Safety Control Synthesis for Time Petri Nets

Guillaume Gardey*, Olivier (F.) Roux† and Olivier (H.) Roux †
*LaBRI - CNRS UMR 5800
Domaine Universitaire
351, cours de la Libération – 33 405 Talence Cedex – France
Email: gardey@labri.fr
†IRCCyN – UMR CNRS 6597
Université de Nantes, École Centrale de Nantes, École des Mines de Nantes, CNRS
1, rue de la Noë B.P. 92101 – 44321 NANTES cedex 3 – France
Email: {olivier.roux,olivier-h.roux}@irccyn.ec-nantes.fr

***Abstract*— We study some control synthesis problems on an extension of Time Petri Nets that model a plant and its environment. The Time Petri Net control model both represents controllable and uncontrollable events, the problem is then to design a function (*controller*) such that a given property is fulfilled. We focus our analysis on safety properties expressed on the markings of the net and we propose a symbolic method to decide the existence of a controller that ensures these properties. Unlike existing methods on Time Petri Nets, that assume the net is bounded, the method is applicable for any Time Petri Nets. A consequence is that it is possible to decide the existence of a controller that k-bounds the plant. A method is then proposed to build a state-based controller and problems raised by the implementation (*Zenoness, sampling*) of the control function on the plant are discussed.**

Works supported by an ACI grant (French Government) on Control and Observation of Real-Time Open Systems (CORTOS, http://www.lsv.ens-cachan.fr/aci-cortos)

INTRODUCTION

The *Supervisory control theory*, firstly developed by Ramadge and Wonham [1], provides a general framework to solve the control problem for Discrete Event Systems. Unlike the *model-checking* problem in which systems are mainly represented as standalone system (*open-loop* systems), the control framework models the interaction between a system and its environment. The problem is to design an agent (frequently named *controller*, *supervisor* or *strategy*) that ensures a given specification is valid whatever the behavior of the system is (*closed-loop* system).

In a real-time framework, the plant is modeled using discrete or continuous clocks. Whereas time discretization generally aims at transforming the model into a Discrete Event System, the state-space of a dense-time model is infinite. Consequently, symbolic methods (time abstractions) were developed to study such systems. They were applied to Time Petri Nets [2] and Timed Automata [3] for model-checking [3]–[5] or control synthesis [6]. Unlike Timed Automata which have a finite discrete state-space structure (*localities*), the set of reachable *markings* of a Time Petri Net is generally infinite and this property is undecidable [7]. Consequently, to our knowledge, all Time Petri Nets results only hold for bounded Time Petri Nets and generally only for Time Petri Nets with a *safe* untimed structure *i.e.* no more than one token by place, reducing so the expressiveness offered by Time Petri Nets.

Since the goal of the control synthesis is to provide a function to be implemented on the plant to be controlled, the implementability of the control function has to be studied accurately. Two main problems are raised when implementing a control function: *non-Zenoness* and *sampling*. A system is said to be *Zenon* if it can perform infinitely many discrete actions that can prevent time from diverging (in a null time for instance). Such systems are not implementable in practice and conditions to ensure the *non-Zenoness* must be found. On the other hand, a real controller is not generally a dense-time controller: the system is sampled and the controller get input values (sensors, clocks values) from the plant to be controlled at these sample times. Consequently, the problem of the existence of a sampled controller has to be solved.

Related Works

Petri Nets: The control theory provided by Ramadge and Wonham [1] not only applies to Finite State Machine but to any formalism that models a Discrete Event System. Petri Net *Generators* (*labeled* Petri Nets) were then introduced to model Discrete Event System and benefit from the Ramadge and Wonham control framework. However, Petri nets are generally used to describe a system by its state (markings) rather than its language and the control specification is expressed on the reachable markings. Generalized Mutual Exclusion Constraints(GMECs) [8], [9] were introduced to formalize properties on the markings of a Petri net. Several solutions were proposed to solve a control problem expressed using GMECs. Holloway and Krogh [10], [11] proposed an efficient method using both off-line and on-line computation for a class of *safe marked graph*. Li and Wonham [12], [13] developed an *Integer Programming Problem*(IPP) equivalent to the control synthesis of *Vector Discrete Event Systems* that are an equivalent model to Petri Nets. However, since IPP is an open problem the method is not realizable for real systems. Giua *et al.* [8] and Moody *et al.* [14], [15] proposed a solution to the GMECs problem using *monitors* to synthesise a Petri Net that models the closed-loop system. Their

works differ in the property of *maximally permissiveness* of their closed-loop systems: Moody *et al.* developed a solution that does not necessarily compute a maximally permissive controller.

