Synthesis of Bounded Integer Parameters for Parametric Timed Reachability Games*

Aleksandra Jovanović, Didier Lime, Olivier H. Roux

LUNAM Université. École Centrale de Nantes - IRCCyN UMR CNRS 6597
Nantes, France

Abstract. We deal with a parametric version of timed game automata (PGA), where clocks can be compared to parameters, and parameter synthesis. As usual, parametrization leads to undecidability of the most interesting problems, such as reachability game. It is not surprising then that the symbolic exploration of the state-space often does not terminate. It is known that the undecidability remains even when severely restricting the form of the parametric constraints. Since in classical timed automata, real-valued clocks are always compared to integers for all practical purposes, we solve undecidability and termination issues by computing parameters as bounded integers. We give a symbolic algorithm that computes the set of winning states for a given PGA and the corresponding set of bounded integer parameter valuations as symbolic constraints between parameters. We argue the relevance of this approach and demonstrate its practical usability with a small case-study.

1 Introduction

Timed game automata (TGA) [4,15] have become a widely accepted formalism for modeling and analyzing control problems on timed systems. They are essentially timed automata (TA) with the set of actions divided into controllable (used by the controller) and uncontrollable (used by the environment) actions. *Reachability game* for TGA is the problem of determining the strategy for the controller such that, no matter what the environment does, the system ends up in the desired location. Such games are known to be decidable [15]. Introduction of this model is followed by the development of the tool support [5] successfully applied to numerous industrial case studies [9].

This model, however, requires complete knowledge of the systems. Thus, it is difficult to use it in the early design stages when the whole system is not fully characterized. Even when all timing constraints are known, if the environment changes or the system is proven wrong, the whole verification process must be carried out again. Additionally, considering a wide range of values for constants allows for a more flexible and robust design.

Parametric reasoning is, therefore, particularly relevant for timed models, since it allows to the designers to use parameters instead of concrete timing

* This work was partially funded by the ANR national research program ImpRo (ANR-2010-BLAN-0317).

values. This approach, however, leads to the undecidability of the most important questions, such as reachability.

Related work.

Parametric timed automata [2], have been introduced as an extension of TA [1], to overcome the limit of checking the correctness of the systems with respect to concrete timing constraints. The central problem for verification purposes, reachability-emptiness, which asks whether there exists a parameter valuation such that the automaton has an accepting run, is undecidable. This naturally lead to the search for a subclasses of the model for which some problems would be decidable. In [11], L/U automata, which use each parameter as either a lower bound or an upper bound on clocks, is proposed. Reachability-emptiness problem is decidable for this model, but the state-space exploration still might not terminate. Decidability of L/U automata is further studied in [6]. The authors give the explicit representation of the set of parameters, when all parameters are integers and of the same type (L-automata and U-automata). In [7], the authors allow parameters both in the model and the property (PTCTL), and they show that the model-checking problem is decidable, in discrete time over a PTA with one parametric clock, if the equality is not allowed in the formulae. A different approach is taken in [3] where the exploration starts from the initial set of parameter values, for which the system is correct, and enlarges the set ensuring that the behaviors of PTA are time-abstract equivalent. They give a conjecture for the termination of the algorithm, being true on the studied examples.

Parametric Timed Game Automata (PGA): In [12], we have introduced an extension of TGA, called parametric timed game automata (PGA) and its subclass for which the reachability-emptiness game, which asks whether there exist a parameter valuation such that a winning strategy exists, is decidable. The subclass is, however, severely restricted in the use of parameters and the symbolic computation [12] of the set of winning states still might not terminate.

Our contribution. In this paper, we propose an orthogonal restriction scheme that we have introduced in [13] for PTA: since in classical timed game automata, real-valued clocks are always compared to integers for all practical purposes, we solve undecidability and termination issues by computing parameters as bounded integers. We give a symbolic algorithm that computes the set of winning states and the winning strategy for the controller for a given PGA and the corresponding set of parameter valuations as bounded integers. Due to the boundedness of parameters, the termination is ensured, and the resulting set of parameter valuations is given as symbolic constraints between parameters. The symbolic algorithm is based on the computation of the integer hull of the bounded parametric symbolic states. It first computes forward the whole reachable state-space, then propagates backwards the winning states. In order to find the winning states, we extend the well known fixed-point backwards algorithm for solving timed

reachability games [15], for the parametric domain. Surprisingly, we do not have to apply an integer hull in the backwards computation, in order to obtain the correct integer solution.

Organization of the Paper. The rest of the paper is organized as follows. Section 2 provides definitions about PGA, the problems we are considering, and recalls some negative decidability results. In Section 3 we first present the algorithm for solving timed games, then we motivate a restriction scheme, introduced in [13] for PTA, and extend the algorithm for the parametric approach and computation of parameters as bounded integers. The practical use of our method is shown with a small case study in Section 4. We conclude with Section 5.

2 Parametric Timed Games

Preliminaries. $\mathbb{R}$ is the set of real numbers ($\mathbb{R}_{\geq 0}$ is the set of non-negative real numbers), $\mathbb{Q}$ the set of rational numbers, and $\mathbb{Z}$ the set of integers. Let $V \subseteq \mathbb{R}$. A V-valuation on some finite set X is a function from X to V. We denote by V^X the set of V-valuations on X.

Let X be a finite set of variables modeling *clocks* and let P be a finite set of *parameters*. A *parametric clock constraint* γ is an expression of the form $\gamma ::= x_i - x_j \backsim p \mid x_i \backsim p \mid \gamma \wedge \gamma$, where $x_i, x_j \in X$, $\backsim \in \{\leq, <\}$, and p is a linear expression of the form $k_0 + k_1 p_1 + ... + k_n p_n$ with $k_0, ...k_n \in \mathbb{Z}$ and $p_1, ...p_n \in P$.

For any parametric clock constraint γ and any parameter valuation v, we note $v(\gamma)$ the constraint obtained by replacing each parameter p_i by its valuation $v(p_i)$. We denote by $G(X, P)$ the set of parametric constraints over X, and $G'(X, P)$ a set of parametric constraints over X of the form $\gamma' ::= x_i \backsim p \mid \gamma' \wedge \gamma'$.

For a valuation v on X and $t \in \mathbb{R}_{\geq 0}$, we write $v+t$ for the valuation assigning $v(x)+t$ to each $x \in X$. For $R \subseteq X$, $v[R]$ denotes a valuation assigning 0 to each $x \in R$ and $v(x)$ to each $x \in X \backslash R$. Further, we define the null valuation $\mathbf{0}_X$ on X by $\forall x \in X, \mathbf{0}_X(x) = 0$.

