Robustness Analysis of Time Petri Nets

Étienne André$^1$ and Shweta Garg$^2$

$^1$ Université Paris 13, Sorbonne Paris Cité, LIPN, France
Étienne.Andre@lipn.univ-paris13.fr

$^2$ Dept. of Computer Science, IIT Bombay, Mumbai, India
ShwetaGarg@cse.iitb.ac.in

Abstract

Given a parametric time Petri net with inhibitor arcs and a valuation of the timing requirements seen as parameters, we propose a method synthesizing a constraint on these parameters guaranteeing the same set of traces as for the reference valuation. This gives a quantitative measure of the robustness of the system for linear time properties.

1 Introduction

Real-time concurrent systems are often characterized with a set of timing requirements. Even when the correctness for a given set of such requirements has been proved (e.g., using UPPAAL [LPY97]), the corresponding implementation may turn incorrect because the requirements in practice may be slightly different. For example, a system where some component performs an action for, say, 2 seconds can be implemented with a time greater but very close to 2 (say, 2.0001 s), in which case the formal guarantee may not hold anymore.

Many approaches in the literature (see [Mar11] for a survey) consider that the clock (or time) is varying, and measure the robustness of the system w.r.t. this varying clock. Here, we consider that the timing requirements of the system may be subject to some drift in practice. We use the formalism of PITPNs (parametric time Petri nets with inhibitor arcs, see, e.g., [TLR09]): this formalism allows one to reason in a parametric manner, by considering that the requirements are parameters (unknown constants). We extend to PITPNs the inverse method (initially defined in the setting of parametric timed automata [ACEF09]); this algorithm synthesizes a set of constraints on the parameters, which gives a quantitative measure of the robustness of a time Petri net with inhibitor arcs.

2 Parametric Time Petri Nets with Inhibitor Arcs

We mostly use here the definition of [TLR09], where a PITPN $P$ is a standard Petri net extended with firing times of the form $[a,b]$ where $a, b$ are taken from a set of temporal parameters $P$, that are unknown constant taking their values among rationals. A firing time $[a,b]$ means that the transition must be fired at least $a$ and at most $b$ units of time after it is enabled, i.e., after enough tokens are available in the input places. PITPNs also feature inhibitor arcs, that disable a transition when a token is present. This notion is somehow similar to stopwatches, since it blocks the elapsing of time, from the transition point of view. $K_{init}$ is a constraint on $P$ giving the initial domain of the parameters, and must at least specify that the minimum bounds of the firing intervals are

![Figure 1: A PITPN](image-url)
inferior to the maximum bounds. Figure 1 shows an example of PITPN, where the firing times of $t_1$, $t_2$ and $t_3$ are $[a, b]$, $[c, d]$, and $[e, f]$, respectively. Transition $t_2$ is inhibited by $A$.

Given a set $X$ of local firing times of transitions and a set $P$ of parameters, a constraint $D$ over $X$ and $P$ is a conjunction of linear inequalities on $X$ and $P$. We denote by $D_{\downarrow P}$ the constraint over $P$ obtained from $D$ after elimination of the firing times.

The reachable states of a PITPN are parametric state-classes $c$, i.e., pairs $(M, D)$ where $M$ is a marking of the net and $D$ is a firing domain, that is, a constraint over $X$ and $P$. We consider a (classical) semantics where transitions must fire if they can; for example, in Figure 1, $t_1$ must fire before $t_3$ if $b < e$, $t_3$ must fire before $t_1$ if $f < a$, and both orders are possible otherwise.

Given a valuation $\pi$, a class $(M, D)$ is said to be $\pi$-compatible if $\pi \models D_{\downarrow P}$, and $\pi$-incompatible otherwise.

Given a run $r$ of $P$ of the form $(M_0, D_0) \xrightarrow{a_0} \cdots \xrightarrow{a_{n-1}} (M_n, D_n)$, the trace associated with $r$ is the alternating sequence of markings and actions $M_0 \Rightarrow a_0 \Rightarrow \cdots \Rightarrow a_{n-1} M_n$. The trace set of $P$ is the set of all traces associated with the runs of $P$. $\text{Post}_{P(K)}(C)$ (resp. $\text{Post}^i_{P(K)}(C)$) is the set of classes reachable from a set $C$ of classes in exactly one step (resp. $i$ steps) under constraint $K$.

