Symbolic state space of Stopwatch Petri nets with discrete-time semantics (theory paper)

Morgan Magnin, Didier Lime, and Olivier (H.) Roux

IRCCyN, CNRS UMR 6597, Nantes, France
{Morgan.Magnin | Didier Lime | Olivier-h.Roux}@irccyn.ec-nantes.fr

Abstract. In this paper, we address the class of bounded Petri nets with stopwatches (SwPNs), which is an extension of T-time Petri nets (TPNs) where time is associated with transitions. Contrary to TPNs, SwPNs encompass the notion of actions that can be reset, stopped and started. Models can be defined either with discrete-time or dense-time semantics. Unlike dense-time, discrete-time leads to combinatorial explosion (state space is computed by an exhaustive enumeration of states). We can however take advantage from discrete-time, especially when it comes to SwPNs: state and marking reachability problems, undecidable even for bounded nets, become decidable once discrete-time is considered. Thus, to mitigate the issue of combinatorial explosion, we now aim to extend the well-known symbolic handling of time (using convex polyhedra) to the discrete-time setting. This is basically done by computing the state space of discrete-time nets as the discretization of the state space of the corresponding dense-time model. First, we prove that this technique is correct for TPNs but not for SwPNs in general: in fact, for the latter, it may add behaviors that do not really belong to the evolution of the discrete-time net. To overcome this problem, we propose a splitting of the general polyhedron that encompasses the temporal information of the net into an union of simpler polyhedra which are safe with respect to the symbolic successor computation. We then give an algorithm that computes symbolically the state space of discrete-time SwPNs and finally exhibit a way to perform TCTL model-checking on this model.

Key words: Verification using nets, Time Petri nets, symbolic state space, stopwatches, dense-time, discrete-time

Introduction

The ever-growing development of embedded informatics requires efficient methods for the verification *a priori* of real-time systems. That is why researches on formalisms that allow engineers to write and check the interactions between CPUs and the communication networks appear as a hot topic. Time Petri Nets [1] are one of such formalisms. They model the temporal specifications of different actions under the form of time intervals. They can be enriched to represent tasks that may be suspended then resumed.

When modeling a system, either dense-time or discrete-time semantics may be considered. In the first one, time is considered as a dense quantity (i.e. the state of the system can change at any moment) and, in the second one, as a discrete variable (time progress is ensured by clock ticks and the global system may evolve only at these peculiar time steps). The physical systems (the processes) follow a dense-time evolution. The observation of the process is however usually performed through an IT command system which pilots it only at some peculiar instants (digitalization or periodic observations). In addition, the command system is composed of tasks that are executed on one (or many) processor(s) for which physical time is discrete. Dense-time is thus an over-approximation of the real system. The major advantage of dense-time lies in the symbolic abstractions it offers: they are easy to put into application and they avoid the combinatorial explosion of states. In this paper, we aim to propose a symbolic method to compute the state space of discrete-time Time Petri Nets (with stopwatches) by adapting the techniques usually dedicated to dense-time.

Time Petri nets with stopwatches

The two main time extensions of Petri nets are Time Petri nets [1] and Timed Petri nets [2]. In this paper, we focus on Time Petri nets (TPNs) in which transitions can be fired within a time interval.

In order to take into account the global complexity of systems, models now encompass the notion of actions that can be suspended and resumed. This implies extending traditional clock variables by "stopwatches". Several extensions of TPNs that address the modeling of stopwatches have been proposed: Scheduling-TPNs [3] , Preemptive-TPNs [4] (these two models add resources and priorities attributes to the TPN formalism) and Inhibitor Hyperarc TPNs (ITPNs) [5]. ITPNs introduce special inhibitor arcs that control the progress of transitions. These three models belong to the class of TPNs extended with stopwatches (SwPNs) [6]. They have been studied in dense-time semantics.

In [6], state reachability for SwPNs has been proven undecidable, even when the net is bounded. As long as dense-time semantics is considered, the state space is generally infinite. Instead of enumerating each reachable state, verification algorithms compute finite abstractions of the state space, e.g. state class graph, that preserve the properties to be verified. But, as a consequence of the undecidability of the reachability problem, the finiteness of the state class graph cannot be guaranteed. In order to ensure termination on a subclass of bounded SwPNs, Berthomieu *et al.* propose *an overapproximation method* based on a quantization of the polyhedra representing temporal information [6]. Nevertheless the methods are quite costly in terms of computation time.

Discrete-time semantics

In the case of SwPNs, the undecidability of major model-checking problems results from dense-time. The use of discrete-time instead (transitions are then

no longer fired at any time but at integer dates) change these results, as we proved in [7]. In this paper, we established the following results:

- The state reachability problem - undecidable with dense-time semantics - is decidable when discrete-time is considered;
- The state space of discrete-time bounded SwPNs can be computed directly by using existing tools for classical Petri nets.

In the case of TPNs (without stopwatches), main works related to discrete-time are due to Popova. Her method consists in analyzing the behavior only at its so-called "integer-states" (i.e. states where the current local time for all enabled transitions are integers), which is sufficient to know the whole behavior of the net [8].

In both cases, discrete-time based approaches suffer from a combinatorial explosion of the state space size. As efficient as the implementation (see [9] for data-structures dedicated to Petri nets and inspired by the Binary Decision Diagrams (BDDs)) could be, it reaches its limits as soon as there are transitions with a large time interval (e.g. $[1, 10000]$) in the model.

That is why we propose here to work out a method that would allow to compute symbolically the state space of discrete-time TPNs with stopwatches. The most natural idea consists in extending the method applied in dense-time to discrete-time, that means: compute the whole state space by the means of state classes that bring together all the equivalent discrete-time behaviors (the notion of equivalence will be precisely defined in section 4). Our approach consists in computing the dense-time state-space *as long as it is significant* and then discretize it to get the discrete-time behavior.

State space computation of dense-time TPNs

For bounded dense-time TPNs, the state reachability problem is decidable. Although the state space of the model is infinite (as the clocks associated to transitions take their values in $\mathbb{R}$), it can be represented by a finite partition under the form of a state class graph [10] or a region graph [11]. The state class graph computes the whole set of reachable markings of a bounded TPN ; the resulting graph preserves the untimed language of the net (i.e. properties of linear temporal logics). However, it does not preserve branching temporal properties ; for this class of properties, we have to consider refinements of the state class graph, e.g. *atomic state classes* [12]. The zone graph can be used to check quantitative properties based on a subset of TPN-TCTL [13]. At the moment, only the first method has however been extended to TPNs with stopwatches [6]. Thus this is the approach we consider in this paper.

