A game approach to the parametric control of real-time systems

Aleksandra Jovanović, Didier Lime, Olivier Henri Roux

INTERNATIONAL JOURNAL OF CONTROL

A Game Approach to the Parametric Control of Real Time Systems

Aleksandra Jovanović[a], Didier Lime[b] and Olivier H. Roux[b]

[a]Department of Computer Science, University of Oxford, Oxford, UK,
[b]Ecole Centrale de Nantes, LS2N (UMR CNRS 6004), Nantes , France

ARTICLE HISTORY
Compiled February 7, 2019

ABSTRACT
We consider parametric reachability control problems for real-time systems. We model the plant as an extension of parametric timed automata, i.e., a finite automaton equipped with real-valued clocks constraining its behavior, in which the timing constraints on these clocks can make use of parameters. This extension, which we call parametric game automata (PGA), allows for partitioning the actions in the model between two antagonistic entities: the controller and the environment. The most general problem we study then consists in synthesizing both a controller and values for the parameters such that some control location of the automaton is reachable.

It is however well-known that most non-tivial problems on parametric timed automata are undecidable and the classical techniques for the verification (and a fortiori for the control) of timed systems do not terminate in that setting. We therefore provide a subclass of PGA called L/U game automata for which it is decidable.

We then consider a backward fixed-point semi-algorithm for solving timed games with reachability objective allowing to compute the most permissive winning strategy.

We argue the relevance of this approach and demonstrate its practical usability with a small case-study.

KEYWORDS
timed automata, control, game theory, parameters, synthesis

1. Introduction

Real-time and embedded systems often interact with the environment in an unpredictable way. They can also be reactive machines that cooperate with the environment in order to provide their service. Such uncertainties in the system introduced by the environment are often difficult to model. Therefore, these systems need a controller to ensure their correct behavior. Its purpose is to regulate the activity and ensure that the system under control meets the specifications, no matter what happens in the environment.

The introduction of the automata-based formalism into the field of control was motivated by the inadequacy of models based on continuous mathematics (differential equations) to describe certain classes of systems. A decision to open a gate or to turn left or right are discrete and most naturally modeled with an automaton. Control

This work was partially funded by the ANR national research program PACS (ANR-14-CE28-0002).
CONTACT Email: olivier-h.roux@ec-nantes.fr

using automata-based approach lies between computer science and control, and brings together these two communities.

Instead of verifying the correctness of a system, we have here the problem of synthesizing the model for the controller. It consists in computing a controller which, based on the current state of the system, restricts the choices of the system, ensuring that the desired property is satisfied. This problem is often modeled as the synthesis of a winning strategy for the controller in a two-player game against the environment.

More precisely, the two players, the *controller* and the *environment*, take actions from their own set and thus make the game progress. In each state, both players choose, at the same time and independently of each other, a move (a delay or an action). The goal is to find the strategy for the controller such that, no matter what the environment does, the system ends up in a given desired state. Such a strategy is called a winning strategy.

A formalism that is commonly used to describe such systems in a timed framework is timed game automata (TGA, (Maler, Pnueli, & Sifakis, 1995)), that explicitly represents the moves of both players, in terms of controllable and uncontrollable edges. They are extended timed automata that distinguish between the actions of the two players, describing at the same time both the capabilities of the controller and the environment.

The *(Reachability) control problem* for TGA is the problem of determining a strategy for the controller such that, no matter what the environment does, the system ends up in the desired location. This problem is known to be decidable (Maler et al., 1995). The introduction of this model has been followed by the development of efficient algorithms (Cassez, David, Fleury, Larsen, & Lime, 2005) and tool support (Behrmann et al., 2007), successfully applied to several industrial case studies (e.g. (Cassez, Jessen, Larsen, Raskin, & Reynier, 2009; Jessen, Rasmussen, Larsen, & David, 2007)).

This model, however, requires complete knowledge of the system. It is thus 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 controller synthesis 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 values. This approach, however, leads to the undecidability of the most important problems, such as reachability verification (Alur, Henzinger, & Vardi, 1993).

1.1. *Related work.*

Parametric timed automata (Alur et al., 1993) extend timed automata (Alur & Dill, 1994) to overcome the limits 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 no parameter valuation such that the automaton has an accepting run, is undecidable. This naturally led to the search for subclasses of the model for which some problems would be decidable. L/U automata (Hune, Romijn, Stoelinga, & Vaandrager, 2002) use each parameter either as a lower bound or as an upper bound on clocks. The reachability-emptiness problem is decidable for this model, but the state-space exploration, which would allow for explicit synthesis of all the suitable parameter valuations, still might not termi-

nate. The decidability of various verification problems for L/U automata is further studied in (Bozzelli & La Torre, 2009). The authors give the explicit representation of the set of parameters, when all parameters are integers and of the same type (all upper bounds or all lower bounds). In (Bruyère & Raskin, 2007), 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 (André, Chatain, Encrenaz, & Fribourg, 2009) where the exploration starts from an 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.

1.2. *Our contribution.*

We first introduce a model of timed games extended with parameters, called parametric timed game automata (PGA). In this setting the most basic problem is: "does there exist values for the parameters such that there exists a controller, such that some control location is reachable whatever the environment does?", which we will call the *parametric control problem*. As the PGA formalism extends PTA, this problem is undecidable. We therefore provide a subclass of PGA called L/U game automata for which it is decidable. The subclass is based on a restricted use of parameters in the clock constraints, in the spirit of the L/U automata (Hune et al., 2002).

We then consider a backward fixed-point algorithm for solving timed games with reachability objective (Maler et al., 1995). We extend this algorithm for the parametric approach to obtain the set of symbolic constraints on the parameters together with the set of winning states for the controller. It consists of two fixed-point parts, a forward exploration of the state-space and a backward propagation of winning states. The termination however, is not guaranteed.

1.3. *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 present the subclass of PGA for which the reachability-emptiness game is decidable and give the proofs. The algorithm for solving timed games, is presented in Section 4. Finally we present a small case-study in Section 5 and we conclude with Section 6.

2. Parametric Timed Game Automata

2.1. *Preliminaries.*

$\mathbb{R}$ is the set of real numbers and $\mathbb{R}_{\geq 0}$ is the set of non-negative real numbers. $\mathbb{Q}$ is the set of rational numbers, $\mathbb{Z}$ 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 $\gamma ::= x_i \backsim$

$p \mid -x_i \backsim p \mid \gamma \wedge \gamma$, 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 linear constraint on clocks obtained by replacing each parameter p_i by its value $v(p_i)$. Similarly, given a linear constraint on clocks γ and a clock valuation w, $w(\gamma)$ is a boolean value, obtained by replacing each clock x by it value $w(x)$. We denote by $G(X, P)$ the set of parametric constraints over X, and by $G'(X, P)$ a set of parametric constraints over X of the form $\gamma' ::= x_i \backsim p \mid \gamma' \wedge \gamma'$.

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

2.2. *Parametric Timed Games*

Definition 2.1. 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, and $\mathsf{Inv} : L \mapsto G'(X, P)$ is a function that assigns a (parametric) invariant to each location.

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.

