Parametric model-checking of time Petri nets with stopwatches using the state-class graph [⋆]

Louis-Marie Traonouez, Didier Lime, and Olivier H. Roux

Institute of Research in Communications and Cybernetics of Nantes,
1 rue de la Noë, BP 92101, 44321 Nantes Cedex 03, France
{Louis-Marie.Traonouez,Didier.Lime,Olivier-h.Roux}@irccyn.ec-nantes.fr

Abstract. In this paper, we propose a new framework for the parametric verification of time Petri nets with stopwatches controlled by inhibitor arcs. We first introduce an extension of time Petri nets with inhibitor arcs (ITPNs) with temporal parameters. Then, we define a symbolic representation of the parametric state space based on the classical state class graph method. The parameters of the model are embedded into the firing domains of the classes, that are represented by convex polyhedra. Finally, we propose semi-algorithms for the parametric model-checking of a subset of parametric TCTL formulae on ITPNs. We can thus generate the set of the parameter valuations that satisfy the formulae.

Key words: Time Petri nets, stopwatches, model-checking, parameters, state-class graph

Introduction

Formal methods are widely used in the conception of real-time systems. Methods such as model-checking allow the verification of a system by exploring the state-space of a model. A popular class of models is Petri nets and their extensions among which Time Petri nets (TPNs) [1] are a widely used time extension in which transitions can be fired within a time interval.

In order to take into account the global complexity of systems, we can use models that encompass the notion of actions that can be suspended and resumed. This implies extending traditional clock variables by "stopwatches". Several extensions of TPNs address this issue, such as Preemptive-TPNs [2] or Inhibitor Hyperarc TPNs (ITPNs) [3]. ITPNs introduce special inhibitor arcs that control the progress of transitions. These models all belong to the class of TPNs extended with stopwatches (SwPNs)[4].

The model-checking of these models has become more and more efficient. It nevertheless requires a complete knowledge of the system. Consequently, the verification of the behavior must be done after the conception when the global system and its environment are known. On the one hand, it increases the complexity of the conception and the verification of systems. For too complex systems

[⋆] Work supported by the French Government under grant ANR-SETI-06-003.

this can lead to a combinatory explosion. Besides, if the system is proven wrong or if the environment changes, this complex verification process must be carried out again. On the other hand, getting a complete knowledge of a system can be impossible. In many important applications, a system is defined by parameters that are in relation with several other systems. In the existing tools for modelling and verification, parameters are often used, however they must be instantiated to perform analyses. The next development step of the technology is to be able to directly analyze a parametric model.

Related works Parametric analysis of real-time systems has been studied in [5]. They introduce Parametric Timed Automata (PTA) and prove that, in the general case, the emptiness problem is undecidable. On this subject, in [6] the authors prove that for a particular class of PTA called L/U automata this emptiness problem is decidable. They also give a model-checking algorithm that use parametric Difference Bound Matrices. Parametric model-checking can be used to generate a set of constraints on the parameters such that the property is verified. In discrete time, parametric model-checking of PTA has been studied in [7, 8], and some decidability results have been found. On hybrid automata, state-space exploration algorithms have been extended to allow a parametric analysis and implemented in the tool HYTECH [9].

Another approach developed in [10] focuses on the verification of parametric TCTL formulae on clock automata. They consider unbounded parameters that take their value among integers. The problem is decidable and one interest of their algorithm is to detect cycles in the region graph and to express the computation time into a linear combination of the duration of cycles. In [11], this approach has been used in parametric TPNs, but with bounded parameters. However, they consider and analyze a region graph for each parameter valuation.

Our contribution In this context, we propose to study the parametric model-checking problem on time Petri nets and more generally on ITPNs. We consider unbounded parameters and thus, we need a proper abstraction of the state-space of the parametric model. In TPNs, considering that the time is dense, the state-space of the model is infinite, but it can be represented by a finite partition as in the state-class graph [12]. We therefore extend the state-class graph construction with parameters and define parametric state-classes that represent at the same time the temporal domain and the parameter domain. Although the state-class graph does not preserve timed properties, there exists methods [13] to verify a subset of TCTL with this abstraction. We consider this subset of formulae and extend it with parameters. Then, we propose and prove semi-algorithms for parametric model-checking.

Outline of the paper In section 1, we present our parametric extension of ITPNs (PITPNs). Then, in section 2, we introduce some decidability and undecidability results. Section 3 defines the parametric state-class graph of PITPNs. In section 4, we study the parametric model-checking of a subset of TCTL with parameters. Finally, in 5 we discuss our solution to the parametric model-checking problem.

1 Parametric time Petri nets with inhibitor arcs

1.1 Notations

The sets $\mathbb{N}$, $\mathbb{Q}^+$ and $\mathbb{R}^+$ are respectively the sets of natural, non-negative rational and non-negative real numbers. An interval I of $\mathbb{R}^+$ is a $\mathbb{Q}$-interval iff its left endpoint $I^\uparrow$ belongs to $\mathbb{Q}^+$ and its right endpoint $I^\downarrow$ belongs to $\mathbb{Q}^+ \cup \{\infty\}$. We denote by $\mathcal{I}(\mathbb{Q})$ the set of $\mathbb{Q}$-intervals of $\mathbb{R}^+$.

1.2 Formal definitions of PITPNs

We parameterize the ITPN model with a set of temporal parameters $Par = \{\lambda_1, \lambda_2, \dots, \lambda_l\}$ by replacing some of the temporal bounds of the transitions by parameters. These parameters are considered as constant variables in the semantics, and take their values among rationals.

Some initial constraints are given on the parameters. These constraints define the domain $D_p \subseteq {\mathbb{Q}^+}^{Par}$ of the parameters which is a convex polyhedron. These constraints must at least specify that for all parameters valuations in D_p, the minimum bounds of the firing intervals of the transitions are inferior to the maximum bounds. Additional linear constraints may of course be given.

A *valuation* of the parameters is a function $\nu : Par \rightarrow \mathbb{Q}^+$, such that $[\nu(\lambda_1)\ \nu(\lambda_2) \dots \nu(\lambda_l)]^\top \in D_p$, which is equivalent to say that ν is a point of D_p. We will also write that $\nu = [\lambda_1\ \lambda_2 \dots \lambda_l]^\top$.

A *linear constraint* over the parameters is an expression $\gamma = \sum_{i=0}^{l} a_i * \lambda_i \sim b$, where $\forall 0 \leq i \leq l,\ a_i, b \in \mathbb{R}$ and $\sim \in \{=, <, >, \leq, \geq\}$. A convex polyhedron is a conjunction of linear constraints.

