Real-Time Control with Parametric Timed Reachability Games

Aleksandra Jovanovic, Sébastien Faucou, Didier Lime, Olivier Henri Roux

Real-Time Control with Parametric Timed Reachability Games [⋆]

A. Jovanović [**] **S. Faucou** [***] **D. Lime** [**] **O. H. Roux** [**]

LUNAM Université.
[**] École Centrale de Nantes (e-mail:
aleksandra.jovanovic@irccyn.ec-nantes.fr,
didier.lime@irccyn.ec-nantes.fr, olivier-h.roux@irccyn.ec-nantes.fr).
[***] Université de Nantes (e-mail: sebastien.faucou@univ-nantes.fr)
IRCCyN UMR CNRS 6597

Abstract: Timed game automata are used for solving control problems on real-time systems. A timed reachability game consists in finding a strategy for the controller for the system, modeled as a timed automaton. Such a controller says when and which of some "controllable" actions should be taken in order to reach "goal" states. We deal with a parametric version of timed game automata. We define parametric timed reachability games and introduce their subclass for which the existence of a parameter valuation, such that there is a strategy for the controller to reach the "goal" state, is decidable. We also propose a semi-algorithm to symbolically compute the corresponding set of parameter valuations.

Keywords: Timed automata, game theory, parameters, control, verification, model-checking

1. INTRODUCTION

Formal methods are widely used in the analysis of time critical systems. For instance, methods such as model-checking allow the verification of a system by exploring the state-space of a (timed) model, e.g. a timed automaton. A prerequisite for these methods is the availability of a complete model of the system. Thus, it can be difficult to use them at early design stages, when the whole system is not fully characterized.

It is sometimes possible to overcome this problem by using parameters for modeling values that are not fully characterized yet. To exploit such models, a parametric approach in automata theory must be used. The analysis of a parametric model produces symbolic constraints on the parameters that ensure the correctness of the system.

Parametric control problem The *verification* problem for a given model of the system S and a model of the specification φ, consists in checking whether S satisfies φ, which is often written $S \models \varphi$ and referred to as the *model-checking problem.* The *parametric model-checking problem* for a parametric system S and a parametric specification φ, consists in checking whether there exists a valuation v of the parameters such that S satisfies φ for this valuation, which is written $[\![S]\!]_v \models [\![\varphi]\!]_v$.

The *control problem* assumes the system is *open* i.e. we can restrict the behaviour of S: some events in S are *controllable* and the others are *uncontrollable*, and we can sometimes disable controllable actions. The *control problem* for a system S and a specification φ asks the following: is there a controller C s.t. $S \times C \models \varphi$? The *parametric control problem* for a parametric system S and a parametric specification φ asks the following: is there a controller C and a valuation v of the parameters s.t. $[\![S]\!]_v \times C \models [\![\varphi]\!]_v$? The associated *parametric controler synthesis problem* asks to compute a witness controller C and the set of valuations V such that $\forall v \in V$, $[\![S]\!]_v \times C \models [\![\varphi]\!]_v$.

The control problem can be formulated as a game in which the controller plays against the environment.

Related Work Parametric timed automaton (PTA) has been introduced in Alur et al. (1993) as an extension of the timed automaton, Alur and Dill (1994), that allows the use of parameters instead of concrete values in the clock constraints. The emptiness problem for PTA asks: is the set of parameter valuations, such that an automaton has an accepting run, empty? The main result is that this problem is undecidable in general except for some restricted cases. In Hune et al. (2002), the authors introduce a subclass of parametric timed automata, called lower bound/upper bound (L/U) automata, for which the emptiness problem is decidable.

Timed game automata (TGA), Maler et al. (1995), were introduced for solving control problems on real-time systems. TGA are based on the framework of Ramadge and Wonham (1989), developed for the control of the discrete event processes. A timed game automaton is essentially a timed automaton whose set of actions is partitioned into controllable and uncontrollable actions. Two players, a controller and an environment, choose at every instant one of the available actions from their own sets and the game progresses. A *timed reachability game* consists in finding a strategy for the controller: when and which of the controllable actions should be taken such that, regardless of what the environment does, the system ends up in

[⋆] This work was supported by project ANR-2010-BLAN-0317.

a desired location. Such timed games are known to be decidable, Maler et al. (1995); Asarin et al. (1998).

Our Contribution We first define parametric timed game automata (PGA) and introduce their subclasses for which we prove the decidability of the emptiness problem for parametric timed reachability game i.e. the existence of a parameter valuation, such that there is a strategy for the controller to reach the "goal" state. We then propose the extension for the parametric case of the algorithm of Cassez et al. (2005) for solving timed games. It leads to a semi-algorithm that symbolically computes the set of corresponding parameter valuations. We end with a case study that illustrates the use of the new subclass and proposed algorithm.

2. PARAMETRIC TIMED GAMES

2.1 Basic Definitions

Parametric constraints Let $X = \{x_1, ...x_m\}$ be a finite set of variables modeling *clocks*. $\mathbb{R}$ is the set of real numbers, $\mathbb{Q}$ the set of rational numbers, and $\mathbb{Z}$ the set of integers. A *clock valuation* is a function $w : X \mapsto \mathbb{R}_{\geq 0}$ assigning a non-negative real value to each clock. Let $P = \{p_1, ...p_n\}$ be a finite set of *parameters*. A *parametric clock constraint* c is an expression of the form $c ::= x_i - x_j \backsim p \mid x_i \backsim p \mid c \wedge c$, 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 $k_0, ...k_n \in \mathbb{Z}$. A *parameter valuation* is a function $v : P \mapsto \mathbb{Q}$ assigning a rational value to each parameter. For any parametric clock constraint c and any parameter valuation v, we note $[\![c]\!]_v$ the constraint obtained by replacing each parameter p_i by its valuation $v(p_i)$. A pair (w, v) of a clock valuation and a parameter valuation satisfies a parametric constraint c, notation $(w, v) \models c$, if the expression $c[w, v]$, obtained by replacing each parameter p_i with $v(p_i)$ and each clock x_i with $w(x_i)$, evaluates to true. We denote by $G(X)$ the set of parametric constraints over X, and $G'(X)$ a set of parametric constraints over X of the form $c' ::= x_i \backsim p \mid c' \wedge c'$. If w is a clock valuation and $t \in \mathbb{R}_{\geq 0}$, we write $w + t$ for the valuation assigning $w(x) + t$ to each clock. For $R \subseteq X$, $w[R]$ denotes a valuation assigning 0 to each clock in R and $w(x)$ to each clock in $X \backslash R$.