Our contribution

In this paper, we address the general class of bounded Petri nets with stopwatches (SwPNs) with strict or weak temporal constraints and a single-server dense-time or discrete-time semantics. Our goal is to prove that, in certain cases, it

is possible to compute the state space and the untimed language of a discrete-time net by simply discretizing the state space of the associated dense-time net. We exhibit an example showing however that, in the general case, this approach can lead to a false analysis. We conclude by giving a method that allies the efficiency of dense-time symbolic approaches with the specificities of discrete-time enumerative methods in order to compute a finite partition of the state space of TPNs with stopwatches. For the sake of simplicity, our results are explained of the ITPN model.

Outline of the paper

Our aim is to compute efficiently the state space of a bounded discrete-time SwPN in order to verify quantitative timing properties. The paper is organized as follows: section 2 introduces TPNs with stopwatches (by the means of ITPNs) and the related semi-algorithm that computes their state space. In section 3, we show that, for classical TPNs, the discretization of the dense-time state space is sufficient to get the discrete-time state space. Section 4 extends the result to some subclasses of SwPNs but shows it is not valid for the general class of SwPNs. We exhibit a SwPN such that the discrete-time behaviors do not encompass all the dense-time behaviors in terms of untimed language and marking reachability. In section 5, we propose a method, for computing the state space of SwPNs: this method combines the advantages of symbolic computations with dense-time specificities.

1 Time Petri Nets with inhibitor arcs

We give an informal presentation of Time Petri Nets with inhibitor arcs in appendix **??**.

1.1 Notations

The sets $\mathbb{N}$, $\mathbb{Q}^+$ and $\mathbb{R}^+$ are respectively the sets of natural, non-negative rational and non-negative real numbers. An interval I of $\mathbb{R}^+$ is a $\mathbb{N}$-interval iff its left endpoint belongs to $\mathbb{N}$ and its right endpoint belongs to $\mathbb{N} \cup \{\infty\}$. We set $I^{\downarrow} = \{x | x \leq y \text{ for some } y \in I\}$, the *downward closure* of I and $I^{\uparrow} = \{x | x \geq y \text{ for some } y \in I\}$, the *upward closure* of I. We denote by $\mathcal{I}(\mathbb{N})$ the set of $\mathbb{N}$-intervals of $\mathbb{R}^+$.

1.2 Formal definitions and semantics of Time Petri nets with inhibitor arcs

Definition 1. *A Time Petri net with Inhibitor Arcs (ITPN) is a n-tuple* $\mathcal{N} = (P, T, {}^{\bullet}(.), (.)^{\bullet}, {}^{\circ}(.), M_0, I)$*, where*

- $P = \{p_1, p_2, \ldots, p_m\}$ *is a non-empty finite set of* places*,*

- $T = \{t_1, t_2, \ldots, t_n\}$ *is a non-empty finite set of* transitions,
- ${}^\bullet(.) \in (\mathbb{N}^P)^T$ *is the* backward incidence function,
- $(.)^\bullet \in (\mathbb{N}^P)^T$ *is the* forward incidence function,
- ${}^\circ(.) \in (\mathbb{N}^P)^T$ *is the* inhibition function,
- $M_0 \in \mathbb{N}^P$ *is the* initial marking *of the net,*
- $I_s \in (\mathcal{I}(\mathbb{N}))^T$ *is the function that associates a firing interval to each transition.*

In [5], the authors extend inhibitor arcs with the notion of hyperarc. Inhibitor hyperarcs make it easier to model systems with priority relations between transitions, but they do not increase the theoretical expressivity of the model compared to inhibitor arcs. That is why we can equivalently work on Time Petri Nets with inhibitor arcs or inhibitor hyperarcs. For the sake of simplicity, we focus on nets with inhibitor arcs (ITPNs) in this paper.

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

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

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

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

A transition t is said to be *firable* when it has been enabled and not inhibited for at least $I(t)^\downarrow$ 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 $\uparrow enabled(t_k, M, t_i)$, 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:

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

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

Let $\mathbb{T}$ be a generic time domain: it may be $\mathbb{N}$, $\mathbb{Q}^+$ or $\mathbb{R}^+$.

Definition 2. *A* state *of a TPN is a pair $q = (M, I)$ in which M is a marking and I is a function called the* interval *function. Function $I \in (\mathcal{I}(\mathbb{N}))^T$ associates a temporal interval with every transition enabled at M.*

We define the semantics of an ITPN as a time transition system. In this model, two kinds of transitions may occur: *time* transitions when time passes and *discrete* transitions when a transition of the net is fired.

Definition 3 (Semantics of an ITPN). *Given a time domain $\mathbb{T}$, the semantics of a Time Petri Net with Inhibitor Arcs $\mathcal{N}$ is defined as a Timed Transition System $\mathcal{S}_{\mathcal{N}}^{\mathbb{T}} = (Q, q_0, \rightarrow)$ such that:*

- $Q = \mathbb{N}^P \times \mathbb{T}^T$;
- $q_0 = (M_0, I_s)$
- $\rightarrow \in Q \times (T \cup \mathbb{T}) \times Q$ *is the transition relation including a time transition relation and a discrete transition relation. The time transition relation is defined $\forall d \in \mathbb{T}$ by:*

$$(M, I) \xrightarrow{d} (M, I') \text{ iff } \forall t_i \in T,$$
$$\begin{cases} I'(t_i) = \begin{cases} I(t_i) \text{ if } t_i \in enabled(M) \\ \quad \text{and } t_i \in inhibited(M) \\ I'(t_i)^{\uparrow} = max(0, I(t_i)^{\uparrow} - d), \text{ and } I'(t_i)^{\downarrow} = I(t_i)^{\downarrow} - d \text{ otherwise,} \end{cases} \\ M \geq^{\bullet} t_i \Rightarrow I'(t_i)^{\downarrow} \geq 0 \end{cases}$$

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

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

In the *dense-time approach*, time is considered as a *continuous* variable whose evolution goes at rate 1. The dense-time semantics of the net $\mathcal{N}$ is thus $\mathcal{S}_{\mathcal{N}}^{dense} = \mathcal{S}_{\mathcal{N}}^{\mathbb{R}^+}$.

By contrast, in the *discrete-time approach*, time is seen as *"jumping" from one integer to the other*, with no care of what may happen in between. The latter is an under-approximation of the former. The discrete-time semantics of $\mathcal{N}$ is $\mathcal{S}_{\mathcal{N}}^{discrete} = \mathcal{S}_{\mathcal{N}}^{\mathbb{N}}$.

Note that $\mathcal{S}_{\mathcal{N}}^{discrete}$ can be straightforwardly reduced to a simple transition system. As long as dicrete-time semantics is considered, any open interval with integer bounds may be turned into a closed interval. That is why, in the following, we only consider closed intervals (that may however be open on ∞).

