Precise Robustness Analysis of Time Petri Nets with Inhibitor Arcs

Étienne André, Giuseppe Pellegrino*, and Laure Petrucci

Université Paris 13, Sorbonne Paris Cité, LIPN
F-93430, Villetaneuse, France

Abstract. Quantifying the robustness of a real-time system consists in measuring the maximum extension of the timing delays such that the system still satisfies its specification. In this work, we introduce a more precise notion of robustness, measuring the allowed variability of the timing delays in their neighbourhood. We consider here the formalism of time Petri nets extended with inhibitor arcs. We use the inverse method, initially defined for timed automata. Its output, in the form of a parametric linear constraint relating all timing delays, allows the designer to identify the delays allowing the least variability. We also exhibit a condition and a construction for rendering robust a non-robust system.

Keywords: Time Petri nets, Quantitative robustness, Parameter synthesis.

1 Introduction

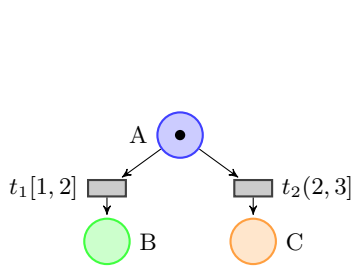
Formalisms for modelling real-time systems, such as time Petri nets [10] or timed automata [3], have been extensively used in the past decades, and led to useful and efficient implementations. Time Petri nets (TPNs for short) are an extension of Petri nets where firing conditions are given in the form of intervals $[a, b]$. Each transition can only fire at least a time units and at most b time units after it is enabled. ITPNs extend TPNs with inhibitor arcs, i.e. arcs that disable their outgoing transition if their incoming place is not empty.

However, these formalisms allow for modelling in theory delays arbitrarily close (or even equal) to zero; this implies that the real system must be arbitrarily fast, which may be unrealistic in practice, where response times may not be neglected. These formalisms also allow for simultaneous occurrence of events, which may not be realistic in practice either, due to slightly different clock rates of several processors. And similarly, they allow for arbitrary precision, which is unrealistic: For example, a system where some component performs an action for e.g. 2 seconds can be implemented with a delay greater but very close to 2 (e.g. 2.0001 s), in which case the formal guarantee may not hold anymore.

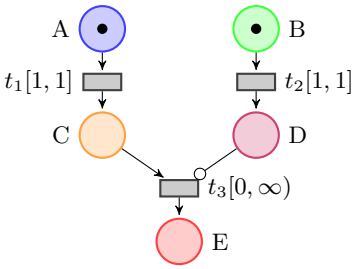
The implementation in practice of a real-time system (modelled, e.g. by an ITPN) can lead in particular to two kinds of undesired consequences: the occurrence of behaviours that were proven impossible in theory, and the unlikely occurrence of behaviours that were proven possible in theory.

* This work is partially supported by an Erasmus grant.

V. Braberman and L. Fribourg (Eds.): FORMATS 2013, LNCS 8053, pp. 1–15, 2013.



(a) Example of undesired reachability



(b) Example of unlikely reachability

Fig. 1. Examples of non-robust ITPNs

Consider the simple TPN in Fig. 1a (from [2]). According to the semantics of TPNs (e.g. defined in [13]), place C is unreachable, that is, there exists no reachable marking such that the number of tokens in C is greater than 0. Indeed, starting from marking A (i.e. a marking with 1 token in place A), t_1 can fire anytime between 1 and 2 time units after the system start. At time 2, t_1 must fire if it has not yet fired, because its associated interval is about to expire and no other transition is firable (t_2 will be firable right after time 2). Hence, C is unreachable. Now suppose that the upper bound of the firing interval of t_1 is increased, even by an infinitesimal duration. Then, t_2 is firable immediately after time 2, and C can be reached in some executions.

Now consider the ITPN in Fig. 1b. According to the semantics of ITPNs, place E is reachable. Indeed, starting from a marking AB (i.e. a marking with 1 token in place A and 1 token in place B), t_1 can fire at time 1, giving marking CB. Then, after a null duration, t_3 can fire due to the absence of token in D. This sequence of transitions is unlikely to happen in practice due to delays exactly equal to zero; if the bounds of t_1 or the lower bound of t_3 become slightly larger, or the bounds of t_2 becomes slightly smaller, E becomes unreachable.

In this work, we use techniques based on parameter synthesis to compute a precise quantitative analysis of the admissible variability of the timing bounds of an ITPN with respect to linear-time properties. We use PITPNs, that is extensions of ITPNs where timing bounds are *parameters*, i.e. unknown constants. Our contributions are as follows:

1. We define the notion of *covering constraint* for parametric time Petri nets with inhibitor arcs (PITPNs), and characterise it;
2. We extend the *inverse method* to PITPNs (initially defined in the setting of parametric timed automata [4]), and prove that it preserves linear-time properties, based on the notion of covering constraint; and
3. We exploit the constraint output to obtain a precise quantitative measure of the system robustness for linear-time properties.

Given in the form of a constraint on the timing bounds seen as parameters, our robustness condition allows a designer (i) to relate the variability of the timing bounds with each other, (ii) to exhibit the critical timing bounds that do not allow any variability, and (iii) to render a system robust under certain conditions.

Related Work. Robustness in the setting of timed automata has received much attention in the past decade (see [9] for a survey). Most previous works (see e.g. [5,9,8,2,12,6]) consider that all timing constraints can be enlarged by a single very small (but positive) variation Δ. This robustness condition considers a unique positive parameter Δ; hence, roughly speaking, the robustness is guaranteed as long as the different clocks remain in intervals $[a-\Delta, b+\Delta]$ instead of $[a, b]$. In a geometrical context, the admissible variability can be seen as a simple hypercube (called "Δ-cube" from now on) in $2*n$ dimensions, with n the number of timing constraints. In contrast, we give a precise measure of the robustness, by considering possible local variations of each lower and upper bound of the firing intervals of a time Petri net. This is given in the form of a *polyhedron* in $2*n$ dimensions, where n is the number of transitions. Hence, each bound can vary independently of the others. Our approach has the following advantages: (1) it identifies the most critical interval bounds, and helps the designer in tuning them (when possible) so that the system becomes robust; (2) it relates bounds in a parametric way, identifying bounds that should, for example, remain smaller than others; (3) it also outputs a constraint even when some bounds cannot tolerate any variation, whereas Δ-based approaches would just classify the system as non-robust (i.e. synthesise a $\Delta = 0$). Since parameter synthesis is undecidable for PITPNs [13], our algorithm may not terminate in the general case; however, we give sufficient termination conditions for subclasses of PITPNs.