A *parametric time interval* is a function $J : D_p \rightarrow \mathcal{I}(\mathbb{Q}^+)$ that associates to each parameter valuation a $\mathbb{Q}$-interval. The set of parametric time intervals over Par is denoted by $\mathcal{J}(Par)$.

Definition 1. *A parametric time Petri net with inhibitor arcs (PITPN) is a tuple $\mathcal{N} = \langle P, T, Par, {}^\bullet(.), (.)^\bullet, {}^\circ(.), M_0, J_s, D_p \rangle$, where:*

- $P = \{p_1, p_2, \dots, p_m\}$ *is a non-empty finite set of* places*,*
- $T = \{t_1, t_2, \dots, t_n\}$ *is a non-empty finite set of* transitions*,*
- $Par = \{\lambda_1, \lambda_2, \dots, \lambda_l\}$ *is a finite set of* parameters*,*
- ${}^\bullet(.) \in (\mathbb{N}^P)^T$ *is the* backward incidence function*,*
- $(.)^\bullet \in (\mathbb{N}^P)^T$ *is the* forward incidence function*,*
- ${}^\circ(.) \in (\mathbb{N}^P)^T$ *is the* inhibition function*,*
- $M_0 \in \mathbb{N}^P$ *is the* initial marking *of the net,*
- $J_s \in (\mathcal{J}(Par))^T$ *is the function that associates a* parametric firing interval *to each transition,*
- $D_p \subseteq {\mathbb{Q}^+}^{Par}$ *is a convex polyhedron that is the* domain of the parameters*.*

A *marking* M of the net is an element of $\mathbb{N}^P$ such that $\forall p \in P, M(p)$ is the number of tokens in the place p.

A transition t is said to be *enabled* by the marking M if $M \geq^{\bullet} t$, (*i.e.* if the number of tokens in M in each input place of t is greater or equal to the value on the arc between this place and the transition). We denote it by $t \in \mathsf{enabled}\,(M)$.

A transition t is said to be *inhibited* by the marking M if the place connected to one of its inhibitor arc is marked with at least as many tokens than the weight of the considered inhibitor arc between this place and t: $0 < {}^{\circ}t \leq M$. We denote it by $t \in \mathsf{inhibited}\,(M)$. Practically, inhibitor arcs are used to stop the elapsing of time for some transitions: an inhibitor arc between a place p and a transition t means that the stopwatch associated to t is stopped as long as place p is marked with enough tokens.

A transition t is said to be *active* in the marking M if it is enabled and not inhibited by M.

Example 1. In the figure 1 an example of PTPN is given that includes three parameters a, b and c.

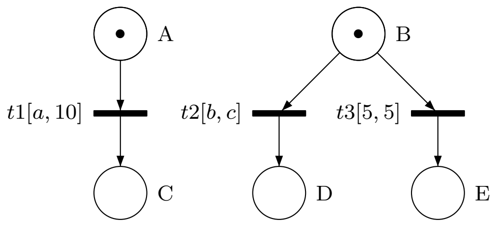

Fig. 1. Parametric time Petri nets

1.3 Semantics of parametric time Petri nets with inhibitor arcs

The semantics of a Parametric Time Petri net with Inhibitor Arcs $\mathcal{N}$ is defined for a parameter valuation $\nu \in D_p$, as the non-parametric ITPN obtained when replacing in $\mathcal{N}$ all the parameters by their valuation.

Definition 2 (Semantics of a PITPN). *Given a PITPN* $\mathcal{N} = \langle P, T, Par, {}^{\bullet}(.), (.)^{\bullet}, {}^{\circ}(.), M_0, J_s, D_p\rangle$, *and a valuation* $\nu \in D_p$, *the semantics* $[\![\mathcal{N}]\!]_{\nu} = \langle P, T, {}^{\bullet}(.), (.)^{\bullet}, {}^{\circ}(.), M_0, I_s\rangle$ *of* $\mathcal{N}$ *is a ITPN such that* $\forall t \in T,\ I_s(t) = J_s(t)(\nu)$.

We now recall the semantics of an ITPN.

A transition t is said to be *firable* when it has been enabled and not inhibited for at least $I_s(t)^{\uparrow}$ time units.

A transition t_k is said to be *newly* enabled by the firing of the transition t_i from the marking M, and we denote it by $\uparrow\mathsf{enabled}\,(t_k, M, t_i)$, if the transition is enabled by the new marking $M -^{\bullet} t_i + t_i^{\bullet}$ but was not by $M -^{\bullet} t_i$, where M is the marking of the net before the firing of t_i. Formally:

$$\begin{aligned}&\uparrow\mathsf{enabled}\,(t_k, M, t_i) = ({}^{\bullet}t_k \leq M -^{\bullet} t_i + t_i^{\bullet})\\ &\wedge((t_k = t_i) \vee ({}^{\bullet}t_k > M -^{\bullet} t_i))\end{aligned}$$

By extension, we will denote by $\uparrow\mathsf{enabled}\,(M, t_i)$ the set of transitions newly enabled by firing the transition t_i from the marking M.

Definition 3. *A* state *of a ITPN is a pair $q = (M, I)$ in which M is a marking and I is a function called the* interval *function. Function $I \in (\mathcal{I}(\mathbb{R}))^T$ associates a temporal interval with every transition enabled at M.*

The semantics of an ITPN is defined as a timed transition system (TTS) [14], in which two kinds of transitions may occur: *time* transitions when time passes and *discrete* transitions when a transition of the net is fired.

Definition 4 (Semantics of an ITPN). *The semantics of a time Petri net with inhibitor arcs $\mathcal{N} = \langle P, T, {}^\bullet(.), (.)^\bullet, {}^\circ(.), M_0, I_s \rangle$ is defined as the TTS $\mathcal{S}_\mathcal{N} = \langle Q, q_0, \rightarrow \rangle$ such that:*

- $Q = \mathbb{N}^P \times (\mathcal{I}(\mathbb{R}))^T$,
- $q_0 = (M_0, I_s)$,
- $\rightarrow \in Q \times (T \cup \mathbb{R}^+) \times Q$ *is the transition relation including a time transition relation and a discrete transition relation.*
  The time transition relation is defined $\forall d \in \mathbb{R}^+$ *by:*
  $(M, I) \xrightarrow{d} (M, I')$ *iff* $\forall t_i \in T$,
  $$\begin{cases} I'(t_i) = \begin{cases} I(t_i) \text{ if } t_i \in \mathsf{enabled}\,(M) \text{ and } t_i \in \mathsf{inhibited}\,(M) \\ I'(t_i)^\uparrow = \max(0, I(t_i)^\uparrow - d), \text{ and } I'(t_i)^\downarrow = I(t_i)^\downarrow - d \text{ otherwise,} \end{cases} \\ M \geq {}^\bullet t_i \Rightarrow I'(t_i)^\downarrow \geq 0 \end{cases}$$
  The discrete transition relation is defined $\forall t_i \in T$ *by:*
  $$(M, I) \xrightarrow{t_i} (M', I') \text{ iff } \begin{cases} t_i \in \mathsf{enabled}\,(M) \text{ and } t_i \notin \mathsf{inhibited}\,(M)\,, \\ M' = M - {}^\bullet t_i + t_i^\bullet, \\ I(t_i)^\uparrow = 0, \\ \forall t_k \in T, I'(t_k) = \begin{cases} I_s(t_k) \text{ if } \uparrow\mathsf{enabled}\,(t_k, M, t_i) \\ I(t_k) \text{ otherwise} \end{cases} \end{cases}$$

