Probabilistic Robust Timed Games *

Youssouf Oualhadj[1], Pierre-Alain Reynier[2], and Ocan Sankur[3]

[1] Université de Mons (UMONS), Belgium
[2] LIF, Université d'Aix-Marseille & CNRS, France
[3] Université Libre de Bruxelles, Belgium

Abstract. Solving games played on timed automata is a well-known problem and has led to tools and industrial case studies. In these games, the first player (Controller) chooses delays and actions and the second player (Perturbator) resolves the non-determinism of actions. However, the model of timed automata suffers from mathematical idealizations such as infinite precision of clocks and instantaneous synchronization of actions. To address this issue, we extend the theory of timed games in two directions. First, we study the synthesis of *robust* strategies for Controller which should be tolerant to adversarially chosen clock imprecisions. Second, we address the case of a stochastic perturbation model where both clock imprecisions and the non-determinism are resolved randomly. These notions of robustness guarantee the implementability of synthesized controllers. We provide characterizations of the resulting games for Büchi conditions, and prove the EXPTIME-completeness of the corresponding decision problems.

1 Introduction

For real-time systems, timed games are a standard mathematical formalism which can model control synthesis problems under timing constraints. These consist in two-players games played on arenas, defined by timed automata, whose state space consists in discrete locations and continuous clock values. The two players represent the control law and the environment. Since the first theoretical works [2], symbolic algorithms have been studied [10], tools have been developed and successfully applied to several case studies.

Robustness Because model-based techniques rely on abstract mathematical models, an important question is whether systems synthesized in a formalism are implementable in practice. In timed automata, the abstract mathematical semantics offers arbitrarily precise clocks and time delays, while real-world digital systems have response times that may not be negligible, and control software cannot ensure timing constraints exactly, but only up to some error, caused by clock imprecisions, measurement errors, and communication delays. A major challenge is thus to ensure that the synthesized control software is *robust*, *i.e.* ensures the specification even in presence of imprecisions [15].

Following these observations there has been a growing interest in lifting the theory of verification and synthesis to take robustness into account. Model-checking problems were re-visited by considering an unknown perturbation

* This work was partly supported by ANR projects ECSPER (ANR-2009-JCJC-0069) and ImpRo (ANR-2010-BLAN-0317), European project Cassting (FP7-ICT-601148), and the ERC starting grant inVEST (FP7-279499).

parameter to be synthesized for several kinds of properties [19, 12, 7], see also [9]. Robustness is also a critical issue in controller synthesis problems. In fact, due to the infinite precision of the semantics, synthesized strategies may not be realizable in a finite-precision environment; the controlled systems synthesized using timed games technology may not satisfy the proven properties at all. In particular, due to perturbations in timings, some infinite behaviors may disappear completely. A first goal of our work is to develop algorithms for *robust* controller synthesis: we consider this problem by studying robust strategies in timed games, namely, those guaranteeing winning despite imprecisions bounded by a parameter.

Adversarial or Stochastic Environments We consider controller synthesis problems under two types of environments. In order to synthesize correct controllers for critical systems, one often considers an *adversarial* (or worst-case) environment, so as to ensure that *all* behaviors of the system are correct. However, in some cases, one is rather interested considering a *stochastic environment* which determines the resolution of non-determinism, and the choice of clock perturbations following probability distributions. We are then interested in satisfying a property *almost-surely*, that is, with probability 1, or *limit-surely*, that is, for every $\varepsilon > 0$, there should exist a strategy for Controller under which the property is satisfied with probability at least $1 - \varepsilon$.

Contributions We formalize the robust controller synthesis problem against an adversarial environment as a (non-stochastic) game played on timed automata with an unknown imprecision parameter δ, between players Controller and Perturbator. The game proceeds by Controller suggesting an action and a delay, and Perturbator perturbing each delay by at most δ and resolving the non-determinism by choosing an edge with the given action. Thus, the environment's behaviors model both uncontrollable moves and the limited precision Controller has. We prove the EXPTIME-completeness of deciding whether there exists a positive δ for which Controller has a winning strategy for a Büchi objective, matching the complexity of timed games in the classical sense. Our algorithm also allows one to compute $\delta > 0$ and a witness strategy on positive instances.

For stochastic environments, we study two probabilistic variants of the semantics: we first consider the case of adversarially resolved non-determinism and independently and randomly chosen perturbations, and then the case where both the non-determinism and perturbations are randomly resolved and chosen. In each case, we are interested in the existence of $\delta > 0$ such that Controller wins almost-surely (resp. limit-surely). We give decidable characterizations based on finite abstractions, and EXPTIME algorithms. All problems are formulated in a parametric setting: the parameter δ is unknown and is to be computed by our algorithms. This is one of the technical challenges in this paper.

Our results on stochastic perturbations can also be seen as a new interpretation of robustness phenomena in timed automata. In fact, in the literature on robustness in timed automata, non-robust behaviors are due to the accumulation of the imprecisions δ along long runs, and in the proofs, one exhibits non-robustness by artificially constructing such runs (e.g. [12, 22]). In contrast, in the present setting, we show that non-robust behaviors either occur almost surely, or can be avoided surely (Theorem 13).

Related Work While several works have studied robustness issues for model-checking, there are very few works on robust controller synthesis in timed systems:

– The (non-stochastic) semantics we consider was studied for *fixed* δ in [11] encoding by timed games; but the parameterized version of the problem was not considered.
– The restriction of the parameterized problem to (non-stochastic) *deterministic* timed automata was considered in [22]. Here, the power of Perturbator is restricted as it only modifies the delays suggested by Controller, but has no non-determinism to resolve. Therefore, the results consist in a robust Büchi acceptance condition for timed automata, but they do not generalize to timed games. Technically, the algorithm consists in finding an *aperiodic* cycle, which are cycles that are "stable" against perturbations. This notion was defined in [3] to study entropy in timed languages. We will also use aperiodic cycles in the present paper.
– A variant of the semantics we consider was studied in [8] for (deterministic) timed automata and shown to be EXPTIME-complete already for reachability due to an implicit presence of alternation. Timed games, Büchi conditions, or stochastic environments were not considered.
– Probabilistic timed automata where the non-determinism is resolved following probability distributions have been studied [16, 4, 17]. Our results consist in deciding almost-sure and limit-sure Büchi objectives in PTAs subject to random perturbations in the delays. Note that PTAs are equipped with a possibly different probability distribution for each action. Although we only consider uniform distributions, the two settings are equivalent for almost-sure and limit-sure objectives. Games played on PTA were considered in [14] for minimizing expected time to reachability with NEXPTIME $\cap$ co-NEXPTIME algorithms.

To the best of our knowledge, this work is the first to study a stochastic model of perturbations for synthesis in timed automata.