Note that given an arbitrary order on $X \cup P$, any valuation (w, v) can be identified to a point in $\mathbb{R}^{|P|+|X|}$, and the set of valuations (w, v) such that some parametric constraint c is true is then a convex polyhedron in that space.

Parametric Timed Automata Parametric clock constraints are used as guards and invariants in parametric timed automata.

Definition 1. A Parametric Timed Automaton (PTA) is a tuple $\mathcal{A} = (L, l_0, X, \Sigma, P, E, I)$, where L is a finite set of locations, $l_0 \in L$ is an initial location, X is a finite set of clocks, Σ is a finite alphabet of actions, P is a finite set of parameters, $E \subseteq L \times \Sigma \times G(X) \times 2^X \times L$ is a finite set of transitions, and $I : L \mapsto G'(X)$ is a function that assigns an invariant to each location.

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

- $Q = \{(l, w) \in L \times (X \mapsto \mathbb{R}_{\geq 0}) \mid (w, v) \models I(l)\}$
- $q_0 = \{(l, w_0) \in Q \mid l = l_0 \wedge w_0 : X \mapsto 0\}$
- the transition relation $\rightarrow$ is defined as: $\forall (l, w), (l', w') \in Q$, $t \geq 0$ and $a \in \Sigma$:

$$\text{delay transition: } (l, w) \xrightarrow{t} (l, w') \text{ if } w' = w + t$$

$$\text{action transition: } (l, w) \xrightarrow{a} (l', w') \text{ if } \exists g \in G(X),$$
$$R \subseteq X : l \xrightarrow{a,g,R} l' \text{ and } (w, v) \models g \text{ and } w' = w[R]$$

A run of a parametric timed automaton $\mathcal{A}$ under a parameter valuation v, is a sequence of alternating delay and action transitions in the semantics of $[\![\mathcal{A}]\!]_v$. $\mathit{Runs}((l, w), [\![\mathcal{A}]\!]_v)$ denotes the set of runs starting in (l, w), and $\mathit{Runs}([\![\mathcal{A}]\!]_v)$ denotes the set of runs starting in the initial state (l_0, w_0). If ρ is a finite run, we denote by $\mathit{last}(\rho)$ the last state of ρ.

A state (l, w) is said to be *reachable* if there exists a finite run $\rho \in \mathit{Runs}([\![\mathcal{A}]\!]_v)$ such that $\mathit{last}(\rho) = (l, w)$.

L/U Automata A parameter p_i occurs positively (resp. negatively) in a parametric constraint $x_i - x_j \sim k_0 + k_1 p_1 + ... + k_n p_n$, with $\sim \in \{\leq, <\}$, if $k_i > 0$ (resp. $k_i < 0$). A *lower bound* (resp. *upper bound*) parameter of a parametric timed automaton $\mathcal{A}$ is a parameter that only occurs negatively (resp. positively) in the parametric constraints of $\mathcal{A}$.

Definition 3. (L/U automaton Hune et al. (2002)). A L/U automaton is a parametric timed automaton where every parameter is either a lower or an upper bound parameter.

Let $\mathcal{A}$ be a non-parametric timed automaton. Weakening the guards and invariants in $\mathcal{A}$ (decreasing the lower bounds and increasing the upper bounds on clocks) yields an automaton whose reachable states include those of $\mathcal{A}$, and strengthening the guards and invariants in $\mathcal{A}$ (increasing the lower bounds and decreasing the upper bounds on clocks) yields an automaton whose reachable states are a subset of those of $\mathcal{A}$. Following this fact, decidability of the reachability-emptiness problem (existence of a parameter valuation such that a certain state in the automaton is reachable from the initial state) has been established for L/U automata (detailed proof in Hune et al. (2002)).

2.2 Parametric Timed Games

We now extend the previous definitions to obtain a more powerful formalism that allows us to express control problems. To this end we will parametrize the classical notion of timed games.

Definition 4. 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 parameter valuation v, we obtain a timed game automaton $[\![\mathcal{G}]\!]_v$ by replacing each parametric constraint c in the guards and invariants by $[\![c]\!]_v$.

In a timed game, each of the two players acts within its set of actions according to a *strategy*. The game being symmetric, we only give the definitions for Player 1 (the controller playing with actions in Σ^c). Since we consider only reachability properties, as in Maler et al. (1995); Cassez et al. (2005), invariants restrict the behavior of the automaton but never force a player to move.

Definition 5. (Strategy). A strategy $\mathcal{F}$ over $[\![\mathcal{G}]\!]_v$ is a partial function from $Runs([\![\mathcal{G}]\!]_v)$ 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 $last(\rho) \xrightarrow{d'} q$.

A strategy therefore tells each player, given the history of the game, whether to play one of its actions or to wait (delay). We consider only memory-less strategies, where $\mathcal{F}(\rho)$ only depends on the current state $last(\rho)$, and which are sufficient for timed reachability games, Asarin et al. (1998), and, as we will show, for our parametric extension.

It should be noted that the uncontrollable actions cannot be relied on to reach the desired state: the controller has to be able to reach the desired locations by itself.

An example, shown in Figure 2, and a case-study, presented in Section 5, illustrate the strategy to reach a desired location.

The restricted behaviour of $[\![\mathcal{G}]\!]_v$ when the controller plays the strategy $\mathcal{F}$ against all possible strategies of the environment is defined by the notion of *outcome.*

Definition 6. (Outcome). Let $\mathcal{G}$ be a PGA, v be a parameter valuation, and $\mathcal{F}$ be a strategy over $[\![\mathcal{G}]\!]_v$. The outcome $Outcome(q, \mathcal{F})$ of $\mathcal{F}$ from state q is the subset of finite runs in $Runs(q, [\![\mathcal{G}]\!]_v)$ defined inductively as:

- the run with no action $q \in Outcome(q, \mathcal{F})$
- if $\rho \in Outcome(q, \mathcal{F})$ then $\rho' = \rho \xrightarrow{\delta} q' \in Outcome(q, \mathcal{F})$ if $\rho' \in Runs(q, [\![\mathcal{G}]\!]_v)$ 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. $last(\rho) \xrightarrow{\delta'} q'' \wedge \mathcal{F}(\rho \xrightarrow{\delta'} q'') = \mathsf{delay}$.
- for an infinite run ρ, $\rho \in Outcome(q, \mathcal{F})$, if all the finite prefixes of ρ are in $Outcome(q, \mathcal{F})$.