2.1 Parametric Timed Games

Definition 1. *A Parametric Timed Automaton (PTA) is a tuple* $\mathcal{A} = (L, l_0, X, \Sigma, P, E, \mathsf{Inv})$*, where* L *is a finite set of locations,* $l_0 \in L$ *is the initial location,* X *is a finite set of clocks,* Σ *is a finite alphabet of actions,* P *is a finite set of parameters,* $E \subseteq L \times \Sigma \times G(X, P) \times 2^X \times L$ *is a finite set of edges: if* $(l, a, \gamma, R, l') \in E$ *then there is an edge from* l *to* l' *with action* a*, (parametric)* guard γ *and set of clocks to reset* R*, and* $\mathsf{Inv} : L \mapsto G'(X, P)$ *is a function that assigns a (parametric) invariant to each location.*

For any $\mathbb{Q}$-valuation v on P, the structure $v(\mathcal{A})$ obtained from $\mathcal{A}$ by replacing each constraint γ by $v(\gamma)$ is a *timed automaton* with invariants [1,10]. The behavior of a PTA $\mathcal{A}$ is described by the behavior of all timed automata obtained by considering all possible valuations of parameters.

Definition 2 (Semantics of a PTA). *The concrete semantics of a PTA $\mathcal{A}$ under a parameter valuation v, notation $v(\mathcal{A})$, is the labelled transition system $(Q, q_0, \rightarrow)$ over $\Sigma \cup \mathbb{R}_{\geq 0}$ where:*

- $Q = \{(l, w) \in L \times \mathbb{R}^X_{\geq 0} \mid w(v(\mathsf{Inv}(l)))$ *is true* $\}$
- $q_0 = \{(l_0, \boldsymbol{0}_X) \in Q\}$
- *delay transition:* $(l, w) \xrightarrow{t} (l, w + t)$ *with* $t \geq 0$, *iff* $\forall t' \in [0, t], (l, w + t') \in Q$
- *action transition:* $(l, w) \xrightarrow{a} (l', w')$ *with* $a \in \Sigma$, *iff* $(l, w), (l', w') \in Q$, *there exists an edge* $(l, a, \gamma, R, l') \in E, w' = w[R]$ *and* $w(v(\gamma))$ *is true.*

A finite run of PTA $\mathcal{A}$, under a parameter valuation v, is a sequence of alternating delay and action transition in the semantics of $v(\mathcal{A})$, $\rho = q_1 a_1 q_2 ... a_{n-1} q_n$, where $\forall i, q_i \in Q, a_i \in \Sigma \cup \mathbb{R}_{\geq 0}$, and $q_i \xrightarrow{a} q_{i+1}$. The last state of ρ is denoted by $last(\rho)$. We denote by $\mathsf{Runs}(v(\mathcal{A}))$ the set of runs starting in the initial state of $v(\mathcal{A})$, and by $\mathsf{Runs}(q, v(\mathcal{A}))$ the set of runs starting in q. A run is maximal if it is either infinite or cannot be extended. A state q is said to be reachable in $\mathcal{A}$ if there exists a finite run $\rho \in \mathsf{Runs}(v(\mathcal{A}))$, such that $last(\rho) = q$.

In [12], we have extended the previous definitions, to obtain a more powerful formalism that allows us to express parametric control problems on timed systems.

Definition 3. *A Parametric (Timed) Game Automaton (PGA) $\mathcal{G}$ is a parametric timed automaton with its set of actions Σ partitioned into controllable (Σ^c) and uncontrollable (Σ^u) actions.*

As for PTA, for any PGA $\mathcal{G}$ and any rational valuation on parameters v, the structure $v(\mathcal{G})$, obtained by replacing each constraint γ by $v(\gamma)$, is a timed game automaton.

In a TGA, two players, a controller and an environment, choose at every instant one of the available actions from their own sets, according to a strategy, and the game progresses. Since the game is symmetric, we give only the definition for the controller playing with actions from Σ^c. At each step, a strategy tells controller to either delay in a location (delay), or to take a particular controllable action.

Definition 4 (Strategy). *A strategy $\mathcal{F}$ over $v(\mathcal{G})$ is a partial function from $\mathsf{Runs}(v(\mathcal{G}))$ to $\Sigma^c \cup \{\mathsf{delay}\}$ such that for every finite run ρ, if $\mathcal{F}(\rho) \in \Sigma^c$ then $last(\rho) \xrightarrow{\mathcal{F}(\rho)} q$ for some state $q = (l, w)$, and if $\mathcal{F}(\rho) = \mathsf{delay}$, then there exists some $d > 0$ such that for all $0 \leq d' \leq d$, there exists some state q such that $last(\rho) \xrightarrow{d'} q$.*

We consider only memory-less strategies, where $\mathcal{F}(\rho)$ only depends on the current $last(\rho)$. Note that the uncontrollable actions cannot be used to reach the desired location, the controller has to be able to reach it by itself.

$\mathsf{Outcome}$ defines the restricted behavior of $v(\mathcal{G})$, when the controller plays some strategy $\mathcal{F}$.

Definition 5 (Outcome). *Let $\mathcal{G}$ be a PGA, v be a parameter valuation, and $\mathcal{F}$ be a strategy over $v(\mathcal{G})$. The outcome* $\mathsf{Outcome}(q, \mathcal{F})$ *of $\mathcal{F}$ from state q is the subset of runs in* $\mathsf{Runs}(q, v(\mathcal{G}))$ *defined inductively as:*

- *the run with no action $q \in \mathsf{Outcome}(q, \mathcal{F})$*
- *if $\rho \in \mathsf{Outcome}(q, \mathcal{F})$ then $\rho' = \rho \xrightarrow{\delta} q' \in \mathsf{Outcome}(q, \mathcal{F})$ if $\rho' \in \mathsf{Runs}(q, v(\mathcal{G}))$ and one of the following three condition holds:*
  1. *$\delta \in \Sigma^u$,*
  2. *$\delta \in \Sigma^c$ and $\delta = \mathcal{F}(\rho)$,*
  3. *$\delta \in \mathbb{R}_{\geq 0}$ and $\forall 0 \leq \delta' < \delta$, $\exists q'' \in S$ s.t. $last(\rho) \xrightarrow{\delta'} q'' \wedge \mathcal{F}(\rho \xrightarrow{\delta'} q'') =$ delay.*
- *for an infinite run ρ, $\rho \in \mathsf{Outcome}(q, \mathcal{F})$, if all the finite prefixes of ρ are in* $\mathsf{Outcome}(q, \mathcal{F})$.

