = \langle q', \kappa' \rangle$, where q' is the unique location such that $(q, a, q') \in E$ and $(\kappa + \Delta) \models \theta(q, a, q')$, and $\kappa' = (\kappa + \Delta)[\rho(q, a, q') := 0]$. A state, a run, and a player i strategy of $\mathcal{A}$ are, respectively, a state, a run, and a player i strategy of $[\![\mathcal{A}]\!]$. We say that player i wins the goal $\Phi \subseteq S^\omega$ from state $s \in S$ in $\mathcal{A}$ if he wins Φ from s in $[\![\mathcal{A}]\!]$. We say that s is *well-formed* in $\mathcal{A}$ if it is so in $[\![\mathcal{A}]\!]$.

Regions. Timed automaton games, similarly to timed automata, can be analyzed with the help of an equivalence relation of finite index on the set of states. Given a timed automaton game $\mathcal{A}$, for each clock $x \in C$, let c_x be the largest constant in the guards and invariants of $\mathcal{A}$ that involve x, where $c_x = 0$ if x does not occur in any guard or invariant of $\mathcal{A}$. Two clock valuations κ_1, κ_2 are *clock equivalent* if (1) for all $x \in C$, either $\lfloor \kappa_1(x) \rfloor = \lfloor \kappa_2(x) \rfloor$ or both $\lfloor \kappa_1(x) \rfloor > c_x$ and $\lfloor \kappa_2(x) \rfloor > c_x$, (2) the ordering of the fractional parts of the clock variables in the set $\{z\} \cup \{x \in C \mid \kappa_1(x) < c_x\}$ is the same in κ_1 and κ_2, and (3) for all $x \in (\{z\} \cup \{y \in C \mid \kappa_1(y) < c_y\})$, the clock value $\kappa_1(x)$ is an integer if and only if $\kappa_2(x)$ is an integer. A *clock region* is a clock equivalence class, and we write $[\kappa]$ for the clock equivalence class of the clock valuation κ. Two states $\langle q_1, \kappa_1 \rangle$ and $\langle q_2, \kappa_2 \rangle$ are *region equivalent*, written $\langle q_1, \kappa_1 \rangle \equiv \langle q_2, \kappa_2 \rangle$, if (1) $q_1 = q_2$ and (2) κ_1 and κ_2 are clock equivalent. A *region* is an equivalence class with respect to $\equiv$; we write $[s]$ for the region containing state s.

4 Structural Properties of Winning Strategies

We now consider structure theorems for strategies in timed automaton games. Throughout this section, a_1 is an action for player 1, and a_2 one for player 2. For a location p in a timed automaton game $\mathcal{A}$ with clock set C, we let $\Diamond p =$

$\Diamond\{\langle p, \kappa\rangle \mid \kappa \in K(C)\}$ and $\Box p = \Box\{\langle p, \kappa\rangle \mid \kappa \in K(C)\}$.[1] Moreover, $\mathbf{0}$ denotes the valuation that assigns 0 to all clocks in C.

Determinacy. A class $\mathcal{C}$ of timed game structures is *strongly determined* (respectively, *weakly determined*) for a class $\mathcal{F}$ of goals if the following holds for every structure $\mathcal{G} \in \mathcal{C}$, every goal $\Phi \in \mathcal{F}$, all well-formed states s, and each player $i \in \{1, 2\}$: if player i cannot win $WC_i(\Phi)$ from s, then there exists a player $\sim i$ strategy $\pi_{\sim i} \in \Pi_{\sim i}$ such that for all player i strategies $\pi_i \in \Pi_i$, we have $Outcomes(s, \pi_1, \pi_2) \cap WC_{\sim i}(\neg\Phi) \neq \emptyset$ (respectively, $Outcomes(s, \pi_1, \pi_2) \not\subseteq WC_i(\Phi)$). Note that this condition is trivially false for non-well-formed states, because one player cannot win the goal S^ω, and the other player surely cannot win the goal $\emptyset$. We let the class of reachability goals be all goals of the form $\Diamond T$.

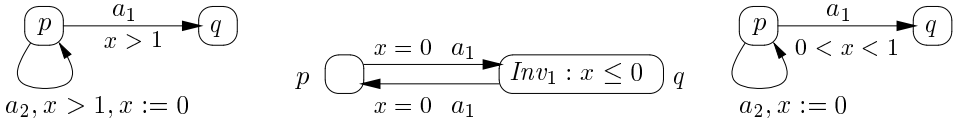
Theorem 1 *The timed automaton games (and hence, the timed game structures) are neither weakly, nor strongly, determined for the class of reachability goals.*

The following example exhibits a timed automaton game and a goal Φ such that player 1 cannot win $\langle 1\rangle\Phi$, but player 2 does not have a strategy to enforce $WC_2(\neg\Phi)$ (strong) or $\neg\, WC_1(\Phi)$ (weak), even if player 2 can use the nondeterministic choices to his advantage.