Organization Definitions are given in Section 2. Preliminaries concerning the notions of regions graphs, orbit graphs and shrunk difference bound matrices are presented in Section 3. The finite-state game abstraction used to characterize the positive instances of the (non-stochastic) problem is presented in Section 4 and the proof of correction is given in Section 5. In Section 6, we consider the two stochastic models for the environment. Due to space limitations, the proofs are omitted, but they are available in [18]

2 Robust Timed Games

Timed automata. Given a finite set of clocks $\mathcal{C}$, we call *valuations* the elements of $\mathbb{R}_{\geq 0}^{\mathcal{C}}$. For a subset $R \subseteq \mathcal{C}$ and a valuation ν, $\nu[R \leftarrow 0]$ is the valuation defined by $\nu[R \leftarrow 0](x) = 0$ if $x \in R$ and $\nu[R \leftarrow 0](x) = \nu(x)$ otherwise. Given $d \in \mathbb{R}_{\geq 0}$ and a valuation ν, the valuation $\nu + d$ is defined by $(\nu + d)(x) = \nu(x) + d$ for all $x \in \mathcal{C}$. We extend these operations to sets of valuations in the obvious way. We write $\mathbf{0}$ for the valuation that assigns 0 to every clock.

An atomic clock constraint is a formula of the form $k \preceq x \preceq' l$ or $k \preceq x - y \preceq' l$ where $x, y \in \mathcal{C}$, $k, l \in \mathbb{Z} \cup \{-\infty, \infty\}$ and $\preceq, \preceq' \in \{<, \leq\}$. A *guard* is a conjunction of atomic clock constraints. A valuation ν satisfies a guard g, denoted $\nu \models g$, if all constraints are satisfied when each $x \in \mathcal{C}$ is replaced with $\nu(x)$. We write $\Phi_{\mathcal{C}}$ for the set of guards built on $\mathcal{C}$. A *zone* is a subset of $\mathbb{R}_{\geq 0}^{\mathcal{C}}$ defined by a guard.

A *timed automaton* $\mathcal{A}$ over a finite alphabet of actions Act is a tuple $(\mathcal{L}, \mathcal{C}, \ell_0, \mathsf{Act}, E)$, where $\mathcal{L}$ is a finite set of locations, $\mathcal{C}$ is a finite set of clocks, $E \subseteq$

$\mathcal{L} \times \Phi_{\mathcal{C}} \times \mathsf{Act} \times 2^{\mathcal{C}} \times \mathcal{L}$ is a set of edges, and $\ell_0 \in \mathcal{L}$ is the initial location. An edge $e = (\ell, g, a, R, \ell')$ is also written as $\ell \xrightarrow{g,a,R} \ell'$. A state is a pair $q = (\ell, \nu) \in \mathcal{L} \times \mathbb{R}^{\mathcal{C}}_{\geq 0}$. An edge $e = (\ell, g, a, R, \ell')$ is enabled in a state (ℓ, ν) if ν satisfies the guard g.

The set of possible behaviors of a timed automaton can be described by the set of its runs, as follows. A *run* of $\mathcal{A}$ is a sequence $q_1 e_1 q_2 e_2 \dots$ where $q_i \in \mathcal{L} \times \mathbb{R}^{\mathcal{C}}_{\geq 0}$, and writing $q_i = (\ell, \nu)$, either $e_i \in \mathbb{R}_{>0}$, in which case $q_{i+1} = (\ell, \nu + e_i)$, or $e_i = (\ell, g, a, R, \ell') \in E$, in which case $\nu \models g$ and $q_{i+1} = (\ell', \nu[R \leftarrow 0])$. The set of runs of $\mathcal{A}$ starting in q is denoted $\mathsf{Runs}(\mathcal{A}, q)$.

Parameterized timed game. In order to define perturbations, and to capture the reactivity of a controller to these, we define the following parameterized timed game semantics. Intuitively, the parameterized timed game semantics of a timed automaton is a two-player game parameterized by $\delta > 0$, where Player 1, also called Controller chooses a delay $d > \delta$ and an action $a \in \mathsf{Act}$ such that every a-labeled enabled edge is such that its guard is satisfied after any delay in the set $d + [-\delta, \delta]$ (and there exists at least one such edge). Then, Player 2, also called Perturbator chooses an actual delay $d' \in d + [-\delta, \delta]$ after which the edge is taken, and chooses one of the enabled a-labeled edges. Hence, Controller is required to always suggest delays that satisfy the guards whatever the perturbations are.

Formally, given a timed automaton $\mathcal{A} = (\mathcal{L}, \mathcal{C}, \ell_0, \mathsf{Act}, E)$ and $\delta > 0$, we define the *parameterized timed game* of $\mathcal{A}$ w.r.t. δ as a two-player turn-based game $\mathcal{G}_\delta(\mathcal{A})$ between players Controller and Perturbator. The state space of $\mathcal{G}_\delta(\mathcal{A})$ is partitioned into $V_C \cup V_P$ where $V_C = \mathcal{L} \times \mathbb{R}^{\mathcal{C}}_{\geq 0}$ belong to Controller, and $V_P = \mathcal{L} \times \mathbb{R}^{\mathcal{C}}_{\geq 0} \times \mathbb{R}_{\geq 0} \times \mathsf{Act}$ belong to Perturbator. The initial state is $(\ell_0, \mathbf{0}) \in V_C$. The transitions are defined as follows: from any state $(\ell, \nu) \in V_C$, there is a transition to $(\ell, \nu, d, a) \in V_P$ whenever $d > \delta$, for every edge $e = (\ell, g, a, R, \ell')$ such that $\nu + d \models g$, we have $\nu + d + \varepsilon \models g$ for all $\varepsilon \in [-\delta, \delta]$, and there exists at least one such edge e. Then, from any such state $(\ell, \nu, d, a) \in V_P$, there is a transition to $(\ell', \nu') \in V_C$ iff there exists an edge $e = (\ell, g, a, R, \ell')$ as before, and $\varepsilon \in [-\delta, \delta]$ such that $\nu' = (\nu + d + \varepsilon)[R \leftarrow 0])$. A *play* of $\mathcal{G}_\delta(\mathcal{A})$ is a finite or infinite sequence $q_1 e_1 q_2 e_2 \dots$ of states and transitions of $\mathcal{G}_\delta(\mathcal{A})$, with $q_1 = (\ell_0, \mathbf{0})$, where e_i is a transition from q_i to q_{i+1}. It is said to be *maximal* if it is infinite or cannot be extended. A *strategy* for Controller is a function that assigns to every non-maximal play ending in some $(\ell, \nu) \in V_C$, a pair (d, a) where $d > \delta$ and a is an action such that there is a transition from (ℓ, ν) to (ℓ, ν, d, a). A strategy for Perturbator is a function that assigns, to every play ending in (ℓ, ν, d, a), a state (ℓ', ν') such that there is a transition from the former to the latter state. A play ρ is *compatible* with a strategy f for Controller if for every prefix ρ' of ρ ending in V_C, the next transition along ρ after ρ' is given by f. We define similarly compatibility for Perturbator's strategies. A play naturally gives rise to a unique run, where the states are in V_C, and the delays and the edges are those chosen by Perturbator.

