

Time Petri Nets with Inhibitor Hyperarcs. Formal Semantics and State Space Computation

Olivier (H.) ROUX and Didier LIME

IRCCyN (Institut de Recherche en Communication et Cybernétique de Nantes)
1, rue de la Noë B.P. 92101
44321 NANTES cedex 3 (France)
{Didier.Lime | Olivier-h.Roux}@irccyn.ec-nantes.fr

Abstract. In this paper, we define Time Petri Nets with Inhibitor Hyperarcs (IHTPN) as an extension of T-time Petri nets where time is associated with transitions. In this model, we consider stopwatches associated with transitions which can be reset, stopped and started by using classical arcs and branch inhibitor hyperarcs introduced by Janicki and Koutny [1]. We give a formal semantics for IHTPNs in terms of Timed Transition Systems and we position IHTPNs with regard to other classes of Petri nets in terms of timed language acceptance. We provide a method for computing the state space of IHTPNs. We first propose an exact computation using a general polyhedron representation of time constraints, then we propose an overapproximation of the polyhedra to allow a more efficient compact abstract representations of the state space based on DBM (Difference Bound Matrix).

Key words: Time Petri nets, inhibitor hyperarc, state space, semantics, real-time systems

1 Introduction

The theory of Time Petri Nets provides a general framework to specify the behavior of reactive and real-time systems. In such applications, it is necessary to model the durations of actions in the form of a time interval and it is sometimes necessary to memorize the progress status of an action when this one is suspended then resumed.

In this class of models, some extensions of time Petri nets have been proposed to model the preemptive scheduling of tasks. Okawa and Yoneda [2] propose an approach with time Petri nets consisting of defining groups of transitions together with rates (speeds) of execution. Roux and Déplanche [3] propose an extension for time Petri nets (scheduling-TPN) that consists of mapping into the Petri net model the way the different schedulers of the system activate or suspend the tasks. For a fixed priority scheduling policy, scheduling-TPN introduce two new attributes associated to each place that respectively represent allocation (processor or resource) and priority (of the modeled task). [4] propose a similar

model: Preemptive Time Petri Net (preemptive-TPN). The two new attributes are associated to transitions instead of places.

However all these models are dedicated to the scheduling problem while stopping then resuming an action is a more general consideration. For example scheduling-TPN and preemptive-TPN are unable to model a circular priority relation which can definitively block the time flow of some transitions while no concurrent transition runs (i.e. the required resources are free). These "timed deadlock" are real and concrete problems in industrial applications, it may thus be necessary to have a model able to express them in order to detect them.

We propose to define a general T-timed extension of Petri nets with the concept of stopwatch. In this model, stopwatches are associated with transitions which can be reset, stopped and started by using classical arcs and branch inhibitor hyperarcs introduced by Janicki and Koutny [1].

We will first discuss timed extensions of Petri nets and then the concept of branch inhibitor hyperarc.

Time Petri nets

The two main timed extensions of Petri Nets are Time Petri Nets (TPN) [5] and Timed Petri Nets [6]. While a transition can be fired within a given interval for TPN, in Timed Petri Nets, temporisation represents the minimal duration firing of transitions (or exact duration firing with an "as soon as possible" firing rule). There are also numerous way of representing time. It could be relative to places, transitions, arcs or tokens. The classes of Timed Petri Nets (time relative to place, transition. . .) are included in the corresponding classes of Time Petri Nets [7] TPN are mainly divided in P-TPN [8], A-TPN [9, 10] and T-TPN [11] where a time interval is relative respectively to places, arcs and transitions. Finally, Time Stream Petri Nets [12] were introduced to model multimedia applications. Since our objective is to use inhibitor arcs to stop the stopwatches associated with the transitions, we will use the T-TPN model.

It is shown in [13] that Petri nets with inhibitor arcs can be described by T-time Petri nets. Time-transitions are then used to model inhibitions. T-TPN do not make it possible to model at once a coherent flow of time and inhibitions.

Priorities, inhibitor arcs and branch inhibitor hyperarcs

A priority net introduced by [14] is a marked net with a binary priority relation on transitions. The priority relation is assumed to be irreflexive and antisymmetric. Moreover, priority is specified only for transitions which are in a local conflict. Petri Nets with inhibitor arcs were introduced by Agerwala [15]. The standard execution rule for inhibitor arcs says that an inhibitor arc between a condition (place) p and an event (transition) t means that t can only be fired if p is unmarked. Several papers aimed at modeling priorities with inhibitor arcs. This can be achieved by using labeled nets [14] where a transition with a lower priority is replaced by a set of transitions with the same label, or by using "invisible transitions" [16]. [1] show with a counterexample that it is impossible to

have a strong (not only isomorphic, but even identical reachability graphs) translation from priority nets to ordinary inhibitor nets. To cope with this problem [1] introduced branch inhibitor hyperarcs. A branch inhibitor hyperarc between places p_1, \dots, p_k and a transition t means that t is not fireable if all the places p_1, \dots, p_k are marked. However, in their example, ordinary inhibitor arcs could be used by adding a single place to the net. Branch inhibitor *hyperarcs* do not increase the expressivity of the model compared to simple inhibitor arcs but nonetheless greatly improve the convenience of the modeling.

Simultaneity and inhibitor arcs

There seems to be no general agreement on an exact definition of a simultaneous step for nets with inhibitor arcs : can a net fire simultaneously two transitions a and b even if the firing of the one (a) inhibits the other (b) ? Both semantics can be found in the literature : models in which the simultaneous firing of $\{a, b\}$ is allowed is often called the " *a priori* concurrent semantics" [17, 1] and those which disallow $\{a, b\}$ the " *a posteriori* concurrent semantics" [18]. Both types are considered in [16, 19]. To be coherent with classical time Petri nets semantics, we will consider in this article a model which does not allow the simultaneous firing of two transitions.

Related works

Cassez and Larsen [20] prove that stopwatch automata (SWA) with unobservable delays are as expressive as linear hybrid automata (LHA) in the sense that any timed language accepted by a LHA is also acceptable by a SWA. The reachability problem is undecidable for LHA and also undecidable for SWA. For time Petri nets as for Petri nets with inhibitor arcs, (and *a fortiori* for time Petri nets with inhibitor arcs (and hyperarcs)) boundedness of marking is undecidable [13, 14], and works on these models report undecidability results, or decidability under the assumption that the Petri net is bounded (as for reachability decidability of Time Petri Nets [21]). Boundedness and other results are obtained by computing the state space.

For dense-time models such as time Petri nets or timed automata, the state space is infinite because of the real-valued clocks; but it is possible to represent the infinite state-space by a finite partitioning in the state class graph or the region graph.