As we are interested in reachability games, we consider only the runs in the outcome that are "long enough" to have a chance to reach the goal location: a run $\rho \in \mathsf{Outcome}(q, \mathcal{F})$ is *maximal* if it is either infinite or there is no delay d and no state q' such that $\rho' = \rho \xrightarrow{d} q' \in \mathsf{Outcome}(q, \mathcal{F})$ and $\mathcal{F}(\rho') \in \Sigma^c$ (the only possible actions from $last(\rho)$ are uncontrollable actions). $MaxOut(q, \mathcal{F})$ notes the set of maximal runs for a state q and a strategy $\mathcal{F}$.

Definition 6 (Winning strategy). *Let $\mathcal{G} = (L, l_0, X, \Sigma^c \cup \Sigma^u, P, E, \mathsf{Inv})$ be a PGA and $l_{goal} \in L$. A strategy $\mathcal{F}$ is winning for the location l_{goal} if for all runs $\rho \in MaxOut(q_0, \mathcal{F})$, where $q_0 = (l_0, \mathbf{0}_X)$, there is some state (l_{goal}, w) in ρ.*

Similarly, a state q is winning (for the controller) if it belongs to a run in the outcome of a winning strategy.

We study the problem of *reachability-emptiness game for PGA*, which is the problem of determining whether the set of parameter valuations, such that there exists a strategy for the controller to enforce the system into the desired location, is empty. We are also interested in the corresponding *reachability-synthesis game* for PGA, which is to compute all parameter valuations such that there exists a winning strategy for the controller.

Reachability-emptiness problem for PTA is undecidable, [1]. As PGA extend PTA, the reachability-emptiness game for PGA is also undecidable [12].

3 Integer Parameter Synthesis

Parametrization leads to undecidability of the most important problems. There exist subclasses of PTA [11,6] (resp. PGA [12]) for which the reachability-emptiness (resp. reachability-emptiness game) is decidable, however, they are severely restricted and their practical usability is unclear.

We advocated in [13] for a different restriction scheme for PTA, to search for parameter values as bounded integers. This makes all the problems decidable, since we can enumerate all the possible valuations. Lifting either one of the two assumptions (boundedness or integer) leads to undecidability [13]. The

explicit enumeration is not practical, and thus we proposed an efficient symbolic method to find the solution. This has the advantage of giving the set of parameter valuations as symbolic constraints between parameters.

3.1 Computing the Winning States in Parametric Timed Games

We first recall an algorithm from [12] to compute the parameter valuations permitting the existence of a winning strategy for the controller. Due to the associated decidability results, its termination is obviously not guaranteed. For the sake of readability, we present it in a simplified version, closer to an extension of the classical algorithm of [15], in which we first compute forward the whole reachable state-space, then propagate backwards the winning states, instead of interleaving the forward and backward computations as done in [12] as an extension of [8]. There would be no problem in restoring that interleaving in the setting proposed here.

The computation consists of two fixed-points on the state-space of the PGA. To handle these sets of states, we define the notion of parametric symbolic state:

Definition 7 (Parametric symbolic state). *A* symbolic state *of a parametric timed (game) automaton $\mathcal{G}$, with set of clocks X and set of parameters P, is a pair (l, Z) where l is a location of $\mathcal{A}$ and Z is a set of valuations v on $X \cup P$.*

For the state-space exploration in the parametric domain, we extend the classical operations on valuation sets:

- projection: for any set of states Z, and any set $R \subseteq P \cup X$, $Z_{|R}$ is the projection of Z on R;
- future: $Z^{\nearrow} = \{v' \mid \exists v \in Z \, s.t. \, v'(x) = v(x) + d, d \geq 0 \text{ if } x \in X; v'(x) = v(x) \text{ if } x \in P\}$
- reset of clocks in set $R \subseteq X$: $Z[R] = \{v[R] \mid v \in Z\}$

We also need the following operators on symbolic states.

- initial symbolic state of the PTA $\mathcal{A} = (L, l_0, \Sigma, X, P, E, \mathsf{Inv})$: $\mathsf{Init}(\mathcal{A}) = (l_0, \{v \in \mathbb{R}^{X \cup P} \mid v_{|X} \in \mathbf{0}_X \cap v_{|P}(\mathsf{Inv}(l_0))_{|X}\})$
- successor by edge $e = (l, a, \gamma, R, l')$: $\mathsf{Succ}((l, Z), e) = (l', (Z \cap \gamma)[R]^{\nearrow} \cap \mathsf{Inv}(l'))$.

We can extend the Succ operator to arbitrary sets of states by defining, for any set of states S and any location l, the subset S^l of S containing the states with location l. S^l is therefore a symbolic state (l, Z) for some set of valuations Z. Then we define $\mathsf{Succ}(S, e)$ as $\mathsf{Succ}(S^l, e)$, with l being the source location of edge e. The reachable state-space of the PGA can be computed by the following fixed-point (when it exists) [13]:

$$S_{n+1} = \mathsf{Init}(\mathcal{A})^{\nearrow} \cup \bigcup_{e \in E} \mathsf{Succ}(S_n, e), \text{ with } S_0 = \emptyset$$

The final fixed-point set is noted S^*. It follows from [11] that all Z are finite unions of convex polyhedra.

In order to compute the winning states, we need a few additional operators:

- past: $v^{\swarrow} = \{v' \mid \exists v \in Z \, s.t. \, v'(x) = v(x) - d, d \geq 0 \text{ if } x \in X; v'(x) = v(x) \text{ if } x \in P\}$
- inverse reset of clocks in set $R \subseteq X$: $Z[R]^{-1} = \{v', v'(x) = 0 \text{ if } v' \in Z[R] \mid \exists v \in Z \text{ s.t. } v'(x) = v(x) \text{ if } x \notin R\}$
- predecessor by edge $e = (l, a, \gamma, R, l')$: $\mathsf{Pred}((l', Z), e) = (l, Z[R]^{-1} \cap \gamma)$.
- controllable (resp. uncontrollable) action predecessors: $\mathsf{cPred}((l', Z))$ (resp. $\mathsf{uPred}((l', Z))$) is the union of all the predecessors of (l', Z) by some edge with target location l' and labelled by a controllable (resp. an uncontrollable) action.

