Robust Controller Synthesis in Timed Automata

Ocan Sankur[1], Patricia Bouyer[1], Nicolas Markey[1], Pierre-Alain Reynier[2]

[1] LSV, ENS Cachan & CNRS, France
[2] LIF, Aix-Marseille Université & CNRS, France

Abstract. We consider the fundamental problem of Büchi acceptance in timed automata in a robust setting. The problem is formalised in terms of controller synthesis: timed automata are equipped with a parametrised game-based semantics that models the possible perturbations of the decisions taken by the controller. We characterise timed automata that are robustly controllable for some parameter, with a simple graph theoretic condition, by showing the equivalence with the existence of an aperiodic lasso that satisfies the winning condition (aperiodicity was defined and used earlier in different contexts to characterise convergence phenomena in timed automata). We then show decidability and PSPACE-completeness of our problem.

1 Introduction

Timed automata [AD94] are a timed extension of finite-state automata, providing an automata-theoretic framework to design, model, verify and synthesise systems with timing constraints. However, the semantics of timed automata is an idealisation of real timed systems; it assumes, for instance, perfect clocks for arbitrarily precise time measures, and instantaneous actions. Thus, properties proven on timed automata may not hold in a real implementation, and similarly, a synthesised controller may not be realisable on a real hardware. This problem has been addressed in several works in the literature, where the goal is to define a convenient notion of *robustness*, so as to define a realistic semantics for timed automata, and also make sure that the verified (or synthesised) behaviour remains correct in presence of small perturbations.

In this work, we consider the fundamental problem of Büchi acceptance of a given timed automaton in a robust setting. Our goal is to distinguish timed automata where a Büchi condition can be satisfied even when the chosen delays are systematically perturbed by an adversary by a bounded parametrised amount. In fact, it has been observed that some timed automata require choosing time delays with infinite precision in order to realise some behaviours. Apart from well-known Zeno behaviours, [CHR02] shows such a convergence phenomenon where an infinite run requires increasing precision at each step. These unrealisable

This work was partly supported by ANR projects ImpRo (ANR-2010-BLAN-0317) and ECSPER (ANR-2009-JCJC-0069), by ERC Starting grant EQualIS (308087), and by European project Cassting (FP7-ICT-601148).

behaviours are discarded in such an adversarial robust setting. Thus, we formalize the problem in a game-theoretic setting. Our objective is to synthesise controllers that are robust while discarding unrealisable behaviours.

More precisely, to define robustness, we equip timed automata with the following game semantics between two players ([CHP11]): *Controller* with a given Büchi objective, and *Perturbator* with the complementary objective. The semantics is a turn-based game parametrised by $\delta > 0$. At each step, Controller chooses an edge, and $d > \delta$, such that the guard of the edge is satisfied after any delay $d' \in [d-\delta, d+\delta]$. Then, the edge is taken after a delay $d' \in [d-\delta, d+\delta]$ chosen by Perturbator. Timed games with parity conditions were studied in [CHP11] for a fixed known parameter $\delta > 0$, and for strictly positive delays with no lower bound. In fact, in this case, one can encode this semantics as a usual timed game, and rely on existing techniques for solving timed games. For an unknown parameter $\delta > 0$, the problem is more complicated, and was left as a challenging open problem in [CHP11].

Our main result is the following: we show that deciding the existence of $\delta > 0$, and of a strategy for Controller in the perturbation game so as to ensure infinite runs satisfying a given Büchi condition is PSPACE-complete, thus no harder than in the exact setting [AD94]. We characterise *robust* timed automata, *i.e.*, those in which Controller has a winning strategy, by showing that Controller can win precisely when the timed automaton has an accepting *aperiodic* lasso. Aperiodicity [Sta12] is a variant of *forgetfulness* introduced in [BA11] in a different context, to study the entropy of timed languages. Our characterisation confirms the suggestion of [BA11] that this notion could be significant in the study of robustness. Our results rely on the non-trivial combination of various techniques used for studying timed automata: forgetful and aperiodic cycles as considered in [BA11,Sta12], the metrics of [GHJ97], shrinking techniques [SBM11,BMS12] and reachability relations of [Pur00]. Last, our proof provides a symbolic representation of Controller's strategy which could be amenable to implementation. A full version of the paper is available in [**?**].

Related works. A similar game semantics was considered in [BMS12], but the winning objectives considered are only reachability. An important consequence is that convergence phenomena and unrealisable strategies are not an issue, since one essentially only considers finite paths. In this paper, we thus need new proof techniques to deal with convergence. In addition, the semantics considered in [BMS12] is less restrictive for Controller: he only needs to suggest delays after which the guard of the chosen edge is satisfied. Hence, the guard may not be satisfied after a perturbation. The emphasis in the resulting semantics is therefore on the newly appearing behaviours. Algorithmically, the semantics of [BMS12] gives rise to more complex problems: reachability is already EXPTIME-complete, whereas we are able to treat richer Büchi objectives in PSPACE in this paper. From a designer's perspective, we believe that both semantics are meaningful in different modelling assumptions. The present semantics is interesting if lower and upper bounds on events, *e.g* task execution times, appear naturally in the model, and need be respected strictly. On the other hand, in other applications, the semantics of [BMS12] allows to model with equality constraints, and then synthesise controllers taking into account additional behaviour due to perturbations.

A related line of work is that of [Pur00,DDR05,DDMR08], which consists in modeling imprecisions by *enlarging* all clock constraints of the automaton by some parameter δ, that is, transforming each constraint of the form $x \in [a, b]$ into $x \in [a - \delta, b + \delta]$. The dual notion of *shrinking* was considered in [SBM11] in order to study whether any significant behaviour is lost when guards are shrunk. Both approaches are interested in model-checking, and the robustness condition is defined on the global behaviour of the enlarged or shrunk timed automaton. This does not capture the system's ability to adapt to perturbations that were observed earlier in a given run. In contrast, the game semantics endows Controller with a *strategy* against perturbations.

Among other robustness notions, [GHJ97] defines the *tube semantics* using a topology on timed automata runs. Our semantics is not related as we have a game semantics and a concrete parameter δ. However we use some results from [GHJ97] in our proofs. [Mar11] surveys different robust semantics for timed automata.

2 Timed Automata and Robust Safety

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 $\nu[R \leftarrow 0](x) = \nu(x)$ for $x \in \mathcal{C} \setminus R$ and $\nu[R \leftarrow 0](x) = 0$ for $x \in R$. 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 *timed automaton* $\mathcal{A}$ is a tuple $(\mathcal{L}, \mathcal{C}, \ell_0, E)$, where $\mathcal{L}$ is a finite set of locations, $\mathcal{C}$ is a finite set of clocks, $E \subseteq \mathcal{L} \times \Phi_{\mathcal{C}} \times 2^{\mathcal{C}} \times \mathcal{L}$ is a set of edges, and $\ell_0 \in \mathcal{L}$ is the initial location. An edge $e = (\ell, g, R, \ell')$ is also written as $\ell \xrightarrow{g,R} \ell'$.