The mainstream approach to compute the state space of T-TPN is the state class graph [13, 11]. The nodes of the state class graph are sets of states (a state class is a pair consisting of a marking and a firing constraint) of the TPN and the edges are labeled with transitions names. If L is the language accepted by the state class graph and L' is the untimed language accepted by the TPN, then $L = L'$. An alternative approach has been proposed by Yoneda [22] in the form of an extension of equivalence classes which allows Computation Tree Logic (CTL) model-checking. Lilius [23] refined this approach so that it becomes possible to apply partial order reduction techniques that have been developed for untimed

systems. Berthomieu and Vernadat [24] propose an alternative construction of the graph of atomic classes of Yoneda applicable to a larger class of nets. The state class method is also improved in [25] to produce a timed automaton which is timed bisimilar to the TPN, allowing TCTL model-checking. Lastly, Gardey and Roux [26] propose to extend the zone based forward algorithm of timed automata to the computation of the state space of time Petri nets. This algorithm differs from the state class graph computation by using the future of a zone instead of shifting the origin of time.

In all these approaches, temporal domains are expressed as a set of linear inequations in the form of a Difference Bound Matrix (DBM). The form of a DBM is not sufficient to express the temporal domain with the presence of stopwatches. For stopwatch automata (SWA) Cassez and Larsen [20] propose an approximate reachability algorithm obtained as a rather immediate extension of the existing reachability algorithm for timed automata. This extension consists of using DBM by the redefinition of the *future* of a zone (DBM). It yields an overapproximation of the reachable state space. In the same way, in the context of scheduling analysis, in [3, 27, 4] authors propose an overapproximation of the state space by using DBM.

Our contribution

We propose to define a T-timed extension of Petri nets with the concept of stopwatch. The stopwatches are associated with the transitions which can be reset, stopped and started by using classical arcs and branch inhibitor hyperarcs introduced by Janicki and Koutny [1].

We give in section 2 the formal semantics of this model in the form of a timed transition system. We give in section 3 the main properties concerning this model (relation with other classes of Petri nets, decidability of boundedness and reachability problems...). We propose in section 4 an exact state space computation (based on the state class graph method) and also a more efficient computation method (at the cost of an overapproximation) that uses compact abstract representations of the state space based on DBM (Difference Bound Matrix). Finally in section 5, we show how to check timed reachability properties on this model by using the classical notion of observer.

2 Time Petri nets with branch Inhibitor Hyperarcs

2.1 Informal presentation

We now present our T-time extension of branch inhibitor nets. We first recall, on examples, the models on which this extension is based *i.e.* branch inhibitor nets with *a posteriori* semantics and time Petri nets.

Two examples of branch inhibitor nets are presented in figure 1. The first net (a) has two inhibitor arcs (*i.e.* two simple hyperarcs) and the second (b) has one hyperarc. The net (a) imposes the firing of transitions t_1 and t_2 before the

firing of the transition t_3 whereas the net (b) imposes only the firing of one of the transitions t_1 or t_2 before the firing of t_3 . Let us recall, that with a *posteriori* semantics, one can fire only one transition at the same time. Reachability graphs are given in figure 2.

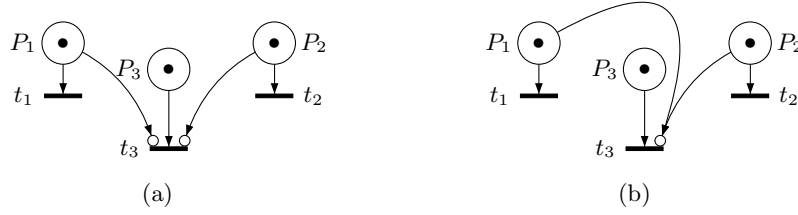


Fig. 1. Two branch inhibitor nets

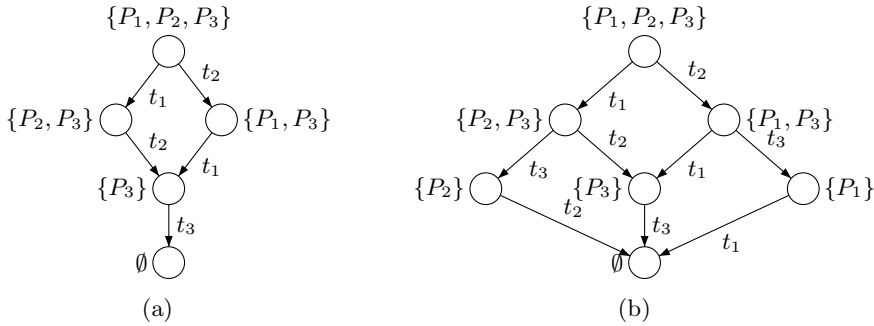


Fig. 2. Reachability graph of branch inhibitor nets of fig.1

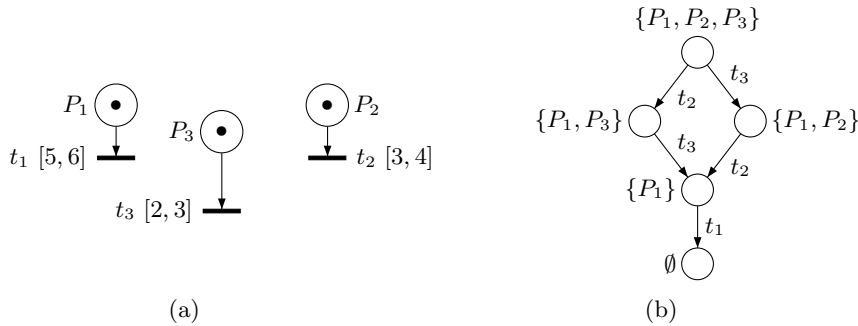


Fig. 3. A time Petri net and its state class graph

If we remove the inhibitor arcs and add time to the underlying structure of those Petri nets, we obtain the time Petri net in figure 3a. t_1 , t_2 and t_3 are enabled at the same time but as the earliest firing time of t_1 (5 time units) is strictly higher than the latest firing time of the other transitions we obtain the reachability graph (computed with the state class graph) in figure 3b.