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

We can now define the notion of *winning strategy.*

Definition 7. (Winning strategy). Let $\mathcal{G} = (L, l_0, X, \Sigma^c \cup \Sigma^u, P, E, I)$ be a PGA and $\mathsf{goal} \in L$. A strategy $\mathcal{F}$ is winning for the location goal if for all runs $\rho \in MaxOut(q_0, \mathcal{F})$, where $q_0 = (l_0, w_0)$, there is some state (goal, w) in ρ.

Similarly, a state s is winning (for the controller) if it belongs to a run in the outcome of a winning strategy. In the parametric case, non-existance of a winning strategy can be solved by the modification of a model, as illustrated in the case-study, Section 5.

Definition 8. Emptiness problem for parametric timed reachability game for PGA is the problem of determining whether the set of parameter valuation, such that there is a strategy for the controller to reach the desired state, is empty.

The emptiness problem for PTA is known to be undecidable, Alur and Dill (1994). As PGA extend PTA, the following theorem stands.

Theorem 1. The emptiness problem for parametric timed reachability game for PGA is undecidable.

We accordingly extend the subclass of L/U automata of Hune et al. (2002) to define a subclass of parametric game automata for which this problem is decidable.

3. L/U REACHABILITY TIMED GAMES

The parameters are partitioned into two sets. The first set P^l contains parameters that are used as lower bounds in the guards on the controllable transitions and as upper bounds in the guards of the uncontrollable transitions. The parameters from the other set, P^u, are used as upper bounds in the controllable and lower bounds in the uncontrollable transitions. This is a natural way of making the controller more powerful by restricting possible behaviors of the environment (and vice-versa). We assume that invariants are non-parametric constraints.

Definition 9. (L/U game automata). A L/U game automaton $\mathcal{G} = (L, l_0, X, \Sigma^c \cup \Sigma^u, P, E, I)$ is a parametric game automaton in which:

- the set of parameters P is partitioned as P^l and P^u;
- each parameter $p \in P^l$ occurs only negatively (resp. positively) in the guards of controllable (resp. uncontrollable) transitions;
- each parameter $p \in P^u$ occurs only positively (resp. negatively) in the guards of controllable (resp. uncontrollable) transitions;
- for each location l, $I(l)$ contains no parameter.

We will now prove that the existence of a parameter valuation, such that the controller has a winning strategy, is decidable for L/U games.

Let (λ, μ) represent a parameter valuation such that λ applies to lower bound parameters, and μ applies to upper bound parameters. Let $\mathcal{G}[(0, \infty)]$ represent the (extended) parameter valuation of $\mathcal{G}$ such that each parameter $p_i^u \in P^u$ is set to ∞, and each parameter $p_i^l \in P^l$ is set to 0. In this way we obtain a timed game automaton, for which the existence of a winning strategy is known to be decidable.

Lemma 1. Let $\mathcal{G}$ be a L/U game automaton. There exists a parameter valuation (λ, μ) such that a goal state is enforceable in $[\![\mathcal{G}]\!]_{(\lambda,\mu)}$ with a strategy $\mathcal{F}$, if and only if a goal state is enforceable in $\mathcal{G}[(0, \infty)]$ using the same strategy $\mathcal{F}$.

Proof. We are searching for the value for upper bound parameters such that all runs of a given strategy $\mathcal{F}$ remain winning. For lower bound parameters, zero valuation is a possible solution. Note that each run of a *winning* strategy necessarily reaches a goal state in finite time and with a finite number of discrete actions.

To find the upper bound value, we introduce in the system a new clock x_0 that serves only to measure the time elapsed from the start. Consider first the untimed runs of $\mathcal{F}$. Since there is a finite number of edges, there is a finite number of possible untimed runs. For each of those runs we measure its *sufficient* duration in a timed case of $\mathcal{G}[(0, \infty)]$. A sufficient duration is obtained when the uncontrollable transitions are taken as late as possible in each state (which means just before we should take a supposed controllable transition in that state) and the controllable transitions are taken when $\mathcal{F}$ says. Since $\mathcal{F}$ is winning, each run i will reach a goal location at some time $x_0^i = T_i$.

We take the maximal value, T_i^{max}, for the upper bound parameters, and 0 for the lower bound parameters. In this valuation, no matter what the environment does, all guards will be satisfied so that we can take the controllable actions at the supposed time. Hence, all runs of the strategy $\mathcal{F}$ remain winning. □

Based on Lemma 1, we can formulate the following theorem.

Theorem 2. The emptiness problem for parametric timed reachability game for L/U game automata is decidable.

If we think in terms of control it may not be very realistic to be allowed to forbid uncontrollable transitions using the values of their parameters. It may actually even seem a bit far-fetched to parametrize uncontrollable actions at all. A consequence of the previous result however, is that for the subclass of L/U game automata with no parameter in the guards of uncontrollable transitions, the problem of the emptiness of the set of valuations such that the controller has a winning strategy is decidable too.

In the context of games however, having a game as symmetric as possible, including parametrization of guards makes sense. We will now explore the case in which we nonetheless impose that the parameter valuations never set the guards of the uncontrollable transitions to false: we can restrict their behaviour but not to the point of uniformly forbidding the transition.

Consequently, all the guards on the uncontrollable transitions that contain a parameter as a lower (resp. upper) bound have to contain a constant as a non-strict upper (resp. lower) bound. Non-strict inequalities are mandatory so that a clock can take the value equal to the constant as a single time point in the emptiness test. The guards on the controllable transitions have no other restriction than the L/U condition.

We also limit the parametric linear expression in the constraints of uncontrollable transitions to just one parameter.

Definition 10. (Restricted L/U game automata). A restricted L/U game automaton is a L/U game automaton in which the guards of the uncontrollable transitions are constraints of the form $k \leq x \leq Ka$ or $Ka \leq x \leq k$, where $x \in X$, $a \in P$, $k \in \mathbb{Q}$ and $K \in \mathbb{Z}$.