As before, we extend all these predecessor operators to arbitrary sets of states. We can now define a *safe-timed predecessors* operator $\mathsf{Pred_t}(S_1, S_2) = \{(l, v) \mid \exists d \geq 0 \text{ s.t. } (l, v) \xrightarrow{d} (l, v'), (l, v') \in S_1 \text{ and } Post_{[0,d]}(l, v) \subseteq S^* \backslash S_2\}$, where $Post_{[0,d]}(l, v) = \{(l, v') \in S^* \mid \exists t \in [0, d] \text{ s.t. } (l, v) \xrightarrow{t} (l, v')\}$.

This corresponds intuitively to the states that can reach S_1 by delay, without going through any state in S_2 along the path.

If we denote by $S_{goal} = \{l_{goal}\} \times \mathbb{R}^X_{\geq 0}$, then the backwards algorithm for solving reachability games is the fixed-point computation of:

$$W_{n+1} = S^* \cap (\mathsf{Pred_t}(W_n \cup \mathsf{cPred}(W_n), \mathsf{uPred}(S^* \backslash W_n)) \cup S_{goal}), \text{ with } W_0 = \emptyset$$

When it exists, the final fixed-point set is noted W^*. We recall the following result from [12]:

Theorem 1 ([12]). *When W^* exists, for any PGA $\mathcal{G}$ and any location l_{goal}, there exists a winning strategy for the controller in $v(\mathcal{G})$, for a parameter valuation v iff $v \in (W^* \cap \mathsf{Init}(\mathcal{G}))_{|P}$.*

The obvious problem with the above approach is that the fixed-point computation of W^* might not terminate. And indeed, already, the fixed-point computation of S^*, the reachable state-space, might not terminate either.

In [13], to solve this problem without restricting the expressiveness of the model too much, we have restricted the problem of parameter synthesis to the search for bounded integer parameters. We wanted to avoid an explicit enumeration of all the possible values of parameters and have therefore proposed a modification of the symbolic computation of S^* that preserves the integer parameter valuations.

The approach is based on the notion of *integer hull*. The integer hull $\mathsf{IntHull}(Z)$ of a convex polyhedron Z in $\mathbb{R}^n$ is the smallest convex subset of Z containing all the elements of Z with integer coordinates. If $\mathsf{Conv}(Z)$ is the smallest convex set containing Z, and $\mathsf{IntVects}(Z)$ the subset of all elements of Z with integer coordinates, then the integer hull of Z is $\mathsf{IntHull}(Z) = \mathsf{Conv}(\mathsf{IntVects}(Z))$. $\mathsf{IntHull}$ is extended to symbolic states as: $\mathsf{IntHull}((l, Z)) = (l, \mathsf{IntHull}(Z))$.

In [13], we have proved that to solve integer parameter synthesis problem it is sufficient to consider the integer hulls of the symbolic states. Therefore, in the standard algorithm for the reachability-synthesis for PTA, we replace all occurrences of the operator Succ with $\mathsf{ISucc}((l, Z), e) = \mathsf{IntHull}(\mathsf{Succ}((l, Z), e))$.

By using the ISucc instead of Succ in the computation of the whole state-space S^* we ensure termination and obtain a subset IS^* of S^* such that $\mathsf{IntVects}(IS^*) = \mathsf{IntVects}(S^*)$, provided we know a bound on the possible values for the parameters and the following assumption holds:

Assumption 1 *Any non-empty symbolic state computed through the* Succ *operator contains at least one integer point.*

From now on we place ourselves in this setting, and like in [13] we can assume w.l.o.g. that all clocks are bounded by some constant.

3.2 Bounded Integer Parameter Synthesis

In [13], in order to prove the correctness of the algorithm that uses the integer hull, we have relied on the convexity of the symbolic states in the forward computation. However, even if S_1 and S_2 are convex, $\mathsf{Pred_t}(S_1, S_2)$ is not in general. By taking the integer hull of a non-convex set, we could include some integer points that do not belong to the original set. Since we want to preserve the integer points, we define a new operator, an *integer shape*, $\mathsf{IntShape}$. As stated before, any set S produced by the backward computation can be expressed as finite unions of convex polyhedra $\bigcup_i Z_i$. For such a finite union; we define $\mathsf{IntShape}(S) = \bigcup_i \mathsf{IntHull}(Z_i)$. We can now extend the needed backwards operators using the notion of integer shape.

We first extend a predecessor by edge operator (Pred) for the computation of integer parameter valuations, similarly to the extension of Succ operator. For a symbolic state (l, Z) and an edge e, an integer predecessor by an edge e is defined as: $\mathsf{IPred}((l, Z), e) = \mathsf{IntShape}(\mathsf{Pred}((l, Z), e))$.

The following lemma states that the computation of the integer shape of a predecessor of a symbolic state (l, Z), results in the same set as if we would compute the integer shape of a symbolic state (l, Z) at first, and then the integer shape of its predecessor by edge.

Lemma 1. *For any symbolic state* (l, Z) *and any edge* e*:*
$\mathsf{IPred}(\mathsf{IntShape}((l, Z), e)) = \mathsf{IntShape}(\mathsf{Pred}((l, Z), e))$

Proof. We will prove both inclusions:

1. $\mathsf{IPred}(\mathsf{IntShape}((l, Z)), e) \subseteq \mathsf{IntShape}(\mathsf{Pred}((l, Z), e))$.

Since $\mathsf{IntShape}((l, Z)) \subseteq (l, Z)$ and $\mathsf{IntShape}$ and Pred are non-decreasing, the first inclusion holds.

2. $\mathsf{IPred}(\mathsf{IntShape}((l, Z)), e) \supseteq \mathsf{IntShape}(\mathsf{Pred}((l, Z), e))$.