A *run* ρ of length $n \geq 0$ in $\mathcal{S}_\mathcal{N}$ is a finite or infinite sequence of alternating time and discrete transitions of the form

$$\rho = q^0 \xrightarrow{d_0} q^0 + d_0 \xrightarrow{t_0} q^1 \xrightarrow{d_1} q^1 + d_1 \xrightarrow{t_1} \cdots q^n \xrightarrow{d_n} \cdots$$

We write *first*(ρ) the first state of a run ρ. A run is *initial* if *first*$(\rho) = q_0$. A run ρ of $\mathcal{N}$ is an initial run of $\mathcal{S}_\mathcal{N}$. For a state q, the set of all the infinite runs starting from q is denoted by $\pi(q)$. The set of all the runs of $\mathcal{N}$ is $\pi(q_0)$.

For a state q^i in ρ the absolute time elapsed (relative to the initial state) is $\mathsf{time}(q) = d_0 + d_1 + \cdots + d_{i-1}$. For a run ρ the total elapsed time in ρ is $\mathsf{time}(\rho) = \sum_{i=0}^{n} d_i$. In this paper we restrict ourselves to non-zeno ITPN, which means that the elapsed time is diverging (i.e. $\forall \rho \in \pi(q_0), \mathsf{time}(\rho) = \infty$), and by extension to non-zeno PITPN (i.e. such that $\forall \nu \in D_p$, $[\![\mathcal{N}]\!]_\nu$ is non zeno).

2 Decidability of Parametric TPNs

In this section, we give some results concerning the decidability of the emptiness and reachability problems for bounded parametric time Petri nets (without inhibitor arcs). The case of PITPNs is of little interest since these problems are already known undecidable for bounded ITPNs [4].

Let us consider lower/upper bound (L/U) bounded parametric TPNs i.e. every parameter occurring in the PTPN is either the lower bound of some of the time intervals or their upper bound, but there exists no pair of intervals I_1, I_2 and no parameter λ such that $\lambda = I_1^{\uparrow}$ and $\lambda = I_2^{\downarrow}$.

Theorem 1. *The emptiness and reachability problems are decidable on bounded L/U parametric TPNs.*

Proof (Theorem 1). The structural and syntactical translation proposed in [15] from a TPN into a bisimilar timed automaton (TA) can straightforwardly be extended from L/U PTPNs to L/U parametric TA [6]. Therefore, since the emptiness and reachability problems are decidable for L/U parametric TA [6], they also are decidable for L/U PTPNs.

Theorem 2. *The emptiness and reachability problems are undecidable on bounded Parametric TPNs.*

Proof (Theorem 2). The structural and syntactical translation preserving timed language acceptance proposed in [16] from a TA into a bounded TPN can straightforwardly be extended to parametric TA. Thus, for every parametric TA, we can compute a parametric TPN that accepts the same timed language. Since the emptiness problem (and then, the reachability problem) is undecidable for parametric TA [5], it is also undecidable for parametric TPNs.

3 The Parametric State-Class Graph of a PITPN

Since the state-space of a non-parametric TPN is generally infinite in dense-time, it is required to abstract the time by merging some states into some equivalence classes. Consequently, symbolic representations of the state-space are used. One of the approaches to partition the state-space in a finite set of infinite state classes is the state-class graph [12]. This approach has been extended for ITPNs in [3].

However, there also exists an infinite number of parameter valuations. Thus, in the same way, we need to use symbolic representations of the parameter domains. In time Petri nets or timed automata, the time domain of an abstract state can be efficiently encoded by a difference bound matrix (DBM). This is why, in the parametric timed automata proposed in [6], the authors define parametric DBMs in which they encode both the time domain and the parameter domain. When considering stopwatch time Petri nets, the firing domain of a class is a general polyhedron and cannot necessarily be represented by a DBM. Consequently, in the parametric state-classes of PITPNs we will use polyhedra, which describe both the transition variable domains and the parameter domains.

3.1 Parametric state-classes

Definition 5. *A* parametric state-class *C of a PITPN is a pair (M, D) where M is a marking of the net and D is a firing domain represented by a (convex) polyhedron involving $l + n$ variables, with n being the number of transitions enabled by the marking of the class and l the number of parameters in the net.*

A point (ν, ν') of the firing domain is constituted by a valuation ν of the parameters in Par and a valuation ν' of the firing times θ of enabled transitions. The set of those variables θ of D will be noted Θ.

We denote by $D_{|Par}$ the projection of a firing domain D on the set of parameters:

$$D_{|Par} = \{\nu \in \mathbb{Q}^{+^l} \mid \exists \nu' \in \mathbb{R}^n \text{ s.t. } (\nu, \nu') \in D\}$$

This definition can be extended to any arbitrary subset of variables of D.

3.2 Computation of the parametric state-class graph

The parametric state-class graph is computed similarly to the non-parametric case. Parameters are embedded into the firing domain of the initial class, and the operations that compute the successor classes do not concern the parameters. However, throughout the computation of the graph, the domain of the parameters in a class will be automatically reduced to consider only the valuations that make this class reachable.