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 (Alur & Dill, 1994; Henzinger, Nicollin, Sifakis, & Yovine, 1994). 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 2.2 (Semantics of a PTA)**.** The concrete semantics of a PTA $\mathcal{A}$ under a parameter valuation v, notation $v(\mathcal{A})$, is the labeled transition system $(Q, q_0, \rightarrow)$ over $\Sigma \cup \mathbb{R}_{\geq 0}$ where:

- $Q = \{(l, w) \in L \times \mathbb{R}^X_{\geq 0} \mid w(v(\mathsf{Inv}(l))) \text{ is true } \}$
- $q_0 = \{(l_0, \mathbf{0}_X) \in Q\}$
- delay: $(l, w) \xrightarrow{t} (l, w + t)$ with $t \geq 0$, iff $\forall t' \in [0, t], (l, w + t') \in Q$
- action: $(l, w) \xrightarrow{a} (l', w')$ with $a \in \Sigma$, iff $(l, w), (l', w') \in Q$, there exists an edge $(l, a, \gamma, R, l') \in E$, such that $w' = w[R]$ 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 transitions in the semantics of $v(\mathcal{A})$, $\rho = q_1 a_1 q_2 ... a_{n-1} q_n$, where $\forall i \in [1..n-1], q_i \in Q, a_i \in \Sigma \cup \mathbb{R}_{\geq 0}$, and $q_i \xrightarrow{a} q_{i+1}$. The last state of ρ is denoted by $\mathsf{last}(\rho)$. 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 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 $\mathsf{last}(\rho) = q$.

We now go one step further and define Parametric Timed Game Automata to model our control problems.

Definition 2.3. 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.

Like 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 (Cassez et al., 2005; Maler et al., 1995).

We formalize our reachability control problem as a timed game, in which the controller player has to reach some distinguished location in a TGA:

Definition 2.4 ((Parametric) Timed Game)**.** A (reachability) *parametric timed game* is a pair $(\mathcal{G}, l_{goal})$ where $\mathcal{G}$ is a PGA and l_{goal} is a location of $\mathcal{G}$.

A (non-parametric) timed game is a parametric timed game $(\mathcal{G}, l_{goal})$ in which $\mathcal{G}$ is a TGA.

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 their 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 the controller to either delay in a location (denote by the delay action), or to take a particular controllable action.

For reachability timed games, one can consider two semantics for uncontrollable actions: either they can only spoil the game and it is up to the controller to do some controllable action to win, or, if at some state s only an uncontrollable action is enabled but forced by time to happen leading to a winning state then, the state s is winning. The usual semantics (Asarin, Maler, Pnueli, & Sifakis, 1998; Cassez et al., 2005; Maler et al., 1995) is the first one where uncontrollable actions cannot help to win and is the one we consider in this paper.

Definition 2.5 (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 $\mathsf{last}(\rho) \xrightarrow{\mathcal{F}(\rho)} q$ 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 $last(\rho) \xrightarrow{d'} q$.

Since we focus on control problems for which the control objective (given a parameter valuation) is to reach a particular location of timed automata, we need only memoryless strategies (Maler et al., 1995), i.e., strategies $\mathcal{F}$ such that $\mathcal{F}(\rho)$ only depends on $\mathsf{last}(\rho)$.

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

Definition 2.6 (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. $\mathsf{last}(\rho) \xrightarrow{\delta'} q'' \wedge \mathcal{F}(\rho \xrightarrow{\delta'} q'') = \mathsf{delay}$.
- 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 out-

come 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 $\mathsf{last}(\rho)$ are uncontrollable actions). $\mathsf{MaxOut}(q, \mathcal{F})$ denotes the set of maximal runs for a state q and a strategy $\mathcal{F}$.

Definition 2.7 (Winning strategy, state, game)**.** Let $\mathcal{G} = (L, l_0, X, \Sigma^c \cup \Sigma^u, P, E, \mathsf{Inv})$ be a PGA and $(\mathcal{G}, l_{goal})$ a parametric timed game. Let v be a parameter valuation.

A strategy $\mathcal{F}$ in the non-parametric game $(v(\mathcal{G}), l_{goal})$ is winning from state q if for all runs $\rho \in \mathsf{MaxOut}(q, \mathcal{F})$, there is some state (l_{goal}, w) in ρ.

A state q is winning if there exists a winning strategy from q.

The timed game $(v(\mathcal{G}), l_{goal})$ is winning if its initial state is winning.

The parametric timed game $(\mathcal{G}, l_{goal})$ is winning if there exists some parameter valuation v such that $(v(\mathcal{G}), l_{goal})$ is winning.

In the non-parametric case, the (reachability) control (resp. synthesis) problem is that of the existence (resp. computation) of a strategy such that, no matter what happens in the environment, the system ends-up in the desired location (for short we say this location is *enforceable*). The control problem is known to be decidable (Maler et al., 1995) and there exists efficient symbolic algorithms for the computation of winning states and strategies (Cassez et al., 2005). We now extend these problems to account for parameters.

Parametric control problem:

INPUTS : A PGA $\mathcal{G}$ and a location l_{goal} of $\mathcal{G}$.

PROBLEM : Is the parametric timed game $(\mathcal{G}, l_{goal})$ winning?

Parametric synthesis problem:

INPUTS : A PGA $\mathcal{G}$ and a location l_{goal} of $\mathcal{G}$.

PROBLEM : Compute the set of valuations v of the parameters and the corresponding winning strategies for $(v(\mathcal{G}), l_{goal})$ to be winning.

The emptiness problem for PTA, i.e. the existence, for a PTA $\mathcal{A}$, of a parameter valuation v such that some location is reachable in $v(\mathcal{A})$ is undecidable (Alur et al., 1993). As the parametric control problem extends the reachability problem (reachability is control with all transitions controllable), the following theorem holds.

Theorem 2.1. *The parametric control problem for PGA is undecidable.*

As a consequence to this negative result we now investigate restrictions to the PGA formalism to make the control problem decidable.

3. L/U Reachability Timed Games

The following syntactic subclass of PTA, called L/U-automaton, has been proposed in Hune et al. (2002) as a decidable subclass for the emptiness problem. It relies on the notion of upper and lower bounds for parameters:

Definition 3.1 (Lower and upper bounds)**.** Let γ be a single conjunct of a parametric clock constraint on the set of clocks X and the set of parameters P. Constraint γ is of the form $sx \sim \Sigma_i a_i p_i$, where $s \in \{-1, 1\}$, $x \in X$, $\sim \in \{<, \leq\}$ and for all i, $a_i \in \mathbb{Z}$

and $p_i \in P$. Parameter p_i is an upper (resp. lower) bound in γ if a_i is positive (resp. negative).

Parameter p is an upper (resp. lower) bound in the PTA $\mathcal{A}$ if for each conjunct γ of each parametric clock constraint in the guards and invariants of $\mathcal{A}$, either p does not occur in γ (its coefficient is zero) or p is an upper (resp. lower) bound in γ.

Definition 3.2 (L/U-automaton)**.** A PTA $\mathcal{A}$ is an L/U-automaton if every parameter is either an upper bound or a lower bound in $\mathcal{A}$.