In [6], it is shown that parameterised robust reachability in timed automata is decidable, again for a single Δ. In [8], computing the greatest acceptable variation Δ is proven decidable for flat timed automata with progressive clocks. In [12], a counter-example refinement approach is used with parametric techniques to evaluate the greatest acceptable variation Δ for parametric timed automata (although not decidable in the general case). These works share similarities with ours in the problem addressed and in the use of parametric techniques. However, beyond the fact that these works consider (a restriction of) timed automata whereas we consider (an extension of) time Petri nets, the main difference lies in the number of dimensions, since they all consider a simple Δ.

Recent work also considered robustness issues in time Petri nets. In [2], the quantification of robustness is performed by considering that the firing intervals can be enlarged by a (positive) parameter. Two problems are considered: the robust boundedness of the net (a bounded net remains bounded even in presence of small time variations) and the robust untimed language preservation (the untimed language remains preserved in presence of small time variations). Our work is close to [2], with notable differences. First, we use here a technique based on parameter synthesis. Second, we give a condition for trace preservation, where traces are defined as alternating markings and actions. Hence, the robustness condition in our work is different from the boundedness and language preservation of [2]. Last but not least, the robustness condition in [2] again considers a unique positive parameter Δ, whereas we compute a polyhedron in $2*n$ dimensions. In [1], a more general notion of robustness is used for time Petri nets, that includes not only a robustness with respect to time, but

also with constraints on the resources (e.g. memory), scheduling schemes (in a multi-processor environment) and possible system failures.

Outline. Section 2 recalls PITPNs and related results. In Section 3, we introduce and characterise covering constraints. In Section 4, we introduce the inverse method for PITPNs and prove its correctness. In Section 5, we exhibit a precise quantitative measure of the system robustness, and use it to turn some non-robust systems robust. We give directions of future research in Section 6.

2 Preliminaries

We denote by $\mathbb{N}$, $\mathbb{Q}_+$ and $\mathbb{R}_+$ the sets of non-negative integers, non-negative rational and non-negative real numbers, respectively.

2.1 Firing Times, Parameters and Constraints

Throughout this paper, we assume a set $\{\theta_1, \theta_2, \dots\}$ of *firing times*. A *firing time* is a variable with value in $\mathbb{R}_+$, encoding the time remaining before a given transition fires. In the following, Θ will denote a finite set $\{\theta_1, \dots, \theta_H\}$ of firing times, for some $H \in \mathbb{N}$. A *firing time valuation* is a function $\nu : \Theta \to \mathbb{R}_+^H$ assigning a non-negative real value with each firing time.

We also assume a set $\{\lambda_1, \lambda_2, \dots\}$ of *parameters*, i.e. unknown constants. In the following, $\Lambda = \{\lambda_1, \dots, \lambda_l\}$ denotes a finite set of parameters for some $l \in \mathbb{N}$. A *parameter valuation* π is a function $\pi : \Lambda \to \mathbb{R}_+$ assigning with each parameter a value in $\mathbb{R}_+$. A valuation π can be seen as a point $(\pi(\lambda_1), \dots, \pi(\lambda_l))$.

Constraints are defined as a set of inequalities. A (linear) inequality over Θ and Λ is $lt \prec lt'$, where $\prec \in \{<, \leq\}$, and lt, lt' are two linear terms of the form $\sum_{1 \leq i \leq N} \alpha_i z_i + d$ where $z_i \in \Theta \cup \Lambda$, $\alpha_i \in \mathbb{Q}_+$, for $1 \leq i \leq N$, and $d \in \mathbb{Q}_+$. We define similarly inequalities over Θ (resp. Λ). A constraint is a conjunction of inequalities. In particular, a constraint over the parameters can be seen as a polyhedron in l dimensions. We denote by $\mathcal{L}(\Lambda)$ the set of all constraints over the parameters. In the sequel, J denotes an inequality over the parameters, E a constraint over the firing times, K a constraint over the parameters, and D a constraint over firing times and parameters. Often, given a PITPN transition t_i, we will denote its parametric lower and upper bounds by λ_i^- and λ_i^+, respectively.

Given an inequality J of the form $lt < lt'$ (respectively $lt \leq lt'$), the *negation* of J, denoted by $\neg J$, is the inequality $lt' \leq lt$ (respectively $lt' < lt$).

Given a constraint E and a firing time valuation ν, $[\![E]\!]_\nu$ denotes the expression obtained by replacing each firing time θ in E with $\nu(\theta)$. A firing time valuation ν *satisfies* constraint E (denoted by $\nu \models E$) if $[\![E]\!]_\nu$ evaluates to true.

Given a parameter valuation π and a constraint D, $[\![D]\!]_\pi$ denotes the constraint over Θ obtained by replacing each parameter λ in D with $\pi(\lambda)$. Likewise, given a firing time valuation ν, $[\![[\![D]\!]_\pi]\!]_\nu$ denotes the expression obtained by replacing each firing time θ in $[\![D]\!]_\pi$ with $\nu(\theta)$. We say that a parameter valuation π *satisfies* a constraint D, denoted by $\pi \models D$, if the set of firing time valuations that satisfy $[\![D]\!]_\pi$ is non-empty.

A parameter valuation π *satisfies* a constraint K over the parameters, denoted by $\pi \models K$, if the expression obtained by replacing each parameter λ in K with $\pi(\lambda)$ evaluates to true. Given two constraints K_1 and K_2, K_1 is *included in* K_2, denoted by $K_1 \subseteq K_2$, if $\forall \pi : \pi \models K_1 \Rightarrow \pi \models K_2$. We consider `true` as a constraint over Λ, corresponding to the set of all possible values for Λ.

We denote by $D{\downarrow_\Lambda}$ the constraint over Λ obtained by projecting D onto Λ, i.e. after elimination of the firing times. Formally, $D{\downarrow_\Lambda} = \{\pi \mid \pi \models D\}$.

We finally define intervals as in [13]. An interval I of $\mathbb{R}_+$ is a $\mathbb{Q}_+$-interval if its left endpoint ${}^{\uparrow}I$ belongs to $\mathbb{Q}_+$ and its right endpoint $I^{\uparrow}$ belongs to $\mathbb{Q}_+ \cup \{\infty\}$. We denote by $\mathcal{I}(\mathbb{Q}_+)$ the set of $\mathbb{Q}_+$-intervals of $\mathbb{R}_+$. A parametric time interval is a function $J : {\mathbb{Q}_+}^{\Lambda} \to \mathcal{I}(\mathbb{Q}_+)$ that associates with each parameter valuation a $\mathbb{Q}_+$-interval. The set of parametric time intervals over Λ is denoted by $\mathcal{J}(\Lambda)$. As for I, we define ${}^{\uparrow}J$ and $J^{\uparrow}$ as the minimum and maximum bounds of J, respectively. They can both be represented using a constraint over Λ.

