Control of Real-Time Systems With Integer Parameters

Aleksandra Jovanovic, Didier Lime, Olivier H Roux

Control of Real-time Systems with Integer Parameters

Aleksandra Jovanović, Didier Lime, Olivier H. Roux

***Abstract*—We consider the problem of synthesizing controllers for real-time systems where some timing features are not known with precision. We model the plant as a parametric timed automaton (PTA), 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. 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-trivial 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 restriction on the use of parameters to ensure the decidability of the control problems. Since in classical timed automata, real-valued clocks are always compared to integers for all practical purposes, we search for parameter values as bounded integers. Hence we solve undecidability and termination issues, we provide terminating symbolic synthesis procedures, and our method retains most of the practical usefulness of PTA for the modeling of real-time systems.

***Index Terms*—Timed Automata, Control, Game Theory, Parameters, Synthesis.**

I. Introduction

Designing real-time systems is a challenging issue and formal models and reasoning are key elements in attaining this objective. In this context, timed automata (TA) [1] are a powerful and popular modeling formalism. They extend finite automata with timing constraints, in which clocks are compared to integer constants that model timing features of the system.

This formalism, however, requires complete knowledge of the system. It is thus difficult to use it in 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 designers to use parameters instead of concrete timing values.

Parametric timed automata [2] extend timed automata [1] 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 a parameter valuation such that the automaton has an accepting run, is undecidable [2]. This naturally led to the search for subclasses of the formalism for which some problems would be decidable. L/U automata [3] 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 state-space exploration, which would allow for explicit synthesis of all the suitable parameter valuations, still might not terminate [4].

Control of timed systems.

The introduction of 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 a finite automaton.

Instead of verifying the correctness of a system, we have here the problem of synthesizing a 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, 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 next state of the system is then determined from both those actions, possibly non-deterministically.

A formalism that is commonly used to describe such systems in a timed framework is timed game automata (TGA, [5]), 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 controller and the environment.

The *(reachability) control problem* for TGA is the problem of determining the strategy for the controller such that, no matter what the environment does, the system ends up in the desired location. This problem is known to be decidable [5]. The introduction of this formalism has been followed by the development of efficient algorithms [6] and tool support [7], successfully applied to several industrial case studies (e.g. [8], [9]).

This work has been partially funded by ANR project ProMiS number ANR-19-CE25-0015.

Aleksandra Jovanović is with Kreatize, Berlin, Germany.

Didier Lime and Olivier H. Roux are with École Centrale de Nantes, LS2N UMR CNRS 6004, Nantes, France.

Manuscript received june 15, 2018

Control of Parametric timed systems.

In [10], we have introduced a formalism for 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 have nonetheless extended the backward fixed-point algorithm for solving timed games with reachability objectives of [5] to the parametric setting [10]. This provides a semi-algorithm, which allows to obtain the set of symbolic constraints on the parameters together with the set of winning states for the controller. The termination, however, is not guaranteed.

Our contribution.

We propose to use a restriction scheme proposed in [4] for PTA: since in classical timed game automata, real-valued clocks are always compared to integers for all practical purposes, we solve undecidability and termination issues by computing parameters as bounded integers. Due to the finite number of parameters values, parameter synthesis can be trivially carried out by an explicit enumeration, but our main contribution is to symbolically compute the resulting set of parameter valuations, without that enumeration. The result is thus given as a symbolic constraint on parameters. The symbolic algorithm is based on the computation of the integer hull of the parametric symbolic states. It builds on the forward exploration of the state-space and backward propagation of winning states of [10], and adds the integer hull operator there to restrict the explored state space to integer parameters, and ensure termination when they are bounded. Surprisingly, we do not have to apply an integer hull in the backwards computation, in order to obtain the correct integer solution. This saves a lot of computational efforts since that operation is quite expensive.

A preliminary version of this work was published in [11]. The present article develops this work, unifies it with the backward fixed-point semi-algorithm for solving timed games proposed in [10] and gives full proofs for all the results.

Organization of the Paper.

The rest of the paper is organized as follows. Section II provides definitions about PGA, the problems we are considering, and recalls some negative decidability results. The semi-algorithm for computing the whole state-space of a PTA is recalled in Section III. In Section IV we motivate and present a new restriction scheme for the parameter synthesis, we give the complexity of the parametric control problem in this setting, we give the bounded integer parameter synthesis algorithm and we prove its correctness, completeness and termination. We present the execution of the algorithm on an example in Section V and we discuss in section VI the performance of the proposed approach implemented in our tool ROMÉO, illustrated on a small but realistic case-study. We conclude with section VII.

II. PARAMETRIC TIMED GAME AUTOMATA

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 $V \subseteq \mathbb{R}$, since set X is finite, and given an arbitrary order on its elements, a V-valuation w_c on $X = \{x_1, \ldots, x_n\}$ can be seen as the vector $(w_c(x_1), \ldots, w_c(x_n))$ of $\mathbb{R}^{|X|}$.

The set of valuations on clocks and parameters that satisfy a parametric clock constraint can then be seen as a convex polyhedron of $\mathbb{R}^{|X \cup P|}$. We use this remark extensively in the subsequent definitions to consider (parametric) clock constraints as sets.

Let us consider a parametric clock constraint γ. For any parameter valuation w_p, we note $w_p(\gamma)$ the linear constraint on clocks obtained by replacing each parameter p_i by its value $w_p(p_i)$. Similarly, for any clock valuation w_c, we note $w_c(\gamma)$ the linear constraint on parameters obtained by replacing each clock x_i by its value $w_c(x_i)$.

Note that by valuating both parameters and clocks in γ, we obtain $w_p(w_c(\gamma)) = w_c(w_p(\gamma))$ that can be seen as a boolean and takes its value in $\{true, false\}$.

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

A. Parametric Timed Games

Definition 1: A Parametric Timed Automaton (PTA) is a tuple $\mathcal{A} = (L, l_0, X, \Sigma, P, E, \mathsf{Inv})$, where L is a finite set of locations, $l_0 \in L$ is the initial location, X is a finite set of clocks, Σ is a finite alphabet of actions, P is a finite set of parameters, $E \subseteq L \times \Sigma \times G(X, P) \times 2^X \times L$ is a finite set of edges, 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 w_p on P, the structure $w_p(\mathcal{A})$ obtained from $\mathcal{A}$ by replacing each constraint γ by $w_p(\gamma)$ is a *timed automaton* with invariants (TA) [1], [12]. The behavior of a PTA $\mathcal{A}$ is described by the behavior of all timed automata obtained by considering all possible valuations of parameters.

Definition 2 (Semantics of a (P)TA): Given a parameter valuation w_p, and a PTA $\mathcal{A}$, the concrete semantics of $w_p(\mathcal{A})$, is the labeled transition system $(Q, q_0, \rightarrow)$ over $\Sigma \cup \mathbb{R}_{\geq 0}$ where:

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

A finite run of PTA $\mathcal{A}$, for a given a parameter valuation w_p, is a sequence of alternating delay and action transitions in the semantics of $w_p(\mathcal{A})$, $\rho = q_1 a_1 q_2 ... a_{n-1} q_n$, where $\forall i \in [1..n-1], q_i \in Q, a_i \in \Sigma \cup \mathbb{R}_{\geq 0}$, and $q_i \xrightarrow{a} q_{i+1}$. Infinite runs are defined similarly. We denote by $\mathsf{Runs}(w_p(\mathcal{A}))$ the set of runs starting in the initial state of $w_p(\mathcal{A})$, and by $\mathsf{Runs}(q, w_p(\mathcal{A}))$ the set of runs starting in q. The last state of a finite run ρ is denoted by $\mathsf{last}(\rho)$. A state q is said to be reachable in $\mathcal{A}$ if there exists a finite run $\rho \in \mathsf{Runs}(w_p(\mathcal{A}))$, such that $\mathsf{last}(\rho) = q$.

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

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

As for PTA, for any PGA $\mathcal{G}$ and any rational valuation on parameters w_p, the structure $w_p(\mathcal{G})$, obtained by replacing each constraint γ by $w_p(\gamma)$, is a timed game automaton [5], [6] (TGA). Note that a TGA can also be seen as a PGA with an empty set of parameters.

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

Definition 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 timed game (i.e. without parameters) is a parametric timed game $(\mathcal{G}, l_{goal})$ in which $\mathcal{G}$ is a TGA.

In a TGA, two players, the controller and the 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 (delay), or to take a particular controllable action.