Definition 6 (Firability). *Let $C = (M, D)$ be a parametric state-class of a PITPN. A transition t_i is said to be* firable *from C iff there exists a solution (ν, ν') of D, such that $\forall j \in \{1, \ldots, n\} - \{i\}$, s.t. $t_j \in \mathsf{enabled}(M)$ and $t_j \notin \mathsf{inhibited}(M)$, $\nu'(\theta_i) \leq \nu'(\theta_j)$. We will write this: $t_i \in \mathsf{firable}(C)$.*

Now, given a parametric class $C = (M, D)$ and a firable transition t_f, the parametric class $C' = (M', D')$ obtained from C by the firing of t_f, which we write $C' = succ(C, t_f)$, is given by

- $M' = M - {}^\bullet t_f + t_f^\bullet$
- D' is computed along the following steps, and noted $\mathsf{next}(D, t_f)$
  1. intersection with the firability constraints : $\forall j$ s.t. t_j is active, $\theta_f \leq \theta_j$
  2. variable substitutions for all enabled transitions that are *active* t_j: $\theta_j = \theta_f + \theta'_j$,
  3. elimination (using for instance the Fourier-Motzkin method) of all variables relative to transitions disabled by the firing of t_f,
  4. addition of inequations relative to newly enabled transitions

$$\forall t_k \in \uparrow\mathsf{enabled}(M, t_f),\ J_s(t_k)^\uparrow \leq \theta'_k \leq J_s(t_k)^\downarrow$$

Case of a point : Let $C = (M, D)$ be a parametric state-class of a PITPN, $x = [\lambda_1 \ldots \lambda_l\ \theta_1 \ldots \theta_n]^\top$ be a point of D and t_f be a transition firable from $(M, \{x\})$. The successor of $\{x\}$ by the firing t_f from marking M is given by

$$\mathsf{next}(\{x\}, t_f) = \left\{ \forall i \in [1..n] \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_l \\ \theta'_1 \\ \vdots \\ \theta'_n \end{bmatrix} \left| \begin{array}{l} \theta'_i \in I_s(t_i) \text{ if } \uparrow\mathsf{enabled}(t_i, M, t_f) \\ \theta'_i = \theta_i \text{ if } t_i \in \mathsf{enabled}(M) \\ \ \text{and } t_i \in \mathsf{inhibited}(M) \\ \ \text{and not } \uparrow\mathsf{enabled}(t_i, M, t_f) \\ \theta'_i = \theta_i - \theta_f \text{ otherwise} \end{array} \right. \right\}$$

The next operator straightforwardly extends to finite or infinite unions of points.

The parametric state-class graph is generated by iteratively applying the function that computes the successors of a state-class:

Definition 7. *Given a PITPN $\mathcal{N}$, the* parametric state-class graph *of $\mathcal{N}$ is the transition system $\mathcal{G}(\mathcal{N}) = \langle \mathcal{C}, \twoheadrightarrow, C_0 \rangle$ such that:*

- $C_0 = (M_0, D_0)$ *is the initial class such that* $D_0 = D_p \wedge \{\theta_k \in J_s(t_k) \mid t_k \in \mathsf{enabled}\,(M_0)\}$
- $C \overset{t}{\twoheadrightarrow} C'$ *iff* $t \in \mathsf{firable}\,(C)$ *and* $C' = succ(C, t)$,
- $\mathcal{C} = \{C | C_0 \twoheadrightarrow^* C\}$, *where* $\twoheadrightarrow^*$ *is the reflexive and transitive closure of* $\twoheadrightarrow$.

3.3 Valuation of the parametric state-class graph

From the parametric state-class graph of a PITPN it is possible to choose a valuation of the parameters and to replace in the graph all the parameters by their value. Then, we obtain a non-parametric graph. However, some firing domains of the classes may become empty, which means that the class is not reachable for this valuation. Those classes must be removed from the non-parametric graph. The graph finally obtained corresponds to the state-class graph of the ITPN obtained for this valuation.

Definition 8 (Valuation of a parametric state-class). *Let $C = (M, D)$ be a parametric state-class of a PITPN $\mathcal{N}$ and let $\nu \in D_p$ be a valuation of the parameters of $\mathcal{N}$. The valuation of C by ν is a non-parametric class $[\![C]\!]_\nu = (M, [\![D]\!]_\nu)$ where*

$$[\![D]\!]_\nu = \{\nu' \in \mathbb{R}^n \mid (\nu, \nu') \in D\}$$

The valuation of the parametric state-class graph is obtained by valuating the classes of the graph, starting from the initial class and stopping if the firing domains become empty.

Definition 9 (Valuation of a parametric state-class graph). *Given a PITPN $\mathcal{N}$ and a valuation $\nu \in D_p$, $[\![\mathcal{G}(\mathcal{N})]\!]_\nu = (\mathcal{C}_\nu, \twoheadrightarrow, [\![C_0]\!]_\nu)$ where:*

- $[\![C_0]\!]_\nu$ *is the valuation of the initial class C_0 of $\mathcal{G}(\mathcal{N})$,*
- $[\![C]\!]_\nu \overset{t}{\twoheadrightarrow} [\![C']\!]_\nu$ *iff* $C = (M, D), C' = (M', D') \in \mathcal{G}(\mathcal{N}) C \overset{t}{\twoheadrightarrow} C'$ *and* $[\![D']\!]_\nu \neq \emptyset$
- $\mathcal{C}_\nu = \{[\![C]\!]_\nu \mid [\![C_0]\!]_\nu \twoheadrightarrow^* [\![C]\!]_\nu\}$, *where* $\twoheadrightarrow^*$ *is the reflexive and transitive closure of* $\twoheadrightarrow$.

The theorem 3 establishes that the valuation of the parametric state-class graph of a PITPN matches the non-parametric state-class graph of the ITPN obtained for the same parameter valuation. Theorem 4 allows to directly determine the accessibility condition of a parametric state-class.

Theorem 3. *Given a PITPN $\mathcal{N}$ and a valuation $\nu \in D_p$, then*

$$[\![\mathcal{G}(\mathcal{N})]\!]_\nu = \mathcal{G}([\![\mathcal{N}]\!]_\nu)$$

Theorem 4. *Given a PITPN $\mathcal{N}$ and a valuation $\nu \in D_p$, let $C = (M, D)$ be a parametric state-class in $\mathcal{G}(\mathcal{N})$. Then*

$$[\![C]\!]_\nu \in [\![\mathcal{G}(\mathcal{N})]\!]_\nu \text{ iff } \nu \in D_{|Par}$$

4 Parametric model checking

The model-checking problem consists in checking that a model $\mathcal{N}$ satisfies a property ϕ expressed in a given logic, which is more formally written $\mathcal{N} \models \phi$. The answer to this problem is either *true* or *false*. Concerning the parametric model-checking problem, given a parametric model $\mathcal{N}$ and a property ϕ, which may also be parameterized, we want to determine the set of parameter valuations $F(\mathcal{N}, \phi)$ such that $\forall \nu \in F(\mathcal{N}, \phi)$ the non-parametric model $[\![\mathcal{N}]\!]_\nu$ obtained for the valuation ν satisfies the non-parametric property $[\![\phi]\!]_\nu$ obtained for the same valuation. This set will be represented by a set of constraints on the parameters of the problem.