Robust timed game problem. Given $\delta > 0$, and a pair of strategies f, g, respectively for Controller and Perturbator, we denote ρ the unique maximal run that is compatible with both f and g. A *Büchi objective* is a subset of the locations of $\mathcal{A}$. A Controller's strategy f is winning for a Büchi objective B if for any Perturbator's strategy g the run ρ that is compatible with f and g is infinite and visits infinitely often a location of B. The *robust timed game problem* asks,

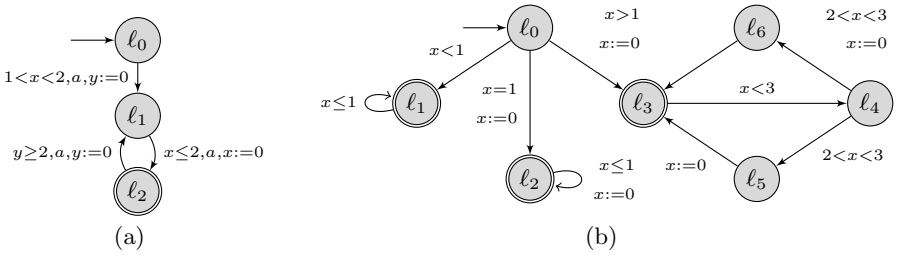

Fig. 1. On the left, a timed automaton from [19] that is not robustly controllable for the Büchi objective $\{\ell_2\}$. In fact, Perturbator can enforce that the value of x be increased by δ at each arrival at ℓ_1, thus blocking the run eventually (see [22]). On the right, a timed automaton that is robustly controllable for the Büchi objective $\{\ell_1, \ell_2, \ell_3\}$. We assume that all transitions have the same label. The cycle around ℓ_1 cannot be taken forever, as value of x increases due to perturbations. The cycle around ℓ_2 can be taken forever, but Controller cannot reach ℓ_2 due to the equality $x = 1$. Controller's strategy is thus to loop forever around ℓ_3. This is possible as for both choices of Perturbator in location ℓ_4, clock x will be reset, and thus perturbations do not accumulate. If one of the two resets were absent, Perturbator could force the run to always take that branch, and would win the game.

for a timed automaton $\mathcal{A}$ and a Büchi objective B, if there exists $\delta > 0$ such that Controller has a winning strategy in $\mathcal{G}_\delta(\mathcal{A})$ for the objective B. When this holds, we say that Controller wins the robust timed game for $\mathcal{A}$, and otherwise that Perturbator does. Note that these games are determined since for each $\delta > 0$, the semantics is a timed game.

Figure 1 shows examples of timed automata where either Controller or Perturbator wins the robust timed game according to our definitions. The main result of this paper for this non-stochastic setting is the following.

Theorem 1. *The robust timed game problem is EXPTIME-complete.*

We focus on presenting the EXPTIME membership in Sections 4 and 5. The algorithm relies on a characterization of winning strategies in a refinement of the region automaton construction.

In order to formally introduce the appropriate notions for this characterization, we need definitions given in the following section.

3 Regions, Orbit Graphs and Shrunk DBMs

Regions and Region Automata. Following [12, 19, 3], we assume that the clocks are bounded above by a known constant [4] in all timed automata we consider. Fix a timed automaton $\mathcal{A} = (\mathcal{L}, \mathcal{C}, \ell_0, \mathsf{Act}, E)$. We define *regions* as in [1], as subsets of $\mathbb{R}^{\mathcal{C}}_{\geq 0}$. Any region r is defined by fixing the integer parts of the clocks, and giving a partition $X_0, X_1, \ldots, X_m$ of the clocks, ordered according to their fractional values: for any $\nu \in r$, $0 = \mathsf{frac}(\nu(x_0)) < \mathsf{frac}(\nu(x_1)) < \ldots < \mathsf{frac}(\nu(x_m))$ for any $x_0 \in X_0, \ldots, x_m \in X_m$, and $\mathsf{frac}(\nu(x)) = \mathsf{frac}(\nu(y))$ for any $x, y \in X_i$.

[4] Any timed automaton can be transformed to satisfy this property.

Here, $X_i \neq \emptyset$ for all $1 \leq i \leq m$ but X_0 might be empty. For any valuation ν, let $[\nu]$ denote the region to which ν belongs. $\mathsf{Reg}(\mathcal{A})$ denotes the set of regions of $\mathcal{A}$. A region r is said to be *non-punctual* if it contains some $\nu \in r$ such that $\nu + [-\varepsilon, \varepsilon] \subseteq r$ for some $\varepsilon > 0$. It is said *punctual* otherwise. By extension, we say that (ℓ, r) is non-punctual if r is.

We define the *region automaton* as a finite automaton $\mathcal{R}(\mathcal{A})$ whose states are pairs (ℓ, r) where $\ell \in \mathcal{L}$ and r is a region. Given $(r', a) \in \mathsf{Reg}(\mathcal{A}) \times \mathsf{Act}$, there is a transition $(\ell, r) \xrightarrow{(r',a)} (\ell', s)$ if r' is non-punctual [5], there exist $\nu \in r$, $\nu' \in r'$ and $d > 0$ such that $\nu' = \nu + d$, and there is an edge $e = (\ell, g, R, \ell')$ such that $r' \models g$ and $r'[R \leftarrow 0] = s$. We write the *paths* of the region automaton as $\pi = q_1 e_1 q_2 e_2 \dots q_n$ where each q_i is a state, and $e_i \in \mathsf{Reg}(\mathcal{A}) \times \mathsf{Act}$, such that $q_i \xrightarrow{e_i} q_{i+1}$ for all $1 \leq i \leq n-1$. The *length* of the path is n, and is denoted by $|\pi|$. If a Büchi condition B is given, a cycle of the region automaton is *winning* if it contains an occurrence of a state (ℓ, r) with $\ell \in B$.

Vertices and Orbit Graphs. A *vertex* of a region r is any point of $\bar{r} \cap \mathbb{N}^{\mathcal{C}}$, where $\bar{r}$ denotes the topological closure of r. Let $\mathcal{V}(r)$ denote the set of vertices of r. We also extend this definition to $\mathcal{V}((\ell, r)) = \mathcal{V}(r)$.

With any path π of the region automaton, we associate a labeled bipartite graph $\Gamma(\pi)$ called the *folded orbit graph of* π [19] (FOG for short). Intuitively, the FOG of a path gives the reachability relation between the vertices of the first and the last regions, assuming the guards are closed. For any path π from state q to q', the node set of the graph $\Gamma(\pi)$ is defined as the disjoint union of $\mathcal{V}(q)$ and $\mathcal{V}(q')$. There is an edge from $v \in \mathcal{V}(q)$ to $v' \in \mathcal{V}(q')$, if, and only if v' is reachable from v along the path π when all guards are replaced by their closed counterparts. It was shown that any run along π can be written as a convex combination of runs along vertices; using this observation orbit graphs can be used to characterize runs along given paths [19]. An important property that we will use is that there is a monoid morphism from paths to orbit graphs. In fact, the orbit graph of a path can be obtained by combining the orbit graphs of a factorization of the path.

When the path π is a cycle around q, then $\Gamma(\pi)$ is defined on the node set $\mathcal{V}(q)$, by merging the nodes of the bipartite graph corresponding to the same vertex. A cycle π is *aperiodic* if for all $k \geq 1$, $\Gamma(\pi^k)$ is strongly connected. Aperiodic cycles are closely related to cycles whose FOG is a complete graph since a long enough iteration of the former gives a complete FOG and conversely, any cycle that has some power with a complete FOG is aperiodic. In the timed automaton of Fig. 1(b), the cycles around locations ℓ_2 and ℓ_3 are aperiodic while that of ℓ_1 is not. Complete FOG are of particular interest to us as they exactly correspond to paths whose reachability relation (between valuations of the initial and last region) is complete [3]. This means that there is no convergence phenomena along the path.