Example 1 Consider Figure 1(a). It is clear that player 1 does not have a winning strategy for $WC_1(\Diamond q)$ from state $\langle p, \mathbf{0}\rangle$. To prove that this game is not strongly determined, we show that no matter which strategy π_2 is played by player 2, player 1 always has a strategy π_1 such that $Outcomes(\langle p, \mathbf{0}\rangle, \pi_1, \pi_2) \cap WC_2(\neg\Diamond q) = \emptyset$. If π_2 proposes a delay $\Delta_2 > 1$, then π_1 plays the move $\langle \Delta_1, a_1\rangle$ for $\Delta_1 = 1 + (\Delta_2 - 1)/2$; if π_2 proposes a delay $\Delta_2 \leq 1$, then π_1 proposes move $\langle 1, \bot\rangle$. Let $\bar{r} \in Outcomes(\langle p, \mathbf{0}\rangle, \pi_1, \pi_2)$. Then, either $\bar{r}$ contains a player 2 move with a positive delay, in which case q is reached, or player 2 plays $\langle 0, \bot\rangle$ moves forever and is not blameless, i.e., $\bar{r} \notin Blameless_2$. In either case, $\bar{r} \notin WC_2(\neg\Diamond q)$. In a similar way, one shows that the game is not weakly determined.

Memoryless Strategies. Memoryless strategies are strategies that only depend on the last state of a run. Formally, a strategy $\pi \in \Pi$ is *memoryless* if, for all $\bar{r}, \bar{r}' \in FRuns$, we have that $last(\bar{r}) = last(\bar{r}')$ implies $\pi(\bar{r}) = \pi(\bar{r}')$. For $i \in \{1, 2\}$, we often treat a memoryless strategy π_i for player i as a function in $S \mapsto M_i$ by writing $\pi_i(last(\bar{r}))$ instead of $\pi_i(\bar{r})$. In the untimed case, memoryless strategies are sufficient to win safety and reachability games. In timed games, memoryless strategies suffice to win safety games, i.e., goals of the form $WC_i(\Box T)$; however, winning strategies in reachability games (goals of the form $WC_i(\Diamond T)$) in general do require memory.

[1] We use the standard LTL operators $\Diamond T$ and $\Box T$ to denote, respectively, the set of traces that eventually reach some state in T, and the set of traces that always stay in T [MP91].



(a) Undetermined. (b) Memory needed. (c) Surprise needed.

Fig. 1. Games with winning condition $WC_1(\Diamond q)$, where $a_1 \in Acts_1$ and $a_2 \in Acts_2$.

Theorem 2

1. *For every well-formed state s of a timed game structure $\mathcal{G}$, and every set T of states of $\mathcal{G}$, if player i has a strategy to win $WC_i(\Box T)$ from s, then player i has a memoryless strategy for winning $WC_i(\Box T)$ from s.*
2. *There exists a timed automaton game $\mathcal{A}$, a state s of $\mathcal{A}$, and a set T of states of $\mathcal{A}$ such that player i has a strategy to win $WC_i(\Diamond T)$ from s, but no memoryless strategy for winning $WC_i(\Diamond T)$ from s.*

The following example proves part 2.

Example 2 Consider the game in Figure 1(b). Player 1 has a winning strategy for $WC_1(\Diamond q)$ from $\langle p, \mathbf{0}\rangle$, but not a memoryless one: to win, he needs to remember whether q has been visited already. If so, then he has to let time pass, and if not, a visit to q has to be made before letting time pass. Let $\pi : S \mapsto M_1$ be a memoryless strategy for player 1. It is easy to see that, if $\pi(\langle p, \mathbf{0}\rangle) = \langle \Delta, \bot\rangle$, then q will never be reached, and otherwise, if $\pi(\langle p, \mathbf{0}\rangle) = \langle 0, a_1\rangle$, then time will not progress, while π does not ensure that player 1 is blameless. Hence, player 1 cannot win $WC_1(\Diamond q)$ with a memoryless strategy.

No-Surprise Strategies. A no-surprise strategy is a strategy that plays only two kinds of moves: either time steps (action $\bot$, with any delay), or actions with delay 0. Formally, a strategy $\pi \in \Pi$ is *no-surprise* if for all $\bar{r} \in FRuns$ either $\pi(\bar{r}) = \langle 0, a\rangle$ with $a \in Acts$, or $\pi(\bar{r}) = \langle \Delta, \bot\rangle$ with $\Delta \in \mathbb{R}_{\geq 0}$. The following theorem shows that there are cases where surprise is necessary to win, even when the goal is a reachability property, and player 2 is restricted to no-surprise strategies as well.