At a given instant, if one player chooses to take an action and the other to delay, then the action is taken. If both choose an action then the result is the non-deterministic choice between the two actions. Finally, if both choose to delay, then nothing happens and we consider the next (different) instant such that one of the player chooses not to delay.

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 [5], [6], [13] is the first one where uncontrollable actions cannot help to win and is the one we consider in this paper.

Definition 5 (Strategy): A strategy $\mathcal{F}$ over $w_p(\mathcal{G})$ is a partial function from $\mathsf{Runs}(w_p(\mathcal{G}))$ 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 $q = (l, w_c)$, 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 $last(\rho) \xrightarrow{d'} q$ and $\mathcal{F}(\rho \xrightarrow{d'} q) = \mathsf{delay}$.

Since we focus on control problems for which the control objective (given a parameter valuation) is to reach a particular location of timed automata, [5] proves that to win, the controller needs only strategies that are memoryless and constant on regions (as defined in [1]). The former means that for a strategy $\mathcal{F}$ and a run ρ, $\mathcal{F}(\rho)$ only depends on $\mathsf{last}(\rho)$. The latter means that even if players are allowed to choose their actions at any instant, with such a strategy, they will not change their mind until either an action has happened or a sufficient duration has passed that makes one of the clock go from a non-integer value to an integer value, or from an integer value to a non-integer value. We assume nothing on the strategies of the environment.

$\mathsf{Outcome}$ defines the restricted behavior of $w_p(\mathcal{G})$, when the controller plays some strategy $\mathcal{F}$, with respect to all the possible strategies of the environment.

Definition 6 (Outcome): Let $\mathcal{G}$ be a PGA, w_p be a parameter valuation, and $\mathcal{F}$ be a strategy over $w_p(\mathcal{G})$. The outcome $\mathsf{Outcome}(q, \mathcal{F})$ of $\mathcal{F}$ from state q is the subset of runs in $\mathsf{Runs}(q, w_p(\mathcal{G}))$ 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 $\rho' \in \mathsf{Runs}(q, w_p\mathcal{G}))$ 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 outcome that are "long enough" to have a chance to reach the goal location: a run $\rho \in \mathsf{Outcome}(q, \mathcal{F})$ is *maximal* if it is has an infinite number of action transitions 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 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 $(w_p(\mathcal{G}), l_{goal})$ is winning from state q if for all runs $\rho \in \mathsf{MaxOut}(q, \mathcal{F})$, there is some state (l_{goal}, w_c) in ρ.

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

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

The parametric timed game $(\mathcal{G}, l_{goal})$ is winning if there exists some parameter valuation w_p such that $(w_p(\mathcal{G}), l_{goal})$ 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 [5] and there exists efficient symbolic algorithms for the computation of winning states and strategies [6]. 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 $(w_p(\mathcal{G}), l_{goal})$ 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 $w_p(\mathcal{A})$ is undecidable [2].

Now remark that in a game in which all transitions are controllable, we trivially have that there exists a winning strategy for the controller to enforce the reachability of some location if and only if that location is reachable. We can therefore check the reachability problem using the control problem, and the following theorem holds:

Theorem 1 ([10]): The parametric control problem for PGA is undecidable.

We have proposed in [10] subclasses of PGA, for which the parametric control problem is decidable. These classes are however severely restricted in their use of parameters. These restrictions are inspired by those defining L/U automata, which to the best of our knowledge are yet the less restrictive subclass of PTA for which the parametric control problem is decidable. Since PGA extend PTA there is little hope that we can do much better than L/U games in terms of syntactic subclasses. Also, even the L/U restriction is not enough in general to ensure the termination of the computation of the whole parametric state-space.

To solve this problem without heavy syntactic restrictions, we have propose (as in [4]) to restrict the problem of parameter synthesis to the search for bounded integer parameter values. This makes all the problems decidable, since we can enumerate all the possible valuations.

Lifting either one of the two assumptions (boundedness or integer) leads to undecidability [4].

To avoid an explicit enumeration of all the possible values of parameters we will propose an efficient symbolic method to find the solution. This has the additional advantage of giving the set of parameter valuations as a symbolic constraint between parameters. As a consequence to this negative result we now investigate restrictions to the PGA formalism to make the control problem decidable.

III. A SYMBOLIC STATE SPACE ABSTRACTION OF 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 [5]. The algorithm is based on the time and action predecessor operators [5], [14], that compute the set of winning states, starting from the goal location. In [5], the authors only use a backwards computation. In order to extend this algorithm to the integer parametric case, we need to compute forward the whole reachable state space, then compute backwards the winning states.

In this section, we present the forward computation of the state space w.r.t. our *bounded integer* approach.

A. State space of a PGA

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 [15].

Definition 8 (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 a valuation v on $X \cup P$, we define its projection $v_{|P}$ on P as the restriction of v to P such that $v_{|P}(x) = v(p)$ for all $p \in P$. We define similarly the projection $v_{|X}$ on X.

Given a set Z of valuations on $X \cup P$ and v a valuation on $X \cup P$, we define $v_{|P}(Z) = \{v'_{|X} | v' \in Z \text{ and } v'_{|P} = v_{|P}\}$. The definition of $v_{|X}(Z)$ is symmetric.

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 [4].

For the computation of the state space, we define the following parametric extensions of the classical operations on valuation sets [4]:

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

Valuations in the initial symbolic states must be such that all clocks are initially equal to 0 and then we saturate with delay while the invariant is satisfied.

For the symbolic successor, we take valuations that satisfy the guard, then we apply resets, and finally we saturate with delay while the invariant is satisfied.

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.

From [10], we know that 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^*.

The obvious problem with this approach is that the fixed-point computation of S^* might not terminate.

We now restrict the problem of parameter synthesis to the search for bounded integer parameter values. To avoid an explicit enumeration of all the possible values of parameters, and following the approach of [4], we now modify the symbolic computation of S^* so that it still preserves the integer parameter valuations.

B. Integer points and integer hulls

For the sake of the simplicity of the presentation, we henceforth consider that all polyhedra representing valuation sets over $X \cup P$ are topologically closed, i.e., all the constraints describing them are non-strict. We invite the reader to refer to [4, Section IV.C] for the explanation of how to compute the integer hull for non-necessarily closed polyhedra.

Let V be a set such that $\mathbb{Z} \subseteq V$. A V-valuation on some set Y is an *integer valuation* if $\forall x \in Y, v(x) \in \mathbb{Z}$. Given a set Z of V-valuations, we denote by $\mathsf{IntVects}(Z)$ the set of integer valuations in Z.

The *convex hull* of a set Z of V-valuations, denoted by $\mathsf{Conv}(Z)$, is the intersection of all the convex sets of V-valuations that contain Z.

The *integer hull* of a set Z of V-valuations, denoted by $\mathsf{IH}(Z)$ is defined as the convex hull of the integer valuations in Z: $\mathsf{IH}(Z) = \mathsf{Conv}(\mathsf{IntVects}(Z))$.

If D is not convex, then $\mathsf{IH}(D)$ can contain strictly more integer valuations than D. Yet, if D can be expressed as a finite union of convex sets $D = \bigcup_i Z_i$, then we can define an extension of the integer hull, which we call *integer shape*, by $\mathsf{IS}(D) = \bigcup_i \mathsf{IH}(Z_i)$. We then have an important property: $\mathsf{IntVects}(\mathsf{IS}(D)) = \mathsf{IntVects}(D)$.

C. (Bounded) Integer parameter state space

For the computation of S^*, as in [4] for the algorithm for the reachability-synthesis for PTA, we now replace all the occurrences of operator Succ with $\mathsf{ISucc}((l,Z),e) = \mathsf{IH}(\mathsf{Succ}((l,Z),e))$.

The reachable integer parameter state-space of the PGA $\mathcal{G}$ can then be computed by the following fixed-point

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

The final fixed point is noted IS^*.

It is easy to see, that all the symbolic states computed by Succ (and therefore ISucc) have convex valuations sets that can actually be represented by convex polyhedra. Actually, as shown in [3], these polyhedra have a special form, called *parametric zone*, that has the important following property: for any integer parameter valuation, the polyhedron obtained by replacing the parameter variables in the parametric zone by their values is a convex polyhedron on clock variables (usually called *zone*) with integer vertices (see [4], Property 3).

Therefore, by using the ISucc instead of Succ in the computation of the whole state-space S^*, we ensure termination and obtain a subset IS^* of S^* such that $\mathsf{IntVects}(IS^*) = \mathsf{IntVects}(S^*)$, provided we know a bound on the possible values for the parameters (from which we can derive a bound on the values of the clocks, see [4]).