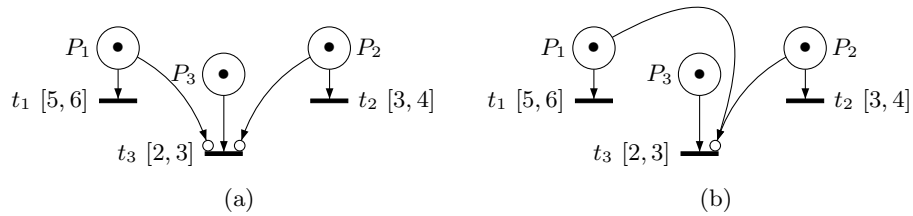


Fig. 4. Two IHTPN

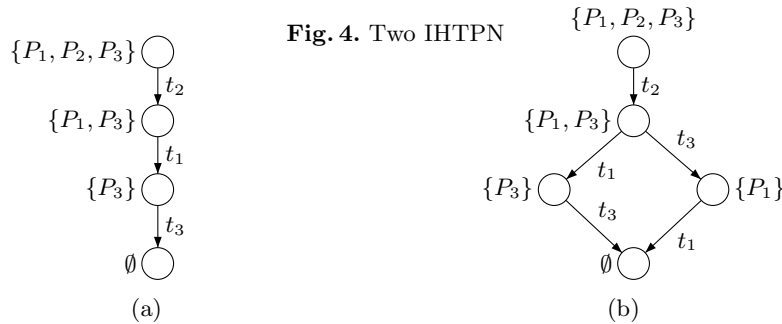


Fig. 5. Reachability graph of IHTPN of fig.4

Now let us add the branch inhibitor arcs of figure 1 to the TPN in figure 3. We obtain two T-time Petri nets with inhibitor hyperarcs in 4. In the two examples (a and b), the inhibitor hyperarcs prohibit the firing of t_3 first and temporisations impose the firing of t_2 before that of t_1 . Let us detail the example (fig.4b): the initial marking of the net is $\{P_1, P_2, P_3\}$. We consider 3 stopwatches $\{x_1, x_2, x_3\}$ associated to respectively $\{t_1, t_2, t_3\}$. A stopwatch x represents the time elapsed since the associated transition t was last enabled (by the marking) and during which t is not inhibited. A stopwatch x is stopped, which is denoted by $\dot{x} = 0$, when the associated transition t is inhibited. A stopwatch x is reset when the associated transition t is *newly enabled* (and inhibited or not).

Since P_1 and P_2 are marked, t_3 is inhibited and $x_1 = 1, x_2 = 1, x_3 = 0$. Time elapses for x_1 and x_2 , and t_2 becomes fireable. After the firing of t_2 , t_3 is not inhibited anymore and $x_1 = 1, x_3 = 1$. We have then $x_1 \in [3, 4]$ and $x_3 = 0$ because time did not pass for t_3 which was inhibited. t_1 can fire in the interval $[1, 3]$ and t_3 must still wait between 2 and 3 time units. Reachability graphs are given in figure 5.

2.2 Formal semantics

Definition 1 (Time Petri net with Inhibitor Hyperarcs). A Time Petri nets with Inhibitor Hyperarcs is a n -tuple

$\mathcal{T} = (P, T, \bullet(\cdot), (\cdot)^\bullet, \alpha, \beta, M_0, I)$, 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,
- $\bullet(\cdot) \in (\mathbb{N}^P)^T$ is the backward incidence function,

- $(\cdot)^\bullet \in (\mathbb{N}^P)^T$ is the forward incidence function,
- $M_0 \in \mathbb{N}^P$ is the initial marking of the net,
- $\alpha \in (\mathbb{Q}^+)^T$ and $\beta \in (\mathbb{Q}^+ \cup \{\infty\})^T$ are functions giving for each transition respectively its earliest and latest firing times ($\alpha \leq \beta$),
- I is a finite set of branch inhibitor hyperarcs:
 - each hyperarc in I is a pair (Q, t) where $t \in T$ is a transition and $Q \in \mathbb{N}^P$ is the backward inhibition valuation of the hyperarc ¹
 - we note $I(t)$ the number of inhibitor hyperarcs of the transition $t \in T$,
 - Let $i \leq I(t)$ the i^{th} inhibitor hyperarc of the transition $t \in T$ such that $\exists Q_i \in \mathbb{N}^P, (Q_i, t) \in I$. We note $\alpha^i = Q_i$.

In the example presented in figure 4b : $I(t_3) = 1, \alpha_3^1 = [1, 1, 0]$ that represents $\{P_1, P_2\}$.

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 valuation on the arc between this place and the transition). We denote it by $t \in \text{enabled}(M)$.

A transition t is said to be *inhibited* by the marking M if there are tokens in all the places of one of its inhibitor hyperarcs: $\exists i \leq I(t), \alpha^i < M$. We denote it by $t \in \text{inhibited}(M)$.

A transition t is said to be *firable* when it has been enabled and not inhibited for at least $\alpha(t)$ 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 $\uparrow \text{enabled}(t_k, M, t_i)$, 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,

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

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

Let us consider the IHTPNs in figure 6. Let us assume that the initial marking is $M(P) = 1, M(P_{inh}) = 1$ and t_1 is firable. t_2 is enabled by the marking M and by the marking $M' = M - \bullet t_1 + t_1^\bullet$ but not by $M - \bullet t_1$. Transitions t_1 and t_2 are newly enabled by the firing of t_1 . The only difference for the example 6b is that the transition t_2 is newly enabled by the firing of t_1 but remains inhibited.

Let us now assume the initial marking is $M(P) = 2, M(P_{inh}) = 1$ and t_1 is firable. t_2 is enabled by the marking M and by the marking $M' = M - \bullet t_1 + t_1^\bullet$ and by $M - \bullet t_1$. t_1 is newly enabled by the firing of t_1 but not t_2 : t_2 remains enabled. The only difference for the example 6b is that the transition t_2 remains enabled and inhibited.

¹ If valuations are equal to zero or one, then Q can be represented by a set of places

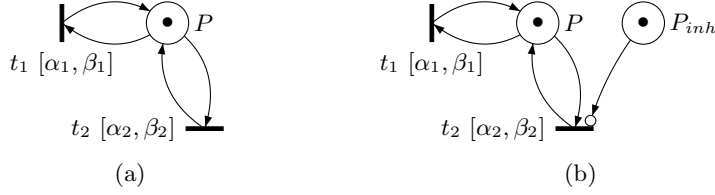


Fig. 6. Example of newly enabled transitions

We define the semantics of time Petri nets with Inhibitor Hyperarcs as *Timed Transition Systems* (TTS) [28]. In this model, two kinds of transitions may occur: *continuous* transitions when time passes and *discrete* transitions when a transition of the net fires.

A *valuation* is a mapping $\nu \in (\mathbb{R}^+)^T$ such that $\forall t \in T, \nu(t)$ is the time elapsed since t was last enabled and during which t remained not inhibited. ($\nu(t)$ represents the value of the stopwatch associated to t). Note that $\nu(t)$ is meaningful only if t is an enabled transition. $\bar{0}$ is the *null valuation* such that $\forall k, \bar{0}_k = 0$.

Definition 2 (Semantics of an IHTPN). *The semantics of a time Petri net with Inhibitor Hyperarcs \mathcal{T} is defined as a TTS $\mathcal{S}_{\mathcal{T}} = (Q, q_0, \rightarrow)$ such that*

- $Q = \mathbb{N}^P \times (\mathbb{R}^+)^T$
- $q_0 = (M_0, \bar{0})$
- $\rightarrow \in Q \times (T \cup \mathbb{R}) \times Q$ is the transition relation including a continuous transition relation and a discrete transition relation.