2.2 Parametric Time Petri Nets with Inhibitor Arcs

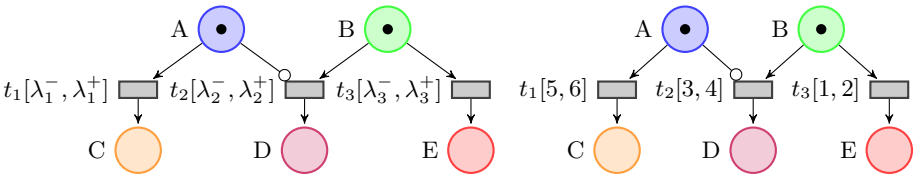
Parametric time Petri nets with inhibitor arcs (PITPNs) are a parametric extension of ITPNs, where the temporal bounds of the transitions can be parameters. We slightly adapt the notations defined in [13] to fit our setting.

Definition 1. *A parametric time Petri nets with inhibitor arcs (PITPN) is a tuple* $\mathcal{N} = \langle P, T, \Lambda, {}^{\bullet}(.), (.)^{\bullet}, (.)^{\circ}, M_0, J_s, K_0 \rangle$ *where*

- $P = \{p_1, \ldots, p_m\}$ *is a non-empty finite set of* places,
- $T = \{t_1, \ldots, t_n\}$ *is a non-empty finite set of* transitions,
- $\Lambda = \{\lambda_1, \ldots, \lambda_l\}$ *is a finite set of* parameters,
- ${}^{\bullet}(.)$ *(resp.* $(.)^{\bullet}$*)* $\in (\mathbb{N}^P)^T$ *is the* backward *(resp.* forward*)* incidence function,
- $(.)^{\circ} \in (\mathbb{N}^P)^T$ *is the* inhibition function,
- $M_0 \in \mathbb{N}^P$ *is the* initial marking,
- $J_s \in \mathcal{J}(\Lambda))^T$ *is the function that associates a* parametric firing interval *with each transition, and*
- $K_0 \in \mathcal{L}(\Lambda)$ *is the* initial constraint *over* Λ.

K_0 is a constraint over Λ giving the initial domain of the parameters, and must at least specify that the minimum bounds of the firing intervals are lower than or equal to the maximum bounds. Additional linear constraints may of course be given. Sometimes, given a constraint K_0, we will denote a PITPN by $\mathcal{N}(K_0)$ when clear from the context, and to emphasise the value of K_0 in $\mathcal{N}$.

Given a PITPN $\mathcal{N}$ and a valuation π, we denote by $[\![\mathcal{N}]\!]_\pi$ the (non-parametric) ITPN where each occurrence of a parameter has been replaced by its constant value as in π. Formally, given $\mathcal{N} = \langle P, T, \Lambda, {}^{\bullet}(.), (.)^{\bullet}, (.)^{\circ}, M_0, J_s, K_0 \rangle$, then $[\![\mathcal{N}]\!]_\pi = \langle P, T, \Lambda, {}^{\bullet}(.), (.)^{\bullet}, (.)^{\circ}, M_0, J_s, K_0 \wedge K_\pi \rangle$, where $K_\pi = \bigwedge_{\lambda \in \Lambda} \big(\lambda = \pi(\lambda)\big)$. For example, the ITPN in Fig. 2b corresponds to the PITPN in Fig. 2a valuated with $\pi = \{\lambda_1^- \to 5, \lambda_1^+ \to 6, \lambda_2^- \to 3, \lambda_2^+ \to 4, \lambda_3^- \to 1, \lambda_3^+ \to 2\}$.



(a) A PITPN $\mathcal{N}$ (b) A valuated (P)ITPN $[\![\mathcal{N}]\!]_\pi$

Fig. 2. A PITPN and its valuation

Semantics. We mostly reuse here the definitions and semantics from [13]. The reachable states of a PITPN are *parametric state-classes* (or simply *classes*), i.e. pairs $c = (M, D)$ where M is a marking of the net and D is a parametric firing domain, that is, a constraint over Θ and Λ. Given a class $c = (M, D)$, a transition t is *enabled* in c if $M \geq {}^\bullet t$ (i.e. if the number of tokens in M in each input place of t is greater than or equal to the value on the arc between this place and the transition). Transition t is *inhibited* 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. Transition t is *active* if it is enabled and not inhibited. Transition t is *firable* if it has been active for at least ${}^\uparrow J_s(t)$ time units.

For a given class, the firing times in Θ correspond to variables encoding the time remaining before an active transition can fire. Hence, these variables *decrease* with time. The initial class of $\mathcal{N}(K)$ is $c_0 = (M_0, D_0)$, with $D_0 = K \wedge \{\theta_k \in J_s(t_k) | (t_k \in \mathit{enabled}(M_0)\}$, where $\mathit{enabled}(M_0)$ denotes the enabled transitions in M_0. For example, suppose that $K = \lambda_1^- \leq \lambda_1^+ \wedge \lambda_2^- \leq \lambda_2^+ \wedge \lambda_3^- \leq \lambda_3^+$; then the initial class of $\mathcal{N}$ in Fig. 2a is:

$$c_0 = (AB, \lambda_1^- \leq \theta_1 \leq \lambda_1^+ \wedge \lambda_2^- \leq \theta_2 \leq \lambda_2^+ \wedge \lambda_3^- \leq \theta_3 \leq \lambda_3^+).$$

We consider a (classical) semantics where a transition must fire before its upper interval bound, unless another transition fires first and disables it; for example, in Fig. 2a, t_1 must fire before t_3 if $\lambda_1^+ < \lambda_3^-$, t_3 must fire before t_1 if $\lambda_3^+ < \lambda_1^-$, and both orders are possible otherwise. Given a class $c = (M, D)$ and a firable transition t_f, $c' = (M', D')$ can be reached from c in one step via transition t_f (denoted by $c \overset{t_f}{\Rightarrow} c'$) if the following holds:

- $M' = M - {}^\bullet t_f + t_f^\bullet$
- D' is computed along the following steps:
  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 t_j that are active, i.e. $\theta_j = \theta_f + \theta'_j$,
  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 inequalities relative to newly enabled transitions[1]: $\forall t_k \in \mathit{NewlyEnabled}(M, t_f), {}^{\uparrow}J_s(t_k) \leq \theta'_k \leq J_s(t_k)^{\uparrow}$, with $\mathit{NewlyEnabled}(M, t_f)$ denoting the set of transitions newly enabled by firing the transition t_f from marking M.