IV. Integer Parameter Synthesis for Control

The well-known backward fixed-point algorithm for solving timed games [5] is based on the time and action predecessor operators [5], [14], that compute the set of winning states, starting from the goal location.

In this section, we will apply our *bounded integer* approach to parameter synthesis for parametric timed games, and show that surprisingly we do not have to apply the integer hull computation when back propagating the winning states.

We first recall some results of [10] on the parametric time and action predecessor operators, and their use in devising a semi-algorithm to solve parametric reachability games.

A. A semi-algorithm to solve parametric reachability games

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 $Post_{[0,d]}(l, v) = \{(l, v') \in S \mid \exists t \in [0, d] \text{ s.t. } (l, v) \xrightarrow{t} (l, v'), v'(x) = v(x) + t, \text{ if } x \in X; v'(x) = v(x) \text{ if } x \in P\}$ is the future operator limited to a maximum time elapsing 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 2 ([6]): 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 3 ([6]): For any two symbolic states $S_1 = \bigcup_i S_{1i}$ and $S_2 = \bigcup_j S_{2j}$:

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

Finally, we will, in the sequel use the following lemma:

Lemma 4 ([10]): 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) $v_{|P}(\mathsf{Pred_t}(Z_1, Z_2)) = \mathsf{Pred_t}(v_{|P}(Z_1), v_{|P}(Z_2))$

Now, if we denote by $S_{goal} = \{l_{goal}\} \times \mathbb{R}^{X \cup P}$, then a backwards algorithm for solving reachability games is given as the fixed-point computation of: $W_0 = \emptyset$ and

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

The computation of $\mathsf{Pred_t}(\mathsf{cPred}(W_n), \mathsf{uPred}(S^* \setminus W_n))$ is illustrated in Figure 1.

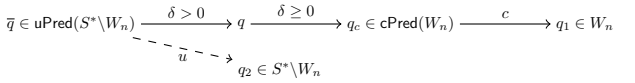

Fig. 1. $q \in \mathsf{Pred_t}(\mathsf{cPred}(W_n), \mathsf{uPred}(S^* \setminus W_n))$: q can reach W_n by delay and controllable action, without going in $S^* \setminus W_n$

Of course, this might not terminate. When it exists, however, the final fixed-point set is noted W^* and we have the following result:

Lemma 5 ([10]): 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 $(l, v) \in W_i$.

B. Integer backward operators

We can now extend the backward operators. From Lemma 2, the result of applying $\mathsf{Pred_t}$ to two parametric zones is a finite union of parametric zones, on which we will be able to apply $\mathsf{Pred_t}$ again with Lemma 3, leading again to finite unions of parametric zones. In the backward computation of winning states, we therefore use IS rather than IH.

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

We then have the following result:

Lemma 6: For any symbolic state (l, Z), with D a finite union of parametric zones, and any edge e:

$$\mathsf{IPred}(\mathsf{IS}((l, D), e)) = \mathsf{IS}(\mathsf{Pred}((l, D), e))$$

Proof: Let us first assume that Z is convex. Then $\mathsf{IS}(Z) = \mathsf{IH}(Z)$. We prove both inclusions:

1) $\mathsf{IPred}(\mathsf{IS}((l, Z)), e) \subseteq \mathsf{IS}(\mathsf{Pred}((l, Z), e))$.
   $\mathsf{IS}((l, Z)) \subseteq (l, Z)$ and since Pred and IS are non-decreasing, we immediately have $\mathsf{IPred}(\mathsf{IS}((l, Z)), e) \subseteq \mathsf{IS}(\mathsf{Pred}((l, Z), e))$.
2) $\mathsf{IPred}(\mathsf{IS}((l, Z)), e) \supseteq \mathsf{IS}(\mathsf{Pred}((l, Z), e))$.
   Let us now consider $v \in \mathsf{IntVects}(\mathsf{Pred}((l, Z), e))$. Then there exists $v' \in Z$ such that $v \in \mathsf{IntVects}(\mathsf{Pred}((l, \{v'\}), e))$. By definition of Pred we also have $v_{|P} = v'_{|P}$. So $v' \in v_{|P}(Z)$. And since v is an integer vector, so is $v_{|P}$ and $v_{|P}(Z)$ is therefore a zone of a classical TA and thus has integer vertices. Consequently, $v_{|P}(Z) = \mathsf{IS}(v_{|P}(Z))$. Moreover, $v_{|P}(Z) \subseteq Z$ (with a slight abuse of notations: each clock valuation in $v_{|P}(Z)$ needs to be combined with the parameter valuation $v_{|P}$ for the dimensions to match) and IS is non-decreasing, so $\mathsf{IS}(v_{|P}(Z)) \subseteq \mathsf{IS}(Z)$ and thus $v' \in \mathsf{IS}(Z)$ and $v \in \mathsf{IntVects}(\mathsf{Pred}((l, \mathsf{IS}(Z)), e))$. Finally, Conv being non-decreasing, we obtain $\mathsf{IS}(\mathsf{Pred}(\mathsf{IS}((l, Z)), e)) \supseteq \mathsf{IS}(\mathsf{Pred}((l, Z), e))$.

Now, suppose that $D = \bigcup_i Z_i$, with all Z_i convex. Then by definition of IS and Pred, we have $\mathsf{IPred}(\mathsf{IS}((l, Z), e)) = \bigcup_i \mathsf{IPred}(\mathsf{IS}((l, Z_i), e))$. Similarly, $\mathsf{IS}(\mathsf{Pred}((l, Z), e)) = \bigcup_i \mathsf{IS}(\mathsf{Pred}((l, Z_i), e))$. And using the result above for each convex Z_i, we complete the proof. ■

We define, in a similar way:

- integer controllable action predecessors: $\mathsf{cIPred}((l, Z)) = \mathsf{IS}(\mathsf{cPred}((l, Z)))$,
- integer uncontrollable action predecessors: $\mathsf{uIPred}((l, Z)) = \mathsf{IS}(\mathsf{uPred}((l, Z)))$.

Lemma 7: For any symbolic state (l, D), with D a finite union of parametric zones:

$$\mathsf{cIPred}(\mathsf{IS}((l, D))) = \mathsf{IS}(\mathsf{cPred}((l, D)))$$

Proof: Immediate with Lemma 6 from the facts that: $\mathsf{cPred}((l, D)) = \bigcup_{c \in \Sigma^c} \mathsf{Pred}(\mathsf{IS}((l, D), c)))$ and $\mathsf{IS}(D_1, D_2) = \mathsf{IS}(D_1) \cup \mathsf{IS}(D_2)$ when D_1 and D_2 are finite unions of parametric zones. ■

The same result obviously holds for uPred and we can finally extend this to the safe-timed predecessor operator by $\mathsf{IPred_t}(D_1, D_2) = \mathsf{IS}(\mathsf{Pred_t}(D_1, D_2))$.

Lemma 8: For any two finite unions of parametric zones D_1 and D_2:

$$\mathsf{IPred_t}(\mathsf{IS}(D_1), \mathsf{IS}(D_2)) = \mathsf{IS}(\mathsf{Pred_t}(D_1, D_2))$$

Proof: Since D_1 and D_2 are finite unions of parametric zones, we can write them as: $D_1 = \bigcup_i D_{1i}$ and $D_2 = \bigcup_j Z_{2j}$. By Lemma 3 we have $\mathsf{Pred_t}(\bigcup_i Z_{1i}, \bigcup_j Z_{2j}) = \bigcup_i \bigcap_j \mathsf{Pred_t}(Z_{1i}, Z_{2j})$. And by Lemma 2, for any i, j, we have: $\mathsf{Pred_t}(Z_{1i}, Z_{2j}) = (Z_{1i}^{\swarrow} \setminus Z_{2j}^{\swarrow}) \cup ((Z_{1i} \cap Z_{2j}^{\swarrow}) \setminus Z_{2j})^{\swarrow}$, because Z_{2j} is convex.

What we need to show then is that, for any two parametric zones Z_1 and Z_2:

- $\mathsf{IS}(Z_1 \cap Z_2) = \mathsf{IS}(\mathsf{IS}(Z_1) \cap \mathsf{IS}(Z_2))$
- $\mathsf{IS}(Z_1 \cup Z_2) = \mathsf{IS}(\mathsf{IS}(Z_1) \cup \mathsf{IS}(Z_2))$
- $\mathsf{IS}(Z_1^{\swarrow}) = \mathsf{IS}(\mathsf{IS}(Z_1)^{\swarrow})$
- $\mathsf{IS}(Z_1 \setminus Z_2) = \mathsf{IS}(\mathsf{IS}(Z_1) \setminus \mathsf{IS}(Z_2))$

These four results are quite straightforward. Let us just prove the first, the rest being similar.