The reachability-emptiness problem is PSPACE-complete for L/U automata (Hune et al., 2002) and, more generally, the emptiness, universality and finiteness of the parameter valuation set are PSPACE-complete for infinite runs acceptance properties, and integer-valued parameters (Bozzelli & La Torre, 2009). These good results are based on a *monotonicity* property that L/U automata have: decreasing lower bounds or increasing upper bounds only *adds* behaviors. So if we set all lower bounds to 0 and all upper bounds to a large enough constant that we can compute, then the resulting timed automaton contains all the possible behaviors. This makes these automata very well-suited for reachability-like properties.

We accordingly extend this subclass to define a subclass of parametric game automata for which the parametric control problem is decidable.

Partition of the Set of Parameters 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 transitions and as lower bounds in the uncontrollable transitions. Increasing an upper bound or decreasing a lower bound both make the controller more powerful and restricts the environment (and vice-versa). We assume that invariants are non-parametric constraints.

Definition 3.3 (L/U game automata (L/U PGA))**.** An L/U game automaton $\mathcal{G} = (L, l_0, X, \Sigma^c \cup \Sigma^u, P, E, \mathsf{Inv})$ 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 as lower bound (resp. upper bound) in the guards of controllable (resp. uncontrollable) transitions;
- each parameter $p \in P^u$ occurs only as upper bound (resp. lower bound) in the guards of controllable (resp. uncontrollable) transitions;
- for each location l, $\mathsf{Inv}(l)$ contains no parameter.

We now prove the following theorem:

Theorem 3.1. *The control problem is decidable for L/U games.*

Proof. Let (λ, μ) represent a parameter valuation such that λ applies to parameters $p \in P^l$, and μ applies to parameters $p \in P^u$, and let $\mathcal{G}[\lambda, \mu]$ the corresponding timed game automaton obtained from a PGA $\mathcal{G}$. Recall that a parameter $p \in P^l$ is a lower bound in the guards of controllable transitions or an upper bound in the guards of uncontrollable transitions. Reciprocally, a parameter $p \in P^u$ is an upper bound in the guards of controllable transitions or a lower bound in the guards of uncontrollable transitions. Let $\mathcal{G}[0, \infty]$ be the TGA obtained from $\mathcal{G}$ when each parameter $p_i^u \in P^u$ is set to ∞, and each parameter $p_i^l \in P^l$ is set to 0. Setting a lower bound parameter to

∞ (in uncontrollable edges) effectively means removing this edge (or setting its guard to False). In this way, we obtain a timed game automaton, for which the existence of a winning strategy is known to be decidable.

We now prove the lemma 3.4 that links this TGA to the original problem:

Lemma 3.4. *Let $\mathcal{G}$ be a L/U game automaton and l_{goal} one of its locations. There exists a parameter valuation (λ, μ) such that $l_{goal} \in L$ is enforceable in $\mathcal{G}[\lambda, \mu]$ with a strategy $\mathcal{F}$, iff it is enforceable in $\mathcal{G}[0, \infty]$ using the same strategy $\mathcal{F}$.*

Proof. First notice that by increasing parametric upper bounds or decreasing parametric lower bounds, we give more power to the controller and less to the environment. Recall that uncontrollable actions cannot help to win, i.e., if the timed game is winning then, in the outcome of any winning strategy, there exists a winning run in which no uncontrollable action is used. So $\mathcal{G}[0, \infty]$ represents the most favorable case for the controller, i.e., if a winning strategy exists in $v(\mathcal{G})$ for some valuation v it certainly also is possible in $\mathcal{G}[0, \infty]$.

Now, in the converse direction suppose some winning strategy $\mathcal{F}$ exists in $\mathcal{G}[0, \infty]$. Given a location, it follows from Maler et al. (1995), that $\mathcal{F}$ can be considered constant on regions without loss of generality. Recall that regions are elementary convex polyhedra that partition the clock-space, and there are a finite number of them for any given TA. By construction of these regions, taking any transition from two states with the same location and clock valuations in the same region leads to states with the same location and clock valuations in the same region (see Alur and Dill (1994) for details).

0 is obviously a suitable value for the lower bound parameters.

We are therefore searching for a finite value for the upper bound parameters such that all runs of a given strategy $\mathcal{F}$ remain winning. Note that each run of a *winning* strategy necessarily reaches the goal location in finite time and with a finite number of discrete actions.

Now, starting from any reachable state, if the environment does not play, the strategy allows the controller to reach the goal in bounded time. Since the maximum time that can be spent in a given region is 1 time unit, and since the strategy is constant on regions, for all starting states with the same location and in the same region, this bounded time differ by at most one. We can therefore compute a uniform bound on all starting regions for the time until the controller reaches the goal by following the strategy (without any disturbance from the environment). Let us call T this bound.

Since the strategy is winning the environment can only play a finite number of time in each run of the outcome of the strategy, and that number can be uniformly bounded. Let us call N this bound.

As a consequence $N * T$ is certainly an upper bound for the time to enforce the goal location with the strategy $\mathcal{F}$.

Finally, we just need to ensure that the upper bounds on each clock, in each guard, are bigger than that constant: consider all the single conjuncts γ_j in the guards of the PGA. Constraint γ_j is of the form $sx \sim \Sigma_i a_i p_i$, where $s \in \{-1, 1\}$, $x \in X$, $\sim \in \{<, \leq\}$ and for all i, $a_i \in \mathbb{Z}$ and $p_i \in P$. Let us call β_j the expression $\sum_i b_i p_i$ with $b_i = a_i$ if a_i is positive and $b_i = 0$ otherwise (i.e. we set all lower bound parameters to 0). The conjunction of the constraints $\beta_j \geq N * T$ is upward-closed (because all the coefficients of the parameters are positive) and, by construction, any solution μ of that system is a valuation such that $\mathcal{F}$ is still a winning strategy in $\mathcal{G}[0, \mu]$. □

The claimed result follows directly from Lemma 3.4 and the decidability of the

control problem for TGA (Maler et al., 1995). □

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 behavior but not to the point of uniformly forbidding the transition.

Consequently, we define a stronger form of the control problem:

Definition 3.5 (T-restricting parameter valuation)**.** Let $\mathcal{G}$ be a PGA and T a subset of its edges. A parameter valuation v is *T-restricting* if there exists some edge in T, with a guard g, such that $v(g)$ is not satisfiable.

A guard being satisfiable means there exists some clock values for which it is true, but not necessarily that these clock values are indeed reachable in some run. To illustrate the definition, suppose we have a guard $x \leq a \wedge x \geq b$, where x is a clock and a and b are parameters, then all the valuations such that $a < b$ are restricting. A contrario, suppose we have a guard $x \leq a \wedge y \geq b$, where x and y are clocks and a and b are parameters, then the valuations such that $a < b$ are non-restricting even if it makes the guard false for some values of x and y (e.g. when $x = y$).

We can now define a more general control problem, in which we are interested only in non-restricting parameter valuations:

Definition 3.6 (T-non-restricting control problem)**.** Let $\mathcal{G}$ be a PGA, l_{goal} one of its location, and T a subset of its edges. The T-non-restricting control problem asks whether there exists a valuation v of the parameters such that v is not T-restricting and $(v(\mathcal{G}), l_{goal})$ is winning.