The set of possible behaviours of a timed automaton can be described by the set of its runs, as follows. A *run* of $\mathcal{A}$ is a sequence $q_1 e_1 q_2 e_2 \ldots$ where $q_i \in \mathcal{L} \times \mathbb{R}_{\geq 0}^{\mathcal{C}}$, and writing $q_i = (\ell, \nu)$, either $e_i \in \mathbb{R}_{>0}$, in which case $q_{i+1} = (\ell, \nu + e_i)$, or $e_i = (\ell, g, R, \ell') \in E$, in which case $\nu \models g$ and $q_{i+1} = (\ell', \nu[R \leftarrow 0])$. We denote by $\mathsf{state}_i(\rho)$ the i-th state of any run ρ, by $\mathsf{first}(\rho)$ its first state, and, if ρ is finite, $\mathsf{last}(\rho)$ denotes the last state of ρ.

In order to define perturbations, and to capture the reactivity of a controller to these, we define the following robust game semantics, defined in [CHP11] (see also [BMS12] for a variant). Intuitively, the robust semantics of a timed automaton is a two-player game parametrised by $\delta > 0$, where Player 1, also called *Controller* chooses a delay $d > \delta$ and an edge whose guard is satisfied after any delay in the set $d + [-\delta, \delta]$. Then, Player 2, also called *Perturbator* chooses an actual delay $d' \in d + [-\delta, \delta]$ after which the edge is taken. Hence, Controller is required to always suggest delays that satisfy the guards whatever the perturbations are.

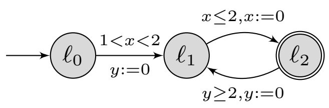
Formally, given a timed automaton $\mathcal{A} = (\mathcal{L}, \mathcal{C}, \ell_0, E)$ and $\delta > 0$, we define the *perturbation game* of $\mathcal{A}$ w.r.t. δ as a two-player turn-based game $\mathcal{G}_\delta(\mathcal{A})$ between players Controller and Perturbator. The state space of $\mathcal{G}_\delta(\mathcal{A})$ is partitioned into $V_C \cup V_P$ where $V_C = \mathcal{L} \times \mathbb{R}_{\geq 0}^{\mathcal{C}}$ belong to Controller, and $V_P = \mathcal{L} \times \mathbb{R}_{\geq 0}^{\mathcal{C}} \times \mathbb{R}_{\geq 0} \times E$ 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 $(\ell, \nu, d, e) \in V_P$ whenever $d > \delta$, $e = (\ell, g, R, \ell')$ is an edge such that $\nu + d + \varepsilon \models g$ for all $\varepsilon \in [-\delta, \delta]$. Then, from any such state $(\ell, \nu, d, e) \in V_P$, there is a transition to $(\ell', (\nu + d + \varepsilon)[R \leftarrow 0]) \in V_C$, for any $\varepsilon \in [-\delta, \delta]$. A pair of states of $V_C \cup V_P$ is said to be *consecutive* if there is a transition between them. A *play* of $\mathcal{G}_\delta(\mathcal{A})$ is a finite or infinite sequence $q_1 e_1 q_2 e_2 \ldots$ 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, e) where $d > \delta$ and e is an edge such that there is a transition from (ℓ, ν) to (ℓ, ν, d, e). A strategy for Perturbator is a function that assigns, to every play ending in (ℓ, ν, d, e), 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 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, the delays are those chosen by Perturbator, and the edges are chosen by Controller.

Given $\delta > 0$, and a pair of strategies f, g, respectively for Controller and Perturbator we let $\mathsf{Outcome}_{\mathcal{A}}^{\delta}(f, g)$ denote the unique maximal run that is compatible with both strategies. We also define $\mathsf{Outcome}_{\mathcal{A}}^{\delta}(f, \cdot)$ (resp. $\mathsf{Outcome}_{\mathcal{A}}^{\delta}(\cdot, g)$) as the set of all maximal runs compatible with f (resp. with g). A *Büchi objective* is a subset of the locations of $\mathcal{A}$. Controller's strategy f is winning for a Büchi objective B, if all runs of $\mathsf{Outcome}_{\mathcal{A}}^{\delta}(f, \cdot)$ are infinite and visit infinitely often some location of B. The *parametrised robust controller synthesis problem* asks, given a timed automaton $\mathcal{A}$ and a Büchi objective B, whether there exists $\delta > 0$ such that Controller has a winning strategy in $\mathcal{G}_\delta(\mathcal{A})$ for the objective B. Note that these games are determined since for each $\delta > 0$, the semantics is a timed game.

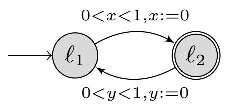
Figure 1 shows examples of controllable and uncontrollable timed automata, in our sense. The main result of this paper is the following.

Theorem 1. *Parametrised robust controller synthesis is PSPACE-complete for Büchi objectives.*

The next section introduces several notions we need to state our main lemma (Lemma 3), which characterises timed automata that are robustly controllable, based on the nature of the lassos of the region automata.



(a)



(b)

Fig. 1. On the left, a timed automaton from [Pur00] 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. On the right, the timed automaton (from [BA11]) is robustly controllable for the Büchi objective $\{\ell_2\}$. In fact, perturbations at a given transition do not affect the rest of the run; they are *forgotten*.

3 Regions, Orbit Graphs, Topology

Regions and Region Automata. We assume that the clocks are bounded above by a known constant in all timed automata we consider. Fix a timed automaton $\mathcal{A} = (\mathcal{L}, \mathcal{C}, \ell_0, \mathcal{E})$. We define *regions* as in [AD94], as subsets of $\mathbb{R}_{\geq 0}^{\mathcal{C}}$. 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 $x, y \in X_i$. 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.