First remark that Z_1 and Z_2 being convex, integer shapes are actually integer hulls. Second $\mathsf{IH}(Z) \subseteq Z$, for any Z and since IH is non-decreasing, $\mathsf{IH}(\mathsf{IH}(Z_1) \cap \mathsf{IH}(Z_2)) \subseteq \mathsf{IH}(Z_1 \cap Z_2)$. Finally, if $v \in \mathsf{IntVects}(Z_1 \cap Z_2)$ then clearly $v \in \mathsf{IntVects}(Z_1) \cap \mathsf{IntVects}(Z_2)$ and consequently $v \in \mathsf{IH}(Z_1) \cap \mathsf{IH}(Z_2))$. Since v is an integer valuation, we further have $v \in \mathsf{IntVects}(\mathsf{IH}(Z_1) \cap \mathsf{IH}(Z_2)$ and thus $\mathsf{IntVects}(Z_1 \cap Z_2) \subseteq \mathsf{IntVects}(\mathsf{IH}(Z_1) \cap \mathsf{IH}(Z_2))$ and we conclude by remarking that Conv is non-decreasing and can therefore be applied on both sides while preserving the inclusion. ∎

C. Computing the Winning States with Integer Parameters

Consider now the following fixed-point computation: $IW_0 = \emptyset$ and

$$IW_{n+1} = IS^* \cap (\mathsf{IPred_t}(\mathsf{cIPred}(IW_n), \mathsf{uIPred}(IS^* \setminus IW_n)) \cup S_{goal})$$

When it terminates, we call IW^* the corresponding fixpoint.

Lemma 9: For any integer parameter valuation v, and any finite union of parametric zones D: $v(\mathsf{IH}(D)) = v(D)$.

Proof: By definition of IS, we have $\mathsf{IS}(D) \subseteq D$. Then, using Lemma 4, we obtain that $v(\mathsf{IS}(D)) \subseteq v(D)$.

Consider for now a single parametric zone Z. Since v is an integer parameter valuation, zone $v(Z)$ has integer vertices so $\mathsf{IH}(v(Z)) = v(Z)$. Let $w \in \mathsf{IntVects}(v(Z))$. Then clearly the valuation combining w and v belongs to Z and is an integer valuation. So it belongs to $\mathsf{IH}(Z)$. So $w \in v(\mathsf{IH}(Z))$ and therefore $\mathsf{IntVects}(v(Z)) \subseteq v(\mathsf{IH}(Z))$. Since Conv is non-decreasing and $v(\mathsf{IH}(Z))$ is convex, we therefore obtain that $\mathsf{IH}(v(Z)) \subseteq v(\mathsf{IH}(Z))$. And then $v(Z) \subseteq v(\mathsf{IH}(Z))$.

To conclude, consider now that $D = \bigcup_i Z_i$, each Z_i being a parametric zone. Then $\bigcup_i v(Z_i) = \bigcup_i v(\mathsf{IH}(Z_i))$ and then, using the basic properties of valuations, $v(\bigcup_i Z_i) = v(\bigcup_i \mathsf{IH}(Z_i))$, which is the expected result. ∎

Lemma 10: For any integer parameter valuation v, any location l, any finite union of parametric zones D, and any edge e: $v(\mathsf{IPred}((l, D), e)) = \mathsf{Pred}((l, v(D)), v(e))$.

Proof: Since D is a finite union of parametric zones, so is $\mathsf{Pred}((l, D), e)$ and by Lemma 4, we have $\mathsf{Pred}((l, v(D)), v(e)) = v(\mathsf{Pred}((l, D), e))$. Furthermore, v is an integer valuation, so by Lemma 9, we have $v(\mathsf{Pred}((l, D), e)) = v(\mathsf{IS}(\mathsf{Pred}((l, D), e)))$ and finally $\mathsf{Pred}((l, v(D)), v(e)) = v(\mathsf{IS}(\mathsf{Pred}((l, D), e)))$. ∎

Lemma 11: For all integer parameter valuation v and all $n \geq 0$, $v(IW_n) = v(W_n)$.

Proof: Let v be an integer valuation. We work by induction on n The case for $n = 0$ is trivial. Now consider some $n \geq 0$ and suppose the property holds:

Recall that

$$IW_{n+1} = IS^* \cap (\mathsf{IPred_t}(\mathsf{cIPred}(IW_n), \mathsf{uIPred}(IS^* \setminus IW_n)) \cup S_{goal})$$

Then, by Lemma 8,

$$IW_{n+1} = IS^* \cap (\mathsf{IS}(\mathsf{Pred_t}(\mathsf{cPred}(IW_n), \mathsf{uPred}(IS^* \setminus IW_n))) \cup S_{goal})$$

Using now Lemma 9, Lemma 4, and Lemma 10,

$$v(IW_{n+1}) = v(IS^*) \cap (\mathsf{Pred_t}(\mathsf{cPred}(v(IW_n)), \mathsf{uPred}(v(IS^*) \setminus v(IW_n))) \cup v(S_{goal}))$$

Using the analogue of Lemma 10 for Succ [4, Lemma 4], since v is an integer valuation, we have $v(\mathsf{ISucc}((l, D), e)) = \mathsf{Succ}((l, v(D)), v(e))$. Through a straightforward induction, this implies that since v is an integer valuation, $v(IS^*) = v(S^*)$.

Putting it all together with the induction hypothesis, this means that $v(IW_{n+1}) = v(W_{n+1})$. ∎

Lemma 12: For a PGA $\mathcal{G}$, location l_{goal}, and a state (l, v) such that v is an integer valuation, it holds that $\forall i$, there exists a winning strategy in at most i controllable steps from $(l, v_{|X})$ in $v_{|P}(\mathcal{G})$ iff $(l, v) \in IW_i$.

Proof: Direct from Lemma 11. ∎

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

Theorem 13 (Termination): For any PGA $\mathcal{G}$ and any desired location l_{goal}, the algorithm terminates.

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

Theorem 14 (Correctness and completeness): Let $\mathsf{Start}(\mathcal{G}) = \{v \in \mathbb{R}^{X \cup P} \mid v_{|X} = \mathbf{0}_X\}$. Let v be an integer parameter valuation. For any PGA $\mathcal{G}$ and any location l_{goal}, upon termination, there exists a winning strategy for the controller in $v(\mathcal{G})$ iff $v \in (IW^* \cap \mathsf{Start}(\mathcal{G}))_{|P}$.

Proof:

We start by proving the right to left implication. Suppose $v \in (IW^* \cap \mathsf{Start}(\mathcal{G}))_{|P}$. Then there exists a state (l_0, v_0) in $IW^* \cap \mathsf{Start}(\mathcal{G})$ such that $v_{0|P} = v$ and $v_{0|X}$ has all coordinates equal to 0, therefore v_0 is an integer valuation on $X \cup P$ and since it belongs IW^*, it belongs to IW^n for some n. We can then apply Lemma 5 to conclude.

Now, we prove the left to right implication. If there exists a winning strategy for the controller to win in $v(\mathcal{G})$ then it means that it can win within a finite number of controllable steps. Then, by Lemma 5, it means that the state (l_0, v_0) such

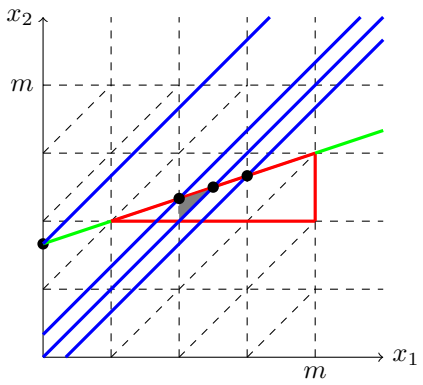

Fig. 2. Alur & Dill's regions (dashed) and the construction of parametric regions

that $v_{0|P} = v$ and $v_{0|X}$ has all coordinates equal to 0 belongs to IW^n for some n, and therefore to IW^*, which concludes the proof. ■

D. Avoiding Integer Hulls in the Backward Computation

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

Consider the fixed-point computation corresponding to this setting: $IW'_0 = \emptyset$ and

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

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

Theorem 15 (Correctness and completeness): For any PGA $\mathcal{G}$, a desired location l_{goal}, and an integer parameter valuation v, upon the termination, there exists a winning strategy for the controller in $v(\mathcal{G})$ iff $v \in (IW'^* \cap \mathsf{Start}(\mathcal{G}))_{|P}$.