- The continuous transition relation is defined $\forall d \in \mathbb{R}^+$ by :

$$(M, \nu) \xrightarrow{d} (M, \nu') \text{ iff } \forall t_i \in T, \begin{cases} \nu'(t_i) = \begin{cases} \nu(t_i) & \text{if } t_i \in \text{enabled}(M) \text{ and } t_i \in \text{inhibited}(M) \\ \nu(t_i) + d & \text{otherwise,} \end{cases} \\ M \geq \bullet t_i \Rightarrow \nu'(t_i) \leq \beta(t_i) \end{cases}$$

- The discrete transition relation is defined $\forall t_i \in T$ by :

$$(M, \nu) \xrightarrow{t_i} (M', \nu') \text{ iff }, \begin{cases} t_i \in \text{enabled}(M) \text{ and } t_i \notin \text{inhibited}(M), \\ M' = M - \bullet t_i + t_i \bullet, \\ \alpha(t_i) \leq \nu(t_i) \leq \beta(t_i), \\ \forall t_k \in T, \nu'(t_k) = \begin{cases} 0 & \text{if } \uparrow \text{enabled}(t_k, M, t_i), \\ \nu(t_k) & \text{otherwise} \end{cases} \end{cases}$$

Note that for transitions which are not enabled, the continuous transition relation of the semantics lets the valuations evolve. They could as well have been stopped.

Note also that, as for untimed Petri nets, branch inhibitor hyperarcs do not increase the expressivity of the model compared to simple inhibitor arcs.

Figure 7 shows an example of how to translate a branch inhibitor hyperarc into simple inhibitor arcs for a safe IHTPN. The general case, for not necessarily bounded IHTPNs, is a bit more complicated and involves explicitly splitting transitions into a discrete and a continuous transitions and adding transitions not to be considered in the reachability graph. Still we are able to define a timed bisimulation between the IHTPN and its translation with simple inhibitor arcs. So, while they do not increase the expressivity of the model, branch inhibitor hyperarcs do allow a more convenient modeling.

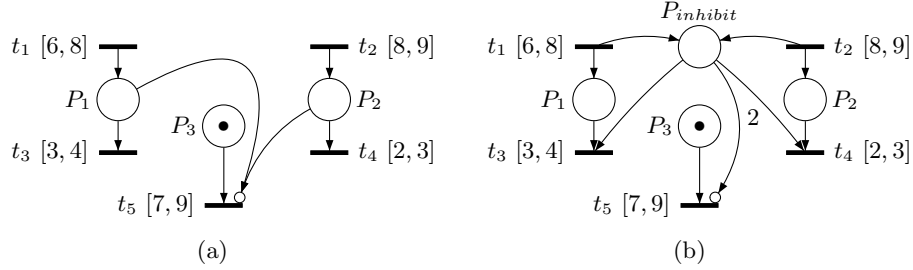


Fig. 7. Two bisimilar safe IHTPN

As for TPN, the behavior of an IHTPN can be defined by timed firing sequences which are sequences of pairs (t, d) where t is a transition of the IHTPN and $d \in \mathbb{R}^+$. Then a sequence $\omega = (t_1, d_1)(t_2, d_2) \cdots (t_n, d_n)$ indicates that t_1 is fired after d_1 time units, then t_2 is fired after d_2 time units and so on, so that transition t_i is fired at absolute time $\sum_{k=1}^i d_k$. A marking M is *reachable* in an IHTPN iff there is a timed firing sequence ω leading from the initial marking M_0 to M . It is common for TPN to define the *untimed* sequence from the timed one: if $\omega = (t_1, d_1)(t_2, d_2) \cdots (t_n, d_n)$ then $Untimed(\omega) = t_1 t_2 \cdots t_n$.

We can now extend the usual notion of accepted language for IHTPN.

Definition 3 (Accepted language). *The language accepted by an IHTPN is the set of all its timed firing sequences.*

Definition 4 (Untimed language). *The untimed language accepted by an IHTPN is the set of all its untimed firing sequences.*

3 Properties

3.1 Relation between some classes of Petri Nets and IHTPN

In this section, we will consider the following classes of Petri nets:

- *PN*: the class of classical Petri Nets,
- *priority-PN*: the class of Petri nets with priority as defined in [14],

- *bih-PN*: the class of Petri Nets with branch inhibitor hyperarcs as defined in [1],
- *TPN*: the class of T-Time Petri Nets as defined in [5],
- *scheduling-TPN*: the class of scheduling extended time Petri nets as defined in [3],
- *preemptive-TPN*: the class of preemptive time Petri nets as defined in [4],
- *IHTPN*: the class of Time Petri Nets with inhibitor Hyperarcs.

In this section, we will consider the timed language subclass definition:

Definition 5 (subclass). *A class A is a subclass of the class B if any timed language accepted by a Petri net of the class A is also acceptable by a Petri net of the class B. It is a strict subclass if the reciprocal is not true.*

As a consequence, classes of models are defined by the expressivity of these models as timed language acceptors.

Property 1 (Relation between TPN and IHTPN). Any TPN can be described by an IHTPN: a TPN is an IHTPN without inhibitor hyperarcs. However, it is easy to find an IHTPN which accepts the language $\forall n \in [0, 1], \forall m \geq 0, (t_1, n)(t_2, m)(t_3, 1 - n)$, while a TPN which accepts it does not exist. Consequently, *TPN* is a strict subclass of *IHTPN*.

Property 2 (Relation between PN and IHTPN). Any PN can be described by a TPN [13], therefore by an IHTPN. *PN* is a strict subclass of *IHTPN*.

Property 3 (Relation between bih-PN and IHTPN). Any PN with branch inhibitor hyperarcs can be described by an IHTPN. It is sufficient to take an IHTPN with the same discrete structure but in which all the firing intervals have a null earliest firing time or an infinite latest firing time. As PN with branch inhibitor hyperarcs is not a timed model, the reciprocal is not true in terms of timed language acceptance. Consequently, *bih-PN* is a strict subclass of *IHTPN*.

Property 4 (Relation between priority-PN and IHTPN). Any priority net can be described by a Petri net with inhibitor hyperarcs [1], therefore, by transitivity, by an IHTPN. *priority-PN* is a strict subclass of *IHTPN*.

Property 5 (Relation between scheduling-TPN or preemptive-TPN and IHTPN). It is easy to show that any scheduling-TPN [3, 27] and any preemptive-TPN [4] can be described by an IHTPN with an inhibitor hyperarc for each priority relation per resource. However, the reciprocal is not true: indeed, with scheduling-TPN and preemptive-TPN, it is not possible to model a circular priority relation which can definitively block the time flow of transitions while no concurrent transition runs (*i.e.* the required resources are free). Consequently, if there is an enabled transition for which time is blocked, that means that there is another one for which time is not blocked. As long as there is an enabled transition, then there will be in the future the firing of a transition. Conversely, IHTPN can model a circular relation of inhibition involving a definitive blocking of the time flow of enabled transition (see fig.8). Consequently, *scheduling-TPN* and *preemptive-TPN* are strict subclasses of *IHTPN*.