We will now establish the decidability of the emptiness problem for this subclass of L/U game automata. Recall that the parameters $p_i^u \in P^u$ (resp. $p_i^l \in P^l$) are used as the lower (resp. upper) bounds in the guards on the uncontrollable transitions. Let $min(p_i^u)$ be the minimal constant that appears as an upper bound in the guards containing p_i^u as a lower bound, and $max(p_i^l)$ be a maximal constant that appears as a lower bound in the guards containing p_i^l as an upper bound. Let $\mathcal{G}_r[max, min]$ represent a valuation of $\mathcal{G}$ that assign 0 (resp. ∞) to each lower (resp. upper) bound parameter that appears only in controllable transitions, and $max(p_i^l)$ (resp. $min(p_i^u)$) to every other p_i^l (resp. p_i^u) bound parameter.

Lemma 2. Let $\mathcal{G}_r$ be a restricted L/U game automaton. There is a parameter valuation (λ, μ) such that a goal state is enforceable in $[\![\mathcal{G}_r]\!]_{(\lambda,\mu)}$ with a strategy $\mathcal{F}$, if and only if the goal state is enforceable in $\mathcal{G}_r[max, min]$ using the same strategy $\mathcal{F}$.

Proof. Again we look for the value for upper bound parameters appearing only in the guards on controllable transitions. Parameter valuations (λ, μ), that assign values $max(p_i^l)$ and $min(p_i^u)$ to parameters p_i^l and p_i^u, that do not appear in controllable transition guards, is a solution. For those parameters, p_k^l and p_k^u, that appear only in guards of controllable transitions, valuations (λ, μ) assign 0 and T_i^{max}, respectively, where the value T_i^{max} is obtained from $\mathcal{G}_r[max, min]$ as in the proof for Lemma 1. □

Similarly, the next theorem follows Lemma 2.

Theorem 3. The emptiness problem for parametric timed reachability game for restricted L/U game automata is decidable.

The use of L/U game automata is shown in the case-study, 5, presenting a copper annealing controller.

4. SYMBOLIC STATE-SPACE EXPLORATION

For timed reachability games, a strategy for the controller is synthesized using the backwards algorithm for solving timed games, Maler et al. (1995). We now define the needed operations.

Let $X \subseteq Q$ and $a \in \Sigma$. The action predecessor of X, $Pred_a(X) = \{(l, w) \mid \exists (l', w') \in X, (l, w) \xrightarrow{a} (l', w')\}$, is extended for the controllable and uncontrollable action predecessors of X in a timed game automaton: $cPred(X) = \bigcup_{c \in \Sigma^c} Pred_c(X)$ and $uPred(X) = \bigcup_{u \in \Sigma^u} Pred_u(X)$, respectively. The action successor is defined as follows $Post_a(X) = \{(l', w') \mid \exists (l, w) \in X, (l, w) \xrightarrow{a} (l', w')\}$.

Timed successors and timed predecessors of X are defined by $X \nearrow = \{(l, w + d) \mid (l, w) \in X \wedge (w + d) \models I(l)\}$, $X \swarrow = \{(l, w - d) \mid (l, w) \in X\}$, respectively.

We also define a *safe-timed predecessors* operator $Pred_t(X_1, X_2)$. A state (l, w) is in $Pred_t(X_1, X_2)$ if from (l, w) we can reach $(l', w') \in X_1$ by time elapsing and along the path from (l, w) to (l', w') avoid X_2, formally:

$Pred_t(X_1, X_2) = \{q' \in Q \mid \exists d \in \mathbb{R}_{\geq 0} \text{ s.t. } q \xrightarrow{d} q', q' \in X_1 \text{ and } Post_{[0,d]}(q) \subseteq Q \backslash X_2\}$, where $Post_{[0,d]}(q) = \{q' \in Q \mid \exists t \in [0, d] \text{ s.t. } q \xrightarrow{t} q'\}$

The *controllable predecessors* operator is defined as follow:

$$\pi(X) = Pred_t(X \cup cPred(X), uPred(X))$$

In practice, the analysis of timed automata is based on the exploration of a finite graph, the *simulation graph*. The nodes of the simulation graph are symbolic states $S = (l, Z)$, where $l \in L$ and Z is a subset of a clock-space $\mathbb{R}^X_{\geq 0}$ defined by a clock constraint, called *zone*. Relation $\longrightarrow$ defines the edges of the simulation graph as $(l, Z) \xrightarrow{a} (l', Z')$, if there is a transition $(l, a, g, R, l') \in E$, and Z' is a zone successor by an edge of Z.

4.1 Algorithm for Solving Timed Games Cassez et al. (2005)

In Cassez et al. (2005), the authors present a symbolic on-the-fly algorithm for solving timed reachability games, which is based on the simulation graph and given in Figure 1. This algorithm consists of a forward computation of the simulation graph and a backward propagation of information of winning states (the states from which there is a strategy to reach the goal location). The *Waiting* set represents a list of edges in the simulation graph waiting to be explored. The *Past* set contains all symbolic states that have already been encountered. The winning status of a symbolic state S is represented by $Win[S]$, a subset of the symbolic state S which is currently known to be winning. The dependency set of S, *Depend*$[S]$, contains a set of edges (predecessors of S), which must be re-added to *Waiting* set when a new information about $Win[S]$ is obtained. Whenever an edge $e = (S, \alpha, S')$ is considered with S' belonging to *Passed*, it is added to a dependency set of S' in order that a possible future information about additional winning states of S' may be back propagated to S. Since zones are used as underlying data structure, all the steps of the algorithm are carried out efficiently. Finally, upon the termination of the algorithm, set Win^* contains all the winning states of the simulation graph.

4.2 Extension for parameter synthesis

The algorithm of Cassez et al. (2005), being on-the-fly, stops as soon as the initial state becomes winning. In the parametric case we are rather interested in computing all the conditions on parameters such that there exists a winning strategy to reach the goal state. That is why our extension of the algorithm removes the condition (of the while loop) $S_0 \in Win[S_0]$, and it is not on-the-fly. Also, we have chosen to "parametrize" the algorithm of Cassez et al. (2005), instead of the plain backwards fixpoint computation of Maler et al. (1995), because the added forward exploration pre-constrains the parameters, with conditions allowing for the reachability of symbolic states, and therefore makes the backward propagation more efficient.