We define the *region automaton* $\mathcal{R}(\mathcal{A})$ as a finite automaton whose states are pairs (ℓ, r) where $\ell \in \mathcal{L}$ and r is a region. There is a transition $(\ell, r) \xrightarrow{\Delta} (\ell, s)$ if there exist $\nu \in r$, $\nu' \in s$ and $d > 0$ such that $\nu' = \nu + d$. There is a transition $(\ell, r) \xrightarrow{e} (\ell', s)$ where $e = (\ell, g, R, \ell')$ if $r \models g$ and $r[R \leftarrow 0] = s$. We write the *paths* of the region automaton as $\pi = q_1 e_1 q_2 e_2 \ldots q_n$ where each q_i is a state, and $e_i \in E \cup \{\Delta\}$, such that $q_i \xrightarrow{e_i} q_{i+1}$ for all $1 \leq i \leq n-1$. We also write $\mathsf{first}(\pi) = q_1$, $\mathsf{last}(\pi) = q_n$, $\mathsf{state}_i(\pi) = q_i$, and $\mathsf{trans}_i(\pi) = e_i$. The *length* of the path is n, and is denoted by $|\pi|$. We denote the subpath of π between states of indices i and j by $\pi_{i\ldots j}$. Given a run ρ of $\mathcal{A}$, its *projection on regions* is the path π in the region automaton s.t. $\mathsf{state}_i(\rho) \in \mathsf{state}_i(\pi)$ for all $1 \leq i \leq n$, and either $\mathsf{trans}_i(\rho) = \mathsf{trans}_i(\pi)$ or $\mathsf{trans}_i(\pi) = \Delta$ and $\mathsf{trans}_i(\rho) \in \mathbb{R}_{\geq 0}$. In this case, we write $\mathsf{first}(\rho) \xrightarrow{\pi} \mathsf{last}(\rho)$ (and say that ρ is *along* π). A *lasso* is a path $\pi_0\pi_1$ where π_1 is a cycle, *i.e.* $\mathsf{first}(\pi_1) = \mathsf{last}(\pi_1)$. A *cycle* of $\mathcal{R}(\mathcal{A})$ is a *progress cycle* if it resets all clocks at least once [Pur00].

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. A path $\pi = q_1 e_1 q_2 e_2 \ldots q_n$ is *non-punctual* if whenever $e_i = \Delta$, q_{i+1} is non-punctual.

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. For any region r, and any clock x, let us denote by $r_{x,0}$ the upper bound (by $-r_{0,x}$ the lower bound) on clock x inside region r.

Then, a vertex v of r is defined by the choice of an index $0 \leq i \leq m$ such that for all $x \in X_1, \ldots, X_i$, we have $v(x) = -r_{0,x}$ and for all $x \in X_{i+1}, \ldots, X_m$, we have $v(x) = r_{x,0}$. Hence, any region has at most $|\mathcal{C}| + 1$ vertices (See e.g. [DDMR08]). We denote by $\inf(r)$ (resp. $\sup(r)$) the vertex of r where all clocks are equal to their lower bounds (resp. upper bounds). Let $\mathcal{V}(r)$ denote the set of vertices of r. We also extend this definition to $\mathcal{V}((\ell, r)) = \mathcal{V}(r)$. Note the following easy properties of regions. Any region r has at most one vertex $v \in \mathcal{V}(r)$ such that both v and $v+1$ belong to $\mathcal{V}(r)$. If these exist, then $v = \inf(r)$ and $v+1 = \sup(r)$. Moreover, $\sup(r) = \inf(r) + 1$ if, and only if r is non-punctual. It has been shown that all valuations in r are convex combinations of $\mathcal{V}(r)$ [Pur00].

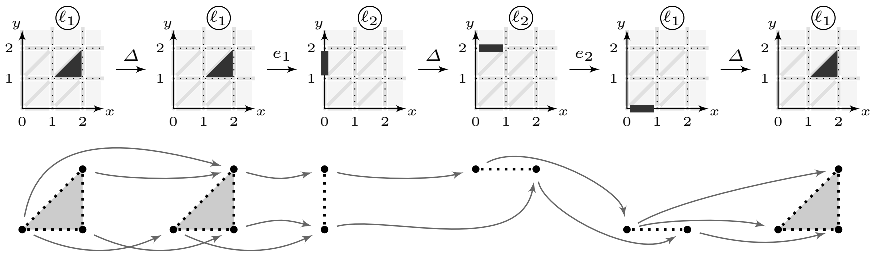
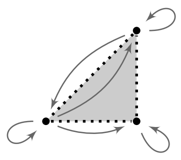


Fig. 2. The orbit graph of a (cyclic) path in the region automaton of the automaton of Fig. 1(a).



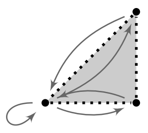


Fig. 3. The folded orbit graph of the (non-forgetful) cycle of Fig. 2.

Fig. 4. The folded orbit graph of a forgetful cycle.

With any path π of the region automaton, we associate a $|\pi|$-partite labelled graph $\gamma(\pi)$ called the *orbit graph of* π [Pur00]. Intuitively, the orbit graph of a path gives the reachability relation between the vertices of the regions visited along the path. Formally, for a transition $\tau = q_1 e_1 q_2$, its orbit graph $\gamma(\tau) = (V_1 \cup V_2, f_G, E)$ is a bipartite graph where $V_1 = \{(1, v)\}_{v \in \mathcal{V}(q_1)}$, and $V_2 = \{(2, v)\}_{v \in \mathcal{V}(q_2)}$. For any $\big((1, u), (2, v)\big) \in V_1 \times V_2$, we have an edge $((1, u), (2, v)) \in E$, if, and only if $u \xrightarrow{\overline{e_1}} v$, where $\overline{e_1} = \Delta$ if $e_1 = \Delta$, and otherwise $\overline{e_1}$ is obtained by replacing the guard by its closed counterpart. Note that each vertex has at least one successor through $\overline{e_1}$ [AD94]. The labelling function f_G maps each i to q_i; we also extend to nodes of G by $f_G((i, v)) = f_G(i)$. In order to extend $\gamma(\cdot)$ to paths, we use a composition operator $\oplus$ between orbit graphs, defined as follows. If $G = (V_1 \cup \ldots \cup V_n, f_G, E)$ and $G' = (V'_1 \cup \ldots \cup V'_m, f_{G'}, E')$ denote two orbit graphs, then $G \oplus G'$ is defined if, and only if, $f_G(n) = f_{G'}(1)$, that is, when the path defining the former graph ends in the first state of the path defining the latter graph. In this case, the graph $G'' = G \oplus G' = (V_1'' \cup \dots V_{n+m-1}'', f_{G''}, E'')$ is defined by taking the disjoint union of G and G', merging each node (n, v) of V_n with the node $(1, v)$ of V_1', and renaming any node $(i, v) \in V_i'$ by $(i + n - 1, v)$, so that we get a $(n + m - 1)$-partite graph. Formally, we let $V_i = V_i''$ for all $1 \leq i \leq n$, and the subgraph of G'' induced on these nodes is equal to G. For any $n + 1 \leq i \leq n + m - 1$, we have $V_i'' = \{(i, v)\}_{(i-n+1,v) \in V_{i-n+1}'}$, and there is an edge $\big((i, v), (i + 1, w)\big) \in E''$ if, and only if, $\big((i - n + 1, v), (i - n, w)\big) \in E'$. Now, we extend $\gamma(\cdot)$ to paths by induction, as follows. Consider any path $\pi = q_1 e_1 \dots q_{n-1} e_{n-1} q_n$, and let $G = (V_1 \cup \dots \cup V_{n-1}, f_G, E)$ be the $(n - 1)$-partite graph $\gamma(q_1 e_1 \dots q_{n-1})$, given by induction. Let $G' = (U \cup U', f_{G'}, E')$ denote the bipartite graph of $q_{n-1} e_{n-1} q_n$. Then, we let $\gamma(\pi) = G \oplus G'$. For any node (i, v) of $\gamma(\pi)$, let $\mathsf{Succ}((i, v))$ denote the set of nodes $(i + 1, w)$ with $\big((i, v), (i + 1, w)\big)$ is an edge. We also extend $\mathsf{Succ}(\cdot)$ to sets of nodes. Fig. 2 displays a path in the region automaton of the automaton depicted on Fig. 1(a) together with its orbit graph. Note that delays of duration zero are allowed when defining orbit graphs.