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

A *run* ρ of length $n \geq 0$ in a TTS is a finite or infinite sequence of alternating time and discrete transitions of the form

$$\rho = q_0 \xrightarrow{d_0} q_0' \xrightarrow{a_0} q_1 \xrightarrow{d_1} q_1' \xrightarrow{a_1} \cdots q_n \xrightarrow{d_n} \cdots$$

We write $first(\rho)$ the first state of a run ρ. A run is *initial* if $first(\rho) = q_0$. A run ρ of $\mathcal{N}$ is an initial run of $\mathcal{S}_{\mathcal{N}}^{\mathbb{T}}$. The timed language accepted by $\mathcal{N}$ with dense-time semantics (respectively with discrete-time semantics) is $\mathcal{L}^{dense}(\mathcal{N}) = \mathcal{L}(\mathcal{S}_{\mathcal{N}}^{dense})$ (resp. $\mathcal{L}^{discrete}(\mathcal{N}) = \mathcal{L}(\mathcal{S}_{\mathcal{N}}^{discrete})$).

In the following, we denote by $\mathcal{Q}^{dense}$ (resp. $\mathcal{Q}^{discrete}$) the set of reachable states of $\mathcal{S}_{\mathcal{N}}^{dense}$ (resp. $\mathcal{S}_{\mathcal{N}}^{discrete}$).

To every TPN (possibly extended with stopwatches) structure $\mathcal{N}$, we can associate either a *dense-time* or a *discrete-time* semantics. We then obtain two different models. In the following, we say that two TPNs are *associated* if, whatever the choice on their semantics has been, they share the same underlying structure $\mathcal{N}$.

Definition 4. *Given an integer $n \in \mathbb{N}$ and a set $D \subseteq \mathbb{R}^n$. We define the* discretization operator *of domain D by: $\mathcal{D}isc(D) = D \cap \mathbb{N}^n$*

Definition 5. *Given an integer $n \in \mathbb{N}$ and a set $D \in \mathbb{R}^n$. A point $\theta = (\theta_1, \theta_2, \ldots, \theta_n) \in D$ is an* integer point *of D if all its coordinates are integers,* i.e. $\forall i, \theta_i \in \mathbb{N}$.

In this paper, we restrict ourselves to the class of *bounded* TPNs and ITPNs. This does not imply that the underlying net is bounded! The converse assertion is however true: the boundedness of the underlying PN ensures the boundedness of a TPN (or ITPN).

1.3 State space computation of dense-time models

In order to analyze a Time Petri Net, the computation of its reachable state space is required. However, the reachable state space of a TPN is obviously infinite.

For models expressed with dense-time semantics, one of the approaches to partition the state space in a finite set of infinite state classes is the state class graph proposed by Berthomieu and Diaz [10]. It has been extended for TPNs with stopwatches [14].

A state class contains all the states of the net between the firing of two successive transitions.

Definition 6. *A* state class *C of a dense-time ITPN is a pair (M, D) where M is a marking of the net and D a (convex) polyhedron with m constraints ($m \in \mathbb{N}$) involving up to n variables, with n being the number of transitions enabled by the marking of the class:*

$$A\Theta \leq B$$

with A and B being rational matrices of respective dimensions (m, n) and $(m, 1)$ and $\overline{\theta}$ being a vector of dimension n. D constrains the firing times θ of transitions.

In the case of TPNs, the firing domain is simpler than a general polyhedron and therefore can be encoded into the efficient Difference Bound Matrix (DBM) datastructure [15, 16]).

We extend the definition of the *discretization operator* of a point in $\mathbb{R}^n$ to state classes by the following definition:

Definition 7. *Let $C = (M, D)$ be a state class of a TPN (with or without inhibitor arcs). We define the* discretization operator *of the state class C by :* $\mathcal{D}isc(C) = (M, \mathcal{D}isc(D))$

In the case of ITPNs, the only firable transitions are the active ones. So we need to define properly the firability of a transition from a class:

Definition 8 (Firability). *Let $C = (M, D)$ be a state class of a ITPN. A transition t_i is said to be* firable *from C iff there exists a solution $(\theta_0^*, \ldots, \theta_{n-1}^*)$ of D, such that $\forall j \in \{0, \ldots, n-1\} - \{i\}$, s.t. t_j is active , $\theta_i^* \leq \theta_j^*$.*

Now, 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. intersection with the firability constraints : $\forall j$ s.t. t_j is active, $\theta_f \leq \theta_j$
  2. variable substitutions for all enabled transitions that are *active* t_j: $\theta_j = \theta_f + \theta'_j$,
  3. elimination (using for instance the Fourier-Motzkin method) of all variables relative to transitions disabled by the firing of t_f,
  4. addition of inequations relative to newly enabled transitions

$$\forall t_k \in \uparrow enabled(M, t_f), I(t_k)^\downarrow \leq \theta'_k \leq I(t_k)^\uparrow$$

The variable substitutions correspond to a shift of time origin for active transitions: the new time origin is the firing time of t_f. t_f is supposed to be firable so the polyhedron constrained by the inequalities $\theta_f \leq \theta_j$ is non empty.

The state class graph is generated by iteratively applying the function that computes the successors of a state class. The computation starts from the initial state class given by $C_0 = (M_0, D_0)$ with $D_0 = \{\theta_k \in I(t_k) \mid t_k \in enabled\,(M_0)\}$.

Definition 9 (Next). *Let $C = (M, D)$ be a state class of a ITPN, let Θ be a point of D and t_f be a transition firable from $(M, \{\Theta\})$. The successor of $(M, \{\Theta\})$ by the firing t_f is given by*

$$next_{t_f}^{dense}(\{\Theta\}) = \left\{ \forall i \in [1..n][\theta'_1 \ldots \theta'_n]^\top \;\middle|\; \begin{array}{l} \theta'_i \in I_s(t_i) \text{ if } \uparrow enabled(t_i, M, t_f) \\ \theta'_i = \theta_i \text{ if } t_i \in enabled(M) \\ \text{ and } t_i \in inhibited(M) \\ \text{ and not } \uparrow enabled(t_i, M, t_f) \\ \theta'_i = \theta_i - \theta_f \text{ otherwise} \end{array} \right\}$$

The $next_{t_f}^{dense}$ operator straightforwardly extends to finite or infinite unions of points.

1.4 State space computation of discrete-time models

To our knowledge, the only existing methods for computing the state space of discrete-time TPNs (with or without stopwatches) are based on an enumeration of states. The computation does not necessarily finish (for example, if the net contains a transition with a latest firing time equal to infinity). It suffers from the combinatorial explosion of the state space size. A way to mitigate this issue consists in working with data-structured inspired from the well-known Binary Decision Diagrams (BDDs) [9]. This approach however reveals its limits when large timing constraints are implicated in the model.

In this paper, we propose a new way to deal with combinatorial explosions. It consists in extending the symbolic methods usually applied to dense-time to discrete-time. We thus define the notion of state classes in the specific context of discrete-time.

The underlying idea is the same as in dense-time: a state class contains all the states of the net between the firing of two successive transitions.