Proof: First remark that, for all n, we have $IW_n \subseteq IW'_n \subseteq W_n$, because integer hulls and shapes only remove points. Let v be an integer parameter valuation. Then, we have $v(IW_n) \subseteq v(IW'_n) \subseteq v(W_n)$. By Lemma 11, $v(IW_n) = v(W_n)$ so all three sets are actually equal and in particular $v(IW'_n) = v(W_n)$. The theorem then follows in the same way as Theorem 14. ■

Proving the termination is much trickier: we construct new objects, that we call *parametric regions*, that refines the standard regions for timed automata of [1] with parametric constraints. We therefore further divide the regions with all the guards from the model and all the constraints defining integer hulls (of the symbolic states) obtained in the forward computation. We prove that these parametric regions are in finite number, that the symbolic states are convex unions of such regions and that the backwards computation preserves such unions. From this we can conclude that the backwards fix-point computation eventually terminates.

Before going into the details of this refinement, let us first briefly recall the main ideas between Alur & Dill's region partition for timed automata.

The aim is to define an equivalence relation on clock valuations such that if two valuations are equivalent they go to equivalent valuations, both by taking some edge, or by letting some time elapse. Regions are then the equivalence classes of that relation. Ensuring that there is a finite number of those regions relies on the fact that if the value of some clock x is above the maximal constant m appearing in all the clock constraints of the automaton (guards and invariants), then the actual value does not matter, only the fact that it is above m: a constraint $x \leq c$ (with by definition $c \leq m$) is false and a constraint $x \geq c$ is true, for any such value.

The dashed part of Fig. 2 outlines the region partition of the clock space in the particular case of two clocks x_1 and x_2. Each region is either a single point with integer coordinates, an open segment strictly between two such points (vertical, horizontal or diagonal), an open triangle, strictly between these segments or points, or, finally, beyond the maximal constant m appearing in the constraints of the system, open and infinite rectangles. We see that for a given value of m there are a finite number of regions.

For timed automata, when clocks are bounded, or by applying an extrapolation operator (see e.g. [16]), the valuation part of symbolic states as computed by (the specialization of) the Succ operator, are convex unions of regions (i.e. zones) for a suitable finite value of m, which ensures that they are in finite number.

In the parametric case, this is more complex because these symbolic states constrain both clocks and parameters. To ensure that they are also in finite number, using the same idea as in [4, Section IV.D.1], we can derive a bound on clocks from the bound on parameters. We could also alternatively use the bound on parameters for an extrapolation operator [17]. Thus we again have a finite number of regions. But the polyhedra defining the state-space are not convex unions of those regions: we therefore also need to refine the regions abstraction by partitioning along all the constraints encountered in the guards and invariants of the PGA, and in the integer hulls of the symbolic states computed in the forward exploration.

Furthermore, such a constraint may create a non-integer point when intersecting the lines $x = k$, for some clock x and $k \in \mathbb{N}$. For each such point, we also further partition along all lines that go though that point and are parallel to the diagonal constraints of the region abstraction (added constraints are of therefore of the form $x_i - x_j = k$ for some clocks x_i, x_j and $k \in \mathbb{Q}$).

Note that if a constraint in the integer hull of some symbolic state intersects a diagonal line defined by $x_i - x_j = k$, for some clocks x_i, x_j and some $k \in \mathbb{Z}$, creating a non-integer point, then the partition already accounts for diagonals going through this point, due to the standard region construction. We now give a formal definition of parametric regions.

Definition 9 (Parametric regions): Let m be the maximal value of parametric expressions occurring in the constraints of the PGA (recall that parameters are bounded so it is possible

to compute that value). Parametric regions are constructed in the following way:

1) the variable-space $\mathbb{R}^{X\cup P}$ is partitioned along the constraints $x \sim k$ and $x - y \sim k$ for all clocks $x, y \in X$, $\sim\in \{<, =, >\}$ and $0 \leq k \leq m$ (this gives standard regions);
2) for any guard or invariant of the automaton and any constraint appearing in the (finite number of) polyhedra defining IS^*, $\gamma \sim 0$, we further partition the variable-space, with the constraints $\gamma \sim' 0$ for every $\sim'\in \{<, =, >\}$;
3) for any (non-integer) point that is the intersection of constraints using an $=$ operator and added in the first two steps, we further refine the variable-space by constraints of the form $x - y + k \sim 0$, with $\sim\in \{<, =, >\}$ for all clocks $x \neq y$ and $k \in \mathbb{Q}$, going through that point.

Fig. 2b shows a two-dimensional example of how the variable-space is partitioned into parametric regions. The dimensions related to parameters are not shown to keep the figure readable. The integer hull of some symbolic state obtained during the forward exploration is drawn in red. Its side that does not overlap with the region graph is extended (in green) as long as it cuts the region graph. Additionally, each non-integer point obtained in the intersection of the constraints defining the integer hull and the region graph, has a diagonal constraint that goes through it (in blue). An example of a parametric region is given in gray.

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

Lemma 16: If $(a_i)_i$ and $(b_j)_j$ are finite families of parametric regions then the following sets are a finite union of parametric regions:

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

Proof: The first three are straightforward using the fact that parametric regions are taken from a finite set.

Given one parametric region, the "past" operator consists in (i) removing the constraints that impose lower bounds on clocks, (ii) add constraints that ensure that clocks stay non-negative: $\forall i, x_i \geq 0$, and (iii) add constraints that ensure that all clocks evolve at the same speed: $\forall i, j, x_i - x_j$ stays in the same fixed interval as in the regions of which we take the past. These constraints we remove or add are all used to partition the variable-space into parametric regions. So, removing or adding them lead to a union of parametric regions. For several parametric regions, the same results holds by finite union.

The fifth is immediate because the constraints of guards and invariants are used to define the parametric regions.

For $\mathsf{Pred_t}$ we need to use once again the two results from [6]: $\mathsf{Pred_t}(\bigcup_i a_i, \bigcup_j b_j) = \bigcup_i \bigcap_j \mathsf{Pred_t}(a_i, b_j)$ and $\mathsf{Pred_t}(a_i, b_j) = (a_i^{\swarrow} \setminus b_j^{\swarrow}) \cup ((a_i \cap b_j^{\swarrow}) \setminus b_j)^{\swarrow}$ (if b_j is convex, which is true by definition of the parametric region), which can equivalently be written as: $\mathsf{Pred_t}(a_i, b_j) = (a_i^{\swarrow} \cap \overline{b_j^{\swarrow}}) \cup (a_i \cap b_j^{\swarrow} \cap \overline{b_j})^{\swarrow}$. We can then conclude by using the first four results. ∎

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

Theorem 17 (Termination): For any PGA $\mathcal{G}$ and any desired location l_{goal} the fixed-point computation of IW'^* terminates.

Proof: Again, we know, from [4], that the forward computation of IS^*, using ISucc and for bounded parameters, terminates. By definition of parametric regions, IS^* can be written as a finite union of symbolic states whose associated valuations can be represented as finite unions of parametric region. When going backwards using $\mathsf{Pred_t}$, by Lemma 16 we know that parametric regions are preserved. Therefore at each step at least one region is added to the set of winning states (otherwise the fixed-point is reached and we can terminate). Since there is a finite number of parametric regions, the computation must terminate. ∎

E. Complexity

As remarked in [4], all of the possible valuations of parameters, that are integer and bounded, can be enumerated in exponential time. Therefore, for a problem that is EXPTIME for TGA, the corresponding bounded integer version for PGA is also EXPTIME. The timed control problem is EXPTIME-complete for TGA [18], and it is a special case of the parametric control problem for PGA. We can thus conclude that the parametric control problem for PGA is EXPTIME-complete for bounded integer parameters.

Theorem 18: The parametric control problem for PGA with bounded integer parameters is EXPTIME-complete.

V. Illustrative Example

We consider the same example as in [6], but we extend and parameterize the model in order to obtain the Parametric Game Automata of Figure 3. The model has two clocks x and y, controllable (c_i) and uncontrollable (u_i) actions and two parameters a and b. The reachability game consists in finding a strategy for the controller that will eventually end up in location $Goal$. We now explain how the algorithm works.

A. Computation of IS^*

We note Z a zone computed with the bounded integer parameter assumption and $\mathcal{Z}$ otherwise.