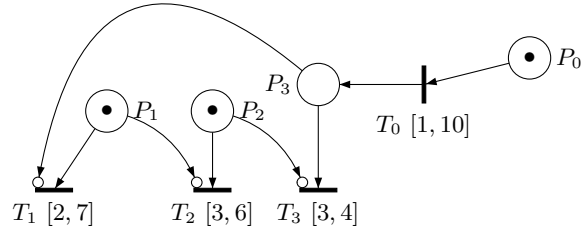


Fig. 8. IHTPN with deadlock

A consequence of the previous property is that IHTPNs are also adapted to the modeling of the real-time tasks scheduling.

3.2 Boundedness and reachability

Property 6 (Reachability and boundedness of marking problems are undecidable for IHTPN). Reachability and boundedness of marking problems are undecidable for TPN and for PN with inhibitor hyperarcs. TPNs and PNs with inhibitor hyperarcs are particular cases of IHTPNs. Consequently, reachability and boundedness of marking problems are undecidable for IHTPNs.

Remark 1. The set of reachable markings of an IHTPN is a subset of the set of reachable markings of the underlying PN. Moreover the set of reachable markings of an IHTPN is also a subset of the set of reachable markings of the underlying PN with inhibitor hyperarcs. However, the set of reachable markings of an IHTPN is not a subset of the set of reachable markings of the underlying TPN. See examples and reachability graph fig.1,2,3,4,5.

Property 7 (Sufficient and decidable condition of IHTPN marking boundedness). A not necessary but sufficient and decidable condition of IHTPN marking boundedness is: the underlying PN is bounded.

Property 8 (Sufficient and undecidable condition of IHTPN marking boundedness). A not necessary but sufficient and undecidable condition of IHTPN marking boundedness is: the underlying PN with inhibitor hyperarcs is bounded.

Definition 6 (k-boundedness). A Petri Net is *k*-bounded iff for any place *p* of the net, no marking *M* such that $M(p) > k$ is reachable.

Property 9 (k-boundedness problems for IHTPN). For classical TPN and for PN with inhibitor hyperarcs, *k*-boundedness is decidable because any *k*-bounded Petri Net has a finite set of reachable marking and for TPN, if the limits of the static firing intervals are rational numbers, it is possible to represent the infinite state space by a finite partitioning. However, for stopwatch automata, for instance, that finite partitioning does not exist in the general case. While IHTPN is a model with stopwatches, it is not proved yet that IHTPNs belong to the same class as stopwatch automata. This means that *k*-boundedness is still an open problem for IHTPN.

The consequence of this section is that as the boundedness of marking problem is undecidable for IHTPN, we can try to compute the state space or to analyse an IHTPN but there is no guarantee for computation termination. In the next section, we will therefore propose a semi-algorithm for IHTPN state space computation.

4 State Space of IHTPN

In order to analyse a time Petri net, the computation of its reachable state space is required. However, the reachable state space of a time Petri net is obviously infinite, so a method has been proposed to partition it into a finite set of infinite *state classes* [11]. This method is briefly explained in the next subsection. The following paragraphs then describe its extension in order to compute the state space of IHTPN.

4.1 State class graph of a Time Petri Net

Given a bounded time Petri net, Berthomieu and Diaz have proposed a method for computing the state space as a finite set of *state classes* [11]. Basically a state class contains all the states of the net between the firing of two consecutive transitions.

Definition 7 (State class). *A state class C , of a time Petri net, is a pair (M, D) where M is a marking of the net and D a set of inequations called the firing domain.*

The inequations in D are of two types [11]

$$\begin{cases} \alpha_i \leq x_i \leq \beta_i \ (\forall i \text{ such that } t_i \text{ is enabled}), \\ -\gamma_{kj} \leq x_j - x_k \leq \gamma_{jk}, \ \forall j, k \text{ such that} \\ j \neq k \text{ and } (t_j, t_k) \in \text{enabled}(M)^2 \end{cases}$$

x_i is the firing time of the enabled transition t_i relatively to the time when the class was entered in.

Because of their particular form, the firing domains may be encoded using DBMs [29], which allow the use of efficient algorithms for the computation of classes.

Given a class $C = (M, D)$ and a fireable transition t_f , computing the successor class $C' = (M', D')$ obtained by firing t_f consists of computing the new marking as usual and the new domain. This involves variable substitutions and eliminations and intersections, modeling the time that elapses, and computation of the canonical form of the DBM allowing efficient checks for equality. The overall complexity is $O(n^3)$ with n being the number of transitions enabled by M [30]. Computation of the state space of the TPN consists merely of the classical building of the reachability graph of state classes. That is to say, that starting from the initial state class, all the successors obtained by firing fireable transitions are computed iteratively until all the produced successors have already been generated.

Example Figure 9 shows an example of a TPN and its state class graph. The domains of the first two classes have been detailed, as well as the markings for each class. Notice that the sets of inequations of the domains fit in a DBM. This form is however insufficient to represent temporal domains with stopwatches.

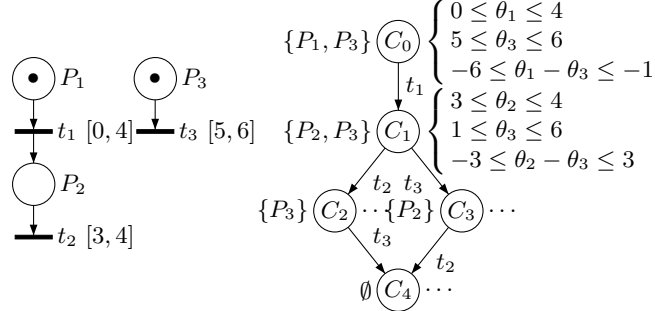


Fig. 9. A TPN and its state class graph

4.2 Exact state space computation of IHTPN using polyhedra

In the method previously presented, temporal domains are expressed as sets of linear inequations in the form of Difference Bound Matrices (DBM). The form of a DBM is not sufficient to express the temporal domain with the presence of stopwatches. Some changes are proposed to allow its construction.

Precisely, the variable substitution in the firing domain is now only done for enabled transitions that are *not inhibited*.

While we still define a state class of an IHTPN as a marking and a domain, the general form of the domain is not preserved. The new general form is that of a polyhedron with constraints involving up to n variables, with n being the number of transitions enabled by the marking of the class:

$$\begin{cases} \alpha_i \leq \theta_i \leq \beta_i, \forall t_i \in \text{enabled}(M), \\ a_{i_1} \theta_{i_1} + \dots + a_{i_n} \theta_{i_n} \leq \gamma_{i_1 \dots i_n}, \\ \forall (t_{i_1}, \dots, t_{i_n}) \in \text{enabled}(M)^n \\ \text{and with } (a_{i_0}, \dots, a_{i_n}) \in \mathbb{Z}^n. \end{cases}$$

Algorithm of computation We will now show the algorithm for computing the successor classes of a given class and then define the state class graph for IHTPN. But first, we need a new definition for the firability of a transition from a class:

Definition 8 (firability). Let $C = (M, D)$ be a state class of an IHTPN. A transition t_i is said to be firable from C iff there exists a solution $(\theta_0^*, \dots, \theta_{n-1}^*)$ of D , such that $\forall j \in [0, n-1] - \{i\}$, s.t. t_j is not inhibited, $\theta_i^* \leq \theta_j^*$.