We define the *folded orbit graph* $\varGamma(\pi)$ for any path π that is not a cycle, as a bipartite graph on node set $\{1\} \times \mathcal{V}(\mathsf{first}(\pi)) \cup \{2\} \times \mathcal{V}(\mathsf{last}(\pi))$. There is an edge $\big((1, v), (2, w)\big)$ in $\varGamma(\pi)$ if, and only if there is a path from $(1, v)$ to (n, w) in $\gamma(\pi)$, where $n = |\pi|$. Nodes are labelled by the regions they belong to. For any cycle π, we define $\varGamma(\pi)$ similarly on the node set $\mathcal{V}(\mathsf{first}(\pi))$. Thus $\varGamma(\pi)$ may contain cycles; an example is given in Fig. 3. We extend the operator $\oplus$ to folded orbit graphs. A strongly connected component (SCC) of a graph is *initial* if it is not reachable from any other SCC.

A *forgetful cycle* of $\mathcal{R}(\mathcal{A})$ is a cycle whose folded orbit graph is strongly connected. A cycle π is *aperiodic* if for all $k \geq 1$, π^k is forgetful. A lasso is said to be aperiodic if its cycle is. Note that there exist forgetful cycles that are not aperiodic [Sta12].

An example of a non-forgetful cycle is given in Fig. 3. The timed automaton of Fig. 1(b) contains a forgetful cycle, shown on Fig. 4.

Some Linear Algebra. For any set of vectors, we denote by $\mathsf{Span}(\mathcal{B})$ the linear span of $\mathcal{B}$, *i.e.* the set of linear combinations of $\mathcal{B}$. In the proofs, we will often use the vertices of a region to define a basis of a vector space that contains the region.

Lemma 2. *Let r be any region, and let $v_0 = \inf(r)$. The set of vectors $\mathcal{B}_{v_0} = \{v - \inf(r)\}_{v \in \mathcal{V}(r) \setminus \{v_0\}}$ is linearly independent. Moreover, the affine space $v_0 + \mathsf{Span}(\mathcal{B}_{v_0})$ contains r.*

Let the dimension of a subset $r \subseteq \mathbb{R}^{\mathcal{C}}$ be the least d such that a affine subspace of $\mathbb{R}^{\mathcal{C}}$ of dimension d contains r. It follows immediately from Lemma 2 that in any region where the partition of the clocks according to their fractional values is written as $X_0, X_1, \dots, X_m$, has dimension m since it has $m + 1$ vertices.

We will consider the usual d_∞ metric on $\mathbb{R}^{\mathcal{C}}$, defined as $d_\infty(\nu, \nu') = \max_{x \in \mathcal{C}} |\nu(x) - \nu'(x)|$. We denote open balls in this metric by $\mathsf{Ball}_{d_\infty}(\nu, \varepsilon)$.

4 Main Lemma and Algorithm

Our main result is based on the following lemma, which gives a characterization of robust timed automata using aperiodic lassos of region automata.

Lemma 3 (Main Lemma). *For any timed automaton $\mathcal{A}$ and Büchi objective B, there exists $\delta > 0$ such that Controller has a winning strategy in $\mathcal{G}_\delta(\mathcal{A})$ for objective B, if, and only if $\mathcal{R}(\mathcal{A})$ has a reachable aperiodic non-punctual B-winning lasso.*

The algorithm for deciding robust Büchi acceptance follows from Lemma 3. It consists in looking for aperiodic non-punctual B-winning lassos in $\mathcal{R}(\mathcal{A})$. These lassos need not be simple, but the following lemma bounds their lengths.

Lemma 4. *Let B be a Büchi objective in a timed automaton $\mathcal{A}$, and π be an aperiodic non-punctual B-winning cycle of $\mathcal{R}(\mathcal{A})$. Then, there exists a cycle π' with the same properties, with length at most $N = 2^{(|\mathcal{C}|+1)^2+1} \times |\mathcal{R}(\mathcal{A})|$.*

The polynomial-space algorithm then consists in guessing an accepting lasso of exponential size in $\mathcal{R}(\mathcal{A})$ on-the-fly, and checking whether its folded orbit graph is aperiodic. The folded orbit graph can also be computed on-the-fly, and aperiodicity can be checked in PSPACE [Sta12]. See Appendix for more details.

The two directions of the main lemma are proved using different techniques; they are presented respectively in Sections 5 and 6. Our results also establish that winning strategies can be represented by regions with a given granularity, depending on δ. An algorithm is described at the end of Section 6 to actually compute the bound δ and a description of the winning strategy for Controller.

5 No Aperiodic Lassos Implies No Robustness

In this section, we prove that Controller loses if there is no aperiodic winning lassos. The idea is that if no accepting lasso of $\mathcal{R}(\mathcal{A})$ is aperiodic, then, as we show, the projection of any play to $\mathcal{R}(\mathcal{A})$ eventually enters and stays in a non-forgetful cycle. Then, we choose an appropriate *Lyapunov function* $L_I(\cdot)$ defined on valuations and taking nonnegative values, and describe a strategy for Perturbator such that the value of $L_I(\cdot)$ is decreased by at least ε at each iteration of the cycle. Hence, Controller cannot cycle infinitely on such cycles: either it reaches a deadlock, or it cycles on non-accepting lassos. In the rest of this section, we describe Perturbator's strategy, study its outcomes, choose a function $L_I(\cdot)$, and prove the first direction of the main lemma.

Our proof is based on several results. First, we consider a result from Puri [Pur00](Lemma 5) on reachability relations between valuations in timed automata, and establish non-trivial properties on it valid along non-punctual paths. We then study the folded orbit graphs of non-punctual progress cycles, and use original proof techniques (*e.g.* using the dimension of sets) to understand the form of these graphs. This allows us to consider the Lyapunov functions of [BA11] in this context, and prove our results as described above.

5.1 Reachability Relations

We already noted that any valuation ν can be written as the convex combination of the vertices of its region, i.e. $\nu = \sum_{v \in \mathcal{V}([\nu])} \lambda_v v$ for some unique coefficients $\lambda_v \geq 0$ with $\sum_v \lambda_v = 1$. When the region is clear from context, we will simply write $\nu = \boldsymbol{\lambda v}$. Given a path π and a vertex v of the region of $\mathsf{first}(\pi)$, let us denote by $R_{\Gamma(\pi)}(v)$ the set of nodes $w \in \mathcal{V}(\mathsf{last}(\pi))$ such that $(v, w) \in E(\Gamma(\pi))$. Thus, this is the "image" of v by the path π. Puri showed in [Pur00] that the reachability along paths can be characterized using orbit graphs.