Without the bounded integer parameter assumption, the initial symbolic state is: $(l_0, \mathcal{Z}_0)$ with $\mathcal{Z}_0 = \big(x = y \wedge x \geq 0 \wedge y \geq 0\big)$. After n loops u_0, we have : $(l_0, \mathcal{Z}_n)$ with $\mathcal{Z}_n = \big(x \geq a \wedge a \leq b + 1 \wedge 0 \leq na \leq y - x \leq n(b + 1)\big)$. At this step, without the bounded integer parameter assumption, the symbolic state graph (see Figure 4) is not finite and the algorithm does not terminate neither for integer parameters nor for bounded parameters.

We now ensure the termination in the integer *and* bounded case. For example, assume all parameters and clocks are less

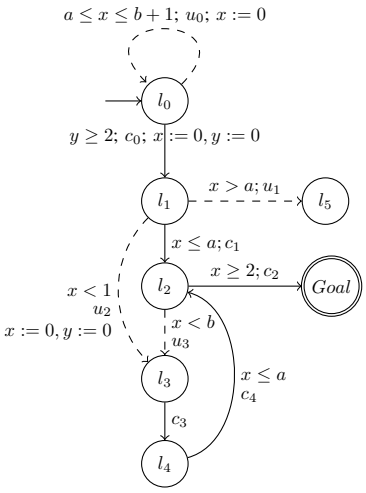

Fig. 3. A Parametric Timed Game Automaton

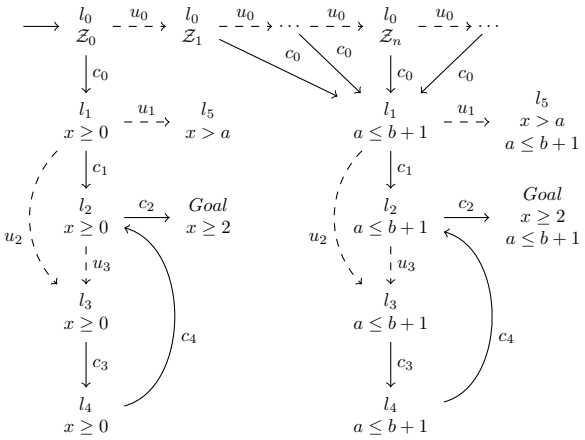

Fig. 4. Infinite Symbolic state graph

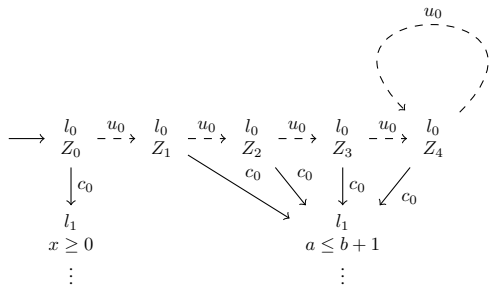

Fig. 5. Symbolic state graph with $a \le 3 \wedge b \le 3$

than or equal to 3 (i.e. in each symbolic state we implicitly have $(x \le 3 \wedge y \le 3 \wedge a \le 3 \wedge b \le 3)$ then:

- After one loop: $Z_1 = \big(x \ge a \wedge a \le 3 \wedge a \le b+1 \wedge a \le y - x \le b+1\big)$
- After two loops: $Z_2 = \big(x \ge a \wedge a \le 1 \wedge a \le b+1 \wedge 2a \le y - x \le 2(b+1)\big)$
- After three loops: $Z_3 = \big(x \ge a \wedge a \le 1 \wedge a \le b+1 \wedge 3a \le y - x \le 3(b+1)\big)$
- After four loops: $Z_4 = \big(x \ge a \wedge a = 0 \wedge a \le b+1 \wedge y - x \le 3\big)$
- $\forall n > 4$, after n loops: $Z_n = Z_4$

The symbolic state graph representing IS^* is given in Figure 5. Only the top part of the figure changes compared to Figure 4, so we have removed the bottom part to make it more readable. After transition c_0, the resets of clocks x and y remove the diagonal constraint involving $y - x$ then from location l_1 on, we have $x = y$ since x and y are, from this location on, always reset simultaneously then, for the sake of readability, we omit clock y in the symbolic states associated with locations l_1 and its successors. Moreover, all the symbolic states obtained by transition c_0 from $(l_0, Z_1) \dots (l_0, Z_4)$ are included in $(l_1, x \ge 0 \wedge a \le b+1)$. We merge them in the symbolic state graph of Figure 5, again for the sake of readability. This is by the way a classical optimisation for timed games, discussed in [6, Section 5.2], and easily extended to the parametric case.

B. Backward algorithm

After the computation of the symbolic states, shown in Figure 5, the backward algorithm starts from the symbolic winning subsets $(Goal, x \ge 2)$ and $(Goal, x \ge 2 \wedge a \le b+1)$. To show more precisely from where the constraints come, we actually describe the back-propagation following the different paths instead of computing explicitly the sets IW'_n (in [6], it is shown that this leads to the same result). From $(Goal, x \ge 2 \wedge a \le b+1)$, by the controllable action (c_2) predecessor, we obtain $(l_2, x \ge 2 \wedge a \le b+1)$. Computing the (timed) past removes the constraint $x \ge 2$, and computing the safe-timed predecessor adds $x \ge b$ in order not to end-up in l_3 by u_3. The resulting state is $(l_2, x \ge b \wedge a \le b+1)$. The controllable action predecessor for c_4 adds the constraint $x \le a$. The constraint on the parameters derived in this state is $a \ge b$. This constraint is back-propagated to the predecessor states. The safe-timed predecessors give us the state $(l_4, x \ge 0 \wedge a \ge b \wedge a \le b+1)$.

We obtain successively the following sets of winning states:

- $(l_3, x \ge 0 \wedge a \ge b \wedge a \le b+1)$,
- $(l_2, \big((x \ge b) \vee (x \ge 0 \wedge a \ge b)\big) \wedge a \le b+1)$,
- $(l_1, (x \le a) \wedge \big((x < 1 \wedge a \ge b) \vee x \ge 1\big) \wedge \big((x \ge b) \vee (x \ge 0 \wedge a \ge b)\big) \wedge a \le b+1)$.

The last one simplifies to $(l_1, x \le a \wedge a \ge b \wedge a \le b+1)$.

From the symbolic winning subset $(Goal, x \ge 2)$, we obtain the same result without the constraint $a \le b+1$.

The constraints are now back-propagated to the states associated with l_0. The constraint $a > 0$ is added in order not to end-up in the symbolic state (l_0, Z_4) by an infinite loop u_0 in null time for symbolic state (l_0, Z_4) which is back-propagated to $(l_0, Z_3) \dots (l_0, Z_0)$. Moreover the back-propagation from $(Goal, x \ge 2)$ to (l_0, Z_0) leads to the constraint $a > b+1$. We finally obtain $(l_0, (a \ge b) \wedge \Big((a > b+1) \vee \big((a \le b+1) \wedge (a > 0)\big)\Big))$. Thus, there exists a winning strategy if and only if $(a > b+1) \vee \big((a > 0) \wedge (b \le a \le b+1)\big)$.

C. Extracting a Winning Strategy

We first recall a classical result (straightforwardly extended to the parametric setting):

Theorem 19 ([5]): 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 memoryless winning strategy in $v(\mathcal{G})$.

With this theorem, and following [5], it is easy to extract a memory-less winning strategy from the set of winning states. We proceed as follows: a controllable action predecessor gives us the state from which a corresponding controllable action should be taken, while safe-timed predecessor further gives us the state where we should delay.

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 function of the parameters.

Let us now illustrate 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 $(Goal, x \geq 2)$ 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 gives two symbolic states:

- $(l_1, x \geq 0 \wedge x \leq a)$. From that state action c_1 should be taken immediately. The controllable action predecessor gives $(l_0, y \geq 2)$ from which action c_0 should be taken immediately. Timed predecessors give the symbolic state $(l_0, y < 2)$ in which we should delay until $y \geq 2$.
- $(l_4, x \geq b \wedge x \leq a)$, 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 $x \geq b$. The controllable action predecessor is $(l_3, x \geq 0)$ from which action c_3 should be taken immediately

We can do the same reasoning by starting from $(Goal, x \geq 2, a \leq b+1)$. Notice that there may be several winning strategies. Algorithmically speaking, the order of exploration of the winning states leads to different winning strategies for different values of parameters.