Definition 10. *A* state class C *of a discrete-time ITPN is a pair* (M, D) *where* M *is a marking of the net and* D *a set (potentially empty) of points of* $\mathbb{N}^n$*, with* n *being the number of transitions enabled by the marking of the class.*

The definition of firability remains the same is the discrete case.

Now, given a class $C = (M, D)$ and a firable transition t_f, the successor class $C' = (M', D')$ obtained from C by the firing of t_f is denoted by $C' = (M', D') = (M', next_{t_f}^{discrete}(D))$ where $next_{t_f}^{discrete}$ is defined for all integer point Θ by

$$next_{t_f}^{discrete}(\{\Theta\}) = \mathcal{D}isc(next_{t_f}^{dense}(\{\Theta\}))$$

. Note that the operator $\mathcal{D}isc$ is necessary here because of the interval of the newly enabled transitions.

The purpose of this paper is to extend symbolic methods of dense-time to discrete-time according the following approach:

- Describe the set of points of a temporal domain D not by an enumeration, but by a convex polyhedron $Poly$ such that $D = \mathcal{D}isc(Poly)$
- Compute $C^{symb'} = (M', Poly')$, successor of $C^{symb} = (M, Poly)$ by the firing of t_f (denoted (M,$next_{t_f}^{dense}(D)$)) by the classical symbolic method and then link the symbolic classes C^{symb}, $C^{symb'}, \ldots$ to the state space of $\mathcal{N}$ considered with its discrete-time semantics $\mathcal{S}_{\mathcal{N}}^{discrete}$

2 Relations between dense-time and discrete-time semantics for Time Petri Nets

2.1 Related work

In [17], Popova proposed an analysis method for TPNs that differs from the classical state class graph of Berthomieu, Diaz and Menasche [15, 10]. Both methods

are based on the computation of a reachability graph. But when Berthomieu et al. deal with classes that are used to describe infinitely many states, Popova restricts herself to one state. She defines a state as the conjunction of a marking and a time vector of the current local time for each of the enabled transitions and a special symbol for disabled transitions. Then she builds a reachability graph that is only based on the so-called "integer states". These are the states where the current local time of all enabled transitions are integers. She proves that the knowledge of the net behavior in these integer states is sufficient to determine the entire behavior of the net. This result, firstly proven for TPNs with finite latest firing times, has been later extended to nets with infinite latest firing times [18].

Following this work, Popova et al. presented, in [19], a parametric description of a firing sequence of a TPN that is based on the notion of integer states.

In the specific context of a comparison between dense and discrete-time, Popova's results establish the following assertion: a discrete-time analysis is sufficient to study a dense-time TPN. In this section, we propose the opposite result (for the state class graph, but our proof can easily be extended to zone graph), that is: the state space of a discrete-time TPN can be directly computed by discretizing its dense-time state space. In fact, this result does not only apply to TPNs. In the next section, we extend it to a more general subclass of TPNs with stopwatches.

2.2 Discretization of the state space of dense-time Time Petri nets

Theorem 1. *For all TPN $\mathcal{N}$, let $\mathcal{S}_{\mathcal{N}}^{dense} = (\mathcal{Q}^{dense}, q_0, \rightarrow^{dense})$ (resp. $\mathcal{S}_{\mathcal{N}}^{discrete} = (\mathcal{Q}^{discrete}, q_0, \rightarrow^{discrete})$) be its dense-time (resp. discrete-time) semantics. Then, $\mathcal{Q}^{discrete} = \mathcal{D}isc(\mathcal{Q}^{dense})$.*

This theorem is a major and immediate corollary of the following one:

Theorem 2. *Let $\mathcal{N}$ be a TPN. Let C be one of the state classes of its dense-time semantics $\mathcal{S}_{\mathcal{N}}^{dense}$ $C = (M, DBM)$. Let t_f be a firable transition from C. Then, $next_{t_f}^{discrete}(\mathcal{D}isc(DBM)) = \mathcal{D}isc(next_{t_f}^{dense}(DBM))$*

Let us omit the proof of this theorem, since it actually is a special case of theorem 4, which will be proved in the next section.

3 Relations between dense-time and discrete-time for Time Petri nets with Stopwatches

3.1 Differences between dense-time and discrete-time semantics in terms of marking reachability and untimed language

We prove here that, in the general case of ITPNs, the state space of a discrete-time ITPN and the discretization of the state space of the associated dense-time net are not equal.

Theorem 3. *There exists a ITPN $\mathcal{N}$, with $\mathcal{S}_{\mathcal{N}}^{dense} = (\mathcal{Q}^{dense}, q_0, \rightarrow^{dense})$ (resp. $\mathcal{S}_{\mathcal{N}}^{discrete} = (\mathcal{Q}^{discrete}, q_0, \rightarrow^{discrete})$) being its dense-time (resp. discrete-time) semantics such that $\mathcal{Q}^{discrete} \subsetneq \mathcal{D}isc(\mathcal{Q}^{dense})$.*

Proof. Let us now exhibit an ITPN that proves this theorem. Consider the net $\mathcal{N} = (P, T, {}^\bullet(.), (.)^\bullet, {}^\circ(.), M_0, I)$ in figure 1.

Let us analyze its behavior. The state space of the dense-time semantics (leading to 31 different state classes) and of the discrete-time semantics can be computed.

In dense-time semantics, the following run is valid:

$$\begin{array}{l} \{p_1, p_8\} \\ \theta(guess) = 0 \\ \theta(r) = 0 \end{array} \xrightarrow{0.5} \begin{array}{l} \{p_1, p_8\} \\ \theta(guess) = 0.5 \\ \theta(r) = 0.5 \end{array} \xrightarrow{guess} \begin{array}{l} \{p_2, p_5, p_8\} \\ \theta(c) = 0 \\ \theta(s) = 0 \\ \theta(r) = 0.5 \end{array} \xrightarrow{3.5} \begin{array}{l} \{p_2, p_5, p_8\} \\ \theta(c) = 3.5 \\ \theta(s) = 3.5 \\ \theta(r) = 4 \end{array} \xrightarrow{r}$$

$$\begin{array}{l} \{p_2, p_5, p_7, p_9\} \\ \theta(c) = 3.5 \\ \theta(s) = 3.5 \\ \theta(t) = 0 \end{array} \xrightarrow{0.5} \begin{array}{l} \{p_2, p_5, p_7, p_9\} \\ \theta(c) = 3.5 \\ \theta(s) = 4 \\ \theta(t) = 0.5 \end{array} \xrightarrow{s} \begin{array}{l} \{p_2, p_6, p_7, p_9\} \\ \theta(c) = 3.5 \\ \theta(t) = 0.5 \\ \theta(flush) = 0 \end{array} \xrightarrow{flush} \begin{array}{l} \{p_2, p_9\} \\ \theta(c) = 3.5 \\ \theta(t) = 0.5 \end{array} \xrightarrow{0.5}$$