DBMs and shrunk DBMs. We assume the reader is familiar with the data structure of *difference-bound matrix (DBM)* which are square matrices over $(\mathbb{R} \times \{<, \leq\}) \cup \{(\infty, <)\}$ used to represent zones. DBMs were introduced in [6, 13] for analyzing timed automata; see also [5]. Standard operations used to explore the state space of timed automata have been defined on DBMs: intersection is

[5] Note this slight modification in the definition of the region automaton.

written $M \cap N$, $\mathsf{Pre}(M)$ is the set of time predecessors of M, $\mathsf{Unreset}_R(M)$ is the set of valuations that end in M when the clocks in R are reset. We also consider $\mathsf{Pre}_{>\delta}(M)$, time predecessors with a delay of more than δ.

To analyze the parametric game $\mathcal{G}_\delta(\mathcal{A})$, we need to express *shrinkings* of zones, *e.g.* sets of states satisfying constraints of the form $g = 1 + \delta < x < 2 - \delta \wedge 2\delta < y$, where δ is a parameter. Formally, a shrunk DBM is a pair (M, P), where M is a DBM, and P is a nonnegative integer matrix called a *shrinking matrix* (SM). This pair represents the set of valuations defined by the DBM $M - \delta P$, for any $\delta > 0$. For instance, M is the guard g obtained by setting $\delta = 0$, and P is made of the integer multipliers of δ.

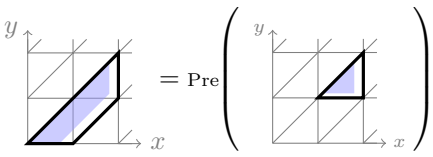

Fig. 2. Time-predecessors operation $(M, P) = \mathsf{Pre}(N, Q)$ applied on a shrunk DBM. Here, the shaded area on the left represents the set $M - \delta P$, while the zone with the thick contour represents M.

We adopt the following notation: when we write a statement involving a shrunk DBM (M, P), we mean that for some $\delta_0 > 0$, the statement holds for $(M - \delta P)$ for all $\delta \in [0, \delta_0]$. For instance, $(M, P) = \mathsf{Pre}_{>\delta}(N, Q)$ means that $M - \delta P = \mathsf{Pre}_{>\delta}(N - \delta Q)$ for all small enough $\delta > 0$. Additional operations are defined for shrunk DBMs: for any (M, P), we define $\mathsf{shrink}_{[-\delta,\delta]}(M, P)$ as the set of valuations ν with $\nu + [-\delta, \delta] \subseteq M - \delta P$, for small enough $\delta > 0$. Figure 2 shows an example of a shrunk DBM and an operation applied on it. Standard operations on zones can also be performed on shrunk DBMs in poly-time [21, 8].

4 Playing in the Region Automaton

In this section, we will define an appropriate abstraction based on region automata in order to characterize winning in the robust timed game. We note that the usual region automaton does not carry enough information for our purpose; for instance, the blocking behavior in Fig.1(a) cannot be detected in the region automaton (which does contain infinite runs). We therefore define, on top of the usual region construction, a complex winning condition $\mathcal{W}$ characterizing accepting runs *along aperiodic cycles*. In order to be able to transfer the condition $\mathcal{W}$ to the continuous semantics, we study the properties of $\mathcal{W}$ on the abstract region game, and derive two necessary and sufficient conditions ($\mathcal{C}_C$ and $\mathcal{C}_P$) for winning which will be used in the next section to derive the algorithm.

Abstract Arena and Strategies We fix a timed automaton $\mathcal{A} = (\mathcal{L}, \mathcal{C}, \ell_0, \mathsf{Act}, E)$ and a Büchi condition ϕ. We define a two-player turn-based game played on the region automaton $\mathcal{R}(\mathcal{A})$. In this game, Controller's strategy consists in choosing actions, while Perturbator's strategy consists in resolving non-determinism.

We consider standard notions of (finite-memory, memoryless) strategies in this game and, given a finite-memory strategy σ, we denote by $\mathcal{R}(\mathcal{A})[\sigma]$ the automaton obtained under strategy σ.

Winning Condition on $\mathcal{R}(\mathcal{A})$ We define set $\mathcal{W}$ of winning plays in the game $\mathcal{R}(\mathcal{A})$: an infinite play is winning iff the following two conditions are satisfied: 1)

an accepting state in ϕ is visited infinitely often 2) disjoint finite factors with complete folded orbit graphs are visited infinitely often.

Proposition 2. *The game $(\mathcal{R}(\mathcal{A}), \mathcal{W})$ is determined, admits finite-memory strategies for both players, and wining strategies can be computed in EXPTIME.*

The above proposition is proved by showing that condition 2) of $\mathcal{W}$ can be rewritten as a Büchi condition: the set of folded orbit graphs constitute a finite monoid (of exponential size) which can be used to build a Büchi automaton encoding condition 2). Using a product construction for Büchi automata, one can define a Büchi game of exponential size where winning for any player is equivalent to winning in $(\mathcal{R}(\mathcal{A}), \mathcal{W})$.

From the abstract game to the robust timed game We introduce two conditions for Perturbator and Controller which are used in Section 5 to build concrete strategies in the robust timed game.

$\mathcal{C}_P$: there exists a finite memory strategy τ for Perturbator such that no cycle in $\mathcal{R}(\mathcal{A})[\tau]$ reachable from the initial state is winning aperiodic.

$\mathcal{C}_C$: there exists a finite-memory strategy σ for Controller such that every cycle in $\mathcal{R}(\mathcal{A})[\sigma]$ reachable from the initial state is winning aperiodic.

Intuitively, determinacy allows us to write that either all cycles are aperiodic, or none is, respectively under each player's winning strategies. We prove that these properties are sufficient and necessary for respective players to win $(\mathcal{R}(\mathcal{A}), \mathcal{W})$:

Lemma 3. *The winning condition $\mathcal{W}$ is equivalent to $\mathcal{C}_P$ and $\mathcal{C}_C$: 1.* Perturbator *wins the game $(\mathcal{R}(\mathcal{A}), \mathcal{W})$ iff property $\mathcal{C}_P$ holds. 2.* Controller *wins the game $(\mathcal{R}(\mathcal{A}), \mathcal{W})$ iff property $\mathcal{C}_C$ holds. In both cases, a winning strategy for $\mathcal{W}$ is also a witness for $\mathcal{C}_C$ (resp. $\mathcal{C}_P$).*

The proof is obtained by the following facts: finite-memory strategies are sufficient to win the game $(\mathcal{R}(\mathcal{A}), \mathcal{W})$, thanks to the previous proposition; given a folded orbit graph γ, there exists n such that γ^n is complete iff γ is aperiodic; last, the concatenation of a complete FOG with an arbitrary FOG is complete.