Instantiating the parameters: Recall that for this example, the parameters are bounded by $(a \leq 3)$ and $(b \leq 3)$ and there exists a winning strategy if and only if $(a > b+1) \vee \big((a > 0) \wedge (b \leq a \leq b+1)\big)$. The possible values for (a, b) are then $(2,0), (3,0), (3,1), (1,1), (2,1), (2,2), (3,2), (3,3)$. We choose valuation $a = 2$ and $b = 1$ which satisfies the constraints. A winning strategy corresponding to this parameter valuation, and extracted from the winning states set, is:

In all states:	Do:
$(l_0, y < 2)$	delay
$(l_0, y \geq 2)$	c_0
$(l_1, x \leq 2)$	c_1
$(l_2, x < 2)$	delay
$(l_2, x \geq 2)$	c_2
$(l_3, x \geq 0)$	c_3
$(l_4, x < 1)$	delay
$(l_4, x \geq 1 \wedge x \leq 2)$	c_4

Implementing the strategy: A systematic implementation of real-time models has been studied in [19]. Practically speaking, implementing such a strategy can be done on different targets such as field programmable gate arrays (FPGA) [20] or microcontrollers [21]. For more complex strategies and in particular for distributed systems, the implementation of distributed timed automata specifications is proposed in [22], allowing to guarantee that the specifications are preserved by the implementation.

VI. Implementation and Case Study

We have implemented the computation of the winning states and the synthesis of the strategy in our tool ROMÉO[1] [23]. With its textual input language, ROMÉO handles a model called Clock Transition Systems (CTS) which encompasses both timed automata and Time Petri Nets. We have extended CTS with controllable and uncontrollable actions in order to model parametric timed games. In this section, we show how the parametric reachability control problem can be used in practice for the synthesis of an offline scheduler.

A. The model

We consider a scheduling problem proposed in [24] and adapted for a periodic non-preemptive and parametric setting. Consider three real-time tasks τ_1, τ_2 and τ_3. Task τ_1 is periodic with period a and has an execution time $C_1 \in [10, b]$ where a and b are parameters. Tasks τ_2 and τ_3 are periodic with periods respectively $2a$ and $3a$, and execution time $C_2 \in [18, 28]$ and $C_3 \in [20, 28]$. These three tasks are scheduled using a non-preemptive[2] policy defined by the controller we want to synthesise.

We model this problem with a network of 4 parametric timed automata given in Figs 6.a, 6.b, 6.c and 6.d. The automaton given in Fig. 6.e will be used later instead of the one from Fig. 6.d. The automata of the network interact with each other by a classical synchronized product *à la Arnold Nivat* where a transition with label $e!$ (respectively $e?$) must be executed simultaneously with one and only one other transition with label $e?$ (respectively $e!$). The synchronization of actions $start_i$ (in blue on the figure Fig. 6) is done as soon as possible (this is an *urgent channel* in the tool UPPAAL [7]).

We assume invariants make it possible to win by taking an uncontrollable action forced by the urgency of the invariant. These "forced" uncontrollable actions can be easily simulated in our framework by adding, for each uncontrollable action from l to l' with a time interval $[\alpha, \beta]$, a controllable action from l to l' with the point time interval $[\beta, \beta]$. The models of the tasks and their activations are given in Figures 6.a, 6.b and 6.c. For example, for task τ_1 in Figure 6.a, as soon as clock x_1 reaches value a, synchronization $start_1$ can be performed if the automaton of the task is in state $task_1$. But if x_1 reaches again value a and exceeds it (meaning that the task τ_1 overruns its deadline) synchronization $start_1$ will definitively be impossible. After the synchronization, the automaton for the

[1] Available at http://romeo.rts-software.org

[2] A running task cannot be interrupted.

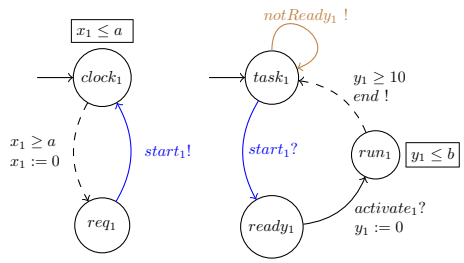

Fig 6.a The periodic request and the model of the task τ_1

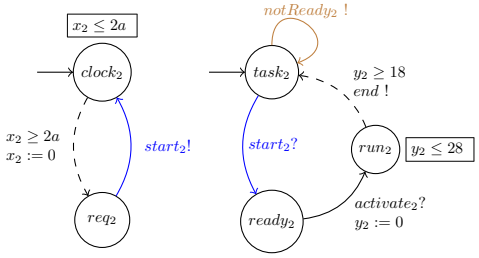

Fig 6.b The periodic request and the model of the task τ_2

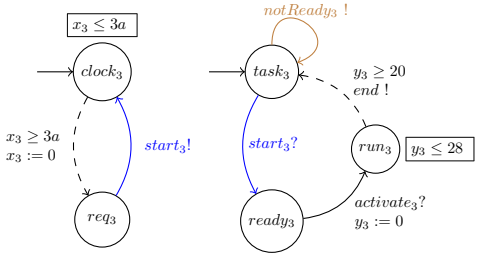

Fig 6.c The periodic request and the model of the task τ_3

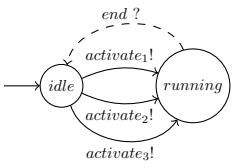

Fig 6.d Non-preemptive scheduling

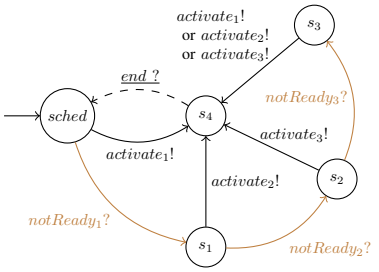

Fig 6.e Non-preemptive scheduling with Rate Monotonic

Fig. 6. A PTA with 6 clocks and 2 parameters for non-preemptive scheduling modeling (actions $start_i$ and $notReady_i$ are urgent)

activation is in location $clock_1$ and the automaton of task τ_1 is in location $ready_1$. The non-preemptive scheduler is given in Figure 6.d.

Moreover, we give in Figure 6.e, the model of the non-preemptive scheduler (to be used instead of the model given in Figure 6.d) with the classical "rate monotonic" priority assignment policy, i.e. static priorities are assigned according to the period of the task, so a shorter period results in a higher task priority. With the scheduler of Figure 6.e, actions $notReady_i$ and $Activate_i$ are urgent. A fixed priority assignment policy is referred to as optimal if it can schedule all tasks sets that are schedulable using a different priority assignment. Rate monotonic priority assignment is not optimal for fixed priority non-preemptive scheduling.

B. Our results

We first use the scheduler of Figure 6.d. where the controller can choose to execute any ready task. Using ROMÉO, and given that all parameters should be non-negative integers, we obtain that there exists a winning strategy (and then an offline scheduler) iff $a - b \geq 28$ and $b \geq 10$. ROMÉO can extract one strategy among the winning strategies corresponding to *infinitely repeating the following sequence of task executions*: $\tau_2, \tau_1, \tau_3, \tau_1, \tau_2, \tau_1, \tau_3, \tau_1, \tau_2, \tau_1, \tau_1$

We now change the model of the scheduler with the rate monotonic priority assignment defined by $priority(\tau_1) > priority(\tau_2) > priority(\tau_3)$ and modelled in Figure 6.e . It leads to the same constraints $a - b \geq 28$ and $b \geq 10$ but the winning strategy is now unique and corresponds to the following repeated sequence: $\tau_1, \tau_2, \tau_1, \tau_3, \tau_1, \tau_2, \tau_1, \tau_3, \tau_1, \tau_2, \tau_1$.

This result is very interesting in practice since it shows that, for this case study, the non-preemptive priority policy is actually optimal for the task sets that correspond to all the possible integer parameter values.

C. Comparison with an enumeration of the parameter values

To the best of our knowledge, the only other approach from the state-of-the-art able to solve a problem similar to ours, is to explicitly enumerate all the possible parameter valuations (assuming there are only a finite number of them), and to solve all the resulting timed games.

We therefore compare now the overall usefulness of our approach to this explicit enumeration scheme. We use Roméo also to solve the timed game obtained for each parameter valuation.

We consider the schedulability problem of figure 6 and we choose values of parameters leading to both true or false results for the controllability of this schedulability problem. The results are given in table I.

In practice, our approach behaves well with respect to the scaling of the bounds on parameters. For an explicit enumeration, we can easily see in table I this is not the case. We use a machine with an Intel Core i7-6500U at 2.5 GHz and 16 Gb RAM. TO means that the computation did not finish within 60 min. We can see that our approach scales much better than the explicit enumeration. Though we show only the result for 1000, on this model, our approach is actually insensitive to the bound on parameters, when it is above 100.

Note that our implementation is not optimized for instantiated parameters. Hence, given a specialized implementation and a not too big set for the possible parameter values, an explicit enumeration might be more efficient. In the general case however, we believe that our approach is more flexible and efficient.