Theorem 3 *There is a timed automaton game $\mathcal{A}$, a state s of $\mathcal{A}$, and a goal Φ such that player 1 has a strategy to win $WC_1(\Phi)$ from s, but there is no no-surprise strategy $\pi_1 \in \Pi_1$ such that for all no-surprise strategies $\pi_2 \in \Pi_2$, we have $Outcomes(s, \pi_1, \pi_2) \subseteq WC_1(\Phi)$.*

The proof is given by the following example.

Example 3 Consider Figure 1(c). Player 1 has a strategy to win $WC_1(\Diamond q)$ from state $\langle p, \mathbf{0}\rangle$. For instance, he can play $\pi_1(\bar{r}) = \langle \frac{1}{2^{n+1}}, a_1\rangle$ if $\bar{r}$ contains n visits

to p and it ends in $\langle p, \kappa \rangle$ with $\kappa(x) + \frac{1}{2^{n+1}} < 1$; and play $\pi_1(\bar{r}) = \langle 1, \bot \rangle$ in all other cases. Let $\pi_2 \in \Pi_2$ and $\bar{r}$ be a run in $Outcomes(\langle p, \mathbf{0} \rangle, \pi_1, \pi_2)$. If one of his moves $\langle \frac{1}{2^n}, a_1 \rangle$ is taken in $\bar{r}$, then player 1 clearly wins, that is, $\bar{r} \in WC_1(\Diamond q)$. Otherwise, if none of these moves is ever carried out in $\bar{r}$, then player 1 is blameless and, as $\sum_{i=1}^{\infty} \frac{1}{2^i} = 1$, time does not diverge, so $\bar{r} \in WC_1(\Diamond q)$ as well.

However, player 1 does not have a no-surprise strategy to win $WC_1(\Diamond q)$ from $\langle p, \mathbf{0} \rangle$. All no-surprise player-1 strategies π_1 lose against player 2 playing the no-surprise strategy π_2 defined by $\pi_2(\bar{r}) = \langle 0, a_2 \rangle$ if $\bar{r} = \bar{r}' \langle m_1, m_2 \rangle s$ and $m_1 = \langle \Delta, \bot \rangle$; and $\pi_2(\bar{r}) = \langle 1, \bot \rangle$ otherwise. This is because, in order to enable a_1, player 1 has to increase x by taking some move $\langle \Delta, \bot \rangle$ first. However, immediately after he does so, player 2 plays $\langle 0, a_2 \rangle$, thus resetting x. As a result, q is never reached, and both players play infinitely often, so π_1 cannot ensure that player 1 is blameless.

Move Independence. A strategy $\pi \in \Pi$ is *move independent* if, for all $\bar{r}, \bar{r}' \in FRuns$, we have that $states(\bar{r}) = states(\bar{r}')$ implies $\pi(\bar{r}) = \pi(\bar{r}')$. We show that move independent strategies suffice to win a timed automaton game. Note that, for ω-regular goals, this result follows immediately from the strategies derived from the μ-calculus solution for these games; see Section 5.

Proposition 1. *Let $\mathcal{A}$ be a timed automaton game and s be a state of $\mathcal{A}$. For every goal Φ, if player i has a strategy to win $WC_i(\Phi)$ from s, then player i has a move independent strategy for winning $WC_i(\Phi)$ from s.*

Persistence. Persistent strategies are strategies that stick with their choices, even if they are interrupted by a move $\langle \Delta, \bot \rangle$ (or another move with the same effect) of the opponent. Formally, a *persistent* player 1 strategy is a strategy $\pi \in \Pi_1$ such that for all finite runs $\bar{r} = \bar{r}' s \langle m_1, m_2 \rangle s'$ with $m_1 = \langle \Delta_1, a_1 \rangle$, $m_2 = \langle \Delta_2, a_2 \rangle$, and $s' = \delta(s, \langle \Delta_2, \bot \rangle)$, we have (1) if $\Delta_2 < \Delta_1$, then $\pi(\bar{r}) = \langle \Delta_1 - \Delta_2, a_1 \rangle$, and (2) if $a_1 \neq \bot$ and $\Delta_1 = \Delta_2$, then $\pi(\bar{r}) = \langle 0, a_1 \rangle$. The persistent player 2 strategies are defined symmetrically. Consider a finite run $\bar{r} = \bar{r}' s \langle m_1, m_2 \rangle s'$. Assume that, in $\bar{r}' s$, player 1 likes to play the move $\pi(\bar{r}' s) = \langle \Delta_1, a_1 \rangle$, but is interrupted because player 2 plays a move $\langle \Delta_2, \bot \rangle$ with $\Delta_2 \leq \Delta_1$. After $\langle \Delta_2, a_2 \rangle$ has been taken, a persistent strategy requires player 1 to play the portion of his previous move $\langle \Delta_1, a_1 \rangle$ which was not carried out; that is, player 1 must play $\langle \Delta_1 - \Delta_2, a_1 \rangle$, unless $\Delta_1 = \Delta_2$ and $a_1 = \bot$. Persistent strategies suffice to win timed games.