Timed Automata: Recent researches have presented symbolic control synthesis algorithms for Timed Automata. In [16], Wong-toi and Hoffman proposed to solve the synthesis control problem by discretizing the Timed Automaton and applying the classical framework for Discrete Event Systems. However, this method suffers from a combinatorial states number explosion due to the time discretization. In [6] Maler *et al.* use a game approach to compute a strategy without needs of discretization for reachability or safety control requirements. A state-base controller can then be extracted in the form of a Timed Automata. This method was generalized or extended to other models (Infinite State Games, Hybrid Automata, Priced Timed Automata).

Time Petri Nets: Despite active researches for untimed models, few studies are available to our knowledge for the control synthesis problem.

Sathaye and Krogh [17], [18]: They define a control extension of Time Petri Nets (Controlled Time Petri Nets) whose transitions are divided into untimed controllable and uncontrollable transitions. They assume the Petri net is safe and *non-zeno* by imposing each transition to be fired after a strict positive delay. They extend the classical State Class Graph of Berthomieu and Diaz [4] to a Control Class Graph.

Chen and Hanish [19]: They propose to compute a state-based controller for safe arc-time Petri nets based on predicate invariance. This method inspired by the approach of Maler *et al.* [6] consists in iteratively computing symbolic states verifying the requirements. In a first time, the control specification is given as a linear predicate on markings and clocks valuations, then, requirements are extended to labeled arc-time Petri nets.

Our Contribution

In this paper, we consider Time Petri Nets for which reachability and boundedness are undecidable. We provide a more general control model for Time Petri Nets than the ones in [17], [19]. A symbolic method based on the notion of controllable predecessors [6] is then shown to prove the decidability of some safety control problems. The controller is synthesized as a function over the state space. The question of the implementability of the function on a real plant is finally discussed (zenoness and sampling).

I. Time Petri Nets

In a first time, we recall the definition and semantics of Merlin's Time Petri Nets [2].

A. Definition

Time Petri nets (TPN) are a time extension of classical Petri nets. Informally, with each transition of the net is associated a clock and a time interval. The clock measures the time since the transition has been enabled and the time interval is interpreted as a firing condition: the transition may fire if the value of its clock belongs to the time interval. If the clock reaches the upper bound of the interval, the transition has to fire.

Formally:

Definition 1 (Time Petri Nets): A Time Petri Net is a tuple $N = (P, T, {}^\bullet(.), (.)^\bullet, \alpha, \beta, M_0)$ defined by:

- $P = \{p_1, p_2, \ldots, p_m\}$ is a non-empty finite set of *places*,
- $T = \{t_1, t_2, \ldots, t_n\}$ is a non-empty finite set of *transitions*,
- ${}^\bullet(.) : T \to \mathbb{N}^P$ is *the backward incidence function*,
- $(.)^\bullet : T \to \mathbb{N}^P$ is *the forward incidence function*,
- $M_0 \in \mathbb{N}^P$ is *the initial marking* of the Petri Net,
- $\alpha : T \to \mathbb{Q}_{\geq 0}$ is the function giving *the earliest firing times* of transitions,
- $\beta : T \to \mathbb{Q}_{\geq 0} \cup \{\infty\}$ is the function giving *the latest firing times* of transitions.

A Petri Net marking M is an element of $\mathbb{N}^P$ such that for all $p \in P$, $M(p)$ is the number of tokens in the place p.

A marking M enables a transition t if : $M \geq {}^\bullet t_i$. The set of transitions enabled by a marking M is noted $enabled\,(M)$.

A transition t_k is said to be *newly enabled* by the firing of a transition t_i if $M - {}^\bullet t_i + t_i^\bullet$ enables t_k and $M - {}^\bullet t_i$ does not enabled t_k. If t_i remains enabled after its firing then t_i is newly enabled. The set of transitions newly enabled by a transition t_i for a marking M is noted $\uparrow enabled\,(M, t_i)$.

$v \in (\mathbb{R}_{\geq 0})^T$ is a valuation of the system. v_i (also noted $v\,(t_i)$) is the time elapsing since the transition t_i has been newly enabled. $\overline{0}$ represents the nul valuation.

B. Semantics

The semantics of a Time Petri Net is defined as a Timed Transition System. Firing a transition is a discrete transition of the Timed Transition System, idling in a marking, the continuous transition.