Lemma 5 ([Pur00]). *Let π be a path from region r to s. Consider any $\nu \in r$ with $\nu = \sum_{v \in \mathcal{V}(r)} \lambda_v v$ for some coefficients $\lambda_v \geq 0$ and $\sum \lambda_v = 1$. If $\nu \xrightarrow{\pi} \nu'$, then for each $v \in \mathcal{V}(r)$, there exists a probability distribution $\{p_{v,w}^{\nu,\nu'}\}_{w \in R_{\Gamma(\pi)}(v)}$ over $R_{\Gamma(\pi)}(v)$ such that $\nu' = \sum_{v \in \mathcal{V}(r)} \lambda_v \sum_{w \in R_{\Gamma(\pi)}(v)} p_{v,w}^{\nu,\nu'} w$. Conversely, if there exist probability distributions $p_{v,w}^{\nu,\nu'}$ satisfying above equalities, then $\nu \xrightarrow{\bar{\pi}} \nu'$.*

Intuitively, the lemma shows that any successor of a point $\nu = \sum_i \lambda_i v_i$ can be obtained by distributing the weight λ_i of each vertex v_i to its successors following a probability distribution.

Example 1. The automaton of Fig. 1(a) contains a cycle on the region $r = [\![1 < x, y < 2 \land 0 < x - y < 1]\!]$. The vertices of r are $v_1 = (1,1), v_2 = (2,1), v_3 = (2,2)$. Consider a point $\nu = \frac{1}{3}v_1 + \frac{1}{3}v_2 + \frac{1}{3}v_3$. Then, Lemma 5 says that $\nu' = \sum_{1 \leq i \leq 3} \lambda_i v_i$ is reachable from ν along the cycle, where $\lambda_1 = \frac{1}{3}0.5 + \frac{1}{3}0.4 = \frac{9}{30}$, $\lambda_2 = \frac{1}{3}1 + \frac{1}{3}0.3 = \frac{13}{30}$, and $\lambda_3 = \frac{1}{3}0.6 + \frac{1}{3}0.2 = \frac{4}{15}$. Here, vertex set $\{v_1, v_3\}$ is an initial SCC I. One can check that L_I is indeed decreasing: $\frac{1}{3} + \frac{1}{3} \geq \frac{9}{30} + \frac{4}{15}$.

For any region r, and any subset $I \subseteq \mathcal{V}(r)$, we define the function $L_I : \bar{r} \to \mathbb{R}_{\geq 0}$ as, $L_I(\nu) = \sum_{v \in I} \lambda_v$, where $\nu = \boldsymbol{\lambda v}$. It is shown in [BA11] that given any cycle π, if I is chosen as the initial strongly connected component of $\Gamma(\pi)$, then for any run $\nu \xrightarrow{\pi} \nu'$, $L_I(\nu') \leq L_I(\nu)$. We will abusively use $L_I(\cdot)$ for a subset I of nodes of $\gamma(\pi)$ or $\Gamma(\pi)$, that correspond to a same region. Notice that $0 \leq L_I(\cdot) \leq 1$.

5.2 A global strategy for Perturbator

Let us call a valuation v *ε-far* if $v + [-\varepsilon, \varepsilon] \subseteq [v]$. A run is ε-far if all delays end in ε-far valuations. We show that Perturbator has a strategy ensuring ε-far runs.

Lemma 6. *Given any $\delta > 0$, and any timed automaton with C clocks, there exists a strategy $\sigma_{\delta,+}^{\mathsf{P}}$ (resp. $\sigma_{\delta,-}^{\mathsf{P}}$) for Perturbator that always perturbs by a positive (resp. negative) amount, and whose all outcomes are $\frac{\delta}{2(C+1)}$-far, and all delays are at least $\frac{\delta}{2(C+1)}$.*

Proof. After any delay $\nu \xrightarrow{d} \nu'$, $d \geq \delta$, chosen by Controller, consider the regions spanned by the set $\nu' + [0, \delta]$. It is easy to see that this set intersects at most

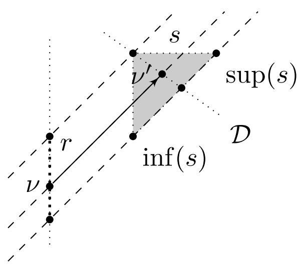
$|\mathcal{C}|+1$ different regions (See also [BMS12, Lemma 6]), all of which must satisfy the guard by definition of the game. So some region r satisfies $\nu' + [\alpha, \beta] \subseteq r$, for some $0 \le \alpha < \beta \le \delta$ with $\beta - \alpha \ge \frac{\delta}{|\mathcal{C}|+1}$. The strategy $\sigma^{\mathsf{P}}_{\delta,+}$ consists in choosing the perturbation as $\frac{1}{2}(\beta - \alpha)$. This guarantees time progress (of at least $\frac{\delta}{2(|\mathcal{C}|+1)}$). Moreover, the resulting valuation is always ε-far in its region, where $\varepsilon = \frac{\delta}{2(|\mathcal{C}|+1)}$. Observe also that all perturbations prescribed by this strategy are positive. The strategy $\sigma^{\mathsf{P}}_{\delta,-}$ is constructed similarly by considering the valuations $\nu' + [-\delta, 0]$.

It turns out that in order to win, Perturbator only needs to ensure ε-far runs, hence either of the strategies defined above is sufficient to win. In the rest, let us fix a strategy $\sigma^{\mathsf{P}}_{\delta}$ as $\sigma^{\mathsf{P}}_{\delta,+}$ or $\sigma^{\mathsf{P}}_{\delta,-}$. In order to prove that ε-far runs are winning for Perturbator, we study the properties of the runs $\mathsf{Outcome}^{\delta}_{\mathcal{A}}(\cdot, \sigma^{\mathsf{P}}_{\delta})$. We prove Propositions 7 and 8 which are a key element of the proof.

Using the ε-far property of the runs, the following proposition derives a bound on the convex combination coefficients of all visited valuations.

Proposition 7. *Let $\rho \in \mathsf{Outcome}_{\mathcal{A}}(\cdot, \sigma^{\mathsf{P}}_{\delta})$. For any $i \ge 1$, if we write $\mathsf{state}_i(\rho) = \boldsymbol{\lambda v}$, then $\lambda_v \ge \varepsilon$ for all vertices $v \in \mathcal{V}([\mathsf{state}_i(\rho)])$.*

Intuitively, a lower bound on the convex coefficients means that the valuation is not close to the borders of the region. We sketch the proof which is by induction. The property is true initially since the region $\mathbf{0}$ has a single vertex. For the induction case, let us mention the easy case of resets. Clock resets map each vertex to a single vertex, so each vertex has a single successor in the orbit graph. Then, it follows from Lemma 5 that the coefficient of each vertex in the target region is at least as large as its predecessor in the source region. For the case of time delays, one needs to consider the geometry of regions. The figure on the right shows the intuition in two dimensions. Given an ε-far delay from ν to ν', with $[\nu] = r$, and $[\nu'] = s$, one shows that the coefficients of $\inf(s)$ and $\sup(s)$ cannot be too small; otherwise the line $\mathcal{D}$ that connects ν' to the third vertex would be close to vertical or to horizontal, requiring ν' to be close to a border of s.