Theorem 4 *Let $\mathcal{G}$ be a timed game structure and s be a state of $\mathcal{G}$. For every goal Φ, if player i has a strategy to win $WC_i(\Phi)$ from s, then player i has a persistent strategy for winning $WC_i(\Phi)$ from s.*

5 Solving Timed Automaton Games

In this section, we show how timed automata games can be solved with respect to ω-regular goals via the equational μ-calculus. We consider a goal that is specified by an parity automaton over the set of locations of the timed automaton game, and based on this, we construct another parity automaton that encodes

the winning condition. Finally, from the automaton that encodes the winning condition we obtain a μ-calculus formula that, evaluated over the timed automaton game, defines the winning states of the game. Since the μ-calculus formula preserves the regions of the timed automaton game, it provides an algorithm for solving timed automaton games.

5.1 Representing Goals and Winning Conditions

Consider a timed automaton game $\mathcal{A}$ with locations Q and clocks C. A goal $\Phi \subseteq (Q \times K(C))^\omega$ of $\mathcal{A}$ is a *location goal* if it is independent of clock valuations; that is, if $\langle q_0, \kappa_0\rangle\langle q_1, \kappa_1\rangle \cdots \in \Phi$, then for all κ_0', κ_1', ..., we have $\langle q_0, \kappa_0'\rangle\langle q_1, \kappa_1'\rangle \cdots \in \Phi$. Since location goals depend only on the sequence of locations, we view, with abuse of notation, a location goal to be a subset of Q^ω. We consider in this section location goals Φ that are *ω-regular* subsets of Q^ω [Tho90]. Such location goals can be specified by means of deterministic parity automata over the alphabet Q [EJ91]. A *parity automaton* (also known as *Rabin-chain automaton*) of order k over the alphabet Σ is a tuple $H = (P, P_0, \Sigma, \tau, \ell, \Omega)$, where P is the set of locations of the automaton, $P_0 \subseteq P$ is the set of initial locations, $\tau : P \mapsto 2^P$ is the transition relation, $\ell : P \mapsto \Sigma$ assigns to each location $p \in P$ a symbol $\ell(p)$ of the alphabet Σ, and $\Omega : P \mapsto \{0, \ldots, 2k-1\}$ assigns to each location $p \in P$ an index $\Omega(p)$.

An *execution* of H from a source location $p_0 \in P$ is an infinite sequence $p_0, p_1, p_2, \ldots$ of automaton locations such that $p_{j+1} \in \tau(p_j)$ for all $j \geq 0$; if $p_0 \in P_0$, then the execution is *initialized*. The execution $\alpha = p_0, p_1, p_2, \ldots$ *generates* the trace $\ell(\alpha) = \ell(p_0), \ell(p_1), \ell(p_2), \ldots$ of symbols of Σ. Given an execution $\alpha = p_0, p_1, p_2, \ldots$, we denote by $\mathit{MaxIndex}(\Omega, \alpha)$ the largest $j \in \{0, \ldots, 2k-1\}$ such that $\Omega(p_i) = j$ for infinitely many i. The execution α is *accepting* if $\mathit{MaxIndex}(\Omega, \alpha)$ is even. The *language* $\mathcal{L}(H)$ is the set of traces $\rho \in \Sigma^\omega$ such that H has an initialized accepting execution α that generates ρ. The automaton H is *deterministic and total* if (1a) for all locations $p_1, p_2 \in P_0$, if $p_1 \neq p_2$, then $\ell(p_1) \neq \ell(p_2)$; (1b) for all symbols $\sigma \in \Sigma$, there is a location $p \in P_0$ such that $\ell(p) = \sigma$; (2a) for all locations $p_1 \in P$ and $p_2, p_3 \in \tau(p_1)$, if $p_2 \neq p_3$, then $\ell(p_2) \neq \ell(p_3)$; (2b) for all locations $p_1 \in P$ and all symbols $\sigma \in \Sigma$, there is a location $p_2 \in \tau(p_1)$ such that $\ell(p_2) = \sigma$. If H is deterministic and total, then we write $\tau(p_1, \sigma)$ for the unique location p_2 with $\ell(p_2) = \sigma$. Deterministic and total parity automata suffice for recognizing all ω-regular languages [Tho90]. We denote by $|H| = |P|$ the *size* of the automaton, measured as its number of locations, and by $|H|_*$ its order k.