Let $v \in \mathsf{IntVects}(\mathsf{Pred}((l, Z), e))$. Then $\exists v' \in Z$ s.t. $v \in \mathsf{IntVects}(\mathsf{Pred}((l, \{v'\}), e))$. By definition of Pred we have that $v_{|P} = v'_{|P}$, and therefore $v' \in v_{|P}(Z)$. $v_{|P}$ is an integer vector (since v is) and $v_{|P}(Z)$ is a zone of a classical TA and thus with integer vertices. Therefore $v_{|P}(Z) = \mathsf{IntShape}(v_{|P}(Z))$. Since $\mathsf{IntShape}$ is non-decreasing and $v_{|P}(Z) \subseteq Z$ we have $\mathsf{IntShape}(v_{|P}(Z)) \subseteq \mathsf{IntShape}(Z)$, and so $v' \in \mathsf{IntShape}(Z)$ and $v \in \mathsf{IntVects}(\mathsf{Pred}((l, \mathsf{IntShape}(Z)), e))$. Again, since $\mathsf{IntShape}$ is non-decreasing, we obtain $\mathsf{IPred}(\mathsf{IntShape}((l, Z)), e) \supseteq \mathsf{IntShape}(\mathsf{Pred}((l, Z), e))$.

We define, in a similar way, an integer controllable (resp. uncontrollable) action predecessors $\mathsf{IcPred}((l,Z)) = \mathsf{IntShape}(\mathsf{cPred}((l,Z)))$ (resp. $\mathsf{IuPred}((l,Z)) = \mathsf{IntShape}(\mathsf{uPred}((l,Z)))$.

Lemma 2. *For any symbolic state (l,Z):*
$\mathsf{IcPred}(\mathsf{IntShape}((l,Z))) = \mathsf{IntShape}(\mathsf{cPred}((l,Z)))$

Proof. Immediate with Lemma 1 from the facts that: $\mathsf{cPred}((l,Z)) = \bigcup_{c\in\Sigma^c} \mathsf{Pred}(\mathsf{IntShape}((l,Z),c)))$ and $\mathsf{IntShape}(S_1,S_2) = \mathsf{IntShape}(S_1) \cup \mathsf{IntShape}(S_2)$ when S_1 and S_2 are finite unions of convex sets.

The same result obviously holds for uPred and we can finally extend this to the safe-timed predecessor operator by $\mathsf{IPred_t}(Z_1,Z_2) = \mathsf{IntShape}(\mathsf{Pred_t}(Z_1,Z_2))$.

Lemma 3. *For any two sets of states S_1 and S_2:*
$\mathsf{IPred_t}(\mathsf{IntShape}(S_1),\mathsf{IntShape}(S_2)) = \mathsf{IntShape}(\mathsf{Pred_t}(S_1,S_2))$

Proof. Recall that S_1 and S_2 are finite unions of convex polyhedra: $S_1 = \bigcup_i Z_{1i}$ and $S_2 = \bigcup_j Z_{2j}$. By using a result from [8], we then have $\mathsf{Pred_t}(\bigcup_i Z_{1i}, \bigcup_j Z_{2j}) = \bigcup_i \bigcap_j \mathsf{Pred_t}(Z_{1i},Z_{2j})$. Then for any i,j, with another result from [8], we have: $\mathsf{Pred_t}(Z_{1i},Z_{2j}) = (Z_{1i}^{\swarrow} \setminus Z_{2j}^{\swarrow}) \cup ((Z_{1i} \cap Z_{2j}^{\swarrow}) \setminus Z_{2j})^{\swarrow}$, because Z_{2j} is convex. What we need to show then is that:
-$\mathsf{IntShape}(Z_1 \cap Z_2) = \mathsf{IntShape}(\mathsf{IntShape}(Z_1) \cap \mathsf{IntShape}(Z_2))$
-$\mathsf{IntShape}(Z_1 \cup Z_2) = \mathsf{IntShape}(\mathsf{IntShape}(Z_1) \cup \mathsf{IntShape}(Z_2))$
-$\mathsf{IntShape}(Z_1^{\swarrow}) = \mathsf{IntShape}(\mathsf{IntShape}(Z_1)^{\swarrow})$
-$\mathsf{IntShape}(Z_1 \setminus Z_2) = \mathsf{IntShape}(\mathsf{IntShape}(Z_1) \setminus \mathsf{IntShape}(Z_2))$
These four results are quite straightforward. Let us just prove the first, the rest being similar.

First remark that Z_1 and Z_2 being convex, integer shapes are actually integer hulls. Second $\mathsf{IntHull}(S) \subseteq S$, for any S and since $\mathsf{IntHull}$ is non-decreasing, $\mathsf{IntHull}(Z_1 \cap Z_2) \supseteq \mathsf{IntHull}(\mathsf{IntHull}(Z_1) \cap \mathsf{IntHull}(Z_2))$.

Now let $v \in \mathsf{IntVects}(\mathsf{IntHull}(Z_1))$ then, by definition, $v \in \mathsf{IntVects}(Z_1)$ and if it also belongs to $\mathsf{IntVects}(\mathsf{IntHull}(Z_1))$ then it is in $\mathsf{IntVects}(Z_1) \cap \mathsf{IntVects}(Z_2)$ or equivalently in $\mathsf{IntVects}(Z_1 \cap Z_2)$ and by taking the convex hull, we have the result.

Now consider the following fixed-point computation:

$$IW_{n+1} = IS^* \cap (\mathsf{IPred_t}(IW_n \cup \mathsf{IcPred}(IW_n), \mathsf{IuPred}(IS^* \setminus IW_n)) \cup S_{goal}),\ IW_0 = \emptyset$$

Lemma 4. *For a PGA $\mathcal{G}$, location l_{goal}, and a state (l,v) such that v is an integer valuation, it holds that for $\forall i$, there exist a winning strategy in at most i controllable steps from $(l,v_{|X})$ in $v_{|P}(\mathcal{G})$ iff $(l,v) \in IW_i$.*

Proof. We proceed by induction. The property obviously holds for IW_0. Now, suppose it holds for some $n \geq 0$. We first prove the left to right implication. Let (l,v) be a state in IW_{n+1} such that v is an integer point. then $(l,v) \in IS^* \cap (\mathsf{IPred_t}(\mathsf{IntShape}(IW_n \cup \mathsf{cPred}(IW_n)), \mathsf{IntShape}(uPr(IS^* \setminus IW_n))) \cup S_{goal})$.

If $(l, v) \in S_{goal}$ we are done, else, by Lemma 3, we know that $(l, v) \in IS^* \cap \mathsf{IntShape}(\mathsf{Pred_t}(IW_n) \cup \mathsf{cPred}(IW_n), \mathsf{uPred}(IS^* \setminus IW_n))$ and then in $\mathsf{Pred_t}(IW_n) \cup \mathsf{cPred}(IW_n), \mathsf{uPred}(IS^* \setminus IW_n)$ and we get the result using the correctness of the $\mathsf{Pred_t}$ operator and the induction hypothesis.