4.1 Parametric TCTL formulae

Like ITPN, we parameterize TCTL formulae by allowing that the bounds of the temporal intervals of the formulae are parameters. The parameters used in the formulae are added to the set of parameters of the PITPN in study. Besides, we consider only a subset of TCTL formulae for which "on-the-fly" model-checking algorithms have already been proposed for TPNs [13]. This subset is sufficient to verify many interesting problems (reachability, safety, bounded liveness...).

First, we recall the syntax and semantics of TCTL formulae in the context of TPNs (or ITPNs).

Definition 10 (TCTL for TPN). *For a TPN $\mathcal{N}$ the grammar of TCTL formulae is*

$$TCTL ::= \mathcal{P} \mid \neg\varphi \mid \varphi \Rightarrow \psi \mid \exists\varphi\mathcal{U}_I\psi \mid \forall\varphi\mathcal{U}_I\psi$$

where $\varphi, \psi \in TCTL$, $I \in \mathcal{I}(\mathbb{Q}^+)$, $\mathcal{P} \in PR$, and $PR = \{\mathcal{P} \mid \mathcal{P} : M \to \{true, false\}\}$ is the set of propositions on the marking on the net.

We use the following abbreviations $\exists\Diamond_I\varphi = \exists\mathbf{true}U_I\varphi$, $\forall\Diamond_I\varphi = \forall\mathbf{true}U_I\varphi$, $\exists\Box_I\varphi = \neg\forall\Diamond_I\neg\varphi$ and $\forall\Box_I\varphi = \neg\exists\Diamond_I\neg\varphi$.

We define the bounded time response by $\varphi \leadsto_I \psi = \forall\Box(\varphi \Rightarrow \forall\Diamond_I\psi)$.

TCTL formulae are interpreted on the states of a model $\mathcal{M} = (\mathcal{S}_\mathcal{N}, \mathcal{V})$, where $\mathcal{S}_\mathcal{N}$ is the state space of the TPN and $\mathcal{V} : \mathcal{S}_\mathcal{N} \to 2^{PR}$ is a function that evaluates the marking of a state, such that $\mathcal{V}(q) = \{\mathcal{P} \in PR \mid \mathcal{P}(M) = true\}$. Now, let $q \in \mathcal{S}_\mathcal{N}$ be a state and $\rho \in \pi(q)$ a run starting from q, such that $\rho = q^0 \xrightarrow{d_0} q^0 + d_0 \xrightarrow{t_0} q^1 \xrightarrow{d_1} q^1 + d_1 \xrightarrow{t_1} \cdots$. We define $\rho^* : \mathbb{R}^+ \to \mathcal{S}_\mathcal{N}$ by $\rho^*(r) = q^i + \delta$ if $r = \sum_{j=0}^{i-1} d_j + \delta$, with $i \geq 0$ and $0 \leq \delta < d_i$.

Definition 11 (Semantics of TCTL). *Given a TPN $\mathcal{N}$ and its model $\mathcal{M} = (\mathcal{S}_\mathcal{N}, \mathcal{V})$, the truth value of a TCTL formula for a state $q \in \mathcal{S}_\mathcal{N}$ is*

- $q \models \mathcal{P}$ *iff* $\mathcal{P} \in \mathcal{V}(q)$,
- $q \models \neg\varphi$ *iff* $q \not\models \varphi$,
- $q \models \varphi \Rightarrow \psi$ *iff* $q \not\models \varphi \ \vee q \models \psi$,
- $q \models \exists\varphi\mathcal{U}_I\psi$ *iff* $\exists\rho \in \pi(q),\ \exists r \in I$ *s.t.* $\rho^*(r) \models \psi$ *and* $\forall r' < r,\ \rho^*(r') \models \varphi$
- $q \models \forall\varphi\mathcal{U}_I\psi$ *iff* $\forall\rho \in \pi(q),\ \exists r \in I,$ *s.t.* $\rho^*(r) \models \psi$ *and* $\forall r' < r,\ \rho^*(r') \models \varphi$

Given a model $\mathcal{M} = (\mathcal{S}_{\mathcal{N}}, \mathcal{V})$, for a marking proposition $\mathcal{P} \in PR$ and a state $q = (M, I) \in \mathcal{S}_{\mathcal{N}}$, we use the notation $M \models \mathcal{P}$ if $\mathcal{P} \in \mathcal{V}(q)$ and $M \not\models \mathcal{P}$ if $\mathcal{P} \notin \mathcal{V}(q)$.

Finally, a TPN $\mathcal{N}$ satisfies a TCTL formula ϕ if and only if $q_0 \models \phi$.

We present now the syntax and semantics of Parametric TCTL (PTCTL) formulae for PITPN.

Definition 12 (PTCTL for PITPN). *For a PITPN $\mathcal{N}$ the grammar of PTCTL formulae is*

$$PTCTL ::= \exists\varphi\mathcal{U}_J\psi \mid \forall\varphi\mathcal{U}_J\psi \mid \exists\Diamond_J\varphi \mid \forall\Diamond_J\varphi \mid \exists\Box_J\varphi \mid \forall\Box_J\varphi \mid \varphi \leadsto_{J_r} \psi$$

where $\varphi, \psi \in PR$, $J, J_r \in \mathcal{J}(Par)$ are parametric time intervals, with the restriction that $J_r = [0, b]$ with $b \in \mathbb{Q}^+ \cup Par$, or $J_r = [0, \infty[$.

The semantics of PTCTL formulae are defined similarly to the semantics of PITPNs. Given a valuation, the parameters in the formulae are replaced by their value to obtain a TCTL formula, which is interpreted on the ITPN obtained for this valuation.

Definition 13 (Semantics of PTCTL). *Let $\mathcal{N}$ be a PITPN and ϕ be a PTCTL formulae and $\nu \in D_p$ be a valuation of the parameters of $\mathcal{N}$ (which are shared with ϕ). $[\![\phi]\!]_\nu$ is the TCTL formula obtained when replacing in ϕ the parametric time interval J (or J_r) by the $\mathbb{Q}$-interval $J(\nu)$ (or $J_r(\nu)$).*
Then $\mathcal{N}$ satisfy ϕ for the valuation ν if and only if $[\![\mathcal{N}]\!]_\nu \models [\![\phi]\!]_\nu$.

4.2 Extending the parametric state-class graph with a global clock

In the state-class graph, the firing domain of a class gives the firing dates of the transitions with the entrance in the class as a time origin. Timed properties are difficult to verify in this context. In order to easily check timed properties with the state class graph abstraction, it is necessary to be able to evaluate the time that has elapsed between classes. For this purpose, we propose to extend the parametric state-classes with an additional variable noted θ_c. This variable is initialized to zero in the initial class, and then decreases when time flows, like a transition variable [1] . However, the variable will not constrain the transitions variables when determining the firability constraints. Then, for all classes, the time elapsed from the initialization of θ_c to the entrance in the class is: $\tau_c = -\theta_c$.