Theorem 3.2. *The T-non-restricting control problem and the control problem are equivalent for general PGA.*

Proof. The T-non-restricting problem is clearly the more general problem since we can always take an empty T set.

Now, remark that the non-restriction constraint can be encoded in the PGA: suppose to begin with that there is only one edge in T and let g be its guard. Then just add an extra location l'_0 that becomes the new initial location. From l'_0 add as many unconstrained self-loops as clocks in the system, each resetting exactly one clock, and each a different one. Then add a controllable edge from l'_0 to the previous initial location l_0, with guard g and resetting all clocks. Clearly, using the self-loops, any combination of values for the clocks can be reached in l'_0 and the transition from l'_0 can therefore be taken if and only if g is satisfiable. After taking this transition, all clocks are reset to zero, we are in l_0, and the execution in the PGA can then proceed exactly as before the transformation. In order to reach the goal, the extra transition must be taken, and therefore the guard must be satsifiable. The only parameter valuations solutions to the control problem on this modified PGA are therefore those that are not T-restricting. Finally, if there are several edges in T, we just need to add the

same widget in sequence at the beginning for each edge to obtain the same result.

In this transformation, if the edges in T are controllable, the extra widgets are compatible with the L/U restriction, and therefore starting from an L/U PGA, we obtain again an L/U PGA. □

The following corollaries are immediate:

Corollary 3.7. *The T-non-restricting problem for PGA is undecidable.*

Corollary 3.8. *The T-non-restricting control problem is decidable for L/U PGA if T contains only controllable edges.*

Proof. In the transformation proposed in the proof of theorem 3.2, if the edges in T are controllable, the extra widgets are compatible with the L/U restriction, and therefore starting from an L/U PGA, we obtain again an L/U PGA for which the control problem is decidable. □

In the previous proof, if some edge in T is uncontrollable, we cannot in general make the same encoding for that edge because the added transitions are controllable and, by the L/U restriction, cannot in general have the same guard as an uncontrollable edge (except if this guard has no parameters but then the non-restriction constraint is useless). Actually, in that case the problem is also undecidable:

Theorem 3.3. *The T-non-restricting control problem is undecidable for L/U game automata.*

Proof. We prove that if T can contain uncontrollable edges, then we can solve the control problem for PGA, which is undecidable, using the T-non-restricting problem for L/U PGA.

Let $\mathcal{G}$ be some PGA. We do the following transformation: for each parameter p that is used both as an upper bound and a lower bound in $\mathcal{G}$, we replace it by two parameters p^- and p^+, using p^- for lower bounds and p^+ for upper bounds. We thus obtain an L/U PGA. To this L/U PGA, we further add a location l^*, which is completely disconnected from the rest, and to that location we add self-loops in the following manner: for each pair p^- and p^+, we add a controllable self-loop with guard $p^- \leq x \leq p^+$, for some arbitrary clock x, and an uncontrollable self-loop with guard $p^+ \leq x \leq p^-$. By construction, we still have an L/U PGA. Let us call it $\mathcal{G}'$. We define T as the set of all these self-loops.

Clearly, a parameter valuation v is not T-restricting if and only if $v(p^-) \leq v(p^+)$ (due to controllable edges in T) and $v(p^+) \leq v(p^-)$ (due to uncontrollable edges in T), that is if and only if $v(p^-) = v(p^+)$. And with this condition, the runs in $v(\mathcal{G})$ and in $v(\mathcal{G}')$ are isomorphic. Consequently, v is a solution to the T-non-restricting control problem for $\mathcal{G}'$ if and only if it is a solution to the control problem for $\mathcal{G}$. □

As a consequence of Theorem 3.3, we need to further restrict L/U PGA to obtain decidability for the non-restricting control problem. As we have seen above we could just enforce non-parameterized guards in uncontrollable transitions, but we can do a bit better: 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 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 3.9 (Simple L/U game automata)**.** A simple L/U game automaton is a L/U game automaton in which the guards of the uncontrollable transitions are constraints of the form $k \leq x \leq p$ or $p \leq x \leq k$, where $x \in X$, $p \in P$, and $k \in \mathbb{Q}$.

Theorem 3.4. *The T-non-restricting control problem for simple L/U game automata is decidable.*

Proof. In order to prove Theorem 3.4, we build from any L/U simple game automaton $\mathcal{G}$ a TGA as follows. First remark that, by using the transformation described in the proof of Theorem 3.2, we can assume that T contains only uncontrollable edges.

Recall that by the L/U restriction:

- parameters $p_i^u \in P^u$ are used as the lower bounds in the guards on the uncontrollable transitions,
- parameters $p_i^l \in P^l$ are used as the 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 the maximal constant that appears as a lower bound in the guards containing p_i^l as an upper bound. Let $\mathcal{G}[\max, \min]$ represent the L/U TGA obtained with the parameter valuation that assigns 0 to each lower bound parameter and ∞ to each upper bound parameter that appears only in controllable transitions or in uncontrollable transitions not in T, and $\max(p_i^l)$ (resp. $\min(p_i^u)$) to every other p_i^l (resp. p_i^u) bound parameter.

Lemma 3.10. *Let $\mathcal{G}$ be simple L/U PGA, l_{goal} one its locations, and T a subset of its edges. The T-non-restricting problem for $\mathcal{G}$ is equivalent to the control problem for the TGA $\mathcal{G}[\min, \max]$.*

Proof. Clearly, by construction, none of the components of this valuation for p^u parameters can be increased because it is already ∞ or because that would give a T-restricting valuation. Similarly, none of the components of this valuation for p^l parameters can be decreased because it is already 0 or because that would give a T-restricting valuation.

So $\mathcal{G}[\max, \min]$ contains all the possible behaviors for a non-restricting valuation and the rest of the proof proceeds as in Lemma 3.4. □

The Theorem follows immediately. □

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

4. A Symbolic Semi-Algorithm to Solve Parametric Timed Games

For timed reachability games, a winning strategy for the controller can be synthesized using a well-known backward fixed-point algorithm for solving timed games (Maler et al., 1995). The algorithm is based on the time and action predecessor operators (Alfaro, Henzinger, & Majumdar, 2001; Maler et al., 1995), that compute the set of

winning states, starting from the goal location. We examine in this section, how this algorithm can be extended to the parametric case.

4.1. *Parametric symbolic state*

In order to represent the infinite state space of PGA, we need an abstraction. We use here an extension of the classical symbolic states abstraction of TA and TGA (Larsen, Pettersson, & Yi, 1995).

Definition 4.1 (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 $X \cup P$.

Given an arbitrary order on clocks and variables, their $\mathbb{R}$-valuations can be seen as points in the $|X \cup P|$-dimensional space $\mathbb{R}^{|X \cup P|}$. Valuation sets that be reached by a given sequence of edges can be represented by convex polyhedra (Jovanović, Lime, & Roux, 2015).