Let $\mathcal{A}$ be a timed automaton game with the set Q of locations, and let Φ be a goal that is specified by means of a deterministic and total parity automaton $H_\Phi = (P, P_0, Q, \tau, \ell, \Omega)$ over the alphabet Q such that $\mathcal{L}(H_\Phi) = \Phi$. The first step towards deriving a μ-calculus formula for computing the winning states of $\mathcal{A}$ with respect to Φ represents the conditions *td* and $\mathit{Blameless}_1$ as ω-regular conditions. To this end, we consider an enlarged state space $\widehat{S} = S \times \{\mathrm{T}, \mathrm{F}\}^2$, and an augmented transition relation $\widehat{\delta} : \widehat{S} \times M_1 \times M_2 \mapsto 2^{\widehat{S}}$. Intuitively, in an augmented state $\langle s, \mathit{tick}, \mathit{bl}\rangle \in \widehat{S}$, the component $s \in S$ is a state of the

original game structure $[\![\mathcal{A}]\!]$, *tick* is true if in the last transition the global clock z has crossed an integer boundary, and bl is true if player 1 is to blame for the last transition. Precisely, we let $\langle\langle q', \kappa'\rangle, tick', bl'\rangle \in \widehat{\delta}(\langle\langle q, \kappa\rangle, tick, bl\rangle, m_1, m_2)$ iff $\langle q', \kappa'\rangle \in \delta(\langle q, \kappa\rangle, m_1, m_2)$, $tick' = \textsc{t}$ iff there is $n \in \mathbb{N}$ such that $\kappa(z) \leq n < \kappa'(z)$, and $bl' = \textsc{t}$ iff $bl_1(\langle q, \kappa\rangle, m_1, m_2, \langle q', \kappa'\rangle)$. The set td corresponds to the runs along which *tick* is true infinitely often, and the set $Blameless_1$ corresponds to the runs along which bl is true only finitely often. Once time divergence and blame are thus encoded, the winning condition $WC_1(\Phi)$ can be specified by a parity automaton $H_{WC_1(\Phi)}$ with the alphabet $\widehat{\Sigma} = Q \times \{\textsc{t}, \textsc{f}\}^2$ and language

$$\mathcal{L}(H_{WC_1(\Phi)}) = \left\{ \begin{array}{c} \langle q_0, tick_0, bl_0\rangle, \\ \langle q_1, tick_1, bl_1\rangle, \\ \dots \end{array} \;\middle|\; \begin{array}{c} (q_0, q_1, \dots \in \mathcal{L}(H_\Phi) \wedge \forall k \in \mathbb{N} \,.\, \exists j \geq k \,.\, tick_j) \\ \vee \\ \exists k \in \mathbb{N}. \forall j \geq k \,.\, (\neg bl_j \wedge \neg tick_j) \end{array} \right\} \tag{1}$$

The automaton $H_{WC_1(\Phi)} = (\widehat{P}, \widehat{P}_0, \widehat{\Sigma}, \widehat{\tau}, \widehat{\ell}, \widehat{\Omega})$ is derived from the automaton H_Φ as follows. Let k be the order of H_Φ. We have $\widehat{P} = P \times \{\textsc{t}, \textsc{f}\}^2 \times \{0, \dots, 2k-1\}$; intuitively, a location $\langle p, tick, bl, h\rangle \in \widehat{P}$ is composed of a location $p \in P$, of two boolean symbols representing the value of *tick* and bl at the location, and of an integer h that keeps track of the maximum index of the locations of H_Φ that have been visited between two occurrences of $tick = \textsc{t}$. For $\langle p, tick, bl, h\rangle \in \widehat{P}$, we define $\widehat{\ell}(\langle p, tick, bl, h\rangle) = \langle \ell(p), tick, bl\rangle$, and we let $\langle p, tick, bl, h\rangle \in \widehat{P}_0$ iff $p \in P_0$. For all $p \in P$, $bl \in \{\textsc{t}, \textsc{f}\}$, and $h \in \{0, \dots, 2k-1\}$, we have $\langle p', tick', bl', h'\rangle \in \widehat{\tau}(\langle p, \textsc{f}, bl, h\rangle)$ iff $p' \in \tau(p)$ and $h' = \max\{h, \Omega(p')\}$, and we have $\langle p', tick', bl', h'\rangle \in \widehat{\tau}(\langle p, \textsc{t}, bl, h\rangle)$ iff $p' \in \tau(p)$ and $h' = \Omega(p')$. The index function $\widehat{\Omega} : \widehat{P} \mapsto \{0, \dots, 2k+1\}$ is defined, for all $p \in P$, all $bl \in \{\textsc{t}, \textsc{f}\}$, and all $h \in \{0, \dots, 2k-1\}$, by $\widehat{\Omega}(\langle p, \textsc{f}, \textsc{f}, h\rangle) = 0$, $\widehat{\Omega}(\langle p, \textsc{f}, \textsc{t}, h\rangle) = 1$, and $\widehat{\Omega}(\langle p, \textsc{t}, bl, h\rangle) = h + 2$. For all executions $\widehat{\alpha} = \langle p_0, tick_0, bl_0, h_0\rangle, \langle p_1, tick_1, bl_1, h_1\rangle, \langle p_2, tick_2, bl_2, h_2\rangle, \dots$ of $H_{WC_1(\Phi)}$, let $\alpha = p_0, p_1, p_2, \dots$ be the corresponding execution in H_Φ. We can show that (a) if there are infinitely many j such that $tick_j = \textsc{t}$, then $MaxIndex(\widehat{\Omega}, \alpha) = MaxIndex(\Omega, \alpha) + 2$; (b) if there is $k \in \mathbb{N}$ such that $tick_j = bl_j = \textsc{f}$ for all $j \geq k$, then $MaxIndex(\widehat{\Omega}, \alpha) = 0$; and (c) in all other cases (i.e., when $tick_j$ holds for only finitely many values of j, but bl_j holds for infinitely many values of j), we have $MaxIndex(\widehat{\Omega}, \alpha) = 1$. Together, these facts lead to (1).