The full semantics can be found in [13].

A *run* of $\mathcal{N}$ is a sequence $c_0 \overset{t_0}{\Rightarrow} \cdots \overset{t_{n-1}}{\Rightarrow} c_n$. Given a run r of $\mathcal{N}$ of the form $(M_0, D_0) \overset{t_0}{\Rightarrow} \cdots \overset{t_{n-1}}{\Rightarrow} (M_n, D_n)$, the *trace associated with* r is the alternating sequence of markings and actions $M_0 \overset{t_0}{\Rightarrow} \cdots \overset{t_{n-1}}{\Rightarrow} M_n$. The *trace set* of $\mathcal{N}$ is the set of all traces associated with the runs of $\mathcal{N}$. This corresponds to the discrete (or time-abstract) behaviour of $\mathcal{N}$. $\mathit{Post}_{\mathcal{N}(K)}(C)$ (resp. $\mathit{Post}^i_{\mathcal{N}(K)}(C)$) is the set of classes reachable from a set C of classes in exactly one step (resp. i steps) in $\mathcal{N}(K)$. Furthermore, we define $\mathit{Post}^*_{\mathcal{N}(K)}(C)$ as $\bigcup_{i \geq 0} \mathit{Post}^i_{\mathcal{N}(K)}(C)$. We define $\mathit{Reach}(\mathcal{N}(K))$ as the set of reachable classes of $\mathcal{N}(K)$, that is $\mathit{Post}^*_{\mathcal{N}(K)}(\{c_0\})$. Finally, we define $\mathcal{G}(\mathcal{N}(K))$ as the parametric reachability graph of $\mathcal{N}(K)$, that is the set of reachable parametric state-classes with the transition relation $\Rightarrow$.

Results. The following lemma, recalled from [13], states that the projection onto the parameters of the constraint associated with a class always gets stronger (i.e. more restricted) along a run of the system.

Lemma 1 (Lemma 14 in [13]). *Given a PITPN $\mathcal{N}$, let $c = (M, D)$ and $c' = (M', D')$ be two classes in $\mathcal{G}(\mathcal{N})$. If $c \overset{t}{\Rightarrow} c'$, then $D'{\downarrow_\Lambda} \subseteq D{\downarrow_\Lambda}$.*

The following result states that the valuation with π of a class c of $\mathcal{N}$ belongs to the graph of $\mathcal{N}$ valuated with π if and only if π belongs to the constraint associated with c.

Theorem 1 (Theorems 12 and 13 in [13]). *Given a PITPN $\mathcal{N}(K)$ and a valuation $\pi \models K$, let $c = (M, D)$ be a class in $\mathcal{G}(\mathcal{N}(K))$. Then: $[\![c]\!]_\pi \in \mathcal{G}([\![\mathcal{N}]\!]_\pi)$ iff $\pi \models D{\downarrow_\Lambda}$.*

3 Covering Constraint

We introduce the notion of covering constraint as the constraint resulting from the intersection of the projection onto the parameters of the constraints associated with all the reachable classes of a PITPN.

Definition 2. *Let $\mathcal{N}$ be a PITPN. The* covering constraint *of $\mathcal{N}$ is:* $\bigcap_{(M,D) \in \mathit{Reach}(\mathcal{N})} D{\downarrow_\Lambda}$.

In the general case, it is possible that the covering constraint of a PITPN will be empty, due to the intersection of disjoint constraints over the parameters. But in the setting of the inverse method (see Section 4), it will not be.

The following lemma relates parametric and non-parametric runs, and derives from Theorem 1.

[1] For sake of simplicity, we only consider here closed intervals of the form $[a, b]$. For open intervals (e.g. $(2, 3]$ in Fig. 1a), one should use strict instead of large inequalities.

Lemma 2. *Let $\mathcal{N}$ be a PITPN, let π be a parameter valuation. Let r be a run of $\mathcal{N}$ reaching a class (M, D) in $\mathcal{G}(\mathcal{N})$. Then there exists an equivalent run in $[\![\mathcal{N}]\!]_\pi$ reaching class $(M, [\![D]\!]_\pi)$ in $\mathcal{G}([\![\mathcal{N}]\!]_\pi)$ iff $\pi \models D{\downarrow_\Lambda}$.*

Proof. Let $(M_0, D_0) \overset{t_0}{\Rightarrow} \dots \overset{t_{k-1}}{\Rightarrow} (M_k, D_k)$ be a run of $\mathcal{N}$. From Theorem 1, we have that $[\![(M_k, D_k)]\!]_\pi \in [\![\mathcal{G}(\mathcal{N}(K))]\!]_\pi$ iff $\pi \models D_k{\downarrow_\Lambda}$. Now consider transition $(M_{k-1}, D_{k-1}) \overset{t_{k-1}}{\Rightarrow} (M_k, D_k)$ in $\mathcal{G}(\mathcal{N})$. Then, from the semantics of PITPNs, for all $\pi \models [\![D_k]\!]_\pi$, then $(M_{k-1}, [\![D_{k-1}]\!]_\pi) \overset{t_{k-1}}{\Rightarrow} (M_k, [\![D_k]\!]_\pi) \in \mathcal{G}([\![\mathcal{N}]\!]_\pi)$. The result then derives from a reasoning by induction on k, with $(M, D) = (M_k, D_k)$. □

Conversely, the following lemma states that, given a PITPN $\mathcal{N}$, a run in a valuation of $\mathcal{N}$ always has an equivalent run in $\mathcal{N}$.

Lemma 3. *Let $\mathcal{N}(K)$ be a PITPN, let π be a parameter valuation such that $\pi \models K$. Let r be a run of $[\![\mathcal{N}]\!]_\pi$. Then there exists an equivalent run in $\mathcal{N}(K)$.*

Proof. $[\![\mathcal{N}]\!]_\pi$ can be seen as a PITPN (hence parametric) with an initial constraint K_π. Since $K_\pi \subseteq K$, from the semantics of PITPNs, the set of behaviours of $\mathcal{N}(K)$ includes the behaviours of $\mathcal{N}(K_\pi)$. Hence any run in $\mathcal{N}(K_\pi)$ has an equivalent in $\mathcal{N}(K)$. □

We now state below a general result that will be used to prove Lemma 5.

Lemma 4. *Let $\mathcal{N}(K)$ be a PITPN. Then for all $(M, D) \in \mathcal{G}(\mathcal{N}(K)), D{\downarrow_\Lambda} \subseteq K$.*

Proof. By induction on Lemma 1, with $K_0 \subseteq K$ as the base case. □

The following result states that, for a PITPN with its own covering constraint K_{cov} as initial constraint, the projection onto the parameters of the constraint associated with a reachable class is always the same, and equal to K_{cov}.