Definition 2 (Semantics of a Time Petri Net): The semantics of a Time Petri Net is defined by the Timed Transition System $\mathcal{S} = (Q, q_0, \to)$:

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

The continuous transition $(M, v) \xrightarrow{d} (M, v')$ is defined $\forall d \in \mathbb{R}_{\geq 0}$ *iff* :

$$\begin{cases} v' = v + d \\ \forall k \in [1, n],\ M \geq {}^\bullet t_k \Rightarrow v'_k \leq \beta(t_k) \end{cases}$$

The discrete transition $(M, v) \xrightarrow{t_i} (M', v')$ is defined $\forall t_i \in T$ *iff* :

$$\begin{cases} M \geq {}^\bullet t_i \wedge M' = M - {}^\bullet t_i + t_i^\bullet \\ \alpha(t_i) \leq v_i \leq \beta(t_i) \\ \forall k \in [1, n]\ v'_k = \begin{cases} 0 \text{ if } t_k \in \uparrow enabled\,(M, t_i) \\ v_k \text{ otherwise} \end{cases} \end{cases}$$

We note $Reach(N)$ the set of reachable markings of a Time Petri Net N and for $X \in Q$, $\mathcal{M}(X)$ the set of distinct

markings of X. A *run* ρ of a Time Petri Net is a path in the Timed Transition System starting from q_0 and ending in a state $q \in Q$ ($q_0 \rightarrow q_1 \rightarrow \cdots \rightarrow q_n = q$). We note Q^+ the set of runs of a Time Petri Net.

C. Main theorems

We recall in this section some main definitions and decidability results for Time Petri Nets.

Theorem 1 (Reachability [7]): Given a Time Petri Net N with initial marking M_0 and a marking M, the existence of a run ρ such that $M_0 \xrightarrow{\rho} M$ is undecidable.

Definition 3 (Boundedness): A Time Petri Net N is *bounded* if it exists an integer k such that for all reachable marking M, $\forall p \in P\ M(p) \leq k$.

Theorem 2 ([7]): The boundedness of a Time Petri Net is undecidable.

In particular, the boundedness of the underlying untimed Petri net is only a sufficient condition of boundedness. There exists bounded Time Petri Nets whose underlying untimed Petri net is not bounded. In the remaining of the paper we will consider Time Petri Net not necessarily bounded.

Definition 4 (k-boundedness): A Time Petri Net N is k-bounded for $k \in \mathbb{N}$ if for every reachable marking M, $M(p) \leq P\ \forall p \in P$.

Theorem 3 ([4]): The k-boundedness is decidable.

However, the state space of the Time Petri Net must be built to decide if it is k-bounded for a given value of k. A method will be proposed in section 3 to build a controller ensuring the boundedness of the closed-loop system.

II. Control synthesis problems for Time Petri Nets

In this section, we present our Time Petri Net control model as well as some control problems.

A. Control model for Time Petri Nets

As usual, the set of transitions is partitioned into *controllable* (T_c) and *uncontrollable* (T_u) transitions: $T = T_c \cup T_u, T_c \cap T_u = \emptyset$. Uncontrollable transitions model actions of the plant that can occur at any time whereas controllable transitions both model: (1) plant's actions that can be forced at a given time or (2) actions specific to the controller (alarm...). In [20], an other classification of transitions is proposed: *prospective* (bounded upper time) and *remote* transitions (unbounded upper time). This classification is used in [19] by assuming that all prospective transitions are uncontrollable. In [19], no time interval is associated with a controllable transition: if an enabled control transition is chosen by the controller the transition is instantaneously fired.

Our model assumes that prospective transitions may be controllable or not and that controllable transitions are associated with a time interval. This more general framework allows to represent different type of transitions:

- *uncontrollable transitions*,
- *totally controllable transitions*: they are associated with the interval $[a, \infty[, a \in \mathbb{Q}_{\geq 0}$, which means that the controller can choose to never fire the transition,
- *partially controllable transitions*: they are associated with a bounded time interval $[a, b]$ (remote transitions), which means that the controller can select some good subintervals to fire the transition but can not delay the firing forever.

Finally, we do not suppose the Time Petri Net to be safe [17]–[19] or bounded .

B. Controller

In its more general form, the controller is defined as a function over the set of runs of the Time Petri Net.

Definition 5: A Time Petri Net controller is a partial function $S : Q^+ \rightarrow 2^{T_c} \cup \{\lambda\}$.