Parametric Symbolic States To modify this algorithm so as to compute parameter valuations, we use an extended notion of symbolic state in which we have a location and a *parametric zone* Z - a polyhedron constraining both clocks and parameters together, i.e., a set of pairs (w, v) satisfying a parametric clock constraint.

```
Initialization:
$Passed \leftarrow \{S_0\}$ where $S_0 = \{(l_0, \mathbf{0})\} \nearrow$;
$Waiting \leftarrow \{(S_0, \alpha, S') \mid S' = \mathrm{Post}_\alpha(S_0) \nearrow\}$;
$Win[S_0] \leftarrow S_0 \cap (\{\mathrm{Goal}\} \times \mathbb{R}^X_{\geq 0})$;
$Depend[S_0] \leftarrow \emptyset$;

Main:
while $((Waiting \neq \emptyset) \wedge (S_0 \notin Win[S_0]))$ do
  $e = (S, \alpha, S') \leftarrow pop(Waiting)$;
  if $S' \notin Passed$ then
    $Passed \leftarrow Passed \cup \{S'\}$;
    $Depend[S'] \leftarrow \{(S, \alpha, S')\}$;
    $Win[S'] \leftarrow S \cap (\{\mathrm{Goal}\} \times \mathbb{R}^X_{\geq 0})$;
    $Waiting \leftarrow Waiting \cup \{(S', \alpha, S'') \mid S'' = \mathrm{Succ}_\alpha(S')\}$;
    if $Win[S'] \neq \emptyset$ then $Waiting \leftarrow Waiting \cup \{e\}$;
    endif
  else (* reevaluate *)
    $Win^* \leftarrow \mathrm{Pred}_t(Win[S] \cup \bigcup_{S \xrightarrow{c} T} \mathrm{Pred}_c(Win[T]),$
                    $\bigcup_{S \xrightarrow{u} T} \mathrm{Pred}_u(T \backslash Win[T])) \cap S$;
    if $(Win[S] \subsetneq Win*)$ then
      $Waiting \leftarrow Waiting \cup Depend[S]$;
      $Win[S] \leftarrow Win^*$;
    endif
    $Depend[S'] \leftarrow Depend[S'] \cup \{e\}$;
  endif
endwhile
```

Fig. 1. Symbolic On-the-fly Algorithm for Timed Reachability Games, Cassez et al. (2005)

The algorithm requires some specific operations on symbolic states. We straightforwardly extend them for this extended notion of symbolic state.

Definition 11. (timed successors, timed predecessors, reset). Let Z be a set of valuations (w, v) for the clocks and the parameters. We define:

- timed successors $Z \nearrow = \{(w', v) \mid \exists t \geq 0 \wedge \exists (w, v) \in Z \text{ s.t. } w' = w + t\}$
- timed predecessors $Z \swarrow = \{(w', v) \mid \exists t \geq 0 \wedge \exists (w, v) \in Z \text{ s.t. } w' = w - t \wedge \forall x_i \in X,\ w(x_i) \leq 0\}$
- reset $Z[R] = \{(w', v) \mid \exists (w, v) \in Z \text{ s.t. } w' = w[R]\}$

Note that if the set of parameters is empty, we have exactly the usual definition of timed successors, timed predecessors and reset.

Let $[\![Z]\!]_v$ define a (non-parametric) zone obtained from a parametric zone Z for a fixed parameter valuation v.

Lemma 3. For any set of valuations on both clocks and parameters Z:

$$[\![Z \nearrow]\!]_v = [\![Z]\!]_v \nearrow \quad (1)$$

$$[\![Z \swarrow]\!]_v = [\![Z]\!]_v \swarrow \quad (2)$$

$$[\![Z[R]]\!]_v = [\![Z]\!]_v[R] \quad (3)$$

Here we state only the proof for timed successors, since the other two are done in the same way.

Proof. We will prove both inclusions:
1. $[\![Z \nearrow]\!]_v \subseteq [\![Z]\!]_v \nearrow$

Suppose that $w' \in [\![Z \nearrow]\!]_v$. Then $(w', v) \in Z \nearrow$ and by definition of a timed successor, $\exists t \geq 0$ such that $w' = w + t$ and $(w, v) \in Z$. Therefore $w \in [\![Z]\!]_v$, and then $\{w\} \nearrow \subseteq [\![Z]\!]_v \nearrow$. Since $w' = w + t \in \{w\} \nearrow$, we finally have $w' \in [\![Z]\!]_v$.

2. $[\![Z]\!]_v \nearrow \subseteq [\![Z \nearrow]\!]_v$

Suppose that $w' \in \llbracket Z \rrbracket_v \nearrow$. Then $\exists t \geq 0$ such that $w' = w + t$ and $w \in \llbracket Z \rrbracket_v$, i.e., $(w, v) \in Z$. By definition of a timed successor, this is exactly $(w', v) \in Z \nearrow$ or, equivalently, $w' \in \llbracket Z \nearrow \rrbracket_v$. □

For the union, intersection and difference set operations we only state the lemma:

Lemma 4. For any set of valuations on both clocks and parameters Z_1 and Z_2:

$$\llbracket Z_1 \cup Z_2 \rrbracket_v = \llbracket Z_1 \rrbracket_v \cup \llbracket Z_2 \rrbracket_v \tag{4}$$

$$\llbracket Z_1 \cap Z_2 \rrbracket_v = \llbracket Z_1 \rrbracket_v \cap \llbracket Z_2 \rrbracket_v \tag{5}$$

$$\llbracket Z_1 \backslash Z_2 \rrbracket_v = \llbracket Z_1 \rrbracket_v \backslash \llbracket Z_2 \rrbracket_v \tag{6}$$

The zone successor by an edge, $Z' = Succ_\alpha(Z)$, in the simulation graph, is obtained, for a transition $(l, a, g, R, l') \in E$, by intersecting a source zone Z with the corresponding transition guard g, reseting clocks in reset set R, letting time elapse, and intersecting with target location invariants. The result, Z', is a zone as well, since zones are closed under intersection, reset, and timed successor operations.

Definition 12. (Zone successor). A zone successor by an edge, $Z' = Succ_\alpha(Z)$, is defined as:

$$Succ_\alpha(Z) = (Z \ \cap \ g)[R] \nearrow \cap \, I(l') \tag{7}$$

Using previous lemmas we can modify the zone successor operator for the parametric zones.