Lemma 5. *Let $\mathcal{N}(K)$ be a PITPN, let K_{cov} be the covering constraint of $\mathcal{N}(K)$. Then for all $(M, D) \in \mathcal{G}(\mathcal{N}(K_{cov})) : D{\downarrow_\Lambda} = K_{cov}$.*

Proof. If K_{cov} is empty, $\mathcal{G}$ is empty too and the result trivially holds. Suppose K_{cov} is non-empty. Let $c = (M, D) \in \mathcal{G}(\mathcal{N}(K_{cov}))$. Let $\pi \models D{\downarrow_\Lambda}$. By Lemma 4, $D{\downarrow_\Lambda} \subseteq K_{cov}$. By construction of K_{cov}, we have that $K_{cov} \subseteq K$. Hence $\pi \models D{\downarrow_\Lambda} \Rightarrow \pi \models K$. Since $\pi \models D{\downarrow_\Lambda}$, from Lemma 2, there exists an equivalent run in $[\![\mathcal{N}]\!]_\pi$ reaching class $(M, [\![D]\!]_\pi)$ in $\mathcal{G}([\![\mathcal{N}]\!]_\pi)$. Since $\pi \models K$, from Lemma 3, there exists an equivalent run in $\mathcal{N}(K)$ reaching class (M, D') for some D'.

Let $\pi' \models K_{cov}$. By construction, $K_{cov} \subseteq D'{\downarrow_\Lambda}$, hence $\pi' \models D'{\downarrow_\Lambda}$. By Lemma 4, $D{\downarrow_\Lambda} \subseteq K$, hence $\pi' \models K$. Since $\pi' \models K$ and $\pi' \models D'{\downarrow_\Lambda}$, applying Theorem 1 to $\mathcal{N}(K)$ gives that $[\![c]\!]_{\pi'} \in \mathcal{G}([\![\mathcal{N}]\!]_{\pi'})$. Since $\pi' \models K_{cov}$ by hypothesis, and $[\![c]\!]_{\pi'} \in \mathcal{G}([\![\mathcal{N}]\!]_{\pi'})$, then applying Theorem 1 to $\mathcal{N}(K_{cov})$ gives that $\pi' \models D{\downarrow_\Lambda}$. Hence $K_{cov} \subseteq D{\downarrow_\Lambda}$. (Lemma 4 gives the other direction.) □

Finally, Theorem 2 states that the trace set of a PIPTN valuated with any parameter valuation satisfying its covering constraint K_{cov} is the same as the trace set of this PITPN with K_{cov} as initial constraint.

Theorem 2. *Let $\mathcal{N}$ be a PITPN, let K_{cov} be the covering constraint of $\mathcal{N}$. Let $\pi \models K_{cov}$. Then the trace sets of $\mathcal{N}(K_{cov})$ and $[\![\mathcal{N}]\!]_\pi$ are equal.*

Proof. Let $\pi \models K_{cov}$. Consider a run of $\mathcal{N}(K_{cov})$ reaching a class (M, D) in $\mathcal{G}(\mathcal{N}(K_{cov}))$. By Lemma 5, it holds that $D{\downarrow_\Lambda} = K_{cov}$. Since $\pi \models K_{cov}$, then $\pi \models D{\downarrow_\Lambda}$. Hence, by Lemma 2, there exists an equivalent run in $\mathcal{G}([\![\mathcal{N}]\!]_\pi)$. Conversely, since $\pi \models K_{cov}$, by lemma 3, any run in $[\![\mathcal{N}]\!]_\pi$ has an equivalent run in $\mathcal{N}(K_{cov})$. □

We can derive from Theorem 2 that the trace set of a PIPTN with any parameter valuation satisfying its covering constraint is always the same. This result will be used to prove the correctness of the inverse method (see Section 4).

Corollary 1. *Let $\mathcal{N}$ be a PITPN, let K_{cov} be the covering constraint of $\mathcal{N}$. Then for all $\pi, \pi' \models K_{cov}$, the trace sets of $[\![\mathcal{N}]\!]_\pi$ and $[\![\mathcal{N}]\!]_{\pi'}$ are equal.*

4 The Inverse Method for Time Petri Nets

We extend to PITPNs the inverse method initially proposed for timed automata [4]. The algorithm relies on the following definition of π-compatibility.

Definition 3. *Given a parameter valuation π, a class (M, D) is said to be* π-compatible *if $\pi \models D{\downarrow_\Lambda}$, and π-incompatible otherwise.*

4.1 Principle

We introduce in Algorithm 1 *IMPN* (i.e. the Inverse Method for time Petri Nets with inhibitor arcs). It uses 3 variables: an integer i measuring the depth of the state space exploration, the current constraint K_c, and the set C of explored classes. Starting from the initial class c_0, *IMPN* iteratively computes classes. When a π-incompatible class is found, an incompatible inequality is non-deterministically selected within the projection of the constraint onto Λ (line 5);

Algorithm 1. *IMPN*$(\mathcal{N}, \pi)$

input : PITPN $\mathcal{N}$ of initial class c_0 and initial constraint K_0, valuation π
output: Constraint K_r

1 $i \leftarrow 0$; $K_c \leftarrow K_0$; $C \leftarrow \{c_0\}$
2 **while** `true` **do**
3 **while** $\exists$ *π-incompatible classes in C* **do**
4 Select a π-incompatible class (M, D) of C
5 Select a π-incompatible J in $D{\downarrow_\Lambda}$
6 $K_c \leftarrow K_c \wedge \neg J$; $C \leftarrow \bigcup_{j=0}^{i} Post^j_{\mathcal{N}(K_c)}(\{c_0\})$
7 **if** $Post_{\mathcal{N}(K_c)}(C) \subseteq C$ **then return** $K_r \leftarrow \bigcap_{(M,D)\in C} D{\downarrow_\Lambda}$
8 $i \leftarrow i + 1$; $C \leftarrow C \cup Post_{\mathcal{N}(K_c)}(C)$

its negation is then added to K_c (line 6). The set of reachable classes is then updated. When all successor classes have already been reached (line 7), *IMPN* returns the intersection K_r of the projection onto Λ of the constraints associated with all the reachable classes.