Definition 14. *An* extended parametric state-class *C of a PITPN is a class whose firing domain D is extended with an additional variable $\theta_c \in \Theta$.*

[1] The value of this variable will always be non-positive. But this is not a problem in the computation of the state-classes. The alternative would be to initialize it, not to zero, but to a sufficiently large value, but this value is hard to determine.

The definition of the firability of an extended class is not modified. The firability constraints indeed only involve the variables θ_i where $\forall i \in \{1, \ldots, n\}$, $t_i \in T$. The next operator is redefined for an extended class such that for a point $x = [\lambda_1 \ldots \lambda_l \; \theta_1 \ldots \theta_n \; \theta_c]^\top$ of D, in $\mathsf{next}((\{x\}, t_f)$ we have $\theta'_c = \theta_c - \theta_f$.

The extended parametric state-class graph $\mathcal{G}_c(\mathcal{N})$ is then computed iteratively in a similar way, starting from the initial class $C_0 = (M_0, D_0)$ where

$$D_0 = D_p \wedge \{\theta_k \in J(t_k) \mid t_k \in \mathsf{enabled}\,(M_0)\} \wedge \{\theta_c = 0\}$$

Finally, given an extended parametric state-class $C = (M, D)$, we are able to determine:

- $\tau_{min}(C)$, the absolute minimum time elapsed when entering the class. This a function of the parameters Par of the net $\tau_{min}(C) : D_p \rightarrow \mathbb{Q}^+$, such that $\tau_{min}(C)(\nu) = \min_{x=(\nu,\nu')\in D}(\tau_c)$. It can be expressed as the maximum between the minimum values of τ_c and it is necessarily positive and finite.
- $\tau_{max}(C)$, the absolute maximum time elapsed when entering the class. This a function on the parameters Par of the net $\tau_{max}(C) : D_p \rightarrow \mathbb{Q}^+ \cup \{\infty\}$, such that $\tau_{max}(C)(\nu) = \max_{x=(\nu,\nu')\in D}(\tau_c)$. It can be expressed as the minimum between the maximum values of τ_c and it is necessarily positive but may be infinite if there is no maximum time.

If $[\![C]\!]_\nu \in [\![\mathcal{G}_c(\mathcal{N})]\!]_\nu$, let $q \in [\![C]\!]_\nu$ be a state. Then the elapsed time of the state is such that $\tau_{min}(C)(\nu) \leq \mathsf{time}(q) \leq \tau_{max}(C)(\nu)$.

Example 2. In the PTPN of the figure 1, we can exhibit the two following extended classes:

$$\begin{array}{lcl}
\mathbf{C_0} = (\mathbf{M_0}, \mathbf{D_0}): & & \mathbf{C_1} = (\mathbf{M_1}, \mathbf{D_1}): \\
M_0 = (A, B) & & M_1 = (B, C) \\
D_0 = \begin{cases} 0 \leq a \leq \theta_1 \leq 10, \\ 0 \leq b \leq \theta_2 \leq c, \\ \theta_3 = 5, \theta_c = 0. \end{cases} & \xrightarrow{t_1} & D_1 = \begin{cases} \theta_3 - \theta_c = 5, \\ b \leq \theta_2 - \theta_c \leq c, \\ -5 \leq \theta_c \leq -a, \\ 0 \leq \theta_2, 0 \leq a, 0 \leq b. \end{cases}
\end{array}$$

Thus, the elapsed time after the firing of t_1 is such that $\tau_{min}(C_1) = a$ and $\tau_{max}(C_1) = \min(5, c)$.

4.3 Principles of parametric model-checking with the state-class graph

Given a PITPN $\mathcal{N}$ and a PTCTL property ϕ, in the parametric model-checking problem we want to characterize the set $F(\mathcal{N}, \phi)$ of all the parameters valuations that resolve the problem, which is defined by:

$$F(\mathcal{N}, \phi) = \{\nu \in D_p \mid [\![\mathcal{N}]\!]_\nu \models [\![\phi]\!]_\nu\}$$

To achieve this we are going to recursively compute, on each extended class $C = (M, D)$, a logical predicate on the parameters that corresponds to the verification of the property on the current class and its successors. This predicate represents the set: $F^{\phi}(C) = \{\nu \in D_{|Par} \mid [\![C]\!]_\nu \models [\![\phi]\!]_\nu\}$

We begin by giving an interpretation of the verification of a PTCTL formula ϕ on an extended parametric state class C, which we write $[\![C]\!]_\nu \models [\![\phi]\!]_\nu$.

Formulae $\boldsymbol{\phi = \exists\varphi\mathcal{U}_J\psi}$ or $\boldsymbol{\phi = \forall\varphi\mathcal{U}_J\psi}$: For a valuation $\nu \in D_{|Par}$ and a state $q \in [\![C]\!]_\nu$, we define $[\![\phi[J - time(q)]]\!]_\nu$ as the TCTL formula obtained after replacing in ϕ, the parametric time interval J by $J(\nu) - time(q)$. Then, according to the form of the PTCTL formula ϕ we define:

– if $\phi = \exists\varphi\mathcal{U}_J\psi$, then $[\![C]\!]_\nu \models [\![\phi]\!]_\nu$ iff $\exists q \in [\![C]\!]_\nu,\ q \models [\![\phi[J - \mathsf{time}(q)]]\!]_\nu$
– if $\phi = \forall\varphi\mathcal{U}_J\psi$, then $[\![C]\!]_\nu \models [\![\phi]\!]_\nu$ iff $\forall q \in [\![C]\!]_\nu,\ q \models [\![\phi[J - \mathsf{time}(q)]]\!]_\nu$

Formulae $\boldsymbol{\phi = \varphi \leadsto_{J_r} \psi}$: We extend the PITPN $\mathcal{N}$ with an additional place named P_{LT} that will be marked if and only if we are looking for ψ. We denote by $\mathcal{N}_{LT}$ the resulting PITPN. In this model, the successor $C' = (M', D') = succ_{LT}(C, t)$ of an extended parametric state-class $C = (M, D) \in \mathcal{G}_c(\mathcal{N}_{LT})$ by a transition $t_f \in \mathsf{firable}\,(C)$, is given by:

– $M' = M - {}^\bullet t_f + t_f^\bullet$ and $\begin{cases} \text{if } (M' \models \varphi \text{ and } M' \not\models \psi) \text{ then } M'(P_{LT}) = 1, \\ \text{else if } (M \models \psi) \text{ then } M'(P_{LT}) = 0, \\ \text{else } M'(P_{LT}) = M(P_{LT}) \end{cases}$
– $D' = \mathsf{next}(D, t_f)$ and
if $(M(P_{LT}) = 0$ or $M \models \psi)$ then the clock variable θ_c is reset to zero.