The special action λ represents a special action that consists in letting the environment evolve on its own without action from the controller. λ is defined as follows:

$$\forall q \in Q^+,\ (\lambda \in S(q)) \implies (\exists t > 0 \mid \forall \delta < t,\ \lambda \in S(q+\delta))$$

i.e. there exists a strict positive amount of time meanwhile the controller can wait and let the plant evolve on its own.

At any state $q_n = (M, v)$, according to the trace $\rho = (q_0 \rightarrow q_1 \rightarrow \cdots \rightarrow q_n) \in Q^+$ leading to that state, the controller chooses whether to fire a controllable transitions or not (λ).

The controller is non-deterministic: a control action is randomly chosen into the set of possible actions.

Definition 6 (State-based controller): A controller is *state-based* (also named *memory-free* or *feedback* controller) if $S : Q \rightarrow 2^{T_c} \cup \{\lambda\}$, that is, only the current state of the system is needed to control it.

The closed-loop system obtained by applying the controller S to the Time Petri Net N is noted S/N.

C. Semantics

The semantics of the closed-loop system is defined as a Timed Transition System:

Definition 7 (Semantics of the closed-loop system): Let N be a Time Petri Net and S a control function. The semantics of the closed-loop system is defined by the Timed Transition System $\mathcal{S} = (Q, q_0, \rightarrow)$:

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

The continuous transition $(M, v) \xrightarrow{d} (M, v')$ is defined $\forall d \in \mathbb{R}_{\geq 0}$ *iff*:

$$\begin{cases} v' = v + d \\ \forall k \in [1, n],\ M \geq^\bullet t_k \Rightarrow v'_k \leq \beta(t_k) \\ \lambda \in S(M, v) \\ \forall t \leq d,\ \lambda \in S(M, v + t) \end{cases}$$

The discrete transition $(M, v) \xrightarrow{t_i} (M', v')$ is defined $\forall t_i \in T$ *iff*:

$$\begin{cases} M \geq^{\bullet} t_i \\ M' = M -^{\bullet} t_i + t_i^{\bullet} \\ \alpha(t_i) \leq v_i \leq \beta(t_i) \\ \forall k \in [1,n]\ v'_k = \begin{cases} 0 \text{ if } t_k \in \uparrow enabled\,(M, t_i) \\ v_k \text{ otherwise} \end{cases} \\ t_i \in S(M,v) \vee t_i \in T_u \end{cases}$$

D. Control Problems

In Petri Nets and Time Petri Nets frameworks, properties are focused on markings and mainly safety properties. We consider control specifications given as finite sets of markings $\mathcal{M}_c$ in which the system must stay in, whatever the environment acts.

Definition 8 (Safety control problem (1)): Given a Time Petri Net N and a finite set of markings $\mathcal{M}_c$, does it exist a controller S such that $Reach(S/N) \subseteq \mathcal{M}_c$?

Since exhaustive enumeration of reachable markings is costly (state space computation), the specification can also be expressed using Generalized Mutual Exclusion Constraints (GMECs) [9].

*Definition 9 (*GMEC*):* Let N be a Petri net with a set of places P, $\overrightarrow{w} : P \to \mathbb{Z}$ a weight of integers and $k \in \mathbb{Z}$ a constant. A single Generalized Mutual Exclusion Constraint $(\overrightarrow{w}, k)$ defines a set of legal markings on N $\mathcal{M}(\overrightarrow{w}, k) = \left\{M \in Reach(N) \mid \overrightarrow{w}^T.M \leq k\right\}$. A set of GMECs $(W, \overrightarrow{k})$ with $W = [\overrightarrow{w_1} \dots \overrightarrow{w_r}]$ and $\overrightarrow{k} = (k_1 \dots k_r)^T$, defines a set of legal markings $\mathcal{M}(W, \overrightarrow{k}) = \left\{M \in Reach(N) \mid W^T.M \leq \overrightarrow{k}\right\}$.

*Definition 10 (*GMECs *control problem (1)):* Let N be a Time Petri Net and $(W, \overrightarrow{k})$ a set of GMECs, does it exist a controller S such that for all $M \in Reach(S/N)$, $W^T.M \leq \overrightarrow{k}$?

Definition 11 (k-boundedness problem): Let N be a Time Petri Net and k an integer, does it exist a controller S such that S/N is k-bounded?

This is a particular case of the GMECs control problem with $\overrightarrow{W} = (1 \dots 1)$ and $\overrightarrow{k} = (k \dots k)$.

III. Controllability

The existence of a controller (and its synthesis) is solved using a backward fixpoint algorithm that computes controllable states *i.e.* states from which only safe states are reachable under the action of the controller. This computation of controllable safe states is based on the definition of controllable predecessors [6].