4.2 Results

Lemma 6. *Let $\mathcal{N}$ be a PITPN, and π be a parameter valuation. Suppose that algorithm IMPN$(\mathcal{N}, \pi)$ terminates with output K_r. It holds that $\pi \models K_r$.*

Proof. By construction, at the end of the inner **while** loop, all classes of C are π-compatible, that is for all $(M, D) \in C, \pi \models D$. As a consequence, $\pi \models D{\downarrow_\Lambda}$. Recall that $K_r = \bigcap_{(M,D)\in C} D{\downarrow_\Lambda}$. Hence $\pi \models K_r$. □

The correctness of *IMPN* mainly relies on the fact that K_r is the covering constraint of $\mathcal{N}$. Hence, the results of Section 3 can be applied.

Theorem 3 (Correctness). *Let $\mathcal{N}$ be a PITPN, and π be a parameter valuation. Suppose IMPN$(\mathcal{N}, \pi)$ terminates with output K_r. Then:*

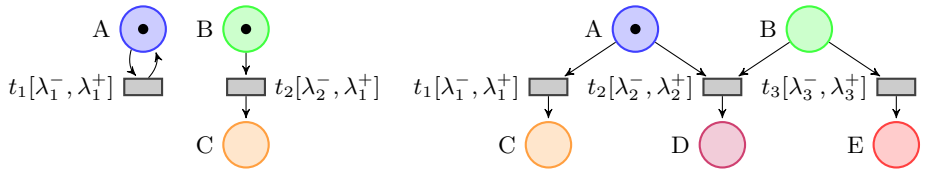
1. *$\pi \models K_r$, and*
2. *$\forall \pi' \models K_r$, $[\![\mathcal{N}]\!]_{\pi'}$ and $[\![\mathcal{N}]\!]_\pi$ have the same trace set.*

Proof. Item 1 comes from Lemma 6. For item 2, since K_r is the covering constraint of $\mathcal{N}$, then we can apply Corollary 1, which gives the result. Also note that the covering constraint cannot be empty since $\pi \models K_r$. □

Non-termination. Parameter synthesis is undecidable for PITPNs [13] and *IMPN* may not always terminate. Consider the PITPN $\mathcal{N}$ in Fig. 3a; then, *IMPN* applied to $\mathcal{N}$ and a reference valuation with all parameters equal to 0 will generate an infinite set of classes with constraints of the form $i * \lambda_1^- \leq \lambda_2^+$, with i infinitely growing. Intuitively, t_1 can fire an arbitrary number of times before t_2 fires. Of course, this is a typical Zeno-behaviour (an infinite number of transitions within a null duration) and, in the case of non-null reference parameter valuations, an inequality $i * \lambda_1^- \leq \lambda_2^+$ will eventually be π-incompatible, thus ensuring termination. Also note $\mathcal{N}$ is a bounded L/U (lower/upper bounds) PTPN [13], showing that termination of *IMPN* is not guaranteed for general bounded L/U PTPNs (although emptiness and reachability problems are decidable in theory). Studying the decidability of this problem, and adapting *IMPN* to ensure termination in this case is the subject of ongoing work.

We can exhibit subclasses for which *IMPN* terminates. This is obviously the case of loopless PITPNs (in which no syntactical loop exists in the model). This is also the case of parametric sequential TPNs [2]; this subclass of TPNs is such that each time a discrete transition is fired, each transition that is enabled in the new/resulting marking is newly enabled. Hence, the problem of infinitely concurrent loops such as in Fig. 3a cannot happen.

Non-confluence and Non-completeness. Due to the non-deterministic selection of an inequality, *IMPN* is non-confluent (i.e. different applications of the



(a) Non-termination (b) Non-completeness

Fig. 3. Counter-examples PITPNs

algorithm can yield different outputs). As a consequence, it is also non-complete (i.e. the resulting constraint may not be the maximal one). Formally:

Proposition 1 (Non-completeness). *There may exist* $\pi' \not\models K_r$ *such that* $[\![\mathcal{N}]\!]_{\pi'}$ *and* $[\![\mathcal{N}]\!]_{\pi}$ *have the same trace set.*

An example for non-completeness is the PITPN $\mathcal{N}$ in Fig. 3b, with the reference parameter valuation $\pi = \{\lambda_1^- \to 5, \lambda_1^+ \to 6, \lambda_2^- \to 1, \lambda_2^+ \to 3, \lambda_3^- \to 2, \lambda_3^+ \to 4\}$. In $[\![\mathcal{N}]\!]_{\pi}$, either t_2 or t_3 can fire first, but not t_1, due to the fact that we have both ${}^{\uparrow}I(t_1) > I^{\uparrow}(t_2)$ and ${}^{\uparrow}I(t_1) > I^{\uparrow}(t_3)$.

When applying the inverse method to $\mathcal{N}$ and π, a class will be generated with BC as a marking, and an associated constraint containing in particular inequalities $\lambda_1^- \leq \lambda_2^+ \wedge \lambda_1^- \leq \lambda_3^+$. Since both are π-incompatible, the algorithm can add to the current constraint either $\lambda_1^- > \lambda_2^+$ or $\lambda_1^- > \lambda_3^+$. Either of them is sufficient to prevent BC to be reachable. Then the result of the application of *IMPN* to $\mathcal{N}$ and π is both non-confluent and non-complete. Also note that, due to the absence of inhibitor arc in $\mathcal{N}$, the non-completeness of *IMPN* also holds for PTPNs.

Nevertheless, it can be shown (as it was the case for timed automata [4]) that a sufficient (but non-necessary) condition for completeness is that *IMPN* does not perform non-deterministic selections of inequalities, i.e. at most one π-incompatible class is met at each iteration.

5 Precise Robustness Analysis

5.1 Local Robustness

Throughout this section, we assume an ITPN N, as well as a parameterised version $\mathcal{N}$ of N where each lower (resp. upper) bound of a transition t_i is replaced with a fresh parameter λ_i^- (resp. λ_i^+). Let π be the reference valuation such that $[\![\mathcal{N}]\!]_{\pi} = N$. We assume that $\mathit{IMPN}(\mathcal{N}, \pi)$ terminates with output K_r.

We will exploit K_r to characterise the precise robustness of the system, i.e. the admissible variability of each timing bound. The original trace set is preserved by any valuation satisfying K_r. Hence, any linear-time (LTL) property that is true in $[\![\mathcal{N}]\!]_{\pi}$ is also true in $[\![\mathcal{N}]\!]_{\pi'}$, for $\pi' \models K_r$. Thus, if the correctness is given in the form of an LTL property, the timing delays can safely vary as long as they satisfy K_r.