For the computation of the state space, we define the following parametric extensions of the classical operations on valuation sets (Jovanović et al., 2015):

- future: $Z^{\nearrow} = \{v' \mid \forall p \in P, v'(p) = v(p) \text{ and } \forall x \in X, v'(x) = v(x) + d, d \geq 0\}$;
- reset of the clock variables 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 PTA $\mathcal{A} = (L, l_0, \Sigma, X, P, E, \mathsf{Inv})$: $\mathsf{Init}(\mathcal{A}) = (l_0, \{v \in \mathbb{R}^{X \cup P} \mid v_{|X} \in \{\vec{0}_X\}^{\nearrow} \cap v_{|P}(\mathsf{Inv}(l_0))_{|X}\})$;
- successor by some edge $e = (l, a, \gamma, R, l')$: $\mathsf{Succ}((l, Z), e) = (l', (Z \cap \gamma)[R]^{\nearrow} \cap \mathsf{Inv}(l'))$

For the backward computation of winning states, we need the following operators:

- past: $Z^{\swarrow} = \{v' \mid \exists v \in Z \, s.t. \, \forall p \in P, v'(p) = v(p) \text{ and } \forall x \in X, v'(x) = v(x) - d, d \geq 0\}$
- inverse reset of clocks in set $R \subseteq X$: $Z[R]^{-1} = \{v' \mid \exists v \in Z \text{ s.t. } \forall p \in P, v'(p) = v(p) \text{ and } \forall x \in X, v(x) = 0 \text{ if } x \in R \text{ and } v'(x) = v(x) \text{ if } x \notin R\}$
- predecessor by edge $e = (l, a, \gamma, R, l')$: $\mathsf{Pred}((l', Z), e) = (l, Z[R]^{-1} \cap \gamma \cap \mathsf{Inv}(l))$.

The predecessor by an edge operation is extended by union to define controllable and uncontrollable action predecessors (predecessors by edge):

- controllable predecessors: $\mathsf{cPred}((l', Z)) = \bigcup_{c \in \Sigma^c} \mathsf{Pred}((l', Z), c)$
- uncontrollable predecessors: $\mathsf{uPred}((l', Z)) = \bigcup_{u \in \Sigma^u} \mathsf{Pred}((l', Z), u)$

We also need to define a *safe-timed predecessors* ($\mathsf{Pred_t}$) operator. Let $S_1, S_2 \subseteq S$, both having the same location, and where S is the set of states in the semantics of a PGA. A state $(l, v) \in S$ is in $\mathsf{Pred_t}(S_1, S_2)$ if from (l, v) we can reach $(l, v') \in S_1$ by time elapsing and along the path from (l, v) to (l, v') avoid S_2, formally:

$$\mathsf{Pred_t}(S_1, S_2) = \{(l, v) \mid \exists d \geq 0 \text{ s.t. } (l, v) \xrightarrow{d} (l, v'), (l, v') \in S_1 \text{ and } Post_{[0,d]}(l, v) \subseteq S \backslash S_2\}$$

where $Post_{[0,d]}(l, v) = \{(l, v') \in S \mid \exists t \in [0, d] \text{ s.t. } (l, v) \xrightarrow{t} (l, v'), v'(x) = v(x) +$

t, if $x \in X; v'(x) = v(x)$ if $x \in P\}$ is the future operator limited to a maximum time elapising of d time units.

This corresponds intuitively to the states that can reach S_1 by delay, without going through any state in S_2 along the path. Safe-timed predecessor operator can also be expressed as follows:

Lemma 4.2 ((Cassez et al., 2005)). *For any two symbolic states S_1 and S_2, such that S_2 is convex:*

$$\mathsf{Pred_t}(S_1, S_2) = (S_1^{\swarrow} \setminus S_2^{\swarrow}) \cup ((S_1 \cap S_2^{\swarrow}) \setminus S_2)^{\swarrow}$$

Also, the following distribution law holds:

Lemma 4.3 ((Cassez et al., 2005)). *For any two symbolic states $S_1 = \bigcup_i S_{1i}$ and $S_2 = \bigcup_j S_{2j}$:*

$$\mathsf{Pred_t}(\bigcup_i S_{1i}, \bigcup_j S_{2j}) = \bigcup_i \bigcap_j \mathsf{Pred_t}(S_{1i}, S_{2j})$$

4.2. *Computing the Winning States in Parametric Timed Games*

Lemma 4.4. *For any location l, any set of valuations on both clocks and parameters Z, Z', and any parameter valuation $v_{|P}$:*

(1) $v_{|P}(Z^{\swarrow}) = v_{|P}(Z)^{\swarrow}$
(2) $v_{|P}(Z \cap Z') = v_{|P}(Z) \cap v_{|P}(Z)$
(3) $v_{|P}(Z[R]^{-1}) = v_{|P}(Z)[R]^{-1}$
(4) $v_{|P}(Z \setminus Z') = v_{|P}(Z) \setminus v_{|P}(Z)$
(5) for any edge e, $v_{|P}(\mathsf{Pred}((l, Z), e)) = \mathsf{Pred}((l, v_{|P}(Z)), v_{|P}(e))$
(6) $v_{|P}(\mathsf{Pred_t}(Z_1, Z_2)) = \mathsf{Pred_t}(v_{|P}(Z_1), v_{|P}(Z_2))$

Proof. The proofs for the first two items can be found in the Lemma 1 of Jovanović et al. (2015). The proof is given for the future operator however. The proof for the past operator is extremely similar and needs just changing the sign for time elapsing.

Consider now the inverse reset operator. Suppose that $v_{|X} \in v_{|P}(Z[R]^{-1})$. Then the valuation v on clocks and parameters obtained by combining $v_{|X}$ and $v_{|P}$ belongs to $Z[R]^{-1}$. Then there exists $v' \in Z$ such that $v'_{|P} = v_{|P}$, and for all clock x, if $x \in R$ then $v'(x) = 0$ and if $x \notin R$ then $v'(x) = v(x)$. It follows that $v'_{|X} \in v_{|P}(Z)$, $v'_{|X}$ and for all clock x, if $x \in R$ then $v'_{|X}(x) = 0$ and if $x \notin R$ then $v'_{|X}(x) = v_{|X}(x)$. This, in turn, means that $v_{|X} \in v_{|P}(Z)[R]^{-1}$. The other direction works similarly.

We now turn to set difference. Consider $v_{|X} \in v_{|P}(Z \setminus Z')$. Then the valuation v on clocks and parameters obtained by combining $v_{|X}$ and $v_{|P}$ belongs to Z but not to Z' and therefore $v_{|X} \in v_{|P}(Z)$ but not in $v_{|P}(Z')$, and finally it belongs to the difference of the two. The other direction is similar.

The proof for Pred follows from the results for intersection and inverse reset. That of $\mathsf{Pred_t}$ follows from Lemma 4.2 and the results for past and intersection. □

We now present a well-known backward fixed-point algorithm for solving timed games with a reachability objective (Maler et al., 1995). We first compute forward the whole reachable state space, then compute backwards the winning states. In Maler et al. (1995), the authors only use the backwards computation. Here, we also intersect with the actual reachable state space.

The computation therefore consists of a sequence of two fixed-points on the state-space of the PGA. To handle these sets of states, we use, as before, the notion of parametric symbolic state from Definition 4.1, $S = (l, Z)$, and the corresponding operators.

We 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.

As a direct consequence of Jovanović et al. (2015), the reachable state-space of the PGA $\mathcal{G}$ can then be computed by the following fixed-point (when it exists):

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

The final fixed-point set is noted S^*.