$$\begin{array}{l} \{p_2, p_9\} \\ \theta(c) = 4 \\ \theta(t) = 1 \end{array} \xrightarrow{c} \begin{array}{l} \{p_3, p_9\} \\ \theta(u) = 0 \\ \theta(t) = 1 \end{array} \xrightarrow{3} \begin{array}{l} \{p_3, p_9\} \\ \theta(u) = 3 \\ \theta(t) = 4 \end{array} \xrightarrow{t} \begin{array}{l} \{p_3, p_{10}\} \\ \theta(u) = 3 \\ \theta(test) = 0 \end{array} \xrightarrow{1} \begin{array}{l} \{p_3, p_{10}\} \\ \theta(u) = 4 \\ \theta(test) = 1 \end{array} \xrightarrow{u}$$

$$\begin{array}{l} \{p_4, p_{10}\} \\ \theta(test) = 1 \\ \theta(too\_early) = 0 \end{array} \xrightarrow{test} \begin{array}{l} \{p_4, p_{11}\} \\ \theta(too\_early) = 0 \\ \theta(too\_late) = 0 \\ \theta(in\_time) = 0 \end{array} \xrightarrow{in\_time} \{p_{goal}\}$$

We aim to prove that, contrary to the dense-time behavior, the analysis of the discrete-time net never leads to the firing of *in_time*. To achieve this goal, we just have to enumerate all the possible runs of the discrete-time net. This is quite easy:

- When *guess* is fired at 0, all the resulting runs end by the firing of *too_early*;
- *A contrario*, when *guess* is fired at 1, all the resulting runs end by the firing of *too_late*.

The specificity of the net we propose lies in the inhibitor arc that stops the stopwatch associated to transition c as long as transition $flush$ (thus s, as $flush$ fires in 0 time) does not fire. This inhibition causes the temporal shift between the upper part of the net and the bottom one (they both have a similar structure): if *guess* fires at 0, then we can say that the upper part is in advance compared to the bottom one. On the contrary, when *guess* fires at 1, the upper part is delayed. The only way to ensure that the two parts evolve in phase is that *guess* fires at 0.5 : the structure of the net then guarantees a parallel and synchronous evolution of the two branches so that p_4 and p_{11} are marked simultaneously, thus allowing the firing of *in_time*.

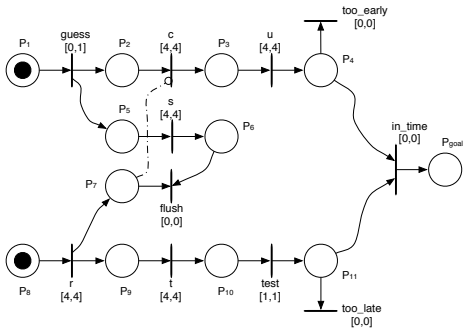

Fig. 1. ITPN showing the problems related to an analysis of the discrete-time PN models

3.2 A sufficient condition on ITPNs such that the discretization of the state space of the dense-time net and the state space of the discrete-time associated net coincide

For $\theta \in \mathbb{R}^+$, we denote by $frac(\theta)$ the fractional part of θ. Let $\Theta = [\theta_1 \cdots \theta_n]^\top$ be a point of $(\mathbb{R}^+)^n$, we denote by $\lceil \Theta \rceil$ the point $[\lceil \theta_1 \rceil \cdots \lceil \theta_n \rceil]^\top$ and $\lfloor \Theta \rfloor$ the point $[\lfloor \theta_1 \rfloor \cdots \lfloor \theta_n \rfloor]^\top$.

Definition 11 (t-thickness). *A class $C = (M, D)$ of an ITPN $\mathcal{N}$ is said to be* t-thick *if for all transition t_f that is firable from C, $\forall \Theta = [\theta_1 \dots \theta_n]^\top \in D$ s.t. for all active transition t_i, $frac(\theta_i) = frac(\theta_f)$ and for all inactive transition t_j, $frac(\theta_j) = 0$,we have: either $\lceil \Theta \rceil \in D$ or $\lfloor \Theta \rfloor \in D$.*

Theorem 4. *Let $\mathcal{N}$ be a ITPN. Let $C = (M, D)$ be one of the state classes of its dense-time semantics $\mathcal{S}_{\mathcal{N}}^{dense}$. Let t_f be a firable transition from C. If (C, D) is t-thick, then $next_{t_f}^{discrete}(\mathcal{D}isc(D)) = \mathcal{D}isc(next_{t_f}^{dense}(D))$*

Proof. The inclusion $next_{t_f}^{discrete}(\mathcal{D}isc(D)) \subseteq \mathcal{D}isc(next_{t_f}^{dense}(D))$ is straightforward. We shall now prove the reverse inclusion.

Let $C' = (M', D')$ be a state class of the TPN with stopwatches $\mathcal{N}$. Let $C = (M, D)$ be its parent class by the firing of transition t_f.

Let $\Theta = [\theta_1 \dots \theta_m]^\top$ be a point of D such that some transition t_f ($f \leq m$) is firable from Θ (i.e. $\forall i \leq m$, st t_i is active, $\theta_f \leq \theta_i$).

To keep the notations simple, let us assume, without loss of generality, that transitions $t_1, \dots, t_m$ are enabled by M (corresponding to variables $\theta_1 \dots \theta_m$ in D) and that the firing of t_f disables transitions $t_1 \dots t_{p-1}$ and newly enables transitions $t_{m+1} \dots t_n$. We also suppose that transitions $t_{p+1}, \dots, t_k$ with $k \leq n$ are inhibited by M and transitions $t_{k+1}, \dots, t_m$ are not.

Then,

$$next_{t_f}^{dense}(\{\Theta\}) = \left\{ [\theta'_1 \dots \theta'_{n-p+1}]^\top = \left| \begin{array}{l} \forall i \in [1..k], \theta'_i = \theta_{p+i} \\ \forall i \in [k+1..m-p], \theta'_i = \theta_{p+i} - \theta_f \\ \forall i \in [m-p+1..n-p+1], \theta'_i \in I(t_i) \} \end{array} \right. \right\}$$

Let Θ' be an integer point of D': $\Theta' \in \mathcal{D}isc(next_{t_f}^{dense}(\{\Theta\}))$ for some $\Theta \in D$. Then, $\forall i, \theta'_i \in \mathbb{N}$, which implies $frac(\theta_f) = frac(\theta_i)$, $\forall i$ s.t. t_i is active in C ($i \in [k+1..m-p]$), and $frac(\theta_i) = 0$ otherwise. As a consequence, since (C, D) is t-thick, either $\lceil\Theta\rceil$ or $\lfloor\Theta\rfloor$ is in D. Let us assume, as both cases are symmetric, that $\lceil\Theta\rceil \in D$.