Successors of a class Given a class $C = (M, D)$ and a firable transition t_f , the class $C' = (M', D')$ obtained from C by the firing of t_f is given by

- $M' = M - \bullet_{t_f} + t_f \bullet$
- D' is computed along the following steps, and noted $next(D, t_f)$
 1. variable substitutions for all enabled transitions that are *not* inhibited
 $t_j: \theta_j = \theta_f + \theta'_j$,
 2. intersection with the set of positive or null reals \mathbb{R}^+ : $\forall i, \theta'_i \geq 0$,
 3. elimination (using for instance the Fourier-Motzkin method [31]) of all variables relative to transitions disabled by the firing of t_f ,
 4. addition of inequations relative to newly enabled transitions

$$\forall t_k \in \uparrow enabled(M, t_f), \alpha(t_k) \leq \theta'_k \leq \beta(t_k).$$

State class graph Knowing how to compute the successors of a class, the state space computation is basically a depth-first or breadth-first graph generation.

For an IHTPN \mathcal{T} , let $\Delta(\mathcal{T}) = (C, C_0, \rightarrow)$ be the transition system such that:

- $C = \mathbb{N}^P \times \mathbb{R}^T$,
- $C_0 = (M_0, D_0)$, where M_0 is the initial marking and $D_0 = \{\alpha(t_i) \leq \theta_i \leq \beta(t_i) \mid t_i \in enabled(M_0)\}$,
- $\rightarrow \in C \times T \times C$ is the transition relation defined by :
$$(M, D) \xrightarrow{t} (M', D') \text{ iff } \begin{cases} M' = M - \bullet_t + t \bullet, \\ t \text{ is firable from } (M, D), \\ D' = next(D, t), \end{cases}$$

Then we can write the state class graph as the quotient of $\Delta(\mathcal{T})$ by a "suitable" equivalence relation. This "suitable" relation may be equality as in definition 9, in which case the state class graph has the same untimed language as its IHTPN. Or it can be inclusion as in definition 10, and then the untimed language is lost but we still have the set of reachable markings.

Given a set of inequations D , we note $\llbracket D \rrbracket$ the set of solutions of D . With this notation, we can now define equality and inclusion of classes:

Definition 9 (Equality of state classes). *Two classes $C_1 = (M_1, D_1)$ and $C_2 = (M_2, D_2)$ are equal if $M_1 = M_2$ and $\llbracket D_1 \rrbracket = \llbracket D_2 \rrbracket$.*

Definition 10 (Inclusion of state classes). *The state class $C = (M_1, D_1)$ is included in the state class $C_2 = (M_2, D_2)$ if $M_1 = M_2$ and $\llbracket D_1 \rrbracket \subset \llbracket D_2 \rrbracket$.*

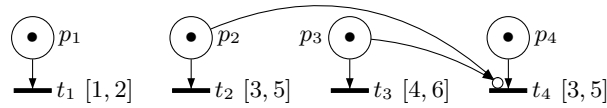


Fig. 10. A time Petri net with inhibitor hyperarcs

Example Figure 10 shows an example of a simple IHTPN. Its initial class is $C = (M, D)$ with $M = \{p_1, p_2, p_3, p_4\}$ and

$$D = \begin{cases} 1 \leq x_1 \leq 2, \\ 3 \leq x_2 \leq 5, \\ 4 \leq x_3 \leq 6, \\ 3 \leq x_4 \leq 5, \\ -4 \leq x_1 - x_2 \leq -1, \\ -5 \leq x_1 - x_3 \leq -2, \\ -4 \leq x_1 - x_4 \leq -1, \\ -3 \leq x_2 - x_3 \leq 1, \\ -2 \leq x_2 - x_4 \leq 2, \\ -1 \leq x_3 - x_4 \leq 3 \end{cases} \quad (1)$$

t_1, t_2, t_3 and t_4 are enabled but t_4 is inhibited, hence:

$$\begin{cases} \dot{x}_1 = 1, \\ \dot{x}_2 = 1, \\ \dot{x}_3 = 1, \\ \dot{x}_4 = 0, \end{cases} \quad (2)$$

t_1 is clearly firable. Let us compute the domain of the class obtained by firing t_1 . The first step is the variable substitution $x_i \leftarrow x'_i + x_1$ for all enabled transitions that are not inhibited but t_1 . The domain becomes

$$\begin{cases} 1 \leq x_1 \leq 2, \\ 3 \leq x_1 + x'_2 \leq 5, \\ 4 \leq x_1 + x'_3 \leq 6, \\ 3 \leq x'_4 \leq 5, \\ -4 \leq x_1 - x_1 - x'_2 \leq -1, \\ -5 \leq x_1 - x_1 - x'_3 \leq -2, \\ -4 \leq x_1 - x_4 \leq -1, \\ -3 \leq x_1 + x'_2 - x_1 - x'_3 \leq 1, \\ -2 \leq x_1 + x'_2 - x_4 \leq 2, \\ -1 \leq x_1 + x'_3 - x_4 \leq 3 \end{cases} \quad (3)$$

The next step is the elimination of the variable x_1 . We use the Fourier-Motzkin method. For that, we rewrite the inequations as follows:

$$\begin{cases} 1 \leq x_1, & x_1 \leq 2, \\ 3 - x'_2 \leq x_1, & x_1 \leq 5 - x'_2, \\ 4 - x'_3 \leq x_1, & x_1 \leq 6 - x'_3, \\ -4 + x_4 \leq x_1 & x_1 \leq -1 + x_4, \\ -2 + x_4 - x'_2 \leq x_1, & x_1 \leq 2 + x_4 - x'_2, \\ -1 + x_4 - x'_3 \leq x_1, & x_1 \leq 3 + x_4 - x'_3, \\ 3 \leq x_4 \leq 5, \\ -3 \leq x'_2 - x'_3 \leq 1, \\ -4 \leq -x'_2 \leq -1, \\ -5 \leq -x'_3 \leq -2 \end{cases} \quad (4)$$

The Fourier-Motzkin method then consists of writing that the system has solutions if and only if the lower bounds of x_1 are less or equal to the upper bounds. The obtained system is then equivalent to the initial one. After a few simplifications we obtain

$$D' = \begin{cases} 1 \leq x'_2 \leq 4, \\ 2 \leq x'_3 \leq 5, \\ 3 \leq x_4 \leq 5, \\ -3 \leq x'_2 - x'_3 \leq 1, \\ 0 \leq x'_2 - x_4 \leq 1, \\ 1 \leq x'_3 - x_4 \leq 2, \\ 4 \leq x'_2 + x_4 \leq 9, \\ 5 \leq x'_3 + x_4 \leq 10, \\ 0 \leq x'_2 + x_4 - x'_3 \leq 6, \\ 2 \leq x'_3 + x_4 - x'_2 \leq 8 \end{cases} \quad (5)$$