5 Solving the Robust Timed Game

In this section, we show that condition $\mathcal{C}_P$ (resp. $\mathcal{C}_C$) is sufficient to witness the existence of a winning strategy in the robust timed game for Perturbator (resp. Controller). By Lemma 3, the robust timed game problem is then reduced to $(\mathcal{R}(\mathcal{A}), \mathcal{W})$ and we obtain:

Theorem 4. *Let $\mathcal{A}$ be a timed automaton with a Büchi condition. Then,* Controller *wins the robust timed game for $\mathcal{A}$ iff he wins the game $(\mathcal{R}(\mathcal{A}), \mathcal{W})$.*

By Proposition 2, the robust timed game can be solved in EXPTIME. In addition, when Controller wins the robust timed game, one can also compute $\delta > 0$ and an actual strategy in $\mathcal{G}_\delta(\mathcal{A})$: Lemma 3 gives an effective strategy σ satisfying $\mathcal{C}_C$ and the proof of Lemma 6 will effectively derive a strategy (given as shrunk DBMs).

5.1 Sufficient condition for Perturbator

We first prove that under condition $\mathcal{C}_P$, Perturbator wins the robust timed game. We use the following observations. Once one fixes a strategy for Perturbator satisfying $\mathcal{C}_P$, intuitively, one obtains a timed automaton (where there is no more non-determinism in actions), such that all accepting cycles are non-aperiodic. As Perturbator has no more non-determinism to resolve, results of [22] apply and the next lemma follows.

Lemma 5. *If $\mathcal{C}_P$ holds, then* Perturbator *wins the robust timed game.*

5.2 Sufficient condition for Controller

Proving that $\mathcal{C}_C$ is a sufficient condition for Controller is the main difficulty in the paper; the proof for the next lemma is given in this section.

Lemma 6. *If $\mathcal{C}_C$ holds, then* Controller *wins the robust timed game.*

Proof outline. We consider the non-deterministic automaton $\mathcal{B} = \mathcal{R}(\mathcal{A})[\sigma]$ which represents the behavior of game $\mathcal{R}(\mathcal{A})$ when Controller plays according to σ, given by condition $\mathcal{C}_C$. Without loss of generality, we assume that σ is a *memoryless strategy* played on the game $\mathcal{R}(\mathcal{A})[\sigma]$ (states of $\mathcal{R}(\mathcal{A})$ can be augmented with memory) and that $\mathcal{B}$ is trim. Given an edge $e = q \to q'$ of $\mathcal{B}$, we denote by $\mathsf{edge}(e)$ the underlying transition in $\mathcal{A}$.

Given a state p of $\mathcal{B}$, we denote by $\mathsf{Unfold}(\mathcal{B}, p)$ the infinite labeled tree obtained as the unfolding of $\mathcal{B}$, rooted in state p. Formally, nodes are labeled by states of $\mathcal{B}$ and given a node x labeled by q, $\sigma(q)$ is defined and there exists q' such that $q \xrightarrow{\sigma(q)} q'$ in $\mathcal{B}$. Then x has one child node for each such q'. We may abusively use nodes to refer to labels to simplify notations.

We first choose states $q_1, \ldots, q_n$ such that every cycle of $\mathcal{B}$ contains one of the q_i's. Let us denote by q_0 the initial state of $\mathcal{B}$, for $i = 0..n$, one can choose a finite prefix t_i of $\mathsf{Unfold}(\mathcal{B}, q_i)$ such that every leaf of t_i is labeled by some q_j, $j = 1..n$. Indeed, as $\mathcal{B}$ is trim and σ is a winning strategy for Controller in the game $(\mathcal{R}(\mathcal{A}), \mathcal{W})$, every branch of $\mathsf{Unfold}(\mathcal{B}, q_i)$ is infinite.

Strategies for standard timed games can be described by means of regions. In our robustness setting, we use shrinkings of regions. Let (ℓ_i, r_i) be the label of state q_i. To build a strategy for Controller, we will identify $\delta > 0$ and zones s_i, $i = 1..n$, obtained as shrinking of regions r_i. These zones satisfy that the controllable predecessors through the tree t_i computed with zones $(s_j)_j$ at leafs contains the zone s_i: this means that from any configuration in (ℓ_i, s_i), Controller has a strategy to ensure reaching a configuration in one of the (ℓ_j, s_j)'s, when following the tree t_i. These strategies can thus be repeated, yielding infinite outcomes. This idea is illustrated in Fig. 3 where the computations along some prefix t are depicted: the shrunk zone at a node represents the controllable predecessors of the shrunk

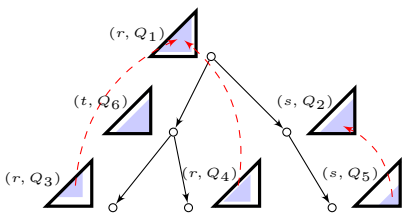

Fig. 3. Proof idea of Lemma 6. Dashed arrows represent cycles.

zones of its children. Here, from the shrunk set of the root node, one can ensure reaching a shrinking of each leaf which is included in the shrinking of the starting state of its cycle, yielding a kind of fixpoint. We have in fact $(r, Q_i) \subseteq (r, Q_1)$ for $i = 3, 4$, and $(s, Q_5) \subseteq (s, Q_2)$.

To identify these zones s_i, we will successively prove the three following facts:
1) Prefixes t_i's can be chosen such that every branch has a complete FOG
2) Controllable predecessors through t_i's of non-empty shrunk zones are non-empty shrunk zones
3) Controllable predecessors through t_i's can be faithfully over-approximated by the intersection of controllable predecessors through branches of t_i

Ensuring branches with complete FOGs. To prove property 1) of the Proof outline, we use condition $\mathcal{C}_C$ and the fact that long enough repetitions of aperiodic cycles yield complete FOGs. We obtain:

Lemma 7. *Under condition $\mathcal{C}_C$, there exists an integer N such that every path ρ in $\mathcal{B}$ of length at least N has a complete folded orbit graph.*

Controllable Predecessors and Merge. In order to compute winning states in $\mathcal{G}_\delta(\mathcal{A})$ through unfoldings, we define two operators. CPre is the standard set of controllable predecessors along a single edge:

Definition 8 (Operator CPre). *Let $e = q \to q_1$ be an edge in some unfolding of $\mathcal{B}$. Let us write $q = (\ell, r)$, $q_1 = (\ell_1, r_1)$, $\sigma(q) = (r', a)$ and $\mathsf{edge}(q \to q_1) = (\ell, g_1, a, R_1, \ell_1)$. Let M_1 be a DBM such that $M_1 \subseteq r_1$ and $\delta \geq 0$. We define the set of δ-controllable predecessors of M_1 through edge e as*
$\mathsf{CPre}^\delta_e(M_1) = r \cap \mathsf{Pre}_{>\delta}\left(\mathsf{Shrink}_{[-\delta,\delta]}\left(r' \cap \mathsf{Unreset}_{R_1}(M_1)\right)\right)$.

The above definition is extended to paths. Intuitively, $\mathsf{CPre}^\delta_e(M_1)$ are the valuations in region r from which M_1 can be surely reached through a delay in r' and the edge e despite perturbations up to δ.

We now consider the case of branching paths, where $\mathsf{Perturbator}$ resolves non-determinism. In this case, in order for $\mathsf{Controller}$ to ensure reaching given subsets in all branches, one needs a stronger operator, which we call CMerge. Intuitively, $\mathsf{CMerge}^\delta_{e_1,e_2}(M_1, M_2)$ is the set of valuations in the region starting r from which $\mathsf{Controller}$ can ensure reaching either M_1 or M_2 by a *single* strategy, whatever $\mathsf{Perturbator}$'s strategy is. The operator is illustrated in Fig. 4.