Moreover, our symbolic computation may terminate even if the parameters are not bounded, while explicit enumeration is impossible in this case. This situation occurs in the present case-study.

Finally, our approach directly gives a symbolic constraint, which is, in our opinion, more useful than the individual values satisfying the property.

Parameter range		$a = 10$ $b = 20$	$a = 50$ $b = 40$	$a = 50$ $b = 20$	$a \in [20, 50]$ $b \in [10, 20]$	$a \in [0, 100]$ $b \in [0, 100]$	$a \in [0, 1000]$ $b \in [0, 1000]$
Controllability result		$false$	$false$	$true$	$a \in [38, 50]$ $b \in [10, 20]$ $a - b \geq 28$	$a \in [38, 100]$ $b \in [10, 72]$ $a - b \geq 28$	$a \in [38, 1000]$ $b \in [10, 972]$ $a - b \geq 28$
Symbolic	Computation time	0.05 s	0.4 s	0.13 s	4.9 s	10.5 s	10.5 s
	Computed states	38	899	139	2924	5275	5275
Enumeration	Computation time	0.05 s	0.4 s	0.13 s	51.6 s	1608 s	TO
	Computed states	38	899	139	99480	$3.13\ 10^6$	TO

TABLE I
COMPARISON WITH AN EXPLICIT ENUMERATION OF PARAMETER VALUES.

VII. CONCLUSION

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.

Since in classical timed game automata, real-valued clocks are always compared to integers for all practical purposes, we solved undecidability and termination issues by computing parameters as bounded integers.

We have proposed a further extension of a well-known fixed-point backward algorithm for solving timed games of [5], first extended for parametric timed games in [10].

The method is symbolic and avoids the explicit enumeration of all possible parameter valuations. It is based on the computation of the integer hulls of the parametric symbolic states. Due to the boundedness of integer parameters, termination is ensured, and the resulting set of parameter valuations is given as symbolic constraints between parameters. We have proved the correctness and completeness of the algorithm and we have proved that the integer hull operator can be avoided when propagating the winning states backwards.

The algorithm is implemented in our tool ROMEO and we have demonstrated its use on a small scheduling case-study.

In future work, we plan to extend this work to other timed models, such as PTA with stopwatches, and look for less restrictive domains of values for parameters.

ACKNOWLEDGMENT

This work is partially supported by the ANR national research program "PACS" (ANR-14-CE28-0002).

REFERENCES

[1] R. Alur and D. Dill, "A Theory of Timed Automata," *Theoretical Computer Science*, vol. 126, no. 2, pp. 183–235, 1994.

[2] R. Alur, T. A. Henzinger, and M. Y. Vardi, "Parametric real-time reasoning," in *ACM Symposium on Theory of Computing*, 1993, pp. 592–601.

[3] T. Hune, J. Romijn, M. Stoelinga, and F. Vaandrager, "Linear parametric model checking of timed automata," *Journal of Logic and Algebraic Programming*, vol. 52-53, pp. 183–220, 2002.

[4] A. Jovanović, D. Lime, and O. H. Roux, "Integer parameter synthesis for real-time systems," *IEEE Transactions on Software Engineering (TSE)*, vol. 41, no. 5, pp. 445–461, 2015.

[5] O. Maler, A. Pnueli, and J. Sifakis, "On the synthesis of discrete controllers for timed systems," in *STACS'95*, ser. LNCS, vol. 900. Springer, 1995, pp. 229–242.

[6] F. Cassez, A. David, E. Fleury, K. Larsen, and D. Lime, "Efficient on-the-fly algorithms for the analysis of timed games," in *CONCUR'05*, ser. LNCS, vol. 3653, 2005.

[7] G. Behrmann, A. Cougnard, A. David, E. Fleury, K. G. Larsen, and D. Lime, "Uppaal-tiga: Time for playing games!" in *CAV*, 2007, pp. 121–125.

[8] F. Cassez, J. J. Jessen, K. G. Larsen, J.-F. Raskin, and P.-A. Reynier, "Automatic synthesis of robust and optimal controllers - an industrial case study," in *HSCC*, 2009, pp. 90–104.

[9] J. J. Jessen, J. I. Rasmussen, K. G. Larsen, and A. David, "Guided controller synthesis for climate controller using uppaal tiga," in *FORMATS'07, Salzburg, Austria*, ser. LNCS, vol. 4763. Springer, 2007, pp. 227–240.

[10] A. Jovanović, D. Lime, and O. H. Roux, "A Game Approach to the Parametric Control of Real Time Systems," *International Journal of Control*, 2018.

[11] ——, "Synthesis of bounded integer parameters for parametric timed reachability games," in *ATVA 2013*, ser. LNCS, vol. 8172. Hanoi, Vietnam: Springer, Oct. 2013, pp. 87–101.

[12] T. A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine, "Symbolic model checking for real-time systems," *Information and Computation*, vol. 111, no. 2, pp. 193–244, 1994.

[13] E. Asarin, O. Maler, A. Pnueli, and J. Sifakis, "Controller synthesis for timed automata," in *Proc. IFAC Symposium on System Structure and Control*. Elsevier, 1998.

[14] L. d. Alfaro, T. A. Henzinger, and R. Majumdar, "Symbolic algorithms for infinite-state games," in *CONCUR'01*, ser. LNCS, London, UK, 2001, pp. 536–550.

[15] K. G. Larsen, P. Pettersson, and W. Yi, "Model-Checking for Real-Time Systems," in *Fundamentals of Computation Theory*, ser. LNCS, vol. 965, 1995, pp. 62–88.

[16] G. Behrmann, P. Bouyer, K. G. Larsen, and R. Pelánek, "Lower and upper bounds in zone-based abstractions of timed automata," *International Journal on Software Tools for Technology Transfer*, vol. 8, no. 3, pp. 204–215, Jun. 2006.

[17] E. André, D. Lime, and O. H. Roux, "Integer-complete synthesis for bounded parametric timed automata," in *The 9th International Workshop on Reachability Problems*, ser. Lecture Notes in Computer Science, vol. 9328. Warsaw, Poland: Springer, Sep. 2015, pp. 7–19.

[18] M. Jurdzinski and A. Trivedi, "Reachability-time games on timed automata," in *ICALP'07*, ser. LNCS, vol. 4596. Springer, Jul. 2007, pp. 838–849.

[19] M. D. Wulf, L. Doyen, and J. Raskin, "Systematic implementation of real-time models," in *FM 2005: Formal Methods, International Symposium of Formal Methods, Newcastle, UK, 2005*, 2005, pp. 139–156.

[20] S. T. Fleming and D. B. Thomas, "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*, 2013, pp. 1–2.

[21] V. Bandur, W. Kahl, and A. Wassyng, "Microcontroller assembly synthesis from timed automaton task specifications," in *Formal Methods for Industrial Critical Systems - 17th International Workshop, FMICS 2012, Paris, France, 2012.*, 2012, pp. 63–77.

[22] R. R. Devillers, J. Didier, and H. Klaudel, "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*, 2013, pp. 226–235.

[23] D. Lime, O. H. Roux, C. Seidner, and L.-M. Traonouez, "Romeo: A parametric model-checker for Petri nets with stopwatches," in *TACAS 2009*, ser. LNCS, vol. 5505. York, UK: Springer, Mar. 2009, pp. 54–57.

[24] G. Bucci, A. Fedeli, L. Sassoli, and E. Vicario, "Time state space analysis of real-time preemptive systems," *IEEE Trans. on Soft. Eng.*, vol. 30, no. 2, pp. 97–111, Feb. 2004.

Aleksandra Jovanović received her PhD degree from École Centrale de Nantes, France, as a member of Research Institute for Communications and Cybernetics of Nantes. After her PhD she became a postdoctoral research assistant at Computer Science Department, University of Oxford. After her postdoc she moved on to other projects in industry.

Didier Lime is an Associate Professor in the computer science department at École Centrale de Nantes. He is the head of the Real Time Systems group of the Laboratory of Digital Sciences of Nantes (LS2N, UMR CNRS 6004). Since 2012 he holds an "habilitation à diriger des recherches" (HDR). His main research interests concern the formal verification and control of timed systems, and their hybrid or parametric extensions.

Olivier H. Roux is full Professor at the École Centrale de Nantes (France) and he carries out his research activity at The laboratory of Digital Sciences of Nantes (LS2N, UMR CNRS 6004). His work deals with the verification and the control of timed systems. He has a particular interest in time Petri nets and timed automata as well as in their stopwatch and parametric extensions.