We need another property of similar spirit stating that all edges of the folded orbit graph receive a probability of at least $\min(\frac{1}{2}, \frac{\varepsilon}{2})$ along ε-far delays, according to the interpretation of Lemma 5. The proof is rather involved, and establishes that there is some degree of freedom in the choice of the probabilities of Lemma 5.

Proposition 8. *Let $\nu = \boldsymbol{\lambda v}$ and $\nu' = \boldsymbol{\lambda' v'}$ denote two valuations satisfying $\boldsymbol{\lambda}, \boldsymbol{\lambda'} \ge \varepsilon$, and s.t. $(\ell, \nu) \xrightarrow{\pi} (\ell, \nu')$ is an ε-far delay of duration at least ε. Then, for each $v \in \mathcal{V}([\nu])$, there exists a probability distribution $\{p_{v,w}\}_{w \in R_{\Gamma(\pi)}(v)}$ over $R_{\Gamma(\pi)}(v)$ s.t. $\nu' = \sum_{v \in \mathcal{V}(r)} \lambda_v \sum_{w \in R_{\Gamma(\pi)}(v)} p_{v,w} w$, and $p_{v,w} \ge \min(\frac{1}{2}, \frac{\varepsilon}{2})$.*

5.3 Decreasing Lyapunov function

In the previous subsection, we established lower bounds on the convex coefficients of the visited valuations, and the probabilities of Lemma 5. We use this property to find a Lyapunov function that strictly decreases at each iteration of a cycle.

In this subsection, we concentrate on progress cycles. In fact, in the proof of Lemma 3, we will show that if Controller is able to win against $\sigma_\delta^{\mathsf{P}}$, then some cycle of $\mathcal{R}(\mathcal{A})$ must be repeated infinitely often. But since $\sigma_\delta^{\mathsf{P}}$ always ensures a time progress of $\frac{\delta}{2(|\mathcal{C}|+1)}$ (see Lemma 6), and because all clocks are bounded, this is only possible in a progress cycle.

The following lemma shows that along non-punctual progress cycles, runs can reach any valuation in a ball around the target state. This gives the dimension of the set of valuations reachable along a progress cycle starting from a given valuation. The proof is somewhat similar to [DDMR08, Lemma 29].

Lemma 9. *Let π be a non-punctual progress cycle, and $(\ell,\nu) \xrightarrow{\pi} (\ell,\nu')$ a run along π. Then, there exists $\varepsilon > 0$ such that there exists a run from (ℓ,ν), along π, to any point in $\{\ell\} \times (\mathsf{Ball}_{d_\infty}(\nu',\varepsilon) \cap [\nu'])$.*

We now prove that the folded orbit graphs of non-punctual progress cycles are always connected. If the cycle is non-forgetful, there are at least two connected SCCs (Corollary 11). The lemma is proved by contradiction: we show that if the graph has disjoint components, then the set of states reachable from a given state cannot have full dimension, which contradicts Lemma 9.

Lemma 10. *The folded orbit graph of a non-punctual progress cycle is connected.*

Corollary 11. *The folded orbit graph of a non-punctual non-forgetful progress cycle π contains at least two strongly connected components that are connected. We associate with each π an initial SCC of $\Gamma(\pi)$, which we denote by $I(\pi)$.*

Hence, for any non-punctual non-forgetful cycle π, we consider the function $L_{I(\pi)}$. The following lemma shows a key property for the proof: $L_{I(\pi)}$ decreases by a fixed amount at each iteration of such a cycle under Perturbator's strategy $\sigma_\delta^{\mathsf{P}}$.

Lemma 12. *Let $\omega \in \mathsf{Outcome}_{\mathcal{A}}^{\delta}(\cdot,\sigma_\delta^{P})$, and ρ be a finite prefix of ω such that π, the projection of ρ to regions, is a cycle. If π is a non-forgetful progress cycle, then, writing $\mathsf{first}(\rho) = \boldsymbol{\lambda v}$ and $\mathsf{last}(\rho) = \boldsymbol{\lambda' v'}$, we have, $\sum_{i\in I(\pi)} \lambda'_i \leq \sum_{i \in I(\pi)} \lambda_i - \varepsilon^2/2$.*

The proof shows that any such run has a transition in which some edge of the folded orbit graph has a successor to a node that is not coreachable from $I(\pi)$. The existence of such an edge is proved using Corollary 11. By Propositions 7 and 8, we know that the convex combination coefficient associated to the node is at least ε, and the probability associated to the edge leaving it is at least $\varepsilon/2$. One then shows that at least $\varepsilon^2/2$ is lost from L_π at each iteration.

The previous lemma already gives an insight into the proof, since it follows that no non-forgetful cycle can be repeated infinitely under strategy $\sigma_\delta^{\mathsf{P}}$. However, one also needs to show that switching between different cycles cannot help Controller win. Thus, the last tool we need for the proof is the following factorization theorem, which allows factoring paths to cycles with the same folded orbit graphs.

Lemma 13. *Let π be a path of $\mathcal{R}(\mathcal{A})$ written as $\pi = \pi_0\pi_1\pi_2 \dots \pi_n$ where each π_i is a cycle that starts in the same state, for $i \geq 1$. Then, one can write $\pi = \pi'_0\pi'_1\pi'_2 \dots \pi'_{m+1}$ such that $m \geq \sqrt{n/r - 2} - 1$, where $r = 2^{(|\mathcal{C}|+1)^2}|\mathcal{R}(\mathcal{A})|$, and for some indices $0 = \alpha_0 < \alpha_1 < \dots$, we have $\pi'_i = \pi_{\alpha_i} \cdot \dots \cdot \pi_{\alpha_{i+1}-1}$ for each $i \geq 0$, and $\Gamma(\pi'_1) = \Gamma(\pi'_i)$ for all $1 \leq i \leq m$.*