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

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

When it exists, the final fixed-point set is noted W^*. We give the following result:

Lemma 4.5. *For a PGA $\mathcal{G}$, a location l_{goal}, and a state (l, v), it holds that for all i, (l, v) is reachable and there exists a winning strategy enforcing l_{goal} in at most i controllable steps from $(l, v_{|X})$ in $v_{|P}(\mathcal{G})$ iff $(l, v) \in W_i$.*

Proof. We proceed by induction. The property obviously holds for W_0. Now, suppose it holds for some $n \geq 0$.

(1) We first prove the left to right implication. Let (l, v) be a state in W_{n+1}. Then $(l, v) \in S^* \cap (\mathsf{Pred_t}(\mathsf{cPred}(W_n), \mathsf{uPred}(S^* \setminus W_n)) \cup S_{goal})$. If $(l, v) \in S_{goal}$ we are done, else, by Lemma 4.4, the latter implies that $(l, v_{|X})$ is in $\mathsf{Pred_t}(\mathsf{cPred}(v_{|P}(W_n)), \mathsf{uPred}(v_{|P}(S^*) \setminus v_{|P}(W_n)))$. By definition of $\mathsf{Pred_t}$, there exists a non-negative delay d such that $v_{|X} + d \in \mathsf{cPred}(v_{|P}(W_n))$ and for all $0 \leq d' \leq d$, $v_{|X} + d' \notin \mathsf{uPred}(v_{|P}(S^*) \setminus v_{|P}(W_n))$. So, by delaying from $v_{|X}$, we can enforce the reachability of $\mathsf{cPred}(v_{|P}(W_n))$. From there, by taking a controllable edge we can enforce the reachability of $v_{|P}(W_n)$ and, using the induction hypothesis, after that we can win in at most n controllable edges. Putting it all together, we can then force the win from v_X, in at most $n+1$ controllable edges.

(2) Now, we prove the right to left implication. If there is a strategy to win in at most $n + 1$ steps, then:
   - Either we need no controllable edges at all, then we are already in S_{goal} and we are done;
   - Or we need at least one controllable edge (possibly after some delay), which can be forced, and then we can win in at most n controllable edges. By the induction hypothesis, this latter statement means we are then in

$v_{|P}(W_n)$. So, from $v_{|X}$, we can force the reachability of $\mathsf{Pred}(v_{|P}(W_n))$, possibly after some delay, which by definition of $\mathsf{Pred_t}$ is exactly $v_{|X} \in \mathsf{Pred_t}(\mathsf{cPred}(v_{|P}(W_n)), \mathsf{uPred}(\mathbb{R}^X_{\geq 0} \setminus v_{|P}(W_n)))$. Since (l, v) is reachable, it is also in S^* and it is therefore also equivalent to only consider uncontrollable transitions in *reachable states* that could divert us from W_n. Finally, we get $v_{|X} \in v_{|P}(S^*) \cap \mathsf{Pred_t}(\mathsf{cPred}(v_{|P}(W_n)), \mathsf{uPred}(v_{|P}(S^*) \setminus v_{|P}(W_n)))$ and we conclude with Lemma 4.4.

□

Now, with Lemma 4.5, we can prove the correctness and completeness of the algorithm.

Theorem 4.1 (Correctness and completeness)**.** *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 $v \in (W^* \cap (l_0, \vec{0}_X \times \mathbb{Q}^{|P|}))_{|P}$.*

Proof. (1) We start by proving the right to left implication. Suppose $v \in (W^* \cap (l_0, \vec{0}_X \times \mathbb{Q}^{|P|}))_{|P}$. Then there exists a state (l_0, v_0) in W^* such that $v_{0|P} = v$ and $v_{0|X}$ has all coordinates equal to 0. Valuation v_0 is on $X \cup P$ and since it belongs W^*, it belongs to W^n for some n. We can then apply Lemma 4.5 to conclude.

(2) 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 4.5, 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 W^n for some n, and therefore to W^*, which concludes the proof.

□

Remark that the non-restriction constraint of Section 2 can be easily enforced in the symbolic computation presented in this section. For each guard, linear constraints on the parameters ensuring that the guard will not be uniformly false can be statically derived. We can then constrain the initial symbolic state of the PGA with these additional constraints and carry out the rest of the computation with no further change. We thus obtain the winning states (and associated parameter valuations) only for non-restricting parameter valuations.

4.3. *Winning Strategy*

In this section we show how to extract the winning strategy from the set of winning states. We first recall a classical result (straightforwardly extended to the parametric setting):

Theorem 4.2 ((Maler et al., 1995))**.** *Let $(\mathcal{G}, l_{goal})$ be a parametric reachability timed game. For all parameter valuations v, if there exists a winning strategy in $v(\mathcal{G})$ then there exists a memory-less winning strategy in $v(\mathcal{G})$.*

With this theorem, and following Maler et al. (1995), it is easy to extract a memory-less winning strategy from the set of winning states. We proceed as follows: controllable action predecessors give us states from which a corresponding controllable action should be taken, while timed predecessors give us states in which we should delay. Since we work on symbolic states, for all the (concrete) states in a given symbolic state of W^*, in particular with the same clock valuations but different parameter

valuations, give the same strategies by this procedure. This thus gives "parametric strategies" defined in reference to the parameters. We illustrate how the computation of a memory-less strategy works with the following example.

4.4. *Example*

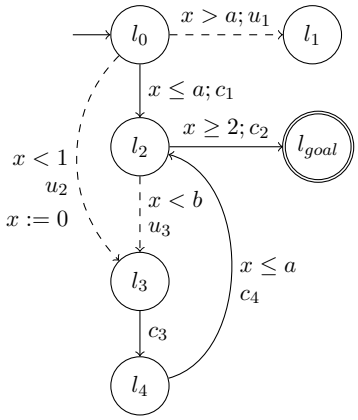

Figure 1. A L/U game automaton

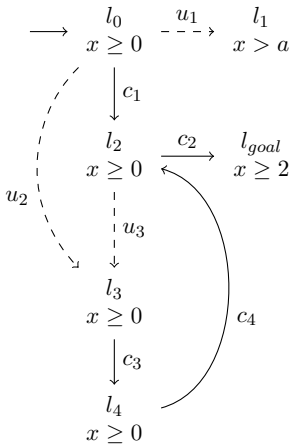

Figure 2. Simulation graph of the L/U game automaton of Figure 1

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 1). 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. The reachability game consists in finding a strategy for the controller that eventually ends up in the location l_{goal}.

We 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 start from the complete symbolic reachability graph and show the back-propagation of winning states.

After the computation of the symbolic reachability graph, shown in Figure 4.4, the backward algorithm starts from the symbolic winning subset $(l_{goal}, x \geq 2)$. By a controllable action (c_2) predecessor, we obtain $(l_2, x \geq 2)$. 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 $(l_2, x \geq b)$. One of the controllable transitions taking us to loc_2 is c_4. A controllable action predecessor (c_4) adds a constraint $x \leq a$. The constraint on the parameters derived in this state is $a \geq b$. This constraint is back-propagated to the preceding states. The (safe) timed predecessors give us the state $(l_4, x \geq 0 \wedge a \geq b)$.