Since t_f is firable from Θ, we have $\forall i \in [1..m]$, st t_i is active, $\theta_f \leq \theta_i$ and then $\theta_f + \delta \leq \theta_i + \delta$ for any δ, in particular for $\delta = 1 - frac(\theta_f) = 1 - frac(\theta_i)$. So t_f is firable from $\lceil\Theta\rceil$.

We have then:

$$next_{t_f}^{dense}(\{\lceil\Theta\rceil\}) = \left\{ [\theta'_1 \dots \theta'_{n-p+1}]^\top = \left| \begin{array}{l} \forall i \in [1..k], \lceil\theta'_i\rceil = \lceil\theta_{p+i}\rceil \\ \forall i \in [k+1..m-p], \theta'_i = \lceil\theta_{p+i}\rceil - \lceil\theta_f\rceil \\ \forall i \in [m-p+1..n-p+1], \theta'_i \in I(t_i)\} \end{array} \right. \right\}$$

For $i \in [k+1..m-p]$, $frac(\theta_f) = frac(\theta_{p+i})$. So we have $\lceil\theta_{p+i}\rceil - \lceil\theta_f\rceil = \theta_{p+i} - \theta_f$.

For $i \in [1..k]$, $frac(\theta_i) = 0$. So we have $\lceil\theta_i\rceil = \theta_i$.

Finally, $next_{t_f}^{dense}(\{\lceil\Theta\rceil\}) = next_{t_f}^{dense}(\{\Theta\})$. As $\lceil\Theta\rceil$ is an integer point of D, we have $\mathcal{D}isc(next_{t_f}^{dense}(\{\lceil\Theta\rceil\})) = next_{t_f}^{discrete}(\{\lceil\Theta\rceil\})$. Therefore, $\mathcal{D}isc(next_{t_f}^{dense}(\{\Theta\})) \in next_{t_f}^{discrete}(\mathcal{D}isc(D))$. □

Theorem 5. *Let $C = (M, D)$ be a class of an ITPN such that D is a DBM. C is t-thick.*

Proof. All constraints in a DBM are either of the form $\theta \leq d$ or of the form $\theta - \theta' \leq d$, with d being an integer. Let $C = (M, D)$ be a class of an ITPN such that D is a DBM. Let t_f be a transition firable from C. Let $\Theta = [\theta_1 \dots \theta_n]^\top \in D$ s.t. for all active transition t_i, $frac(\theta_i) = frac(\theta_f)$ and for all inactive transition t_j, $frac(\theta_j) = 0$.

Due to the particular form of constraints in DBM, it is sufficient to consider the two-dimensional projections of D for our reasoning. So, let us consider any two-dimensional plane corresponding to variables θ_i and θ_j. An example of the projection of a DBM on such a plane is given in Fig 2.

1. If both θ_i and θ_j correspond to inhibited transitions, then due to the above constraints the projection of Θ must be an integer point.
2. If only θ_j corresponds to an inhibited transition, then the projection of Θ must be on one of the verticals (Fig 2(b)).
3. If both variables correspond to active transitions, then the projection of Θ must be on one of the diagonals (Fig 2(a)).

In the first case, the projection of $\lceil\Theta\rceil$ is the same as the projection of Θ so it is inside the projection of the DBM. So let us consider the other two cases.

Suppose that some constrain $\theta_i - \theta_j \leq d$ intersects the segment formed by the projections of Θ and $\lceil\Theta\rceil$. In case 3, the constraint must be parallel to the segment. And since the constraints are large, this constraint is not excluding the projection of $\lceil\Theta\rceil$. In case 2, the intersections of all verticals with diagonal

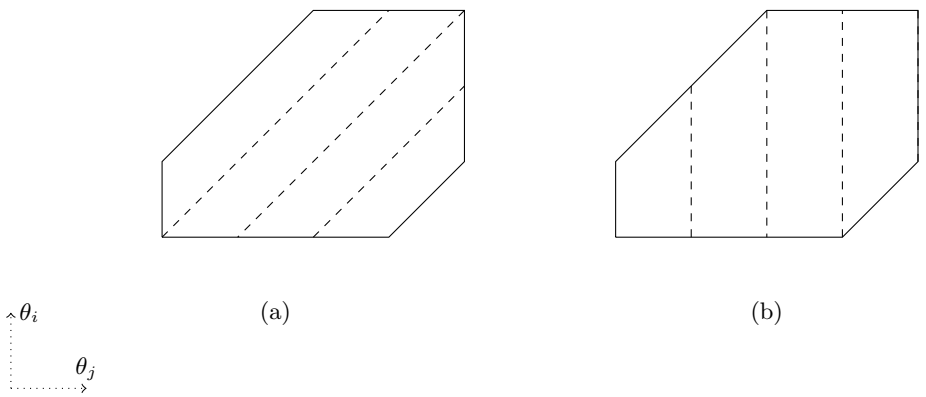

Fig. 2. Projection on a plane of a DBM: (a) θ_i and θ_j both correspond to active transitions (b) θ_j corresponds to an inhibited transition.

constraints are obviously integer points, so again the constraint cannot exclude $\lceil \Theta \rceil$.

The case of the constraint $\theta_i \leq d$ is similar. □

An immediate corollary of theorems 4 and 5 is the following generalization of theorem 2:

Corollary 1. *Let $\mathcal{N}$ be a ITPN. Let C be one of the state classes of its dense-time semantics $\mathcal{S}_{\mathcal{N}}^{dense}$ $C = (M, D)$ such that D is a DBM. Let t_f be a firable transition from C. Then, $next_{t_f}^{discrete}(\mathcal{D}isc(D)) = \mathcal{D}isc(next_{t_f}^{dense}(D))$*

In [20], we exhibited a proper TPN with stopwatches such that the firing domain of all its dense-time state classes are simple DBMs. So the class of ITPNs for which the discretization of the dense classes is exact is non-empty.

4 Symbolic approach for the computation of the state space of discrete-time Time Petri nets with Stopwatches

In order to build a symbolic method for state space computation in discrete-time, we need to define the notion of *symbolic state classes* for discrete-time TPNs (with stopwatches):

Definition 12. *Let $\mathcal{N} = (P, T, {}^\bullet(.), (.)^\bullet, {}^\circ(.), M_0, I)$ be a TPN (with or without inhibitor arcs), $n \in \mathbb{N}$ and $C^{symb} = (M, Poly)$ where $M \in \mathbb{N}^P$ and $Poly$ is a convex polyhedron of $\mathbb{R}^n$. $C = (M, Poly)$ is a* symbolic state class *of the discrete-time semantics $\mathcal{S}_{\mathcal{N}}^{discrete}$ of the net if, for all $\nu \in \mathcal{D}isc(Poly)$, (M, ν) is a state of $\mathcal{S}_{\mathcal{N}}^{discrete}$.*

We are going to symbolically compute the state space of discrete-time nets by using these symbolic state classes. At the end of the computation process, we get a set of symbolic state classes: the discretization of this set gives the state space of the discrete-time net.