Now, we prove the right to left implication. If there is a strategy to win in at most $n+1$ steps, there is one to reach some state (l', v') in one step and win in at most n steps. Then by the induction hypothesis, $(l', v') \in IW_n$ and by the correctness of the $\mathsf{Pred_t}$ operator, (l, v) belongs to $\mathsf{Pred_t}(IW_n) \cup \mathsf{cPred}(IW_n), \mathsf{uPred}(IS^* \setminus IW_n)$. Since v is an integer valuation, (l, v) belongs to $\mathsf{IntShape}(\mathsf{Pred_t}(IW_n) \cup \mathsf{cPred}(IW_n), \mathsf{uPred}(IS^* \setminus IW_n))$ and we can conclude by Lemma 3.

We now prove that the fixed-point computation IW_n terminates and that its result IW^* is correct and complete.

Theorem 2 (Termination). *For any PGA $\mathcal{G}$ and any desired location l_{goal}, the algorithm terminates.*

Proof. We proved, in [13], that the forward computation of IS^* terminates. When going backwards, each time we apply $\mathsf{IPred_t}$ we know that we have added to the winning set of states at least one integer point (otherwise we can terminate). Since there is only a finite number of integer points to add (due to the boundedness of clocks and parameters), the computation terminates.

Theorem 3 (Correctness and completeness). *Let v be an integer parameter valuation. For any PGA $\mathcal{G}$ and any location l_{goal}, upon termination, there exists a winning strategy for the controller in $v(\mathcal{G})$ iff $v \in (IW^* \cap \mathsf{Init}(\mathcal{G}))_{|P}$.*

Proof. We start by proving the right to left implication. Suppose $v \in (IW^* \cap \mathsf{Init}(\mathcal{G}))_{|P}$. Then there exists a state (l_0, v_0) in $IW^* \cap \mathsf{Init}(\mathcal{G})$ such that $v_{0|P} = v$ and $v_{0|X}$ has all coordinates equal to 0. So v_0 is an integer valuation on $X \cup P$ and since it belongs IW^*, it belongs to IW^n for some n. So we can apply Lemma 4 to conclude.

Now, we prove the left to right implication. If there exists a winning strategy for the controller to win in $v(\mathcal{G})$ then it means that it can win within a finite number of controllable steps. Then, by Lemma 4, it means that the state (l_0, v_0) such that $v_{0|P} = v$ and $v_{0|X}$ has all coordinates equal to 0 belongs to IW^n for some n, and therefore to IW^*, which concludes the proof.

3.3 Avoiding Integer Hulls in the Backward Computation

We have shown how we can symbolically compute the bounded integer parameter valuations that permit the controller to win. We will now prove that, surprisingly, we actually do not have to apply an integer hull in the backwards computation, in order to obtain the correct integer solution and ensure termination.

Consider the fixed-point computation corresponding to this setting:

$$IW'_{n+1} = IS^* \cap (\mathsf{Pred_t}(IW'_n \cup \mathsf{cPred}(IW'_n), \mathsf{uPred}(IS^* \setminus IW'_n)) \cup S_{goal}),\ IW'_0 = \emptyset$$

Let us first show that if this procedure terminates, it is sound and complete.

Theorem 4 (Correctness and completeness). *For any PGA $\mathcal{G}$, a desired location l_{goal}, and an integer parameter valuation v, upon the termination, there exists a winning strategy for the controller in $v(\mathcal{G})$ iff $v \in (IW'^* \cap \mathsf{Init}(\mathcal{G}))_{|P}$.*

Proof. First remark that, for all n, we have $IW_n \subseteq IW'_n \subseteq W_n$, because integer hulls and shapes only remove points. Now, $\mathsf{IntVects}$ is a non-decreasing operator so $\mathsf{IntVects}(IW_n) \subseteq \mathsf{IntVects}(IW'_n) \subseteq \mathsf{IntVects}(W_n)$. By Lemma 3 and correctness of the W_n computation, we know that $\mathsf{IntVects}(IW_n) = \mathsf{IntVects}(W_n)$. So all three sets are equal.

Now, as seen in the proof of Theorem 3, any initial state in $v(\mathcal{G})$ for an integer parameter valuation v certainly has an integer valuation of both clocks and parameters, which permits us to conclude.

Proving the termination is much trickier: we shall construct a new object, that we call *parametric region graph*, that refines the standard region graph for timed automata of [1] with parametric constraints. We therefore further divide the region graph with all the guards from the model and all the constraints defining integer hulls (of the symbolic states) obtained in the forward computation.

A constraint of an integer hull may create a non-integer vertex when intersecting the region graph. For each such vertex, we add constraints that go though the vertex and are parallel to the diagonal constraints of the region graph (added constraints are of the form $x_i - x_j + k = 0$ for all clocks x_i, x_j and $k \in \mathbb{Z}$).

Note that if a constraint of an integer hull intersects a diagonal constraint creating a non-integer vertex, there is already a diagonal constraint going through that vertex. We now give a formal definition of the parametric region graph.

Definition 8 (Parametric regions). *Let m be the maximal value of parametric expressions occurring in the constraints of the PGA (recall that parameters are bounded so it is possible to compute that value). The parametric region graph is constructed in the following way:*

- *the variable-space $\mathbb{R}^{X \cup P}$ is partitioned along the constraints $x \sim k$ and $x - y \sim k$ for all clocks $x, y \in X$, $\backsim \in \{<, =, >\}$ and $0 \leq k \leq m$ (this gives the standard region graph);*
- *for any guard of the automaton and any constraint appearing in the (finite number of) convex polyhedra defining IS^*, $\gamma \backsim 0$, we further partition the variable-space, with the constraints $\gamma \backsim' 0$ for every $\backsim' \in \{<, =, >\}$;*
- *for any (non-integer) vertex of the "pre-regions" defined by the above partition, we further refine the variable-space by constraints of the form $x - y + k \backsim 0$, with $\backsim \in \{<, =, >\}$ for all clocks $x \neq y$ and $k \in \mathbb{Z}$, going through that vertex.*

Informally, these constraints partition the variable-space $\mathbb{R}^{X \cup P}$ into a finite number of a *parametric regions*, which are either a vertex, a line fragment between two vertices, or a sub-space between line fragments and vertices that does not contain either a line fragment nor a vertex.