Definition 9 (Operator CMerge). *Let $e_1 = q \to q_1$ and $e_2 = q \to q_2$ be edges in some unfolding of $\mathcal{B}$, and write $q = (\ell, r)$, $\sigma(q) = (r', a)$ and for $i \in \{1, 2\}$, $q_i = (\ell_i, r_i)$, $\mathsf{edge}(q \to q_i) = (\ell, g_i, a, R_i, \ell_i)$. Let M_i be a DBM such that $M_i \subseteq r_i$ for $i \in \{1, 2\}$. For any $\delta \geq 0$, define the set of δ-controllable predecessors of M_1, M_2 through edges e_1, e_2 as* $\mathsf{CMerge}^\delta_{e_1,e_2}(M_1, M_2) = r \cap \mathsf{Pre}_{>\delta}\left(\mathsf{Shrink}_{[-\delta,\delta]}\left(r' \cap \bigcap_{i\in\{1,2\}} \mathsf{Unreset}_{R_i}(M_i)\right)\right)$.

We extend CMerge by induction to finite prefixes of unfoldings of $\mathcal{B}$. Consider a tree t and shrunk DBMs $(M_i, P_i)_i$ for its leaves, $\mathsf{CMerge}^\delta_t\big((M_i, P_i)_i\big)$ is the set of states for which there is a strategy ensuring reaching one of (M_i, P_i). Because $\mathsf{CMerge}^\delta_{e_1,e_2}$ is more restrictive than $\mathsf{CPre}^\delta_{e_1} \cap \mathsf{CPre}^\delta_{e_2}$, we always have $\mathsf{CMerge}^\delta_t \subseteq \bigcap_\beta \mathsf{CPre}^\delta_\beta$, where β ranges over all branches of t (See Fig. 4).

The following lemma states property 2) of the proof outline. Existence of the SM Q follows from standard results on shrunk DBMs. Non-emptiness of (M, Q) follows from the fact that every delay edge leads to a *non-punctual* region. Define a *full-dimensional subset of a set* $r \subseteq \mathbb{R}^n$ is a subset $r' \subseteq r$ such that there is $\nu \in r'$ and $\varepsilon > 0$ satisfying $\mathsf{Ball}_{d_\infty}(\nu, \varepsilon) \cap r \subseteq r'$, where $\mathsf{Ball}_{d_\infty}(\nu, \varepsilon)$ is the standard ball of radius ε for the infinity norm on $\mathbb{R}^n$.

Lemma 10. *Let t be a finite prefix of $\mathsf{Unfold}(\mathcal{B}, q)$, r the region labeling the root, and $r_1, \ldots, r_k$ those of the leafs. $M, N_1, \ldots, N_k$ be non-empty DBMs that are full dimensional subsets of $r, r_1, \ldots, r_k$ satisfying $M = \mathsf{CMerge}^0_t((N_j)_j)$. We consider shrinking matrices P_j, $1 \leq j \leq k$, of DBM N_j such that $(N_j, P_j) \neq \emptyset$. Then, there exists a SM Q such that $(M, Q) = \mathsf{CMerge}^\delta_t((N_j, P_j)_j)$, and $(M, Q) \neq \emptyset$.*

Over-Approximation of CMerge. Given a prefix t where each branch β_i ends in a leaf labeled with (r_i, P_i), we see $\cap_{\beta_i} \mathsf{CPre}^0_{\beta_i}((r_i, P_i))$ as an over-approximation of $\mathsf{CMerge}^0_t((r_i, P_i)_i)$. We will show that both sets have the same "shape", *i.e.* any facet that is not shrunk in one set, is not shrunk in the other one. This is illustrated in Fig. 4.

We introduce the notion of *0-dominance* as follows: for a pair of SMs P, Q, Q *0-dominates* P, written $P \preceq_0 Q$, if $Q[i,j] \leq P[i,j]$, and $Q[i,j] = 0$ implies $P[i,j] = 0$ for all i, j. Informally, a set shrunk by P is smaller than that shrunk by Q, but yields the same shape. The 0-dominance is the notion we use for a "precise" over-approximation of CMerge:

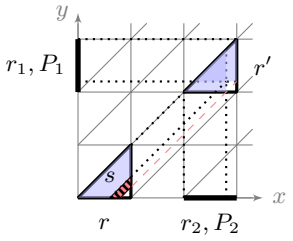

Fig. 4. We have $s = \mathsf{CMerge}^0((r_i, P_i)_i)$, which is strictly included in $\cap_i \mathsf{CPre}^0(r_i, P_i)$ but has the same shape.

Lemma 11. *Let t be a finite prefix of $\mathsf{Unfold}(\mathcal{B}, q)$, with $q = (\ell, r)$, and let (ℓ_i, r_i), $1 \leq i \leq k$ denote the (labels of) leafs of t. We denote by β_i, $1 \leq i \leq k$, the branches of t. Consider SMs P_i, $1 \leq i \leq k$, for regions r_i. Let us denote $(r, P) = \mathsf{CMerge}^0_t((r_i, P_i)_{1 \leq i \leq k})$ and $(r, Q) = \bigcap_{i=1}^k \mathsf{CPre}^0_{\beta_i}(r_i, P_i)$, then $P \preceq_0 Q$.*

Putting everything together. In order to complete the proof of Lemma 6, we first recall the following simple lemma:

Lemma 12 ([22]). *For any DBM M, there is a SM P_0 s.t. $(M, P_0) \neq \emptyset$, and is fully dimensional, and for any SM P and $\varepsilon{>}0$ with $(M, P) \neq \emptyset$ and $M - \varepsilon P_0 \neq \emptyset$, we have $M - \varepsilon P_0 \subseteq (M, P)$.*

Remember we have identified states q_i and trees t_i, $i = 0..n$. Denote (ℓ_i, r_i) the label of q_i. For each $i = 1..n$, we denote by P_i the SM obtained by Lemma 12 for r_i. Consider now some tree t_i, $i = 0..n$, with $((r_j, P_j)_j)$ at leafs. Let β_j be a branch of t_i and denote by (r_j, P_j) its leaf. By Lemma 7, the FOG of β_j is complete, and thus from *any* valuation in r_i, one can reach every valuation in the target region r_j along β_j (see [3]), and thus $r_i = \mathsf{CPre}^0_{\beta_j}(r_j, P_j)$. This holds for every branch and we obtain $r_i = \bigcap_j \mathsf{CPre}^0_{\beta_j}(r_j, P_j)$. By Lemma 11, this entails $r_i = \mathsf{CMerge}^0_{t_i}((r_j, P_j)_j)$. We can choose $\varepsilon > 0$ small enough such that the zone $s_i = r_i - \varepsilon P_i$ is non-empty for every $i = 1..n$ and we obtain $r_i = \mathsf{CMerge}^0_{t_i}((s_j)_j)$. We can then apply Lemma 10, yielding some SM Q_i of r_i such that $\emptyset \neq (r_i, Q_i) = \mathsf{CMerge}^\delta_{t_i}((s_j)_j)$. There are two cases:

– $i = 0$: as r_0 is the singleton $\{\mathbf{0}\}$, we have $(r_0, Q_0) = r_0$, and thus $r_0 = \mathsf{CMerge}^{\delta}_{t_0}((s_j)_j)$. In other terms, for small enough δ's, Controller has a strategy in $\mathcal{G}_\delta(\mathcal{A})$ along t_0 to reach one of the (ℓ_j, s_j)'s starting from the initial configuration $(\ell_0, \mathbf{0})$.
– $i \geq 1$: Lemma 12 entails $s_i \subseteq \mathsf{CMerge}^{\delta}_{t_i}((s_j)_j)$, which precisely states that for small enough δ's, Controller has a strategy in $\mathcal{G}_\delta(\mathcal{A})$ along t_i, starting in (ℓ_i, s_i), to reach one of the (ℓ_j, s_j)'s.