Proof: No winning aperiodic lassos implies no robust safety. To get a contradiction, fix any winning strategy σ for Controller and let ρ be the infinite run $\mathsf{Outcome}^{\delta}_{\mathcal{A}}(\sigma, \sigma^{\mathsf{P}}_{\delta})$, and π its projection on regions. By definition of $\sigma^{\mathsf{P}}_{\delta}$, π is a non-punctual path. Let us write $\pi = \pi_0\pi_1 \dots \pi_n \dots$ such that all π_i, $i \geq 1$, are accepting cycles from a same state. Let $r = 2^{(|\mathcal{C}|+1)^2} \times |\mathcal{R}(\mathcal{A})|$, $n = \lceil 2/\varepsilon^2 \rceil + 1$ and N large enough so that $\lceil \sqrt{N/r - 2} \rceil - 1 \geq n2^{(|\mathcal{C}|+1)^2}$. We extract π' the prefix of π with N factors, and apply Lemma 13 which yields $\pi' = \pi'_0\pi'_1 \dots \pi'_{n'}\pi'_{n'+1}$, with $n' = n2^{(|\mathcal{C}|+1)^2}$, where $\Gamma(\pi'_1) = \dots = \Gamma(\pi'_{n'})$, and π'_i are obtained by concatenating one or several consecutive π_i. By hypothesis, some power k of π'_i is non-forgetful, with $k \leq 2^{(|\mathcal{C}|+1)^2}$ since this is the number of folded orbit graphs for a fixed labelling function. But since the folded orbit graphs are the same for all π'_i, this power is the same for all factors, and $\Gamma({\pi'_i}^k) = \Gamma(\pi'_i\pi'_{i+1} \dots \pi'_{i+k-1})$. Hence, we can factorize π' again into $\pi' = \pi'_0\pi''_1\pi''_2 \dots \pi''_n\pi''_{n+1}$, where $\Gamma(\pi''_1) = \dots = \Gamma(\pi''_n)$ and all are non-forgetful; while π'_0 and π''_{n+1} are arbitrary non-punctual paths. Moreover, π''_i, for $1 \leq i \leq n$, must be progress cycles too. In fact, otherwise there is some clock $x \in \mathcal{C}$ that is never reset along π'. But because $\sigma^{\mathsf{P}}_{\delta}$ ensures a time progress of ε at each delay, this means that π' contains delays of duration at least $n\varepsilon^2/2 > 1$, so we cannot have $\mathsf{first}(\pi'_1) = \mathsf{last}(\pi'_n)$ since the integer part of the clock x changes, and so does the region since all clocks are assumed to be bounded. Now, we have $I(\pi''_i) = I(\pi''_j)$ for any $1 \leq i, j \leq n$, so the functions $L_{I(\pi''_i)}$ are the same for all $1 \leq i \leq n$. If we write ρ_i the state reached in ρ following $\pi'_0\pi''_1 \dots \pi''_i$, then we get, by Lemma 12, $L_{I(\pi''_1)}(\rho_n) \leq L_{I(\pi''_1)}(\rho_0) - n\varepsilon^2/2$. This is a contradiction since $0 \leq L_{I(\pi'_1)} \leq 1$ and $n\varepsilon^2/2 > 1$.

6 Aperiodic Lassos Implies Robustness

We now prove that if $\mathcal{R}(\mathcal{A})$ contains an aperiodic non-punctual B-winning lasso, then there exists $\delta > 0$ and a strategy for Controller in $\mathcal{G}_\delta(\mathcal{A})$ ensuring robust safety. The idea of the proof is to observe that aperiodic cycles do not constrain runs in the way non-forgetful ones do, and show that this is still the case in the perturbation game semantics. More precisely, along an aperiodic lasso, Controller is able to always come back in a set at the middle of the starting region.

6.1 Zones and Shrunk Zones

A *zone* is a subset of $\mathbb{R}^{\mathcal{C}}_{\geq 0}$ defined by a guard. A *difference-bound matrix (DBM)* is a $|\mathcal{C}_0| \times |\mathcal{C}_0|$-matrix over $(\mathbb{R} \times \{<, \leq\}) \cup \{(\infty, <)\}$. We adopt the following notation: for any DBM M, we write $M = (\mathsf{M}, \prec^M)$, where M is the matrix made of the first components, with elements in $\mathbb{R} \cup \{\infty\}$, while $\prec^M$ is the matrix of the second components, with elements in $\{<, \leq\}$. A DBM M naturally represents a zone (which we abusively write M as well), defined as the set of valuations v such that, for all $x, y \in \mathcal{C}_0$, it holds $v(x) - v(y) \prec^M_{x,y} \mathsf{M}_{x,y}$ (where $v(0) = 0$). Standard operations used to explore the state space of timed automata have been defined on DBMs: intersection is 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 $\mathsf{Pre}_{>\delta}(M)$, the set of time predecessors with a delay of more than δ. DBMs were introduced in [BM83,Dil90] for analyzing timed automata; we refer to [BY04] for details.

A parametrised extension, namely *shrunk DBMs* were introduced in [SBM11] in order to study the parametrised state space of timed automata. Intuitively, our goal is to express *shrinkings* of guards, *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 to be chosen. 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 given $\delta > 0$. Considering the example g, M is the guard g obtained by setting $\delta = 0$, and P is made of the integer multipliers of δ.

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 such that $\nu + [-\delta, \delta] \subseteq M - \delta P$, for small enough $\delta > 0$.

Shrunk DBMs are closed under standard operations on zones:

Lemma 14 ([SBM11,BMS12]). *Let $M = f(N_1, \ldots, N_k)$ be an equation between normalized DBMs $M, N_1, \ldots, N_k$, using the operators $\mathsf{Pre}_{>\delta}$, $\mathsf{Unreset}_R$, $\cap$, and $\mathsf{shrink}_{[-\delta,\delta]}$, and let $P_1, \ldots, P_k$ be SMs. Then, there exists a shrunk DBM (M', Q) with $\mathsf{M}' = \mathsf{M}$, $M \subseteq M'$ and $(M', Q) = f\big((N_1, P_1), \ldots, (N_k, P_k)\big)$. The shrunk DBM (M', Q) and an upper bound on δ can be computed in poly-time.*

6.2 Controllable Predecessors

Consider an edge $e = (\ell, g, R, \ell')$. For any set $Z \subseteq \mathbb{R}^{\mathcal{C}}_{\geq 0}$, we define the *controllable predecessors of Z* as follows: $\mathsf{CPre}^\delta_e(Z) = \mathsf{Pre}_{>\delta}(\mathsf{shrink}_{[-\delta,\delta]}(g \cap \mathsf{Unreset}_R(Z)))$. Intuitively, $\mathsf{CPre}^\delta_e(Z)$ is the set of valuations from which Controller can ensure reaching Z in one step, following the edge e. In fact, it can delay in $\mathsf{shrink}_{[-\delta,\delta]}(g \cap \mathsf{Unreset}_R(Z))$ with a delay of more than δ, where under any perturbation in $[-\delta, \delta]$, the valuation satisfies the guard, and it ends, after reset, in Z. We extend this operator to paths as expected. Note that CPre^0_e is the usual predecessor operator without perturbation: If $N = \mathsf{CPre}^0_\pi(M)$ for some sets N, M and path π, then, N is the set of all valuations that can reach some valuation in M following π.