What we can see here is that eight inequations, given on the last four lines, are generated, which cannot be expressed with a DBM. Furthermore, we can easily see that those new inequations may give even more complex inequations (*i.e.* involving more variables) when firing another transition from the obtained class. However, in this example, those extra eight inequations are redundant with the "simple" inequations: for instance, using "simple" inequations $x'_2 \leq 4$ and $x'_4 \leq 5$ we find again $x'_2 + x'_4 \leq 9$. Formally, this means that:

$$\llbracket D' \rrbracket = \left[\left[\begin{cases} 1 \leq x'_2 \leq 4, \\ 2 \leq x'_3 \leq 5, \\ 3 \leq x_4 \leq 5, \\ -3 \leq x'_2 - x'_3 \leq 1, \\ 0 \leq x'_2 - x_4 \leq 1, \\ 1 \leq x'_3 - x_4 \leq 2 \end{cases} \right] \right] \quad (6)$$

This redundancy occurs for many classes in the general case (but unfortunately not always) so it makes sense to ignore the non-DBM form inequations for a faster computation of the state class graph. This may lead, of course, to an overapproximation.

4.3 Overapproximation of the state space

Manipulating polyhedra in the general case involves a very important computing cost. In order to be able to keep our algorithms efficient for IHTPN, we approximate the polyhedron representing the firing domain to the smallest DBM containing it. By doing this, we clearly add states in our classes that should not be reachable and thus we do an overapproximation. This is illustrated by the net in Figure 11.

After the firing sequence t_4, t_1, t_5 , the transition t_6 is not fireable because either t_2 or t_3 (depending on the firing time of t_4) must be fired first. The class

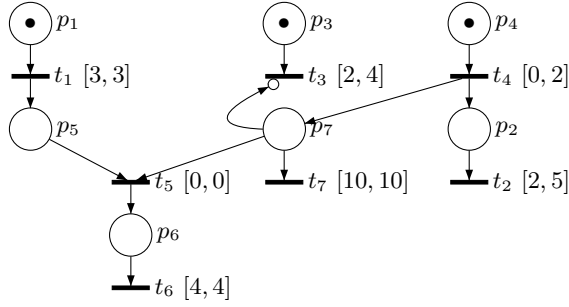


Fig. 11. A time Petri Net with Inhibitor Hyperarcs

obtained is:

$$\left\{ \begin{array}{l} \{p_2, p_3, p_6\}, \\ 0 \leq x_2 \leq 4, \\ 0 \leq x_3 \leq 4, \\ 4 \leq x_6 \leq 4, \\ -4 \leq x_2 - x_3 \leq 4, \\ -4 \leq x_2 - x_6 \leq 0, \\ -4 \leq x_3 - x_6 \leq 0, \\ 1 \leq x_2 + x_3 \leq 6 \end{array} \right.$$

We can easily see that t_6 is indeed not firable, for this implies $x_2 = x_3 = x_6 = 4$ and thus $x_2 + x_3 = 8$. But if we remove the $x_2 + x_3 \leq 6$ constraint in order to keep a DBM form, t_6 becomes firable. So we have here an overapproximation.

However, for the verification of safety properties the overapproximation is not a too big concern. Since we want to ensure that something "bad" never happens, we only need to check a set of states which contains the actual state space of the IHTPN. Of course, there is still a risk of being pessimistic.

5 Verification of timed reachability properties

A consequence of the state space computation is that using the classical observer notion, we can verify varied timed reachability properties. An observer consists of adding places and transitions, which model the property (discrete or quantitative) to check, to the Petri net. Checking the property is transformed in checking the reachability of a given marking of the observer. In [32], the authors present generic TPN observers to model properties like absolute or relative time between the firing of transitions, causality or simultaneity. It is then possible to check properties on IHTPN with the same generic observers (thanks to property 1 in 3).

Let us consider the net in figure 12. It represents a simple IHTPN and an observer in dash lines. The observer allows to check the property: "2 successive occurrences of t_0 always happen in less than 12 time units". The property is *false* iff a marking M such that $M(FALSE) > 0$ is reachable (i.e. iff a run such that the place *FALSE* has a token exists).

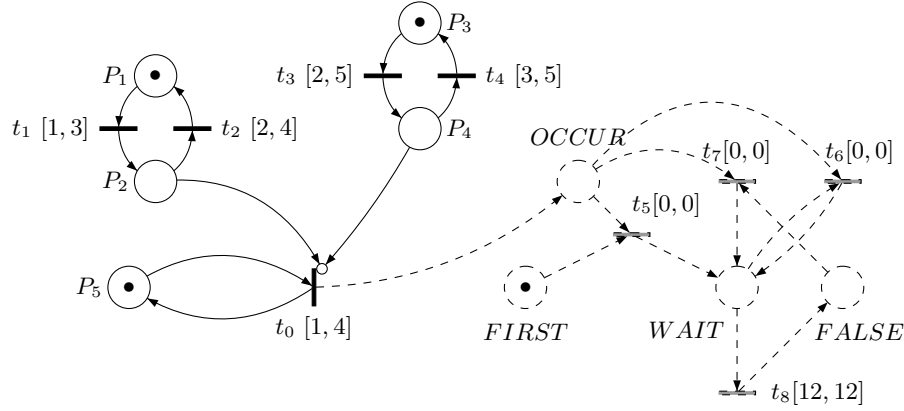


Fig. 12. Example of an IHTPN and an observer (dash lines)

Constructing the state space and looking for a run with a token in *FALSE* allows to conclude that the property is false. Indeed, let us consider, for instance, the run² $s_0 \xrightarrow{2}_{t_0} s_1 \xrightarrow{0}_{t_5} s_2 \xrightarrow{0}_{t_1} s_3 \xrightarrow{0}_{t_3} s_4 \xrightarrow{4}_{t_2} s_5 \xrightarrow{0}_{t_4} s_6 \xrightarrow{2}_{t_1} s_7 \xrightarrow{0}_{t_3} s_8 \xrightarrow{4}_{t_2} s_9 \xrightarrow{1}_{t_4} s_{10} \xrightarrow{1}_{t_8} s_{11} \xrightarrow{0}_{t_0} s_{12}$.

The transition t_0 is inhibited in s_4 and s_8 ; no time elapse in s_1, s_2, s_3, s_5, s_7 and s_{11} . In this run, two successive occurrences of t_0 happen in exactly 12 time units. This run led to $M(\text{FALSE}) = 1$ in s_{11}, s_{12} .

Generally there is no need to compute the whole state space: the computation of the state space can be stopped at the first marking verifying a property.

6 Conclusion

In this paper, we have defined an extension of T-time Petri nets with inhibitor hyperarcs (IHTPN). In this model, we can consider that stopwatches are associated with transitions. A stopwatch can be reset, stopped and started by using classical arcs and inhibitor hyperarcs. It allows the memorisation of the progress status of an action that is stopped and resumed. We have also proposed a method for the computation of the state space of an IHTPN. This includes an exact computation using a general polyhedron representation of timing constraints, and an overapproximation of the polyhedra to allow more efficient compact abstract representations of the state space based on DBM (Difference Bound Matrix).