On this model we define that $[\![C]\!]_\nu \models [\![\phi]\!]_\nu$ if and only if $\forall q \in [\![C]\!]_\nu,\ \forall \rho \in \pi(q)$,

$$M(P_{LT}) = 1 \Rightarrow \begin{cases} \exists 0 \leq r_1 \leq J_r(\nu)^{\downarrow} - \mathsf{time}(q) \text{ s.t. } \rho^*(r_1) \models \psi \text{ and} \\ \forall r_2 \geq r_1\ \rho^*(r_2) \models M(P_{LT}) = 1 \Rightarrow \exists r_3 \geq r_2 \\ \qquad\qquad \text{s.t. } r_3 - r_2 \leq J_r(\nu)^{\downarrow} \text{ and } \rho^*(r_3) \models \psi \end{cases}$$
$$M(P_{LT}) = 0 \Rightarrow \begin{cases} \forall r_2 \geq 0\ \rho^*(r_2) \models M(P_{LT}) = 1 \Rightarrow \exists r_3 \geq r_2 \\ \qquad \text{s.t. } r_3 - r_2 \leq J_r(\nu)^{\downarrow} \text{ and } \rho^*(r_3) \models \psi \end{cases}$$

In this model, $\mathsf{time}(q)$ refers to the time elapsed since the last reinitialization of θ_c. We notice that when the time has been reset (then $\mathsf{time}(q) = 0$) the two definitions above are equivalent and correspond to $q \models [\![\phi]\!]_\nu$.

Finally, the theorem 5 states that we are able to resolve the parametric model-checking problem if we compute the set of solutions on the initial class.

Theorem 5. *Given a PITPN $\mathcal{N}$ and a PTCTL formula ϕ, $F(\mathcal{N}, \phi) = F^{\phi}(C_0)$, where C_0 is the initial class of the extended parametric state class graph of $\mathcal{N}$.*

4.4 Parametric model-checking semi-algorithms

To verify PTCTL formulae we propose three semi-algorithms according to the form of the formulae. These algorithms recursively characterize, for each class C, the set $F^{\phi}(C)$. This set is represented by conjunctions or disjunctions of linear constraints on the parameters. We use a disjunctive normal form (i.e. a disjunction of convex polyhedra).

Algorithm EU: This semi-algorithm is designed for formulae whose form is $\phi = \exists\varphi\mathcal{U}_J\psi$, where $J \in \mathcal{J}(Par)$. Let be $C = (M, D) \in \mathcal{G}_c(\mathcal{N})$, we compute:

$$\begin{aligned}
F^{\phi}_{EU}(C) =& D_{|Par} \ \wedge\ \{\tau_{min}(C) \leq J^{\downarrow}\} \ \wedge\ \Bigg(\Big(M \models \psi \ \wedge \{\tau_{max}(C) \geq J^{\uparrow}\}\Big) \\
&\vee \Big(M \models \varphi \ \wedge\ M \models \psi \ \wedge\ \big(\mathsf{firable}\,(C) = \emptyset \vee \\
&\qquad (\bigvee_{\substack{t\in\mathsf{firable}(C)\\ C'=(M',D')=succ(C,t)}} (\{\tau_{max}(C') \geq J^{\uparrow}\} \wedge\ D'_{|Par})))\Big) \\
&\vee \Big(M \models \varphi \ \wedge\ \mathsf{firable}\,(C) \neq \emptyset \ \wedge\ (\bigvee_{\substack{t\in\mathsf{firable}(C)\\ C'=succ(C,t)}} F^{\phi}_{EU}(C'))\Big)\Bigg)
\end{aligned}$$

This formulae establishes three conditions in disjunction to prove the formula ϕ:

- The first disjunction is used when C verifies ψ but not ϕ. Thus, the elapsed time must be entailed in the interval $\mathcal{J}$ as soon as we get into the class.
- The second case is when both ϕ and ψ are verified. Comparing to the first one it allows to wait in the class.
- The third disjunction is used when only ϕ is verified. In this case we have to compute the successors of C.

Example 3. In the net of the figure 1 we check the formula:
$\phi = \exists\Diamond_{[0,inf]}(M(D) = 1)$. The result is $F^{\phi}_{EU}(C_0) = \{b <= 5\}$.

Algorithm AU: This semi-algorithm is designed for formulae whose form is $\phi = \forall\varphi\mathcal{U}_J\psi$, where $J \in \mathcal{J}(Par)$. Let be $C = (M, D) \in \mathcal{G}_c(\mathcal{N})$, we compute:

$$\begin{aligned}
F^{\phi}_{AU}(C) =& D_{|Par} \ \wedge\ \begin{Bmatrix} \tau_{max}(C) \leq J^{\downarrow} \\ \tau_{max}(C) \neq \infty \end{Bmatrix} \wedge\ \Bigg(\Big(M \models \psi \ \ \wedge \{\tau_{min}(C) \geq J^{\uparrow}\}\Big) \\
&\vee \Big(M \models \varphi \ \wedge\ M \models \psi \ \wedge\ \big(\mathsf{firable}\,(C) = \emptyset \vee \\
&\qquad (\bigwedge_{\substack{t\in\mathsf{firable}(C)\\ C'=(M',D')=succ(C,t)\\ D''=D'\wedge\{\theta_c > -J^{\uparrow}\}}} (F^{\phi}_{AU}(M', D'') \vee \neg D''_{|Par})))\Big) \\
&\vee \Big(M \models \varphi \ \wedge\ \mathsf{firable}\,(C) \neq \emptyset \ \wedge\ \big(\bigwedge_{\substack{t\in\mathsf{firable}(C)\\ C'=(M',D')=succ(C,t)}} \big(F^{\phi}_{AU}(C') \vee \neg D'_{|Par}\big)\big)\Big)\Bigg)
\end{aligned}$$

Similarly, there are three conditions in disjunction. Unlike previously,. in the second one successors are computed, but only on the points of the class for which the property has not been verified yet.

Example 4. In the net of the figure 1 we check the formula:
$\phi = \forall\Diamond_{[0,inf]}(M(D) = 1)$. The result is $F^{\phi}_{AU}(C_0) = \{c < 5\}$.