Fig. 1 shows a two-dimensional example of the variable-space partitioned into parametric regions. An integer hull obtained in a forward computation is

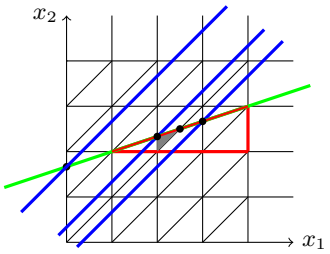

Fig. 1. Parametric region partition

drawn in red. Its side (that does not overlap with the region graph) is extended (in green) as long as it cuts the state-space. Additionally, each non-integer vertex obtained in the intersection of the constraints defining the integer hull and the region graph, has a diagonal constraint that goes through it (in blue). An example of a parametric region is given in gray.

In order to prove the termination when going backwards, we have to show that all the operators preserve the parametric regions.

Lemma 5. *If $(a_i)_i$ and $(b_j)_j$ are finite families of parametric regions then the following sets are a finite union of parametric regions:*

1. $\bigcup_i a_i \cup \bigcup_j b_j$;
2. $\bigcup_i a_i \cap \bigcup_j b_j$;
3. *the complement of* $\bigcup_i a_i$;
4. $(\bigcup_i a_i)^{\swarrow}$;
5. $\mathsf{Pred}((l, \bigcup_i a_i), e)$, *for any location l and edge e;*
6. $\mathsf{Pred_t}(\bigcup_i a_i, \bigcup_j b_j)$.

Proof. The first three are straightforward using the fact that parametric regions are taken from a finite set. The fourth uses the fact that these regions are defined using the diagonal constraints $x - y - k \backsim 0$ going through any vertex. The fifth is immediate. For $\mathsf{Pred_t}$ we need to use once again the two results from [8]: $\mathsf{Pred_t}(\bigcup_i a_i, \bigcup_j b_j) = \bigcup_i \bigcap_j \mathsf{Pred_t}(a_i, b_j)$ and $\mathsf{Pred_t}(a_i, b_j) = (a_i^{\swarrow} \setminus b_j^{\swarrow}) \cup ((a_i \cap b_j^{\swarrow}) \setminus b_j)^{\swarrow}$ (if b_j is convex, which is true by definition of the parametric region), which can equivalently be written as: $\mathsf{Pred_t}(a_i, b_j) = (a_i^{\swarrow} \cap \overline{b_j^{\swarrow}}) \cup (a_i \cap b_j^{\swarrow} \cap \overline{b_j})^{\swarrow}$. We can then conclude by using the first four results.

We can now get back to the termination of the IW'_n fixed-point computation:

Theorem 5 (Termination). *For any PGA $\mathcal{G}$ and any desired location l_{goal} the fixed-point computation of IW'^* terminates.*

Proof. Again we know, from [13], that the forward computation of IS^*, using ISucc and for bounded parameters, terminates. By definition of parametric regions, IS^* can be written as a finite union of symbolic states whose associated

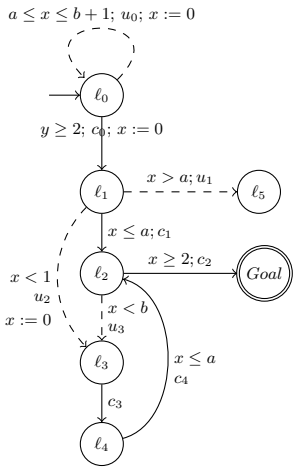

Fig. 2. A Parametric Timed Game Automaton

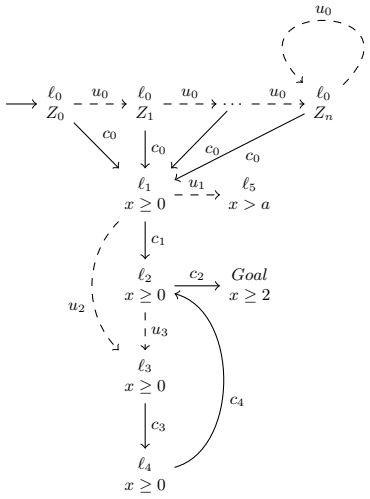

Fig. 3. Symbolic state graph of PGA of Figure 2

valuations can be represented as finite unions of parametric region. When going backwards using $\mathsf{Pred_t}$, by Lemma 5 we know that parametric regions are preserved. Therefore at each step at least one region is added to the set of winning states (otherwise the fixed-point is reached and we can terminate). Since there is a finite number of parametric regions, the computation must terminate.

3.4 Complexity

As remarked in [13], all of the possible valuations of parameters, that are integer and bounded, can be enumerated in exponential time. Therefore, a problem that is EXPTIME for TGA, the corresponding bounded integer version for PGA is also EXPTIME. The reachability game is EXPTIME-complete for TGA [14], and it is a special case of the reachability-emptiness game for PGA. We can thus conclude that the reachability-emptiness (synthesis) game for PGA is EXPTIME-complete for bounded integer parameters.

4 Example

We consider an extension of the example proposed in [8]. The model has two clocks x and y, controllable (c_i) and uncontrollable (u_i) actions and two parameters a and b. The reachability game consists in finding a strategy for the controller that will eventually end up in the location *Goal*. We will now explain how the algorithm works.

A PGA is given in Figure 2 and its symbolic state graph (graph with nodes (l, Z)) in Figure 3. The initial symbolic state is: (ℓ_0, Z_0) with $Z_0 = \{x = y, x \ge$

$0, y \geq 0\}$. After n loops u_0, we have : (ℓ_0, Z_n) with $Z_n = \{x \geq 0, a \leq b+1, 0 \leq na \leq y - x \leq n(b+1)\}$.

We now ensure the termination in the bounded case. For example, assume all parameters and clocks are less than or equal to 3 (i.e. in each symbolic state we implicitly have $x \leq 3, y \leq 3, a \leq 3, b \leq 3$) then:

-After one loop: $Z_1 = \{a \leq 3, a \leq b+1, a \leq y - x \leq b+1\}$

-After three loops: $Z_3 = \{a \leq 1, a \leq b+1, 3a \leq y - x \leq 3(b+1)\}$

-After $n > 3$ loops: $Z_n = Z_{n+1} = \{a = 0, a \leq b+1, y - x \leq 3\}$

After the transition c_0, the reset of the clock x removes the diagonal constraint involving $y - x$ in $Z_0 \ldots Z_n$ and the new constraints $y \geq 2$ and $y - x \geq 2$ are added . All theses zones obtained from $Z_0 \ldots Z_n$ are included in $(x \geq 0) \wedge (y \geq 2) \wedge (y - x \geq 2)$. For the sake of conciseness, in the sequel, we omit $y \geq 2$ and $y - x \geq 2$ in the symbolic states associated with locations ℓ_1 and its successors.