4.1 The case of discrete-time TPNs

Let $\mathcal{N}$ be a TPN. We first compute the state space of $\mathcal{N}$ with its dense-time semantics $\mathcal{S}_{\mathcal{N}}^{dense}$. The discretization of each class identified during the computation leads to the state space of $\mathcal{S}_{\mathcal{N}}^{discrete}$. This results from the theorems we give in section 2.

4.2 The case of discrete-time TPNs with stopwatches

Let $\mathcal{N}$ be a ITPN. We aim to compute the state space of its discrete-time semantics $\mathcal{S}_{\mathcal{N}}^{discrete}$. Let us consider (M_0, D_0) the initial state class of $\mathcal{N}$ with its discrete-time semantics $\mathcal{S}_{\mathcal{N}}^{discrete}$. It is obviously a symbolic state class of $\mathcal{S}_{\mathcal{N}}^{discrete}$ that we denote C_0^{symb}. We compute the successors of this class the same way we would compute its successors for the associated dense-time model. We repeat the process as long as the on-the-fly computed state space do not need general non-DBM polyhedra to be described. As soon as a non-DBM polyhedron appears in the firing domain $Poly$ of a state class $C^{poly} = (M^{poly}, Poly)$, then we decompose it into a union of DBMs $DBM\_split(Poly) = \bigcup D_i^{split}$. In fact, in [20], we identified a necessary and sufficient condition (this condition is quite long; so, in the following, we denote it simply by condition 3.2 of [20]) that establishes the cases when a non-DBM polyhedron appears in the state space computation for a dense-time model. So we use this condition to know when we have to split the polyhedron into a union of DBMs.

The splitting procedure $DBM\_split(.)$ of a polyhedron into a union of DBMs (with preservation of the property $\mathcal{D}isc(Poly) = \bigcup \mathcal{D}isc(D_i^{split})$) is not unique. A rather obvious (but really not efficient) algorithm consists in decomposing the polyhedron $Poly$ into the union of all its integer points $\mathcal{D}isc(Poly)$. A more subtle approach consists in decomposing the polyhedron according to the variables that are part of non-DBM constraints of the polyhedron. In figure 3, we illustrate the stakes related to $DBM\_split(.)$: the problem is to split the polyhedron $Poly$ into a union of DBMs. We propose a solution among others that appears with dashed lines. Many other solutions may be considered but this study is not the scope of this paper.

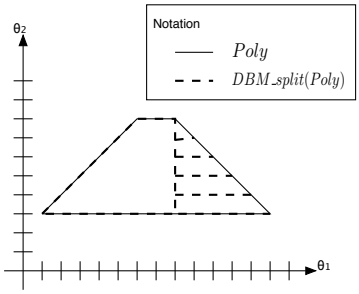

Fig. 3. Illustration of the effects of a splitting procedure. *Poly* represents the temporal domain associated to a symbolic class of a discrete-time ITPN. $DBM\_split(Poly)$ corresponds to a potential decomposition of this polyhedron into a union of DBMs such that $\mathcal{D}isc(Poly) = \mathcal{D}isc(DBM\_split(Poly))$.

$Passed = \emptyset$
$Waiting = \{(M_0, D_0)\}$
While $(Waiting \neq \emptyset)$ **do**
  $(M, D) = pop(Waiting)$
  $Passed = Passed \cup (M, D)$
  For t_f firable from (M, D) **do**
    $M' = M - {}^\bullet t_f + t_f^\bullet$
    If (condition 3.2 of [20] is not satisfied) **then**
      $D' = next^{dense}(D, t_f)$ [5]
      If $((M', D') \notin passed)$ **then**
        $Waiting = Waiting \cup \{(M', D')\}$
      end If
    else
      $\bigcup (D'_i)_{i \in [1,\ldots,n]} = DBM\_split(next^{dense}(D, t_f)$ [5])
      For $i \in [1, \ldots, n]$ **do**
        If $((M', D'_i) \notin Passed)$ **then**
          $Waiting = Waiting \cup \{(M', D'_i)\}$
        end If
      end For
    end If
  end For
done

Algorithm 1: Symbolic algorithm for the computation of the state space of discrete-time TPNs with stopwatches

This method is summarized by the formal algorithm 1.

Theorem 6. *For discrete-time ITPNs, the algorithm 1 is correct w.r.t. marking and state reachability and language. The termination is ensured for discrete-time bounded ITPNs.*

To prove this algorithm, we first have to introduce a corollary of theorem 4.

In [20], we studied the conditions such that general polyhedra (that cannot be written under the form of simple DBMs) appear in the state class graph of dense-time TPNs with stopwatches. We then deduce the following corollary :

Corollary 2. *Let $\mathcal{N}$ be a ITPN. We aim to determine the state space of the discrete-time semantics $\mathcal{S}_{\mathcal{N}}^{discrete}$ of the net by linking it to the on-the-fly computation of the state space of $\mathcal{S}_{\mathcal{N}}^{dense}$. As long as no class of $\mathcal{S}_{\mathcal{N}}^{dense}$ satisfies the necessary and sufficient condition (condition 3.2) exhibited in [20], we have the following properties:*

- *The discretization of the resulting state space of $\mathcal{S}_{\mathcal{N}}^{dense}$ gives states that all belong to the state space of $\mathcal{S}_{\mathcal{N}}^{discrete}$.*
- *Every untimed run of $\mathcal{S}_{\mathcal{N}}^{dense}$ that is identified is included in the set of all untimed runs of $\mathcal{S}_{\mathcal{N}}^{discrete}$.*

Then we deduce the correctness of the algorithm:

Proof. The correctness of algorithm 1 follows from the previous theorems of our paper, especially from theorem 4 and corollary 2. The convergence of the algorithm is a consequence of the decidability of the reachability problem for discrete-time bounded ITPNs.

Such an abstraction has many practical implications: it enables us to use algorithms and tools based on DBMs developed for TPNs to check properties on ITPNs with discrete time semantics. We use the tool Roméo [21] that has been developed for the analysis of TPN (state space computation and "on the fly" model-checking of reachability properties and TCTL properties) . For instance, it is possible to check real-time properties expressed in TCTL on bounded discrete-time ITPNs by a very simple adaptation of this tool.

On the one hand, in [22], the authors consider TCTL model-checking on TPNs. In fact, their algorithms apply on all Petri nets extended with time such that the firing domains of all state classes are DBMs. On the other hand, algorithm 1 states how to compute the state space of discrete-time ITPNs by using only DBMs. The combination of the two procedures leads to an elegant way to model-check TCTL formulae on discrete-time ITPNs. The implementation in Roméo leads to quite nice results. To illustrate the merits of our work, we provide a benchmark that compares the efficiency, in terms of computation speed, of the state space computation using the enumerative technique we introduced in [7] with the symbolic algorithm we propose in this paper.