Lemma 1 *Given H_Φ, we can construct a deterministic and total parity automaton $H_{WC_1(\Phi)}$ satisfying (1) such that $|H_{WC_1(\Phi)}| = 4 \cdot |H_\Phi| \cdot |H_\Phi|_*$ and $|H_{WC_1(\Phi)}|_* = |H_\Phi|_* + 1$.*

5.2 A μ-calculus Formula for the Winning States

For all $\langle p, tick, bl, h\rangle \in \widehat{P}$, we let $\widehat{\ell}_Q(\langle p, tick, bl, h\rangle) = \ell(p) \in Q$, $\widehat{\ell}_t(\langle p, tick, bl, h\rangle) = tick$, and $\widehat{\ell}_b(\langle p, tick, bl, h\rangle) = bl$. The fixpoint formula ψ_Φ that solves the game with goal Φ is constructed as follows [dAHM01a]. The formula ψ_Φ is composed of blocks $\mathcal{B}_0, \dots, \mathcal{B}_{2k+1}$, where $\mathcal{B}_0$ is the innermost

block and $\mathcal{B}_{2k+1}$ the outermost block. The formula uses the set of variables $\{x_j^{\widehat{p}} \mid \widehat{p} \in \widehat{P}, j \in \{0,\ldots,2k+1\}\} \cup \{y\}$, which take values in 2^S, where S is the set of states of the game structure $\mathcal{A}$. The block $\mathcal{B}_0$ is a ν-block which consists of all equations of the form

$$x_0^{\widehat{p}} = (\widehat{\ell}_Q(\widehat{p}) \times K(C)) \cap \mathit{CPre}_1\left(\bigvee_{\widehat{p}' \in \tau(\widehat{p})} x^{\widehat{p}'}_{\widehat{\Omega}(\widehat{p})} \times \widehat{\ell}_t(\widehat{p}') \times \widehat{\ell}_b(\widehat{p}')\right)$$

for $\widehat{p} \in \widehat{P}$, where C is the set of clocks of $\mathcal{A}$. For $0 < j < 2k+1$, the block $\mathcal{B}_j$ is a μ-block if j is odd, and a ν-block if j is even; in either case it consists of the set of equations $\{x_j^{\widehat{p}} = x_{j-1}^{\widehat{p}} \mid \widehat{p} \in \widehat{P}\}$. The block $\mathcal{B}_{2k+1}$ consists of the set of equations $\{x_{2k+1}^{\widehat{p}} = x_{2k}^{\widehat{p}} \mid \widehat{p} \in \widehat{P}\} \cup \{y = \bigvee_{\widehat{p} \in \widehat{P}_0} x_{2k+1}^{\widehat{p}}\}$. The output variable is y.

The operator $\mathit{CPre}_1 : \widehat{S} \mapsto S$ is the *controllable predecessor operator,* defined by $\exists m_1 \in \Gamma_1(s) \,.\, \forall m_2 \in \Gamma_2(s) \,.\, \widetilde{\delta}(s, m_1, m_2) \in X$. Intuitively, for $s \in S$ and $\widehat{X} \subseteq \widehat{S}$, we have that $s \in \mathit{CPre}_1(\widehat{X})$ if player 1 can force the augmented game to $\widehat{X}$ in one move. As an example, consider the set $\widehat{X} = (X_1 \times \{\textsc{f}\} \times \{\textsc{t}\}) \cup (X_2 \times \{\textsc{f}\} \times \{\textsc{f}\})$ for some $X \subseteq S$. Then, $s \in \mathit{CPre}_1(\widehat{X})$ if player 1 has a move such that, whatever the move played by player 2: either (a) the game proceeds to X_1, the global clock z does not advance beyond an integer boundary ($\mathit{tick} = \textsc{f}$), and player 1 is blamed ($\mathit{bl} = \textsc{t}$); or (b) the game proceeds to X_2, the global clock z does not advance beyond an integer boundary, and player 1 is not blamed. The implementation and properties of operator CPre_1 are discussed below. Note that the formula ψ_Φ depends only on H_Φ, but not on the timed game structure over which it is evaluated (except trivially via the product with $K(C)$, which is simply the set of all clock valuations). Denote by $[\![y]\!]_{\mathcal{A}}^{\psi_\Phi} \subseteq S$ the fixpoint valuation of y over the timed game structure $[\![\mathcal{A}]\!]$. Lemma 2 enables the computation of the winning states of the game with respect to player 1; the winning states with respect to player 2 can be computed in a symmetrical fashion.