After the computation of the symbolic states, shown in Figure 3, the backward algorithm starts from the symbolic winning subset $(Goal, x \geq 2)$. By a controllable action (c_2) predecessor, we obtain $(\ell_2, x \geq 2)$. Computing the (timed) past removes the constraint $x \geq 2$, and computing the safe-timed predecessor adds $x \geq b$ in order not to end-up in ℓ_3 by $u3$. The resulting state is $(\ell_2, x \geq b)$. One of the controllable transitions taking us to ℓ_2 is c_4. A controllable action predecessor (c_4) adds a constraint $x \leq a$. A constraint on the parameters derived in this state is $a \geq b$. This contraint is back-propagated to the preceding states. The safe-timed predecessors give us the state $(\ell_4, x \geq 0 \wedge a \geq b)$.

We obtain successively the following sets of winning states: $(\ell_3, x \geq 0 \wedge a \geq b)$, $(\ell_2, (x \geq b) \vee (x \geq 0 \wedge a \geq b))$ and $(\ell_1, (x \leq a) \wedge \big((x < 1 \wedge a \geq b) \vee x \geq 1\big) \wedge \big((x \geq b) \vee (x \geq 0 \wedge a \geq b)\big)$. The last one simplifies to $(\ell_1, (x \leq a \wedge a \geq b))$. The constraints are now back-propagated to the states associated with ℓ_0. The constraint $a > 0$ is added in order not to end-up in the symbolic state (ℓ_0, Z_n) by u_0 then the winning states obtained from (ℓ_0, Z_n) is $(\ell_0, \overline{(a \leq b+1) \wedge (a = 0)})$, from (ℓ_0, Z_3) is $(\ell_0, \overline{(a \leq b+1) \wedge (a \leq 1)}) \ldots$ and from (ℓ_0, Z_0) is $(\ell_0, \overline{(a \leq b+1)})$. We finally obtain $(\ell_0, (a \geq b) \wedge \Big((a > b+1) \vee \big((a \leq b+1) \wedge (a > 0)\big)\Big))$. Thus, there exists a winning strategy if and only if $(a \geq b) \wedge \big((a > b+1) \vee (a > 0)\big)$.

It is now easy to extract the memory-less winning strategy from the set of winning states as follows: a controllable action predecessor gives us the state from which a corresponding controllable action should be taken, while safe-timed predecessor further gives us the state where we should delay. Thus, a whole winning strategy consists in: delaying in all states $(\ell_0, \{y < 2\})$, doing c_0 in all states $(\ell_0, \{y \geq 2\})$, doing c_1 in all states $(\ell_1, \{x \leq a\})$, delaying in all states $(\ell_2, \{x < 2\})$, doing c_2 in all states $(\ell_2, \{x \geq 2\})$, doing c_3 in all states $(\ell_3, \{x \geq 0\})$, delaying in all states $(\ell_4, \{x < b\})$, doing c_4 in all states $(\ell_4, \{x \geq b \wedge x \leq a\})$.

5 Conclusion

In this paper we have proposed an extension of a method introduced in [13], for the computation of winning states and synthesis of bounded integer parameters

for parametric timed game automata. The method is symbolic and it is based on the computation of the integer hull of the bounded parametric symbolic states. In order to find the winning states, we have extended the standard fixed-point backwards algorithm for solving timed reachability games, for the parametric domain. Surprisingly, we do not have to apply an integer hull in the backwards computation, in order to obtain the correct integer solution. In future, we plan to extend this work to other timed models, such as PTA with stopwatches, as well as to look for less restrictive codomains for parameter valuations.

References

1. R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.
2. R. Alur, T. A. Henzinger, and M. Y. Vardi. Parametric real-time reasoning. In *ACM Symposium on Theory of Computing*, pages 592–601, 1993.
3. E. André, T. Chatain, E. Encrenaz, and L. Fribourg. An inverse method for parametric timed automata. In *RP workshop on Reachability Problems*, volume 223, pages 29–46, Liverpool, U.K., 2008.
4. E. Asarin, O. Maler, A. Pnueli, and J. Sifakis. Controller synthesis for timed automata. In *Proc. IFAC SSSC*. Elsevier, 1998.
5. G. Behrmann, A. Cougnard, A. David, E. Fleury, K. G. Larsen, and D. Lime. Uppaal-tiga: Time for playing games! In *CAV*, pages 121–125, 2007.
6. L. Bozzelli and S. L. Torre. Decision problems for lower/upper bound parametric timed automata. *Formal Methods in System Design*, 35(2):121–151, 2009.
7. V. Bruyère and J.-F. Raskin. Real-time model-checking: Parameters everywhere. *Logical Methods in Computer Science*, 3(1):1–30, 2007.
8. F. Cassez, A. David, E. Fleury, K. Larsen, and D. Lime. Efficient on-the-fly algorithms for the analysis of timed games. In *CONCUR'05*, volume 3653 of *LNCS*, 2005.
9. F. Cassez, J. J. Jessen, K. G. Larsen, J.-F. Raskin, and P.-A. Reynier. Automatic synthesis of robust and optimal controllers - an industrial case study. In *HSCC*, pages 90–104, 2009.
10. T. A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Inform. and Computation*, 111(2):193–244, 1994.
11. T. Hune, J. Romijn, M. Stoelinga, and F. Vaandrager. Linear parametric model checking of timed automata. *Journal of Logic and Algebraic Programming*, 52-53:183–220, 2002.
12. A. Jovanović, S. Faucou, D. Lime, and O. H. Roux. Real-time control with parametric timed reachability games. In *Proc. of WODES'12*, pages 323–330. IFAC, Oct. 2012.
13. A. Jovanović, D. Lime, and O. H. Roux. Integer parameter synthesis for timed automata. In *Proc. of TACAS'13*, volume 7795 of *LNCS*, pages 391–405. Springer, Mar. 2013.
14. M. Jurdzinski and A. Trivedi. Reachability-time games on timed automata. In *Proc. of ICALP 2007*, volume 4596 of *LNCS*, pages 838–849. Springer, July 2007.
15. O. Maler, A. Pnueli, and J. Sifakis. On the synthesis of discrete controllers for timed systems. In *STACS '95*, 1995.