Lemma 5. For any set of valuations on both clocks and parameters Z:

$$\llbracket Succ_\alpha(Z) \rrbracket_v = Succ_\alpha(\llbracket Z \rrbracket_v) \tag{8}$$

Proof. Following Lemma 3 and Lemma 4 : $\llbracket Succ_\alpha(Z) \rrbracket_v = \llbracket (Z \cap g)[R] \nearrow \cap I(l') \rrbracket_v = \llbracket (Z \cap g)[R] \nearrow \rrbracket_v \cap \llbracket \ I(l') \rrbracket_v = \llbracket (Z \cap g)[R] \rrbracket_v \nearrow \cap \llbracket \ I(l') \rrbracket_v = \llbracket (Z \cap g) \rrbracket_v [R] \nearrow \cap \llbracket Inv(l') \rrbracket_v = (\llbracket Z \rrbracket_v \cap \llbracket g \rrbracket_v)[R] \nearrow \cap \llbracket I(l') \rrbracket_v = Succ_\alpha(\llbracket Z \rrbracket_v)$ □

Now we can modify, in the same manner, the operators needed for solving timed games.

Lemma 6. For any set on valuation on both clocks and parameters Z:

$$\llbracket Pred_a(Z) \rrbracket_v = Pred_a(\llbracket Z \rrbracket_v) \tag{9}$$

Proof.

We will prove both inclusions:
1. $\llbracket Pred_a(Z) \rrbracket_v \subseteq Pred_a(\llbracket Z \rrbracket_v)$

Suppose that $w \in \llbracket Pred_a(Z) \rrbracket_v$. Then $(w, v) \in Pred_a(Z)$ and by definition of action predecessor, $\exists w'$ such that $(l, w) \xrightarrow{a} (l', w')$ and $(w', v) \in Z$. Therefore $w' \in \llbracket Z \rrbracket_v$, and then $w \in Pred_a(\llbracket Z \rrbracket_v)$.

2. $Pred_a(\llbracket Z \rrbracket_v) \subseteq \llbracket Pred_a(Z) \rrbracket_v$

Suppose that $w' \in Pred_a(\llbracket Z \rrbracket_v)$. Then $\exists w$ such that $(l', w') \xrightarrow{a} (l, w)$ and $w \in \llbracket Z \rrbracket_v$, i.e., $(w, v) \in Z$. By definition of action predecessor, we have $(w', v) \in Pred_a(Z)$, which gives us $w' \in \llbracket Pred_a(Z) \rrbracket_v$. □

The safe-timed predecessors operator can be expressed as (detailed proof in Cassez et al. (2005)):

$$Pred_t(Z_1, Z_2) = (Z_1 \swarrow \backslash Z_2 \swarrow) \cup ((Z_1 \cap Z_2 \swarrow) \backslash Z_2) \swarrow \tag{10}$$

Lemma 7. For any set of valuations on both clocks and parameters Z_1 and Z_2:

$$\llbracket Pred_t(Z_1, Z_2) \rrbracket_v = Pred_t(\llbracket Z_1 \rrbracket_v, \llbracket Z_2 \rrbracket_v) \tag{11}$$

The proof is similar to the one of Lemma 5, and it applies to a safe-timed predecessor expressed by equation (10).

Now we have all the necessary operations modified for the parametric symbolic state-space, and we can prove the correctness of the parametric algorithm. Upon the termination of our semi-algorithm the following theorem stands:

Theorem 4. For a PGA $\mathcal{G}$, and a desired state goal, there exists a winning strategy for a parameter valuation function v if and only if $(l_0, w_0) \in \llbracket Win[S_0] \rrbracket_v$.

Proof. $Win(\mathcal{G})$ is a set of winning states in $\mathcal{G}$. The iterative process of the algorithm is given by $Win^0 = \mathsf{goal}$ and $Win^{n+1} = \pi(Win^n)$. We have that $Win^0 = \llbracket Win^0 \rrbracket_v$, because $\mathsf{goal} = \{\mathsf{goal} \times \mathbb{R}^X_{\geq 0}\}$, and therefore is not affected by a parameter valuation function.

We apply a timed predecessors operator in a parametric domain and by Lemma 7 we have: $\pi(\llbracket Win^n \rrbracket_v) = \llbracket \pi(Win^n) \rrbracket_v = \llbracket Win^{(n+1)} \rrbracket_v$. The least fix point obtained is $\llbracket Win^* \rrbracket_v$, and $\llbracket Win^* \rrbracket_v = \llbracket Win(\mathcal{G}) \rrbracket_v$ (proved in Maler et al. (1995)).

A state (l, w) is winning if it belongs to $\llbracket Win(\mathcal{G}) \rrbracket_v$, thus there is a winning strategy for $\mathcal{G}$ if $(l_0, w_0) \in \llbracket Win(\mathcal{G}) \rrbracket_v$.

An invariance property of the algorithm for solving timed games Cassez et al. (2005), $Win\llbracket S \rrbracket_v \subseteq Win(\llbracket \mathcal{G} \rrbracket_v)$, has been proven in Cassez et al. (2005). We can adapt the proof for the parametric case and show that $\llbracket Win[S] \rrbracket_v \subseteq \llbracket Win(\mathcal{G}) \rrbracket_v$, for some $S = (l, Z)$, where Z is a parametric zone. We have that $Win\llbracket S \rrbracket_v = \llbracket Win[S] \rrbracket_v$, because $Win[S] \subseteq S$. By the induction hypothesis, we may assume, in the non-parametric case, that $Win\llbracket S' \rrbracket_v \subseteq Win\llbracket \mathcal{G} \rrbracket_v$, when $S \xrightarrow{a} S'$. Zone Z', of the state S', is the successor of the zone Z. Since we have, by Lemma 5, that $Succ_\alpha(\llbracket Z \rrbracket_v) = \llbracket Succ_\alpha(Z) \rrbracket_v$, we may also assume that $\llbracket Win[S'] \rrbracket_v \subseteq \llbracket Win[\mathcal{G}] \rrbracket_v$. Then, by the monotonicity of $Pred_t$ it follows that $\llbracket Win^* \rrbracket_v \subseteq \llbracket (\pi(Win(\mathcal{G})) \rrbracket_v \subseteq \llbracket Win(\mathcal{G}) \rrbracket_v$. Then, the property holds when we update $\llbracket Win^* \rrbracket_v \leftarrow \llbracket Win[S] \rrbracket_v$.