Lemma 2 *We have* $\langle 1 \rangle \Phi = [\![y]\!]_{\mathcal{A}}^{\psi_\Phi}$.

5.3 The Controllable Predecessor Operator

The operator CPre_1 can be computed as follows. For $X \subseteq \widehat{S}$, write $X = (X_{\textsc{t}} \times \{\textsc{t}\}) \cup (X_{\textsc{f}} \times \{\textsc{f}\})$, for $X_{\textsc{t}}, X_{\textsc{f}} \subseteq S \times \{\textsc{t}, \textsc{f}\}$. Intuitively, $X_{\textsc{t}}$ (resp. $X_{\textsc{f}}$) represents the portion of X that corresponds to the case where bl is $\textsc{t}$ (resp. $\textsc{f}$). Then, $s \in \mathit{CPre}_1(X)$ if and only if:

$$\begin{aligned}
&\exists \langle \Delta_1, a_1 \rangle \in \Gamma_1(s) \,. \\
&\quad \forall \langle \Delta_2, a_2 \rangle \in \Gamma_2(s) \,. \Big(\Delta_2 \leq \Delta_1 \implies \big(\delta(s, \langle \Delta_2, a_2 \rangle), \mathit{tick}(s, \Delta_2)\big) \in X_{\textsc{f}} \Big) \wedge \\
&\quad \Big(\big(\delta(s, \langle \Delta_1, a_1 \rangle), \mathit{tick}(s, \Delta_1)\big) \in X_{\textsc{t}} \vee \forall \langle \Delta_2, a_2 \rangle \in \Gamma_2(s) \,.\, \Delta_2 < \Delta_1 \Big),
\end{aligned}$$

where $\mathit{tick}(\langle q, \kappa \rangle, \Delta)$ is $\textsc{t}$ iff $\kappa(z) \leq n < \kappa(z) + \Delta$, for some integer n. In words, the above formula states that there is a player 1 action that, played with delay

Δ_1, leads to X_{T}; moreover, all actions of player 2, if played with delay up to Δ_1, lead to X_{F}. The following lemma states that the controllable predecessor operator preserves regions for timed automaton games.

Lemma 3 *For $n \geq 0$, consider $X = \bigcup_{j=1}^{n}(X_j \times \{tick_j\} \times \{bl_j\})$, where for $1 \leq j \leq n$, the set X_j is a region, and $tick_j, bl_j \in \{\mathrm{T}, \mathrm{F}\}$. Then, $CPre_1(X)$ is a union of regions.*

5.4 Putting It All Together

From the constructions of the previous subsections, we obtain the following decidability result for timed automaton games with ω-regular location goals.

Theorem 5 *Consider a timed automaton game $\mathcal{A}$ with the set Q of locations, and a parity automaton H_Φ that specifies a location goal $\Phi \subseteq Q^\omega$. Let C be the set of clocks of $\mathcal{A}$, let $m = |C|$, and let $c = \max\{c_x \mid x \in C\}$. Then, the set of winning states $\langle 1 \rangle \Phi$ can be computed in time $O((|Q| \cdot m! \cdot 2^m \cdot (2c+1)^m \cdot |H_\Phi| \cdot |H_\Phi|_*)^{(|H_\Phi|_*+1)})$.*

Corollary 1. *The problem of solving a timed automaton game for a location goal specified by a parity automaton is EXPTIME-complete.*

EXPTIME-hardness follows from the EXPTIME-hardness for alternating reachability on timed automata [HK99]. Membership in EXPTIME is shown by the exponential-time algorithm outlined above. The algorithm for solving timed automaton games can also be used to simultaneously construct a winning strategy for player 1, as in [dAHM01b]. The winning strategies thus constructed have the following finitary structure. Two finite runs $\bar{r} = s_0, \langle m_1^1, m_1^2 \rangle, s_1, \ldots, s_k$ and $\bar{r}' = s'_0, \langle m_1'^1, m_1'^2 \rangle, s'_1, \ldots, s'_k$ of the same length are *region equivalent*, written $\bar{r} \equiv \bar{r}'$, if for all $0 \leq j \leq k$, we have $[s_j] = [s'_j]$. A strategy is a *region strategy* if, for region equivalent finite runs, it prescribes moves to the same region. Formally, a strategy $\pi \in \Pi$ is a *region strategy* if for all $\bar{r}, \bar{r}' \in FRuns$, we have that $\bar{r} \equiv \bar{r}'$ implies $\delta(last(\bar{r}), \pi(\bar{r})) \equiv \delta(last(\bar{r}), \pi(\bar{r}'))$. Since the $CPre_1$ operator preserves regions, we can show that the strategy constructed by the above algorithm does not distinguish between region equivalent runs, and hence, the constructed strategy is a region strategy.