We obtain successively the following sets of winning states: $(l_3, x \geq 0 \wedge a \geq b)$, $(l_2, (x \geq b) \vee (x \geq 0 \wedge a \geq b))$ and $(l_0, (x \leq a) \wedge \big((x < 1 \wedge a \geq b) \vee x \geq 1\big) \wedge \big((x \geq b) \vee (x \geq 0 \wedge a \geq b)\big)$. The last one simplifies to $(l_0, (x \leq a \wedge a \geq b))$.

We stop here with the details of the computation and give the set of winning states obtained upon the termination of the algorithm: $(l_0, (x \leq a) \wedge \Big((x < 1 \wedge a \geq b) \vee (x \geq$

$b) \vee (x \geq 0 \wedge a \geq b)\big)$, $(l_2, (x \geq b) \vee (x \geq 0 \wedge a \geq b))$, $(l_3, x \geq 0 \wedge a \geq b)$, $(l_4, x \geq 0 \wedge a \geq b)$ and $(l_{goal}, x \geq 2)$.

Let us now show how to extract a winning strategy from the winning set of states. The symbolic state $(l_2, x \geq 2)$ is the controllable action predecessor of $(l_{goal}, x \geq 2)$ by action c_2. Then the winning strategy is: in all states of $(l_2, x \geq 2)$ the controllable transition c_2 should be taken immediately, and in $(l_2, x \geq 0)$, we should delay until $x \geq 2$. The controllable action predecessor from l_2 takes us to the symbolic state $(l_4, x \geq b \wedge x \leq a)$, deriving a constraint $a \geq b$. From that state action c_2 should be taken immediately, and timed predecessors give the symbolic state $(l_4, x \geq 0, a \geq b)$ in which we should delay until $x \geq b$. A whole winning strategy consists in:

In all states:	Do:
$(l_0, 0 \leq x \leq a)$	c_1
$(l_2, 0 \leq x < 2)$	delay
$(l_2, x \geq 2)$	c_2
$(l_3, x \geq 0)$	c_3
$(l_4, 0 \leq x < b)$	delay
$(l_4, x \geq b \wedge x \leq a)$ (recall that $b \leq a$)	c_4

Most Permissive Strategy Notice that there may be several winning strategies. Algorithmically speaking, the order of exploration of the winning states leads to different winning strategies. As an example, applying controllable predecessor from $(l_2, (x \geq b) \vee (x \geq 0 \wedge a \geq b))$ to l_0 can lead to both strategies from l_0:

(1) doing c_1 in all states (l_0, x) with $x \leq a$;
(2) delaying in all states (l_0, 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 $b \leq a$).

The most permissive strategy can be defined as the union of all these winning strategies.

5. Case study

The applications of parametric real-time control cover a wide range of domains. Among them are manufacturing systems, traffic systems, communication protocols, logistic systems, hardware circuits and embedded systems. For the latter, our approach allows of course the synthesis of controllers of the environment, but also the synthesis of device drivers, as well as the synthesis of real-time schedulers. For such systems it is often impossible to obtain the complete knowledge of the system, especially in the early design stages and even when all the timing constants are known, if the execution of the system slightly deviates from the expected behavior, the system synthesized by a non-parametric approach might not satisfy the expected properties anymore. The interest of the use of parameter has been shown for the design of a concrete industrial aerial video tracking system made by Thales (Parquier et al., 2016). Finally, considering a wide range of values for timing constants jointly with a control approach allows for a more flexible and robust design.

We now show on a case study how the parametric reachability control problem modeled by a simple L/U game automata can be used for the design of a system.

We will first show that, regardless of the values of the parameters, there is no winning strategy that eventually end up in the goal location. We then modify the

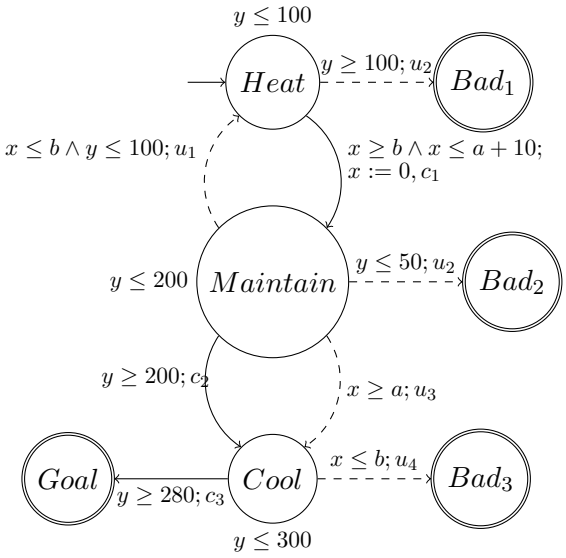

Figure 3. A Copper Annealing Controller model

model of the system in order to obtain a winning strategy that we generate with our algorithm.

The Copper Annealing Controller Let us consider the Copper Annealing Controller depicted in Figure 3. 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 3 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.

We assume that the parameters are non-negative.

The parameter a means that a heating stage is followed by a maintaining stage whose duration can be either longer than the heating duration or shorter, but no more than 10 time units shorter. 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.

No winning strategy 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 location $Maintain$ and then the location 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 location and two controllable actions).

Winning states Thus, on the corrected model, 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 location $Heat$ can be left before the condition $y \geq 100$ becomes true; the condition $a > 40$ allows to avoid Bad_2 by ensuring that the location $Maintain$ can be reached with $y > 50$ and the condition $a > b$ allows to avoid Bad_3.

T-non-restricting parameter valuations Let T be the set of edges of the L/U PGA of the Figure 3. All the valuations of the model such that $b < a + 10$ are T-restricting since they make the guard $x \geq b \wedge x \leq a+10$ not satisfiable. However all the valuations such that there exists a winning strategy (i.e. $(b < 100) \wedge (a > 40) \wedge (a > b)$) are T-non-restricting.

Instantiating the parameters The constraints on the parameters are $a > 40$, $b < 10$ and $a > b$. A sensible choice in this domain is $a = 60$ and $b = 50$ which indeed satisfies the constraints and also ensures a good level of robustness, with a being small enough not to delay the process too much. For more complex constraints, linear programming could be used to optimize the design.

Winning strategy A winning strategy corresponding to this parameter valuation, and extracted from the winning states set, consists in:

In all states (recall that $x \geq 0$ and $y \geq 0$):	Do:
$(Heat, \{y \leq 50 \text{ or } x < 50\})$	delay
$(Heat, \{y > 50 \text{ and } 50 \leq x \leq 70\})$	c_1
$(Maintain, \{y < 200\})$	delay
$(Maintain, \{y = 200\})$	c_2
$(Cool, \{y < 280\})$	delay
$(Cool, \{280 \leq y\})$	c_3

Implementing the strategy Implementing such a strategy has been studied in Wulf, Doyen, and Raskin (2005). It can be done on different targets such as field programmable gate arrays (FPGA) (Fleming & Thomas, 2013) or microcontroller (Bandur, Kahl, & Wassyng, 2012). For more complex strategy and in particular for distributed system, the implementation of distributed timed automata specification is proposed in Devillers, Didier, and Klaudel (2013) allowing to guarantee that the specification are preserved by the implementation.