If $S = S_0$ and $(l_0, w_0) \in \llbracket Win[S_0] \rrbracket_v$ then $(l_0, w_0) \in \llbracket Win^* \rrbracket_v$, hence, there is a strategy to reach a goal state in $\llbracket \mathcal{G} \rrbracket_v$, and it can be extracted from $\llbracket Win^* \rrbracket_v$. □

The termination of our parametrization of the algorithm of Cassez et al. (2005) is not guaranteed. In the case of termination however, if the initial state belongs to a set of winning states, the correct set of constraints on the parameters is obtained and a winning strategy can be extracted from the set of winning states.

Example We consider the same example as in Cassez et al. (2005), but we parametrize the model in order to obtain a L/U game automaton (Figure 2). It has one clock x, controllable (c_i) and uncontrollable (u_i) actions and two parameters a and b: a appears positively in the guards of the controllable transitions c_1 and c_4 and negatively in the guard of the uncontrollable transition u_1; b appears positively in the guard of the uncontrollable transition u_3.

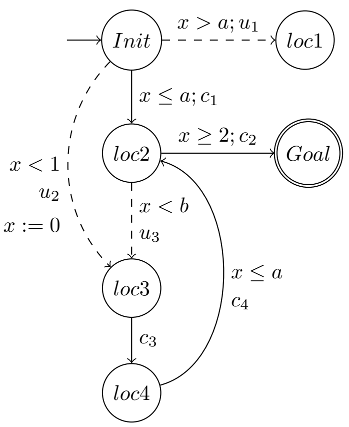

Fig. 2. A L/U Game Automaton

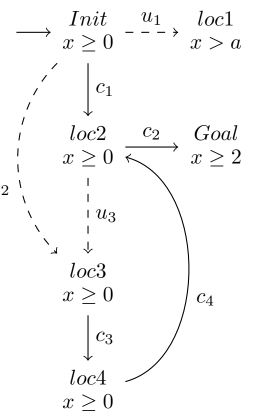

Fig. 3. Simulation graph of PGA of Figure2

The reachability game consists in finding a strategy for the controller that will eventually ends up in the location *Goal*.

We will now explain how the algorithm works. Although the algorithm of Cassez et al. (2005) is an interleaved combination of a forward computation and a backward propagation, for the sake of simplicity, we will start from a simulation-graph and show the back-propagation of winning states.

After the computation of a simulation graph, 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 $(loc2, x \geq 2)$. Computing the timed predecessors removes the constraint $x \geq 2$, and computing the *controllable predecessors* adds $x \geq b$ in order not to end-up in loc_3 by $u3$. The resulting state is $(loc2, x \geq b)$. One of the controllable transitions taking us to loc_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 $(loc4, x \geq 0 \wedge a \geq b)$.

We obtain successively the following sets of winning states: $(loc3, x \geq 0 \wedge a \geq b)$, $(loc2, (x \geq b) \vee (x \geq 0 \wedge a \geq b))$ and $(Init, (x \leq a) \wedge \big((x < 1 \wedge a \geq b) \vee x \geq 1\big) \wedge \big((x \geq b) \vee (x \geq 0 \wedge a \geq b)\big)$. The last one simplifies to $(Init, (x \leq a \wedge a \geq b))$.

4.3 Winning Strategy

In this section we show how to extract the winning strategy from the set of winning states.

Theorem 5. If there exists a winning strategy for the parametric timed game automaton $\mathcal{G}$, then there exists a memory-less winning strategy for $\mathcal{G}$.

Proof. If there exists a winning strategy for the parametric timed game automaton then there exists a parameter valuation v such that this winning strategy exists. For this parameter valuation v, we obtain a timed game automaton $[\![\mathcal{G}]\!]_v$, for which the winning strategy obtained by the algorithm is memory-less, since it suggests to a controller to either delay or take a controllable action in each state, Maler et al. (1995). □

Thanks to this theorem, it is easy to extract the memoryless winning strategy from the set of winning states as follows: a controllable action predecessor give us the state from which a corresponding controllable action should be taken, while timed predecessor further gives us the state where we should delay.

Let us go back to the previous example. The set of states $(loc2, x \geq 2)$ is the controllable action predecessor from $(Goal, x \geq 2)$ by action c_2. Then the winning strategy is: in all states $(loc2, x \geq 2)$ the controllable transition c_2 should be taken immediately, and in $(loc2, x \geq 0)$, we should delay until $x \geq 2$. The controllable action predecessor from $loc2$ takes us to a state $(loc4, x \geq b \wedge x \leq a)$, deriving a constraint $a \geq b$. From that state an action c_2 should be taken immediately, and timed predecessor gives the state $(loc4, x \geq 0, a \geq b)$ in which we should delay until $x \geq b$.

Since we can not influence the uncontrollable transitions if we end-up in $loc4$, controllable transition c_4 should be taken as soon as $x \geq b$ is satisfied, and with the condition $a \geq b$ we are sure that u_3 cannot be fired again.

Notice that the order of exploration of the winning states leads to different winning strategies. As an example, applying controllable predecessor from $(loc2, (x \geq b) \vee (x \geq 0 \wedge a \geq b))$ to $Init$ can lead to both strategies from $Init$:

(1) doing c_1 in all states $(Init, x)$ with $x \leq a$;
(2) delaying in all states $(Init, x)$ with $x < b$ and $x \leq a$ and doing c_1 for all states with $x \geq b$ and $x \leq a$ (recall that $b \leq a$).

Thus, a whole winning strategy consists in:

- doing c_1 in all states $(Init, x)$ with $x \leq a$
- delaying in all states (loc_2, x) with $x < 2$
- doing c_2 in all states (loc_2, x) with $x \geq 2$
- doing c_3 in all states (loc_3, x)
- delaying in all states (loc_4, x) with $x < b$
- doing c_4 in all states (loc_4, x) with $x \geq b$ and $x \leq a$ (recall that $b \leq a$).

5. CASE STUDY

Let us consider the Copper Annealing Controller depicted in Figure 4. Annealing, in metallurgy and materials science, is a heat treatment wherein a material is altered, causing changes in its properties such as strength and hardness. It is a process that produces conditions by heating to above the critical temperature, maintaining a suitable temperature, and then cooling.

The parametric timed automaton shown in Figure 4 has two clocks x and y, two parameters a and b, controllable (c_i) and uncontrollable (u_i) actions. Action c_1 stops the heater to maintain the temperature. Actions c_2 and c_3 start and stop the cooler, respectively. The copper is observed by sensors that produce uncontrollable actions: u_1 is raised when the copper could be softer: it should be heated a bit more; u_2 is raised when the copper is too hard: the process must stop; u_3 is raised when the copper is soft enough: it should be cooled as soon as possible; u_4 is raised when the copper is too soft: the process must stop.