We use here several examples in order to better illustrate the notions. For the PITPN in Fig. 2a, with $\pi = \{\lambda_1^- \to 5, \lambda_1^+ \to 6, \lambda_2^- \to 3, \lambda_2^+ \to 4, \lambda_3^- \to 1, \lambda_3^+ \to 2\}$ as a reference valuation, *IMPN* outputs the constraint $K_r = \lambda_1^- \leq \lambda_1^+ \wedge \lambda_2^- \leq \lambda_2^+ \wedge \lambda_3^- \leq \lambda_3^+ \wedge \lambda_3^+ < \lambda_1^-$. For a parameterised version of the ITPN in Fig. 1a, *IMPN* outputs the constraint $K_r = \lambda_1^- \leq \lambda_1^+ \wedge \lambda_2^- \leq \lambda_2^+ \wedge \lambda_2^- \geq \lambda_1^+$. For a parameterised version of the ITPN in Fig. 1b, *IMPN* outputs the constraint $K_r = \lambda_1^- \leq \lambda_1^+ \wedge \lambda_2^- \leq \lambda_2^+ \wedge \lambda_3^- = 0 \wedge 0 \leq \lambda_3^+ \wedge \lambda_1^+ = \lambda_2^-$.

Definition 4. *An ITPN N is* robust with respect to linear-time properties *(or* LT-robust*) if there exists $\gamma > 0$ such that for any linear time property φ, $N' \models \varphi$ if and only if $N \models \varphi$, where N' is an ITPN similar to N where each timing bound c can be replaced with any value within $[c - \gamma, c + \gamma]$.*

For example, the ITPN in Fig. 2a is LT-robust (with e.g. $\gamma = 1$), whereas the ITPNs in Fig. 1 are not.

Local Robustness. The resulting constraint K_r is given in the form of a convex (possibly unbounded) polyhedron. For each interval bound λ_i in $\mathcal{N}$, its *local robustness* $LR(\lambda_i)$ is defined as the distance between $\pi(\lambda_i)$ and the closest border of the polyhedral representation of K_r. For example, in Fig. 2a, $LR(\lambda_1^-) = 1$. In Fig. 1a, $LR(\lambda_1^-) = 1$ whereas $LR(\lambda_1^+) = 0$, showing that this latter bound renders the system non-robust. The following lemma follows from Definition 4, from the definition of LR and the correctness of *IMPN*.

Lemma 7. *If for each parameter λ in $\mathcal{N}$, $LR(\lambda) > 0$, then N is LT-robust.*

Ranging Interval. For each interval bound λ_i in $\mathcal{N}$, its *ranging interval* $RI(\lambda_i)$ is defined as its minimum and maximum admissible values within K_r. It is computed by valuating all parameters but λ_i in K_r, and converting the resulting inequality in the form of an interval. For example, in Fig. 2a, $RI(\lambda_1^-) = (2, 6]$. In Fig. 1a, $RI(\lambda_1^+) = [1, 2]$.

The local lower (resp. upper) variability is defined as the distance between the parameter valuation and the lower (resp. upper) bound of RI; formally, given $RI(\lambda_i) = (a, b)$, $LLV(\lambda_i) = \pi(\lambda_i) - a$ and $LUV(\lambda_i) = b - \pi(\lambda_i)$. Note that the local robustness can be obtained from the local variability: $LR(\lambda_i) = \min(LLV(\lambda_i), LUV(\lambda_i))$.

Computation of Δ. Our approach also allows to retrieve the value of the "Δ" of Δ-based approaches. It is defined as the minimum over the set of parameters of the distance between a parameter and the closest border of the polyhedron. Formally, $\Delta = \min\left(\min_{i \in \Delta^-} LLV(\lambda_i), \min_{i \in \Delta^+} LUV(\lambda_i)\right)$, where Δ^- (resp. Δ^+) denotes the set of parameters appearing in an interval lower (resp. upper) bound. This distinction is necessary, since Δ-based approaches only consider the positive enlarging of intervals. For the ITPN in Fig. 2a, the maximum possible Δ is 1.5 (see Section 5.3). And, obviously, $\Delta = 0$ for the ITPNs in Fig. 1.

5.2 Improving the System Robustness

Identifying Critical Timing Bounds. Our approach allows to exhibit *critical* timing bounds: critical timing bounds are those rendering the system non-robust, i.e. with a null local robustness. For example, in Fig. 1a, λ_1^+ and λ_2^- are the critical timing bounds. In Fig. 1b, λ_1^+, λ_2^- and λ_3^- are the critical timing bounds.

Relaxing Bounds. For some systems, it is possible to refine the values of the critical timing bounds so that the system becomes robust, with the same discrete behaviour. In practice, this may in particular be the case of hardware systems, where the timing bounds come from the traversal time of micro components: One can change the timing bounds by replacing a component with another one. In software, one can also refine the values of some timers if needed.

In that case, one can exploit the precise robustness analysis to synthesise values for the timing bounds so that the system is robust. A system is said to be potentially robust if all timing bounds λ_i have a ranging interval non-reduced to a point (even if their local robustness may possibly be null, i.e. $LR(\lambda_i) = 0$).

Definition 5. *An ITPN N is* potentially robust *if, for all timing bounds λ_i, $LLV(\lambda_i) \neq LUV(\lambda_i)$.*

This notion of potential robustness is a sufficient condition so that an ITPN becomes robust with the same discrete behaviour.

Theorem 4. *If N is potentially robust, then there exists π_R such that $[\![\mathcal{N}]\!]_{\pi_R}$ is LT-robust, and has the same trace set as N.*

Proof. By Lemma 7, only the timing bounds λ_i such that $LR(\lambda_i) = 0$ render $\mathcal{N}$ non-LT-robust. For all λ_i such that $LR(\lambda_i) > 0$, we set $\pi_R(\lambda_i) = \pi(\lambda_i)$. Now consider a λ_i such that $LR(\lambda_i) = 0$. By definition of LR, either $LLV(\lambda_i) = 0$ or $LUV(\lambda_i) = 0$. Consider the former case (the latter case is dual). Let $\pi_R(\lambda_i) = \pi(\lambda_i) + \big(LLV(\lambda_i) + LUV(\lambda_i)\big)/2$. Since $\mathcal{N}$ is potentially robust, $LLV(\lambda_i) \neq LUV(\lambda_i)$; hence $LLV(\lambda_i) < \pi_R(\lambda_i) < LUV(\lambda_i)$. As a consequence, in π_R, we have $LR(\lambda_i) > 0$. By construction, and from the convexity of K_r, $\pi_R(\lambda_i)$ is in K_r; hence, from Theorem 3, $[\![\mathcal{N}]\!]_{\pi_R}$ and $[\![\mathcal{N}]\!]_{\pi}$ have the same trace set. □

Note that this is a sufficient but non-necessary condition, since the notion of potential robustness is based on LLV and LUV, that come from K_r, which is non-complete. Furthermore, one can find further conditions (and constructions) to render a system robust. For example, the ITPN in Fig. 1b is not potentially robust; but it can be made robust with the same discrete behaviour, e.g. by replacing the intervals associated with both t_1 and t_2 with $[0, 1]$.