It immediately follows from Lemma 14 that the controllable predecessors of shrunk DBMs are shrunk DBMs, which are computable.

Corollary 15. *Let $e = (\ell, g, R, \ell')$ be an edge. Let M and N be non-empty DBMs such that $N = \mathit{CPre}^0_e(M)$. Then, for any SM P, there exists an SM Q such that $(N', Q) = \mathit{CPre}^\delta_e((M, P))$ for some $N \subseteq N'$ and $\mathsf{N} = \mathsf{N}'$.*

The set (N', Q) given by the previous lemma can be empty in general. However, as the following lemma shows, it turns out that in the case of non-punctual paths, controllable predecessors of open sets are non-empty, for small enough $\delta > 0$.

Lemma 16. *Let π be a non-punctual path from region r to s. Let $s' \subseteq s$ such that there exists $\nu' \in s'$ and $\varepsilon > 0$ with $\mathit{Ball}_{d_\infty}(\nu', \varepsilon) \cap s \subseteq s'$. Then, $\mathit{CPre}^\delta_\pi(s')$ is non-empty for small enough $\delta > 0$.*

6.3 Winning Under Perturbations

Let $\pi_0\pi$ denote a non-punctual aperiodic lasso, where π is the cycle; $\Gamma(\pi^n)$ is strongly connected for $n \geq 1$. We prove that the graph of some power is complete:

Lemma 17. *Let π be an aperiodic cycle. Then, there exists $n \leq |\mathcal{C}_0| \cdot |\mathcal{C}_0|!$ such that $\Gamma(\pi^n)$ is a complete graph.*

Let us assume, by the previous lemma, that $\Gamma(\pi)$ is a complete graph (one can consider the lasso $\pi_0\pi^n$ for an appropriate n). Let s be a region with smaller granularity inside r, obtained so that the Hausdorff distance between r and s is positive. In this case, s can be chosen so that it can be expressed by a DBM (with rational components). The construction is defined in the following lemma.

Lemma 18. *For any non-empty DBM M, there exists a non-empty DBM N such that $\mathit{Ball}_{d_\infty}(\nu, \varepsilon) \cap M \subseteq N$ for some $\nu \in M$, and for any shrinking matrix P with $(M, P) \neq \emptyset$, $N \subseteq (M, P)$. Moreover, N is computable in polynomial time.*

The last element we need for our proof is an observation from [BA11]: If π is a cycle of $\mathcal{R}(\mathcal{A})$ such that $\Gamma(\pi)$ is a complete graph, then for all $(\ell, \nu), (\ell, \nu') \in \mathsf{first}(\pi)$, there is a run $(\ell, \nu) \xrightarrow{\pi} (\ell, \nu')$, hence the reachability relation is complete.

Proof: Winning aperiodic lassos implies robust safety. We write $r = \mathsf{first}(\pi)$. Let $s \subseteq r$ given by Lemma 18 applied to r. Because $\Gamma(\pi)$ is complete, we have, by previous remark $r = \mathsf{CPre}^0_\pi(s)$. By Lemma 15, there exists a SM Q such that $(r, Q) = \mathsf{CPre}^\delta_\pi(s)$. By Lemma 16, (r, Q) is non-empty. By definition of s, for small enough $\delta > 0$, $s \subseteq (r, Q)$, so Controller has a strategy to always move inside s, at each iteration of π. Similarly, $\mathsf{CPre}^\delta_{\pi_0}((r, Q))$ is also non-empty and therefore contains the initial state. Hence, Controller wins.

Now, to actually compute δ and a winning strategy in exponential time, given an aperiodic lasso, one iterates the cycle so as to obtain a complete folded orbit graph (Lemma 17), then picks a subset s as in Lemma 18, and uses Corollary 15 to compute the controllable predecessors of s. This also provides the greatest δ under which the strategy along given lasso is valid.

7 Future works

We intend to investigate possible extensions of this work to timed games, where Perturbator entirely determines the move in some locations. This could require generalizing the notion of aperiodicity from paths to trees since Controller can no more ensure to follow a given path. Another interesting future work is studying infinite runs in the semantics of [BMS12].

References

[AD94] R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[BA11] N. Basset and E. Asarin. Thin and thick timed regular languages. In FORMATS'11, LNCS 6919, p. 113–128. Springer, 2011.

[BM83] B. Berthomieu and M. Menasche. An enumerative approach for analyzing time Petri nets. In WCC'83, p. 41–46. North-Holland/IFIP, 1983.

[BMS12] P. Bouyer, N. Markey, and O. Sankur. Robust reachability in timed automata: A game-based approach. In ICALP'12, LNCS 7392, p. 128–140, Warwick, UK, 2012. Springer.

[BY04] J. Bengtsson and W. Yi. Timed automata: Semantics, algorithms and tools. In *Lectures on Concurrency and Petri Nets*, LNCS 2098, p. 87–124. Springer, 2004.

[CHP11] K. Chatterjee, T. A. Henzinger, and V. S. Prabhu. Timed parity games: Complexity and robustness. *Logical Methods in Computer Science*, 7(4), 2011.

[CHR02] F. Cassez, T. A. Henzinger, and J.-F. Raskin. A comparison of control problems for timed and hybrid systems. In HSCC'02, LNCS 2289, p. 134–148. Springer, 2002.

[DDMR08] M. De Wulf, L. Doyen, N. Markey, and J.-F. Raskin. Robust safety of timed automata. *Formal Methods in System Design*, 33(1-3):45–84, 2008.

[DDR05] M. De Wulf, L. Doyen, and J.-F. Raskin. Almost ASAP semantics: From timed models to timed implementations. *Formal Aspects of Computing*, 17(3):319–341, 2005.

[Dil90] D. L. Dill. Timing assumptions and verification of finite-state concurrent systems. In AVMFSS'89, LNCS 407, p. 197–212. Springer, 1990.

[GHJ97] V. Gupta, Th. A. Henzinger, and R. Jagadeesan. Robust timed automata. In HART'97, LNCS 1201, p. 331–345. Springer, 1997.

[Mar11] N. Markey. Robustness in real-time systems. In SIES'11, p. 28–34, Västerås, Sweden, 2011. IEEE Comp. Soc. Press.

[Pur00] A. Puri. Dynamical properties of timed automata. *Discrete Event Dynamic Systems*, 10(1-2):87–113, 2000.

[SBM11] O. Sankur, P. Bouyer, and N. Markey. Shrinking timed automata. In FSTTCS'11, LIPIcs 13, p. 375–386. Leibniz-Zentrum für Informatik, 2011.

[SBMR13] O. Sankur, P. Bouyer, N. Markey, and P.-A. Reynier. Robust controller synthesis in timed automata. Research Report LSV-13-08, Laboratoire Spécification et Vérification, ENS Cachan, France, 2013. 27 pages.

[Sta12] A. Stainer. Frequencies in forgetful timed automata. In FORMATS'12, LNCS 7595, p. 236–251. Springer, 2012.