The parameter a means that a heating stage is followed by a maintaining stage whose duration is at least 10 time

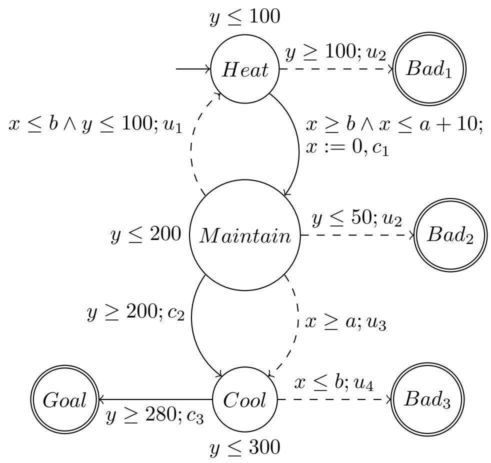

Fig. 4. A Copper Annealing Controller model

units longer than the heating duration. The parameter b comes from the dynamics of the system. For a copper wire heated during at least b time units, the values given by sensors u_1 and u_4 are relevant and guaranteed during b time units after the end of the heating stage.

The reachability game consists in finding a strategy, that will eventually end up in the location *Goal*. Actually, for this model, we obtain that there is no winning strategy for this game since it is impossible to prevent the transition u_1 from the locality $Maintain$ and then the locality Bad_1 is always reachable after some loops $(c_1.u_1)^*$ followed by u_2.

Then the model of the controller must be corrected. Since heating a bit more the copper, when it is possible, is not necessary, we can delete the transition u_1 (another way would consist in controlling the transition from $Maintain$ to $Heat$ when action u_1 occurs by adding a locality and two controllable actions). Thus, there exists a winning strategy if and only if $(b < 100) \wedge (a > 40) \wedge (a > b)$ and the set of winning states obtained by the algorithm is:

- $(Heat, (x \geq 0) \wedge (y \geq 0) \wedge (y < 100) \wedge (b < 100) \wedge (a > 40) \wedge (a > b))$,
- $(Maintain, (x \geq 0) \wedge (y > 50) \wedge (y \leq 200) \wedge (b \leq y - x \leq a + 10) \wedge (y - x < 100) \wedge (a > b))$,
- $(Cool, (x > b) \wedge (0 \leq y \leq 300) \wedge (b \leq y - x \leq a + 10) \wedge (y - x < 100))$,
- $(Goal, y \geq 280 \wedge (b \leq y - x \leq a + 10) \wedge (y - x < 100))$.

Intuitively, the condition $b < 100$ allows to avoid Bad_1 by ensuring that the locality $Heat$ can be left before the condition $y \geq 100$ becomes true; the condition $a > 40$ allows to avoid Bad_2 by ensuring that the locality $Maintain$ can be reached with $y > 50$ and the condition $a > b$ allows to avoid Bad_3. A winning strategy extracted from the winning set consists in:

- delaying in all states $(Heat, x, y)$ with $y \leq 50$ or $x < b$
- doing c_1 in all states $(Heat, x, y)$ with $y > 50$ and $x \geq b$
- delaying in all states $(Maintain, x, y)$ with $y < 200$
- doing c_2 in all states $(Maintain, x, y)$ with $y = 200$
- delaying in all states $(Cool, x, y)$ with $y < 280$
- doing c_3 in all states $(Cool, x, y)$ with $280 \leq y$

6. CONCLUSIONS AND FUTURE WORK

In this paper we have introduced parametric timed game automata and their subclasses (L/U and restricted L/U game automata), for which the emptiness problem of parametric timed reachability game is decidable. We have also adapted the algorithm for solving timed games from Cassez et al. (2005) for the parametric case. When the initial state is winning, a set of constraints on the parameters is obtained together with a set of winning states.

Although the subclass of PGA proposed might seem overly restricted, in practice it is not rare that only one or two values are parametrized and we search for all their possible valuations, as shown in our case-study.

Our model can be put in use for deciding timed alternating simulation in parametric domain, which serves to model real-time controller synthesis problems. Simulation can also be accompanied by a logical characterization, Bozzelli et al. (2009): if TGA $\mathcal{A}$ is simulated by TGA $\mathcal{B}$ then whenever a formula holds in $\mathcal{A}$, it also holds in $\mathcal{B}$. In Bulychev et al. (2009), a notion of weak alternating simulation, that preserves controllability with respect to ATCTL, between two timed game automata has been introduced. Problem of checking given notion of simulation has been reduced to solving timed reachability game and an on-the-fly algorithm for solving this game is proposed. In the parametric case, this algorithm could give the constraints on the parameters such that the weak alternating simulation holds between two timed game automata (for example, an abstract model and its refinement).

REFERENCES

Alur, R. and Dill, D. (1994). A theory of timed automata. *Theoretical Computer Science B*, 126, 183–235.

Alur, R., Henzinger, T.A., and Vardi, M.Y. (1993). Parametric real-time reasoning. In *ACM Symposium on Theory of Computing*, 592–601.

Asarin, E., Maler, O., Pnueli, A., and Sifakis, J. (1998). Controller synthesis for timed automata. In *Proc. IFAC Symposium on System Structure and Control*. Elsevier.

Bozzelli, L., Pinchinat, S., and Legay, A. (2009). On timed alternating simulation for concurrent timed games. In *FSTTCS 2009*, Leibniz Int. Proceedings in Informatics.

Bulychev, P., Chatain, T., David, A., and Larsen, K. (2009). Efficient on-the-fly algorithm for checking alternating timed simulation. In *FORMATS '09*, 73–87.

Cassez, F., David, A., Fleury, E., Larsen, K., and Lime, D. (2005). Efficient on-the-fly algorithms for the analysis of timed games. In *CONCUR'05*, volume 3653 of *LNCS*.

Hune, T., Romijn, J., Stoelinga, M., and Vaandrager, F. (2002). Linear parametric model checking of timed automata. *Journal of Logic and Algebraic Programming*, 52-53, 183–220.

Maler, O., Pnueli, A., and Sifakis, J. (1995). On the synthesis of discrete controllers for timed systems. In *STACS '95*.

Ramadge, P.J.G. and Wonham, W.M. (1989). The control of discrete event systems. *Proceedings of the IEEE*, 77(1), 81–98.