Theorem 6 *Let $\mathcal{A}$ be a timed automaton game and s a state of $\mathcal{A}$. For every ω-regular location goal Φ, if player i has a strategy to win $WC_i(\Phi)$ from s, then player i has a region strategy for winning $WC_i(\Phi)$ from s.*

References

[AD94] R. Alur and D.L. Dill. A theory of timed automata. *Theor. Comp. Sci.*, 126:183–235, 1994.

[AdAHM99] R. Alur, L. de Alfaro, T.A. Henzinger, and F.Y.C. Mang. Automating modular verification. In *Concurrency Theory*, Lect. Notes in Comp. Sci. 1664, pages 82–97. Springer, 1999.

[AH97] R. Alur and T.A. Henzinger. Modularity for timed and hybrid systems. In *Concurrency Theory*, Lect. Notes in Comp. Sci. 1243, pages 74–88. Springer, 1997.

[AHK02] R. Alur, T.A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *J. ACM*, 49:672–713, 2002.

[AMPS98] E. Asarin, O. Maler, A. Pnueli, and J. Sifakis. Controller synthesis for timed automata. In *Proc. IFAC Symp. System Structure and Control*, pages 469–474. Elsevier, 1998.

[Chu63] A. Church. Logic, arithmetics, and automata. In *Proc. Int. Congress of Mathematicians, 1962*, pages 23–35. Institut Mittag-Leffler, 1963.

[dAHM01a] L. de Alfaro, T.A. Henzinger, and R. Majumdar. From verification to control: Dynamic programs for omega-regular objectives. In *Proc. Symp. Logic in Comp. Sci.*, pages 279–290. IEEE, 2001.

[dAHM01b] L. de Alfaro, T.A. Henzinger, and R. Majumdar. Symbolic algorithms for infinite-state games. In *Concurrency Theory*, Lect. Notes in Comp. Sci. 2154, pages 536–550. Springer, 2001.

[dAHS02] L. de Alfaro, T.A. Henzinger, and M.I.A. Stoelinga. Timed interfaces. In *Embedded Software*, Lect. Notes in Comp. Sci. 2491, pages 108–122. Springer, 2002.

[EJ91] E.A. Emerson and C.S. Jutla. Tree automata, mu-calculus, and determinacy. In *Proc. Symp. Foundations of Comp. Sci.*, pages 368–377. IEEE, 1991.

[FLM02] M. Faella, S. La Torre, and A. Murano. Dense real-time games. In *Proc. Symp. Logic in Comp. Sci.*, pages 167–176. IEEE, 2002.

[HHM99] T.A. Henzinger, B. Horowitz, and R. Majumdar. Rectangular hybrid games. In *Concurrency Theory*, Lect. Notes in Comp. Sci. 1664, pages 320–335. Springer, 1999.

[HK99] T.A. Henzinger and P.W. Kopke. Discrete-time control for rectangular hybrid automata. *Theor. Comp. Sci.*, 221:369–392, 1999.

[MP91] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification*. Springer, 1991.

[MPS95] O. Maler, A. Pnueli, and J. Sifakis. On the synthesis of discrete controllers for timed systems. In *Theor. Aspects of Comp. Sci.*, Lect. Notes in Comp. Sci. 900, pages 229–242. Springer, 1995.

[PR89] A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proc. Symp. Principles of Programming Languages*, pages 179–190. ACM, 1989.

[RW89] P.J.G. Ramadge and W.M. Wonham. The control of discrete-event systems. *IEEE Transactions on Control Theory*, 77:81–98, 1989.

[SGSAL98] R. Segala, G. Gawlick, J. Søgaard-Andersen, and N. Lynch. Liveness in timed and untimed systems. *Info. and Comput.*, 141:119–171, 1998.

[Tho90] W. Thomas. Automata on infinite objects. In J. van Leeuwen, ed., *Handbook Theor. Comp. Sci.*, vol. B, pages 135–191. Elsevier, 1990.

[Yi90] W. Yi. Real-time behaviour of asynchronous agents. In *Concurrency Theory*, Lect. Notes in Comp. Sci. 458, pages 502–520. Springer, 1990.