We have shown that Petri nets with inhibitor hyperarcs, time Petri nets, preemptive-TPN and scheduling-TPN are all strict subclasses of IHTPN. As a consequence, IHTPN can be used in the context of scheduling verification but IHTPN are also able to express more general problems including circular relation

² We note $S \xrightarrow{d}_e S'$ to indicate a discrete step e preceded by a time flow of d time units.

of inhibition involving a definitive blocking of the time flow of enabled transition. It can thus be important to have a model able to express this kind of deadlocks in order to detect them.

Finally, we have shown how to check complex timing properties on IHTPN by using observers.

Future works include the study of the model-checking of properties expressed in temporal logics such as Timed Computation Tree Logic (TCTL). This can be carried out when the state space is computable, and when it is not the case, it can be done on the fly for some properties.

We also plan to tackle some of the problems that remain open such as the decidability of k -boundedness for IHTPN.

References

1. Janicki, R., Koutny, M.: On causality semantics of nets with priorities. *Fundamenta Informaticae* **38** (1999) 223–255
2. Okawa, Y., Yoneda, T.: Schedulability verification of real-time systems with extended time Petri nets. *International Journal of Mini and Microcomputers* **18** (1996) 148–156
3. Roux, O.H., Déplanche, A.M.: A t-time Petri net extension for real time-task scheduling modeling. *European Journal of Automation (JESA)* **36** (2002)
4. Bucci, G., Fedeli, A., Sassoli, L., Vicario, E.: Modeling flexible real time systems with preemptive time Petri nets. In: 15th Euromicro Conference on Real-Time Systems (ECRTS'2003). (2003) 279–286
5. Merlin, P.M.: A study of the recoverability of computing systems. PhD thesis, Department of Information and Computer Science, University of California, Irvine, CA (1974)
6. Ramchandani, C.: Analysis of asynchronous concurrent systems by timed Petri nets. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA (1974) Project MAC Report MAC-TR-120.
7. Pezze, M., Toung, M.: Time Petri nets: A primer introduction. Tutorial presented at the Multi-Workshop on Formal Methods in Performance Evaluation and Applications, Zaragoza, Spain (1999)
8. Khansa, W., Denat, J.P., Collart-Dutilleul, S.: P-Time Petri Nets for manufacturing systems. In: International Workshop on Discrete Event Systems, WODES'96, Edinburgh (U.K.) (1996) 94–102
9. de Frutos Escrig, D., Ruiz, V.V., Alonso, O.M.: Decidability of properties of timed-arc Petri nets. In: 21st International Conference on Application and Theory of Petri Nets (ICATPN'00). Volume 1825 of Lecture Notes in Computer Science., Aarhus, Denmark, Springer-Verlag (2000) 187–206
10. Abdulla, P.A., Nylén, A.: Timed Petri nets and BQOs. In: 22nd International Conference on Application and Theory of Petri Nets (ICATPN'01). Volume 2075 of Lecture Notes in Computer Science., Newcastle upon Tyne, United Kingdom, Springer-Verlag (2001) 53–72
11. Berthomieu, B., Diaz, M.: Modeling and verification of time dependent systems using time Petri nets. *IEEE transactions on software engineering* **17** (1991) 259–273
12. Diaz, M., Senac, P.: Time stream Petri nets: a model for timed multimedia information. *Lecture Notes in Computer Science* **815** (1994) 219–238

13. Menasche, M.: Analyse des réseaux de Petri temporisés et application aux systèmes distribués. PhD thesis, Université Paul Sabatier, Toulouse, France (1982)
14. Hack, M.: Petri net language. Computation Structures Group Memo 127, MIT (1975)
15. Agerwala, T.: A complete model for representing the coordination of asynchronous processes. Technical report, John Hopkins University, Baltimore, Maryland. (1974)
16. Chiola, G., Donatelli, S., Franceschinis, G.: Priorities, inhibitor arcs and concurrency in PT nets. In: 12th international conference on Application and Theory of Petri Nets. (1991) 182–205
17. Janicki, R., Koutny, M.: Invariant semantics of nets with inhibitor arcs. In: In CONCUR'91, Incs vol.527. (1991) 317–331
18. Peterson, J.: Petri net theory and the modeling of systems. Prentice-Hall, New-York (1981)
19. Janicki, R., Koutny, M.: Semantics of inhibitor nets. Information and Computation **123** (1995) 1–15
20. Cassez, F., Larsen, K.: The impressive power of stopwatches. In Palamidesi, C., ed.: 11th International Conference on Concurrency Theory, (CONCUR'2000). Number 1877 in Lecture Notes in Computer Science, University Park, P.A., USA, Springer-Verlag (2000) 138–152
21. Popova, L.: On time Petri nets. Journal on Information Processing and Cybernetics, EIK **27** (1991) 227–244
22. Yoneda, T., Ryuba, H.: CTL model checking of time Petri nets using geometric regions. IEICE Transactions on Information and Systems **E99-D** (1998) 297–396
23. Lilius, J.: Efficient state space search for time Petri nets. In: MFCS Workshop on Concurrency '98. Volume 18 of ENTCS., Elsevier (1999)
24. Berthomieu, B., Vernadat, F.: State class constructions for branching analysis of time Petri nets. In: 9th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'2003), Springer-Verlag (2003) 442–457
25. Lime, D., Roux, O.H.: State class timed automaton of a time Petri net. In: 10th International Workshop on Petri Nets and Performance Models, (PNPM'03), IEEE Computer Society (2003)
26. Gardey, G., Roux, O.H., Roux, O.F.: A zone-based method for computing the state space of a time Petri net. In: In Formal Modeling and Analysis of Timed Systems, (FORMATS'03). Volume LNCS 2791., Springer (2003)
27. Lime, D., Roux, O.H.: Expressiveness and analysis of scheduling extended time Petri nets. In: 5th IFAC International Conference on Fieldbus Systems and their Applications, (FET'03), Elsevier Science (2003)
28. Larsen, K.G., Pettersson, P., Yi, W.: Model-checking for real-time systems. In: Fundamentals of Computation Theory. (1995) 62–88
29. Dill, D.L.: Timing assumptions and verification of finite-state concurrent systems. In: Workshop Automatic Verification Methods for Finite-State Systems. Volume 407. (1989) 197–212
30. Berthomieu, B.: La méthode des classes d'états pour l'analyse des réseaux temporels. In: 3e congrès Modélisation des Systèmes Réactifs (MSR'2001), Toulouse, France, Hermes (2001) 275–290
31. Dantzig, G.B.: Linear programming and extensions. IEICE Transactions on Information and Systems (1963)
32. Toussaint, J., Simonot-Lion, F., Thomesse, J.P.: Time constraint verification methods based on time Petri nets. In: 6th Workshop on Future Trends in Distributed Computing Systems (FTDCS'97), Tunis, Tunisia (1997) 262–267