Given a PITPN $P$ and a valuation $\pi$, we denote by $P[\pi]$ the ITPN where each occurrence of a parameter has been replaced by its constant value as in $\pi$.

3 The Inverse Method for Time Petri Nets

3.1 Principle

We introduce in Algorithm 1 the inverse method $IM$ for PITPNs. Given a PITPN $P$ and a reference parameter valuation $\pi_0$, $IM$ synthesizes a constraint $K_0$ on $P$ such that, for all $\pi \models K_0$, $P[\pi_0]$ and $P[\pi]$ have the same trace sets. Starting from the initial class $c_0$, $IM$ iteratively computes state classes. When a $\pi_0$-incompatible class is found, an incompatible inequality is nondeterministically selected within the projection of the constraint onto $P$ (line 5); its negation is then added to $K$ (line 6). The set of reachable states is then updated. When all successor classes have been reached before (line 7), $IM$ returns the intersection of the projection onto the parameters of the constraints associated with all the reachable classes.

3.2 Application to a Simple Example

Let us apply $IM$ to the PITPN of Figure 1 with a $\pi_0$ where $a, b, c, d, e, f$ take as values $5, 6, 3, 4, 1, 2$ respectively. We have $K_{\text{init}} = a \leq b \land c \leq d \land e \leq f$. For sake of conciseness, we always project $D$ onto $P$ (hence we omit the firing times in $X$). The initial class is $c_0 = \{AB, K_{\text{init}}\}$, where $AB$ denotes that places $A$ and $B$ both contain a token. From $c_0$, one can reach $c_1 = \{AE, K_{\text{init}} \land b \geq e\}$, and a class $(CB, K_{\text{init}} \land f \geq a)$. The former is $\pi_0$-compatible, but not the latter because $f \geq a$ is not true in $\pi_0 (2 \geq 5)$. Hence, we add $f < a$ to $K$, and go on. From $\{c_0, c_1\}$, one can reach $c_2 = \{CE, K_{\text{init}} \land b \geq e \land f < a\}$, which is $\pi_0$-compatible. Then $IM$ reaches a fixpoint and returns $K_0 = K_{\text{init}} \land b \geq e \land f < a$.
For any parameter valuation satisfying $K_0$, the trace set (actually reduced to a single trace) is the one given in Figure 2. Beyond the application to robustness or optimization of some constants, the resulting constraint also gives valuable information about the relationship between the timing requirements. For example, $c$ and $d$ do not appear within $K_0$ (apart from $c \leq d$): this can be explained as follows. Initially, the token in $A$ inhibits $t_2$; then, $t_3$ must fire first (because $f < a$) and, when $t_1$ finally fires (thus disinhibiting $t_2$), there is no token anymore in $B$; hence $t_2$ can never fire if $f < a$, whatever are the values of $c$ and $d$.

3.3 Results

Theorem 1 (Correctness). Let $P$ be a PITPN, and $\pi_0$ be a reference valuation. Let $K_0 = IM(P, \pi_0)$. Then: (1) $\pi_0 \models K_0$ and (2) $\forall \pi \models K_0$, $P[\pi]$ and $P[\pi_0]$ have the same trace set.

This result allows one to quantify the robustness of the system: the equality of trace sets implies that any linear-time (LTL) property that is true in $P[\pi_0]$ is also true for $P[\pi]$. Hence, if the correctness is given under the form of an LTL property, the timing requirements can safely vary as long as they satisfy $K_0$. Observe that timed properties (involving quantitative information) can also be modeled as properties on traces, by adding an observer to the system.

Due to the non-deterministic selection of an inequality, $IM$ is non-confluent. As a consequence, it is also non-complete. Formally:

Proposition 1. There may exist $\pi \not\models K_0$ such that $P[\pi]$ and $P[\pi_0]$ have the same trace set.

Future Work. The inverse method is not guaranteed to terminate in the setting of parametric timed automata. However, the counter-example used in [ACEF09] cannot be used for PITPNs; proving (non-)termination is the subject of ongoing work. An implementation (for instance in IMITATOR [AFKS12]) should be performed. We also plan to extend the method to other classes of Petri nets, such as timed extensions of colored Petri nets [JK09]. Furthermore, this work could be combined with the modular state space exploration for timed Petri nets [LP07].

References