Algorithm LT: This semi-algorithm is designed for formulae whose form is $\phi = \varphi \leadsto_{J_r} \psi$, where $J_r \in \mathcal{J}(Par)$ such that $J_r = [0, b]$ with $b \in \mathbb{Q}^+ \cup Par$, or $J_r = [0, \infty[$. Let be $C = (M, D) \in \mathcal{G}_c(\mathcal{N}_{LT})$, we compute:

$$\begin{aligned} F^{\phi}_{LT}(C) = & D_{|Par} \ \wedge \ \left(M(P_{LT}) = 0 \ \vee \ \begin{Bmatrix} \tau_{max}(C) \le J_r^{\downarrow} \\ \tau_{max}(C) \neq \infty \end{Bmatrix} \right) \\ & \wedge \left(\left(\mathsf{firable}\,(C) = \emptyset \wedge \left(M(P_{LT}) = 0 \vee M \models \psi \right) \right) \vee \right. \\ & \left. \left(\mathsf{firable}\,(C) \neq \emptyset \ \wedge \left(\bigwedge_{\substack{t \in \mathsf{firable}(C) \\ C' = (M', D') = succ_{LT}(C, t)}} \left(F^{\phi}_{LT}(C') \vee \neg D'_{|Par} \right) \right) \right) \right) \end{aligned}$$

This algorithm is similar to F_{AU} when $J^{\uparrow} = 0$. However the analysis only stop when no successors are found.

5 Discussion

The semi-algorithms presented in this paper have been implemented in the tool Romeo [17], a software for time Petri nets analysis. For polyhedra manipulation, the Parma Polyhedra Library [18] is used to represent the firing domains of the parametric state-classes and the logical formulae computed by the model-checking algorithms. These formulae are represented as powersets of convex polyhedra, that is to say a finite disjunction of polyhedra.

As mentioned before, the parametric model-checking problem is undecidable. Indeed the parametric state-class graph of a PITPN may be infinite. Additionally, to determine the whole set of parameters valuations that satisfy a formula, it would be in general necessary to analyze every parametric state-class. Nevertheless, some methods can help with the termination. In this way, if a parametric state-class $C = (M, D)$ is included in another class $C' = (M', D')$ (i.e. $M = M'$ and $D \subseteq D'$), it can be shown that $F^{\phi}_{EU}(C) \subseteq F^{\phi}_{EU}(C')$, and on the contrary that $F^{\phi}_{AU}(C') \subseteq F^{\phi}_{AU}(C) \vee \neg D_{|Par}$ and $F^{\phi}_{LT}(C') \subseteq F^{\phi}_{LT}(C) \vee \neg D_{|Par}$. As a result, in our "on-the-fly" model-checking approach it will not be necessary to analyze the whole state-class graph, but we will be able to stop the analysis of successors when finding included parametric state-classes.

Conclusion

In this paper, we have introduced a parametric extension of time Petri nets with stopwatches where the temporal bounds of the firing intervals are replaced by temporal parameters. We have proposed a symbolic representation of the state-space of these parametric models which is based on a parametric extension of the state-class graph. Upon this abstraction we have developed semi-algorithms for the parametric model-checking of parametric TCTL formulae.

In our future works we want to integrate this parametric approach in the development cycle of real-time systems through the functional decomposition of the systems. On concrete examples, a parametric decomposition combined with a projection of the formulae to verify can be useful in the development process. We hope to succeed in the elaboration of a formal framework for this method so that the process could be automated.

References

1. Merlin, P.: A study of the recoverability of computing systems. PhD thesis, Department of Information and Computer Science, Univ. of California, Irvine (1974)
2. Bucci, G., Fedeli, A., Sassoli, L., Vicario, E.: Time state space analysis of real-time preemptive systems. IEEE trans. on Soft. Eng. **30**(2) (February 2004) 97–111
3. Roux, O.H., Lime, D.: Time Petri nets with inhibitor hyperarcs. Formal semantics and state space computation. In: ICATPN'04. Volume 3099 of LNCS., Bologna, Italy, Springer (June 2004) 371–390
4. Berthomieu, B., Lime, D., Roux, O.H., Vernadat, F.: Reachability problems and abstract state spaces for time petri nets with stopwatches. Discrete Event Dynamic Systems **17**(2) (2007) 133–158
5. Alur, R., Henzinger, T.A., Vardi, M.Y.: Parametric real-time reasoning. In: ACM Symposium on Theory of Computing. (1993) 592–601
6. Hune, T., Romijn, J., Stoelinga, M., Vaandrager, F.W.: Linear parametric model checking of timed automata. In: TACAS. Volume 2031 of LNCS., Springer (2001)
7. Bruyère, V., Raskin, J.F.: Real-time model-checking: Parameters everywhere. CoRR **abs/cs/0701138** (2007)
8. Bruyère, V., Dall'olio, E., Raskin, J.F.: Durations and parametric model-checking in timed automata. ACM Trans. Comput. Logic **9**(2) (2008) 1–23
9. Henzinger, T.A., Ho, P.H., Wong-Toi, H.: HYTECH: A model checker for hybrid systems. Int. Journal on Soft. Tools for Technology Transfer **1**(1–2) (1997) 110–122
10. Wang, F.: Parametric timing analysis for real-time systems. Inf. Comput. **130**(2) (1996) 131–150
11. Virbitskaite, I., Pokozy, E.: Parametric behaviour analysis for time petri nets. In: PaCT '999: Proceedings of the 5th Int. Conference on Parallel Computing Technologies, London, UK, Springer-Verlag (1999) 134–140
12. Berthomieu, B., Diaz, M.: Modeling and verification of time dependent systems using time Petri nets. IEEE trans. on Soft. Eng. **17**(3) (1991) 259–273
13. Hadjidj, R., Boucheneb, H.: On-the-fly tctl model checking for time petri nets using state class graphs. In: ACSD, IEEE Computer Society (2006) 111–122
14. Larsen, K.G., Pettersson, P., Yi, W.: Model-checking for real-time systems. In: Fundamentals of Computation Theory. (1995) 62–88
15. Cassez, F., Roux, O.H.: Structural translation from Time Petri Nets to Timed Automata – Model-Checking Time Petri Nets via Timed Automata. The journal of Systems and Software **79**(10) (2006) 1456–1468
16. Bérard, B., Cassez, F., Haddad, S., Lime, D., Roux, O.H.: Comparison of the expressiveness of timed automata and time Petri nets. In: FORMATS 05. Volume 3829 of LNCS., Uppsala, Sweden, Springer (2005)
17. Olivier (H.) Roux, Didier Lime, G.G., Magnin, M.: Roméo. Available at http://romeo.rts-software.org (2006)
18. Bagnara, R., Hill, P.M., Zaffanella, E.: The Parma Polyhedra Library. Quaderno 457, Dipartimento di Matematica, Università di Parma, Italy (2006)