These strategies can thus be combined and repeated, yielding the result.

6 Probabilistic Semantics

In some systems, considering the environment as a completely adversarial opponent is too strong an assumption. We thus consider stochastic environments by defining two semantics as probabilistic variants of the robust timed games. The first one is the *stochastic game semantics* where Perturbator only resolves the non-determinism in actions, but the perturbations are chosen independently and uniformly at random in the interval $[-\delta, \delta]$. The second semantics is the *Markov decision process (MDP) semantics*, where the non-determinism is also resolved by a uniform distribution on the edges, and there is no player Perturbator.

6.1 Stochastic Game Semantics

Formally, given $\delta > 0$, the state space is partitioned into $V_C \cup V_P$ as previously. At each step, Controller picks a delay $d \geq \delta$, and an action a such that for every edge $e = (\ell, g, a, R, \ell')$ such that $\nu + d \models g$, we have $\nu + d + \varepsilon \models g$ for all $\varepsilon \in [-\delta, \delta]$, and there exists at least one such edge e. Perturbator then chooses an edge e with label a, and a perturbation $\varepsilon \in [-\delta, \delta]$ is chosen independently and uniformly at random. The next state is determined by delaying $d + \varepsilon$ and taking the edge e. To ensure that probability measures exist, we restrict to *measurable strategies*.

In this semantics, we are interested in deciding whether Controller can ensure a given Büchi objective almost surely, for some $\delta > 0$. It turns out that the same characterization as in Theorem 4 holds in the probabilistic case.

Theorem 13. *It is EXPTIME-complete to decide whether for some $\delta > 0$,* Controller *has a strategy achieving a given Büchi objective almost surely in the stochastic game semantics. Moreover, if $\mathcal{C}_C$ holds then* Controller *wins almost-surely; if $\mathcal{C}_P$ holds then* Perturbator *wins almost-surely.*

This is a powerful result showing a strong distinction between robust and non-robust timed games: in the first case, a controller that ensures the specification almost surely can be computed, while in non-robust timed games, *any* controller will fail *almost surely*. Thus, while in previous works on robustness in timed automata (e.g. [19]) the emphasis was on additional behaviors that might appear in the worst-case due to the accumulation of perturbations, we show that in our setting, this is inevitable. Note that this also shows that limit-sure winning (see next section) is equivalent to almost-sure winning.

6.2 Markov decision process semantics

The *Markov decision process semantics* consists in choosing both the perturbations, and the edges uniformly at random (and independently). Formally, it consists in restricting Perturbator to choose all possible edges uniformly at random in the stochastic game semantics. We denote by $\mathcal{G}_\delta^{\mathsf{MDP}}(\mathcal{A})$ the resulting game, and $\mathbb{P}^\sigma_{\mathcal{G}_\delta^{\mathsf{MDP}}(\mathcal{A}),s}$ the probability measure on $\mathsf{Runs}(\mathcal{A},s)$ under strategy σ.

For a given timed Büchi automaton, denote ϕ the set of accepting runs. We are interested in the two following problems: (we let $s_0 = (\ell_0, \mathbf{0})$)

Almost-sure winning: does there exist $\delta > 0$ and a strategy σ for Controller such that $\mathbb{P}^\sigma_{\mathcal{G}_\delta^{\mathsf{MDP}}(\mathcal{A}),s_0}(\phi) = 1$?

Limit-sure winning: does there exist, for every $0 \leq \varepsilon \leq 1$, a perturbation upper bound δ, and a strategy σ for Controller such that $\mathbb{P}^\sigma_{\mathcal{G}_\delta^{\mathsf{MDP}}(\mathcal{A}),s_0}(\phi) \geq 1-\varepsilon$?

Observe that if almost-sure winning cannot be ensured, then limit-sure winning still has a concrete interpretation in terms of controller synthesis: given a quantitative constraint on the quality of the controller, what should be the precision on clocks measurements to be able to synthesize a correct controller? Consider the timed automaton depicted on the right. It is easy to see that Controller loses the (non-stochastic) robust game, the stochastic game and in the MDP semantics with almost-sure condition, but he wins in the MDP semantics with limit-sure condition.

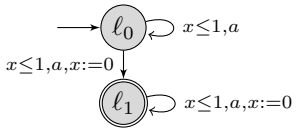

Fig. 5. This automaton is losing in the MDP semantics for the almost-sure winning but winning under the same semantics for the limit-sure winning. In fact, a blocking state (ℓ_0, x) with $x > 1-\delta$ is reachable with positive probability for any δ.

Theorem 14. *It is EXPTIME-complete to decide whether* Controller *wins almost-surely (resp. limit-surely) in the MDP semantics of a timed Büchi automaton.*

To prove this theorem, we will define decidable characterizations on $\mathcal{R}(\mathcal{A})$ which we will see as a finite Markov decision process. In this MDP, the nondeterminism of actions is resolved according to a uniform distribution. Given a strategy $\hat{\sigma}$ for Controller and a state v, we denote by $\mathbb{P}^{\hat{\sigma}}_{\mathcal{R}(\mathcal{A}),v}$ the resulting measure on $\mathsf{Runs}(\mathcal{R}(\mathcal{A}), v)$. The initial state of $\mathcal{R}(\mathcal{A})$ is v_0. We will use well-known notions about finite MDPs; we refer to [20].

Almost-sure winning We introduce the winning condition $\mathcal{W}'$: Controller's strategy $\hat{\sigma}$ in $\mathcal{R}(\mathcal{A})$ is winning in state v iff $\mathbb{P}^{\hat{\sigma}}_{\mathcal{R}(\mathcal{A}),v}(\phi) = 1$ and all runs in $\mathsf{Runs}(\mathcal{R}(\mathcal{A})[\hat{\sigma}], v)$ contain infinitely many disjoint factors whose FOGs are complete. Observe that this combines an almost-sure requirement with a sure requirement. This winning condition is our characterization for almost-sure winning:

Proposition 15. Controller *wins almost-surely in the MDP semantics of a timed Büchi automaton $\mathcal{A}$ iff* Controller *wins the game $(\mathcal{R}(\mathcal{A}), \mathcal{W}')$ in v_0.*

Intuitively, the first condition is required to ensure winning almost-surely, and the second condition allows to forbid blocking behaviors. Notice the resemblance with condition $\mathcal{W}$; the difference is that ϕ only needs to be ensured almost-surely rather than surely. We prove the decidability of this condition.