5.3 Comparison with Δ-Based Approaches

The main drawback of our approach is that it does not terminate in the general case, although we exhibited cases for which termination is guaranteed (see

(a) Representation of K_r (b) Representation of the Δ-cube

Fig. 4. Graphical comparison for the example in Fig. 2a

Section 4.2). In contrast, related work show that deciding only whether a system is robust is decidable in most cases. However, beside the fact that we give a quantitative measure of the robustness in the form of a constraint in $2*n$ dimensions (with n the number of transitions), our approach is particularly interesting in the case of a non-robust system. First, we exhibit which timing bounds are responsible for the non-robustness. Second, we give a condition to render the system robust without changing its discrete behaviour.

Furthermore, our approach may output a significantly larger constraint than the Δ-cube output by Δ-based approaches. Actually, when the result of *IMPN* is complete, the resulting polyhedron is necessarily at least as large as the Δ-cube. Consider again the example in Fig. 2a. In order to enable a graphical comparison in 2 dimensions, we assign all parameters but λ_1^- and λ_3^+ to their value as in π. Hence the constraint becomes $\lambda_1^- \leq 6 \wedge 1 \leq \lambda_3^+ \wedge \lambda_3^+ < \lambda_1^-$. This constraint is depicted in Fig. 4a. As of Δ-based approaches, they cannot compute a value for Δ greater than 1.5 in this situation. Indeed, with $\Delta = 1.5$, λ_3^+ becomes $\lambda_3^+ + \Delta = 3.5$, λ_1^- becomes $\lambda_1^- - \Delta = 3.5$, in which case the discrete behaviour becomes different (t_1 can fire before t_3). This Δ is given in Fig. 4b.

The interpretation of the much larger parametric domain covered by K_r compared to the Δ-cube can be explained as follows: (1) The parametric domain below $\lambda_3^+ = 2$ and above $\lambda_1^- = 5$ is not covered by the Δ-cube, because Δ-based approaches consider a positive parameter $\Delta \geq 0$. Hence, it is not possible to study, e.g. by how much an upper bound can be decreased. (2) The constraint K_r allows to relate parameters. Whereas the value of Δ prevents λ_1^- and λ_3^+ to vary by more than 1.5, the inequality $\lambda_3^+ < \lambda_1^-$ states that λ_1^- may vary by more than 1.5, as long as λ_3^+ varies less (i.e. $\lambda_3^+ < \lambda_1^-$). This is of particular interest in systems where some bounds are more likely to vary than others. (3) This small example is a "good" example for Δ-based approaches. In the case where at least one parameter cannot vary, Δ would be inevitably equal to 0, whereas K_r would still give an output for other dimensions. This is the case of the ITPNs in Fig. 1.

6 Final Remarks

In this paper, we extended the inverse method to PITPNs and showed how to exploit its output to obtain a precise quantitative measure of the system

robustness for linear-time properties. This paper considers the quantification of the system robustness with respect to linear-time (hence time-abstract) properties only. Nevertheless, timed properties can also be considered, by adding an observer net. This observer synchronises with the system ITPN, and can reduce timed properties to time-abstract properties.

Our algorithms should be implemented and compared with similar tools, such as Shrinktech [11]. Finally, we only addressed here the variability of the timing delays (Δ), but not the admissible variations of the clock speed (usually called "ϵ"). Our approach could be extended to this setting using extensions of the inverse method for parameterised hybrid systems [7], by adding for each clock two additional parameters ϵ_i^- and ϵ_i^+ measuring the admissible decrease and increase speed rate.

Acknowledgment. We are grateful to an anonymous reviewer for his/her very detailed comments.

References

1. Akshay, S., Hélouët, L., Jard, C., Lime, D., Roux, O.H.: Robustness of time petri nets under architectural constraints. In: Jurdziński, M., Ničković, D. (eds.) FORMATS 2012. LNCS, vol. 7595, pp. 11–26. Springer, Heidelberg (2012)
2. Akshay, S., Hélouët, L., Jard, C., Reynier, P.-A.: Robustness of time Petri nets under guard enlargement. In: Finkel, A., Leroux, J., Potapov, I. (eds.) RP 2012. LNCS, vol. 7550, pp. 92–106. Springer, Heidelberg (2012)
3. Alur, R., Dill, D.L.: A theory of timed automata. Theoretical Computer Science 126(2), 183–235 (1994)
4. André, É., Soulat, R.: The Inverse Method. FOCUS Series in Computer Engineering and Information Technology. ISTE Ltd and John Wiley & Sons Inc. (2013)
5. Bouyer, P., Larsen, K.G., Markey, N., Sankur, O., Thrane, C.: Timed automata can always be made implementable. In: Katoen, J.-P., König, B. (eds.) CONCUR 2011. LNCS, vol. 6901, pp. 76–91. Springer, Heidelberg (2011)
6. Bouyer, P., Markey, N., Sankur, O.: Robust reachability in timed automata: A game-based approach. In: Czumaj, A., Mehlhorn, K., Pitts, A., Wattenhofer, R. (eds.) ICALP 2012, Part II. LNCS, vol. 7392, pp. 128–140. Springer, Heidelberg (2012)
7. Fribourg, L., Kühne, U.: Parametric verification and test coverage for hybrid automata using the inverse method. IJFCS 24(2), 233–249 (2013)
8. Jaubert, R., Reynier, P.-A.: Quantitative robustness analysis of flat timed automata. In: Hofmann, M. (ed.) FOSSACS 2011. LNCS, vol. 6604, pp. 229–244. Springer, Heidelberg (2011)
9. Markey, N.: Robustness in real-time systems. In: SIES, pp. 28–34. IEEE Computer Society Press (2011)
10. Merlin, P.M.: A study of the recoverability of computing systems. PhD thesis, University of California, Irvine, CA, USA (1974)
11. Sankur, O.: Shrinktech: A tool for the robustness analysis of timed automata. In: CAV. LNCS. Springer (to appear, 2013)
12. Traonouez, L.-M.: A parametric counterexample refinement approach for robust timed specifications. In: FIT. EPTCS, vol. 87, pp. 17–33 (2012)
13. Traonouez, L.-M., Lime, D., Roux, O.H.: Parametric model-checking of stopwatch Petri nets. Journal of Universal Computer Science 15(17), 3273–3304 (2009)