All examples depict bounded nets. When the intervals associated to transitions are small (example 2), the enumerative method is more efficient than the symbolic algorithm. This is due to the power of BDDs based techniques available on the enumerative approach. Nevertheless, when the net contains one (or

Net	Symbolic algorithm - Roméo		Enumerative algorithm - Markg	
	Time	Memory	Time	Memory
Ex 1	0.12 s	1 320 KB	1.03 s	96 032 KB
Ex 2	34.42 s	1 320 KB	2.95 s	111 700 KB
Ex 3	45.12 s	20 012 KB	NA	NA
Ex 4	0.52 s	2 940 KB	1.07 s	95 796 KB
Ex 5	0.15 s	1 320 KB	1 148.18 s	139 800 KB

Table 1. Comparison between symbolic algorithm (with Roméo) and enumerative techniques (with Markg) to compute the state space of discrete-time nets (Pentium ; 2 GHz; 2GB RAM

more) transitions with a wide range betweebn earliest and latest firing times (all examples, except the second one, contain a transition whose firing interval is $[0, 1000]$), the enumerative method suffers from combinatorial explosion, while our symbolic algorithm leads to good results.

5 Conclusion

In this paper, we have considered an extension of T-time Petri nets with stopwatches (SwPNs). In this model, stopwatches are associated with transitions: a stopwatch can be reset, stopped and started by using inhibitor arcs. This allows the memorisation of the progress status of an action that is stopped and resumed. Our methods and results have been illustrated on Time Petri Nets with Inhibitor arcs (ITPNs). They are however general enough to be applied on the whole class of SwPNs.

We aimed to extend to discrete-time semantics the symbolic methods usually applied for the computation of the state space of a dense-time TPN (with stopwatches) and then perform TCTL model-checking. The approach we introduce consists in linking the state space of a discrete-time net to the discretization of the state space of the associated dense-time model. This method is correct for a subclass of SwPNs (including TPNs), but not for the general class of SwPNs. We thus propose a more general method for computing symbolically the state space of discrete-time nets. It is based on the decomposition of the general polyhedra that encompass the temporal information of the net into a union of simpler polyhedra. The subtlety of the symbolic algorithm depends on the function that splits the polyhedra into a union of DBMs: shall the polyhedron be split in a minimum number of DBMs? Are some splitting procedure more efficient than others for a long-term computation? What is the cost of such algorithms? These questions are the basis of our current investigations.

Further work consist in investigating the efficiency of several algorithms that split a general polyhedra in unions of DBMs.

References

1. Merlin, P.: A study of the recoverability of computing systems. PhD thesis, Department of Information and Computer Science, University of California, Irvine, CA (1974)
2. 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.
3. Roux, O., Déplanche, A.M.: A t-time Petri net extension for real time-task scheduling modeling. European Journal of Automation (JESA) **36** (2002) 973–987
4. Bucci, G., Fedeli, A., Sassoli, L., Vicario, E.: Time state space analysis of real-time preemptive systems. IEEE transactions on software engineering **30** (2004) 97–111
5. Roux, O.H., Lime, D.: Time Petri nets with inhibitor hyperarcs. Formal semantics and state space computation. In: The 25th International Conference on Application and Theory of Petri Nets, (ICATPN'04). Volume 3099 of Lecture Notes in Computer Science., Bologna, Italy, Springer (2004) 371–390
6. Berthomieu, B., Lime, D., Roux, O.H., Vernadat, F.: Reachability problems and abstract state spaces for time petri nets with stopwatches. Journal of Discrete Event Dynamic Systems (DEDS) (2007) To appear.
7. Magnin, M., Molinaro, P., Roux, O.H.: Decidability, expressivity and state-space computation of stopwatch petri nets with discrete-time semantics. In: 8th International Workshop on Discrete Event Systems (WODES'06), Ann Arbor, USA (2006)
8. Popova, L.: On time petri nets. Journal Inform. Process. Cybern., EIK (formerly: Elektron. Inform. verarb. Kybern.) **27** (1991) 227–244
9. Molinaro, P., Delfieu, D., Roux, O.H.: Improving the calculus of the marking graph of Petri net with bdd like structure. In: 2002 IEEE international conference on systems, man and cybernetics (SMC 02), Hammamet, Tunisia (2002)
10. Berthomieu, B., Diaz, M.: Modeling and verification of time dependent systems using time Petri nets. IEEE transactions on software engineering **17** (1991) 259–273
11. Gardey, G., Roux, O.H., Roux, O.F.: State space computation and analysis of time Petri nets. Theory and Practice of Logic Programming (TPLP). Special Issue on Specification Analysis and Verification of Reactive Systems (2006) to appear.
12. 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). Volume 2619 of Lecture Notes in Computer Science., Warsaw, Poland, Springer Verlag (2003) 442–457
13. Boucheneb, H., Gardey, G., Roux, O.H.: TCTL model checking of time Petri nets. Technical Report IRCCyN number RI2006-14 (2006)
14. Lime, D., Roux, O.: Expressiveness and analysis of scheduling extended time Petri nets. In: 5th IFAC International Conference on Fieldbus Systems and their Applications, (FET 2003), Aveiro, Portugal, Elsevier Science (2003)
15. Berthomieu, B., Menasche, M.: An enumerative approach for analyzing time Petri nets. IFIP Congress Series **9** (1983) 41–46
16. Dill, D.: Timing assumptions and verification of finite-state concurrent systems. In: Workshop Automatic Verification Methods for Finite-State Systems. Volume 407. (1989) 197–212
17. Popova, L.: On time petri nets. Journal Information Processing and Cybernetics **27** (1991) 227–244

18. Popova-Zeugmann, L.: (Essential states in time petri nets)
19. Popova-Zeugmann, L., Schlatter, D.: Analyzing paths in time petri nets. Fundamenta Informaticae **37** (1999) 311–327
20. Magnin, M., Lime, D., Roux, O.: An efficient method for computing exact state space of Petri nets with stopwatches. In: third International Workshop on Software Model-Checking (SoftMC'05). Electronic Notes in Theoretical Computer Science, Edinburgh, Scotland, UK, Elsevier (2005)
21. Gardey, G., Lime, D., Magnin, M., Roux, O.H.: Roméo: A tool for analyzing time Petri nets. In: 17th International Conference on Computer Aided Verification (CAV'05). Volume 3576 of Lecture Notes in Computer Science., Edinburgh, Scotland, UK, Springer (2005)
22. Hadjidj, R., Boucheneb, H.: On-the-fly tctl model checking for time petri nets using state class graphs. In: ACSD, IEEE Computer Society (2006) 111–122