Lemma 16. *The game $(\mathcal{R}(\mathcal{A}), \mathcal{W}')$ admits finite-memory strategies, and winning strategies can be computed in EXPTIME.*

The proof of Prop. 15 uses the following ideas. We first assume that Controller wins the abstract game using some strategy $\hat{\sigma}$. We derive from $\hat{\sigma}$ a strategy σ in the MDP semantics by concretizing the delays chosen by σ. To do so, we consider the automaton $\mathcal{R}(\mathcal{A})[\hat{\sigma}]$ and proceed as in Section 5, which results in a strategy defined by means of shrinking matrices. Using the results of Section 5, we can prove that the outcomes of σ are never blocked, and thus the probabilities of paths in $\mathcal{R}(\mathcal{A})$ under $\hat{\sigma}$ are preserved by σ. As a consequence, σ wins almost-surely.

Conversely, by contradiction, we assume that Controller does not satisfy $\mathcal{W}'$ in $\mathcal{R}(\mathcal{A})$ while there exists an almost-sure strategy σ for the MDP semantics. We build from σ a strategy $\hat{\sigma}$ in $\mathcal{R}(\mathcal{A})$, and prove that it satisfies ϕ almost-surely. This entails the existence of a run ρ in $\mathcal{R}(\mathcal{A})[\hat{\sigma}]$ such that ρ eventually does not contain factors with a complete FOG. We finally show that, with positive probability, perturbations ensure that the run gets blocked along a finite prefix of this path, which ensures that σ is not almost-surely winning.

Limit-sure winning As illustrated in Fig. 5, it is possible, for any $\varepsilon > 0$, to choose the parameter $\delta > 0$ small enough to ensure a winning probability of at least $1 - \varepsilon$. The idea is that in such cases one can ensure reaching the set of almost-sure winning states with arbitrarily high probability, although the run can still be blocked with small probability before reaching this set.

To characterize limit-sure winning, we define condition $\mathcal{W}''$ as follows. If $\textsc{Win}'$ denotes the set of winning states for Controller in the game $(\mathcal{R}(\mathcal{A}), \mathcal{W}')$, then $\mathcal{W}''$ is defined as the set of states from which one can almost surely reach $\textsc{Win}'$.

Proposition 17. Controller *wins limit-surely in the MDP semantics of a timed Büchi automaton $\mathcal{A}$ from s_0 iff* Controller *wins the game $(\mathcal{R}(\mathcal{A}), \mathcal{W}'')$ in v_0.*

The proof of this proposition relies on the following lemma, and uses techniques similar as those introduced to prove Proposition 15.

Lemma 18. *The game $(\mathcal{R}(\mathcal{A}), \mathcal{W}'')$ admits finite-memory strategies, and winning strategies can be computed in EXPTIME.*

7 Conclusion

In this paper, we defined robust timed games with Büchi conditions and unknown imprecision parameters. Our formalism allows one to solve robust controller synthesis problems both against an adversarial (or worst-case) environment, and two variants of probabilistic environments. The procedures we have developed allow, when they exist, to effectively build a bound $\delta > 0$ on the perturbation and a winning strategy for Controller. Some questions remain open including the generalization of these results to concurrent timed games with parity conditions considered in [11]. We believe it is possible to derive symbolic algorithms but this will require extending the theory to detect aperiodic cycles in zone graphs rather than in the region graph.

References

1. Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.
2. Eugene Asarin, Oded Maler, Amir Pnueli, and Joseph Sifakis. Controller synthesis for timed automata. In SSC'98, p. 469–474. Elsevier Science, 1998.
3. Nicolas Basset and Eugene Asarin. Thin and thick timed regular languages. In FORMATS'11, LNCS 6919, p. 113–128. Springer, 2011.
4. Danièle Beauquier. On probabilistic timed automata. *Theor. Comput. Sci.*, 292(1):65–84, January 2003.
5. Johan Bengtsson and Wang Yi. Timed automata: Semantics, algorithms and tools. In *Lectures on Concurrency and Petri Nets*, LNCS 2098, p. 87–124. Springer, 2004.
6. Bernard Berthomieu and Miguel Menasche. An enumerative approach for analyzing time Petri nets. In WCC'83, p. 41–46. North-Holland/IFIP, September 1983.
7. Patricia Bouyer, Nicolas Markey, and Pierre-Alain Reynier. Robust analysis of timed automata via channel machines. In FoSSaCS'08, LNCS 4962, p. 157–171. Springer, 2008.
8. Patricia Bouyer, Nicolas Markey, and Ocan Sankur. Robust reachability in timed automata: A game-based approach. In ICALP'12, LNCS 7392, p. 128–140. Springer, 2012.
9. Patricia Bouyer, Nicolas Markey, and Ocan Sankur. Robustness in timed automata. In RP'13, LNCS 8169, p. 1–18. Springer, 2013.
10. Franck Cassez, Alexandre David, Emmanuel Fleury, Kim G. Larsen, and Didier Lime. Efficient on-the-fly algorithms for the analysis of timed games. In *CONCUR 2005*, p. 66–80. Springer, 2005.
11. Krishnendu Chatterjee, Thomas A. Henzinger, and Vinayak S. Prabhu. Timed parity games: Complexity and robustness. *Logical Methods in Computer Science*, 7(4), 2011.
12. Martin De Wulf, Laurent Doyen, Nicolas Markey, and Jean-François Raskin. Robust safety of timed automata. *Formal Methods in System Design*, 33(1-3):45–84, 2008.
13. David L. Dill. Timing assumptions and verification of finite-state concurrent systems. In AVMFSS'89, LNCS 407, p. 197–212. Springer, 1990.
14. Vojtęch Forejt, Marta Kwiatkowska, Gethin Norman, and Ashutosh Trivedi. Expected reachability-time games. In *Formal Modeling and Analysis of Timed Systems*, LNCS 6246, p. 122–136. Springer Berlin Heidelberg, 2010.
15. Thomas A. Henzinger and Joseph Sifakis. The embedded systems design challenge. In FM'06, LNCS 4085, p. 1–15, Hamilton, Canada, 2006. Springer.
16. Henrik E Jensen. Model checking probabilistic real time systems. In *Proc. 7th Nordic Workshop on Programming Theory*, p. 247–261. Citeseer, 1996.
17. Marta Kwiatkowska, Gethin Norman, Roberto Segala, and Jeremy Sproston. Automatic verification of real-time systems with discrete probability distributions. *Theor. Comput. Sci.*, 282(1):101–150, June 2002.
18. Youssouf Oualhadj, Pierre-Alain Reynier, and Ocan Sankur. Probabilistic robust timed games. 2014.
19. Anuj Puri. Dynamical properties of timed automata. *Discrete Event Dynamic Systems*, 10(1-2):87–113, 2000.
20. Martin L. Putterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley and Sons, New York, NY, 1994.
21. Ocan Sankur, Patricia Bouyer, and Nicolas Markey. Shrinking timed automata. In FSTTCS'11, LIPIcs 13, p. 375–386. Leibniz-Zentrum für Informatik, 2011.
22. Ocan Sankur, Patricia Bouyer, Nicolas Markey, and Pierre-Alain Reynier. Robust controller synthesis in timed automata. In CONCUR'13, LNCS 8052, p. 546–560. Springer, 2013.