6. Conclusion

In this paper we have studied control problems for timed automata extended with timing parameters, expressed in terms of parametric timed games. In that setting, the existence of parameter values such that a controller enforcing the reachability of some control location exists is undecidable. We have therefore proposed a decidable subclasses based on a restricted use of parameters in the clock constraints, in the spirit of the L/U automata (Hune et al., 2002).

We have also proposed an extension of a well-known fixed-point backward algorithm for solving timed games of Maler et al. (1995), for the parametric approach. In that parametric setting, our algorithm consists of two parts, the fixed-point forward exploration of the state-space and the fixed-point backward computation of winning states. Its termination is not guaranteed but when it does terminate, it gives the set of symbolic constraints on parameters and the set of winning states, from which it is easy to extract a winning strategy and, in turn, a controller.

In future work, we plan to implement the algorithm and for this we will focus on different restrictions on the use of parameters to ensure the decidability of the control problems and the termination of the synthesis procedures. In particular, since in practice the timing features of systems are given as integers, we will apply the *bounded integer* approach of Jovanović, Lime, and Roux (2013); Jovanović et al. (2015). Moreover, to avoid an explicit enumeration of all the possible values of parameters we will implement a modification of the symbolic computation of S^* that preserves the integer parameter valuations which will be given as symbolic constraints between parameters.

In order to cover a wider range of real-time systems, in particular those using preemptive scheduling, we also plan to extend this work to Petri Nets with stopwatches Berthomieu, Lime, Roux, and Vernadat (2007). In such a formalism, we allow clocks to be stopped and resumed later on, while memorizing their values. We will rely on existing prior work on such formalisms (e.g. Berthomieu et al. (2007); Lime and Roux (2009) for stopwatch Petri nets or Alur et al. (1995) for hybrid automata). As already noted in Henzinger, Ho, and Wong-toi (1997), for stopwatch automata, a non initialized stopwatch with a null timed derivative can be interpreted as a timing parameter. The basic reachability problems for stopwatch automata or Petri nets are undecidable however, even without using parameters. We nonetheless want to investigate the design of incomplete algorithms, or efficient semi-algorithms for stopwatch Petri nets with parameters.

References

Alfaro, L. d., Henzinger, T. A., & Majumdar, R. (2001). Symbolic algorithms for infinite-state games. In *Proceedings of the 12th Int. Conference on Concurrency Theory (CONCUR '01)* (pp. 536–550). London, UK.

Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T., Ho, P.-H., Nicollin, X., ... Yovine, S. (1995). The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, *138*(1), 3 - 34.

Alur, R., & Dill, D. (1994). A Theory of Timed Automata. *Theoretical Computer Science*, *126*(2), 183-235.

Alur, R., Henzinger, T. A., & Vardi, M. Y. (1993). Parametric real-time reasoning. In *ACM symposium on theory of computing* (p. 592-601).

André, É., Chatain, T., Encrenaz, E., & Fribourg, L. (2009). An inverse method for parametric timed automata. *Int. Journal of Foundations of Computer Science*, *20*(5), 819-836.

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

Bandur, V., Kahl, W., & Wassyng, A. (2012). Microcontroller assembly synthesis from timed automaton task specifications. In *Formal methods for industrial critical systems - 17th international workshop, FMICS 2012, Paris, France, 2012.* (pp. 63–77).

Behrmann, G., Cougnard, A., David, A., Fleury, E., Larsen, K. G., & Lime, D. (2007). Uppaal-tiga: Time for playing games! In *Conf. on Computer Aided Verification (CAV 2007)* (p. 121-125).

Berthomieu, B., Lime, D., Roux, O. H., & Vernadat, F. (2007). Reachability problems and abstract state spaces for time Petri nets with stopwatches. *Journal of Discrete Event Dynamic Systems (jDEDS)*, *17*(2), 133–158.

Bozzelli, L., & La Torre, S. (2009). Decision problems for lower/upper bound parametric timed automata. *Formal Methods in System Design*, *35*(2), 121-151.

Bruyère, V., & Raskin, J.-F. (2007). Real-time model-checking: Parameters everywhere. *Logical Methods in Computer Science*, *3*(1), 1–30.

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

Cassez, F., Jessen, J. J., Larsen, K. G., Raskin, J.-F., & Reynier, P.-A. (2009). Automatic synthesis of robust and optimal controllers - an industrial case study. In *Hscc* (p. 90-104).

Devillers, R. R., Didier, J., & Klaudel, H. (2013). Implementing timed automata specifications: The "sandwich" approach. In *13th International Conference on Application of Concurrency to System Design, ACSD 2013, Barcelona, Spain, 8-10 July, 2013* (pp. 226–235).

Fleming, S. T., & Thomas, D. B. (2013). FPGA based control for real time systems. In *23rd International Conference on Field programmable Logic and Applications, FPL 2013, Porto, Portugal, September 2-4, 2013* (pp. 1–2).

Henzinger, T. A., Ho, P.-H., & Wong-toi, H. (1997). Hytech: A model checker for hybrid systems. *Software Tools for Technology Transfer*, *1*, 460–463.

Henzinger, T. A., Nicollin, X., Sifakis, J., & Yovine, S. (1994). Symbolic model checking for real-time systems. *Information and Computation*, *111*(2), 193–244.

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

Jessen, J. J., Rasmussen, J. I., Larsen, K. G., & David, A. (2007). Guided controller synthesis for climate controller using uppaal tiga. In *Formal Modeling and Analysis of Timed Systems, 5th International Conference, FORMATS 2007, Salzburg, Austria, October 3-5, 2007, LNCS 4763* (p. 227-240). Springer.

Jovanović, A., Lime, D., & Roux, O. H. (2013, October). Synthesis of bounded integer parameters for parametric timed reachability games. In *11th International Symposium on Automated Technology for Verification and Analysis (ATVA 2013)* (Vol. 8172, pp. 87–101). Hanoi, Vietnam: Springer.

Jovanović, A., Lime, D., & Roux, O. H. (2015). Integer parameter synthesis for real-time systems. *IEEE Transactions on Software Engineering (TSE)*, *41*(5), 445–461.

Larsen, K. G., Pettersson, P., & Yi, W. (1995). Model-Checking for Real-Time Systems. In *Fundamentals of computation theory, lncs 965* (pp. 62–88).

Lime, D., & Roux, O. H. (2009). Formal verification of real-time systems with preemptive scheduling. *Journal of Real-Time Systems*, *41*(2), 118–151.

Maler, O., Pnueli, A., & Sifakis, J. (1995). On the synthesis of discrete controllers for timed systems. In *Proc. STACS'95, LNCS 900* (pp. 229–242). Springer.

Parquier, B., Rioux, L., Henia, R., Soulat, R., Roux, O. H., Lime, D., & André, E. (2016, November). Applying parametric model-checking techniques for reusing real-time critical systems. In *5th Int. Workshop on Formal Techniques for Safety-Critical Systems (FTSCS 2016), vol. 694 of CCIS.* Tokyo, Japan: Springer.

Wulf, M. D., Doyen, L., & Raskin, J. (2005). Systematic implementation of real-time models. In *FM 2005: Formal Methods, international symposium of formal methods, Newcastle, Uk, july 18-22, 2005* (pp. 139–156).