A. Preliminaries: Controllable Predecessors

Let $X \subseteq Q$ be a set of states of a Time Petri Net N. The set of controllable predecessors of X is defined by:

$$\begin{aligned} \pi(X) = \{q \in Q \mid ((&\exists \delta \in \mathbb{R}_{\geq 0}, t \in T, q' \in X\ q \xrightarrow{\delta} \xrightarrow{t} q') \\ &\vee (\exists \delta \in \mathbb{R}_{\geq 0}, q' \in X\ q \xrightarrow{\delta} q')) \\ &\wedge \forall \delta_u \in \mathbb{R}_{\geq 0} \text{ if } \exists t_u \in T_u, q'_u \notin X\ q \xrightarrow{\delta_u} \xrightarrow{t_u} q'_u \\ &\text{then } \exists \delta_c < \delta_u, t_c \in T_c, q'_c \in X\ q \xrightarrow{\delta_c} \xrightarrow{t_c} q'_c\} \end{aligned}$$

Informally, q is a controllable predecessor of X iff:

- a state q' of X is reachable by time elapsing and firing of a transition (figure 1),
- for any uncontrollable transition diverging from X, the controller can take a decision at an earlier date to constraint the system in X (figure 2).

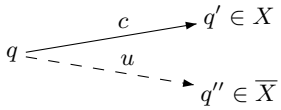

Fig. 1. Discrete predecessors

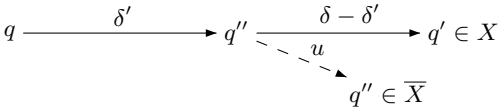

Fig. 2. Controllable time predecessors

Now, let $S \subseteq Q$, we consider and note $\mathcal{A}_s(S)$ the semi-algorithm that computes the greatest fixpoint of $\pi(X)$, *i.e.* the set of all states that are controllable safe predecessors of S.

Algorithm 1 $\mathcal{A}_s(S)$

$X_0 \leftarrow S$
repeat
  $X_{i+1} \leftarrow X_i \cap \pi(X_i)$
until $X_{i+1} = X_i$

B. Decidability Results

In this section, we prove the decidability of the control problems presented in section 2.

First, we note $\mathcal{R}^*$ the set of *regions* [3]. This set of regions defines a finite partition of $\mathbb{R}^T_{\geq 0}$ and was used to prove the decidability of TCTL model-checking on Timed Automata by Alur and Dill [3].

We first prove the following lemma:

Lemma 1: Given a Time Petri Net N and $X \subseteq \mathbb{N}^P \times \mathcal{R}^*$ a finite set of couples (marking, regions) then, if the set of distinct markings of X ($\mathcal{M}(X)$) is finite :

1) $\pi(X)$ is a finite set of $\mathbb{N}^P \times \mathcal{R}^*$.
2) $\pi(X)$ is defined over $Pre_T(\mathcal{M}(X)) \times \mathcal{R}^*$ where $\mathcal{M}(X)$ is the set of markings of X and $Pre_T(Y)$ the set of markings from which a marking of $M(X)$ is reachable by the firing of a transition of T ($Pre_T(Y) = \{M \in \mathbb{N}^P \mid \exists t \in T, \exists M' \in Y\ M -^{\bullet} t + t^{\bullet} = M'\}$).

Proof: Since $\mathcal{M}(X)$ and T are supposed to be finite sets, $Pre_T(\mathcal{M}(X))$ is a finite set of markings.

Let $r \subseteq \mathcal{R}^*$ be a finite set of regions, it was proven in [21] that $\pi(r)$ can be expressed as a finite subset of $\mathcal{R}^*$.

Consequently, $\pi(X)$ is a finite set of $\mathbb{N}^P \times \mathcal{R}^*$. ■

We can then prove the following theorem :

Theorem 4 (Safety Control Problem): The existence of a solution to the *Safety Control Problem* is decidable.

Proof: Let $S \subseteq \mathbb{N}^P \times \mathcal{R}^*$ a finite set of safe states. $\pi(X)$ is a monotone function ($X_1 \subseteq X_2 \Rightarrow \pi(S_1) \subseteq \pi(S_2)$) and according to lemma 1, all elements $((\pi(X_i))_i$ are defined over the finite domain $Pre_T(\mathcal{M}(X)) \times \mathcal{R}^*$. Consequently, the fix-point algorithm terminates in a finite number of steps. If the set of initial states Q_0 intersects the solution X^* then there exists a controller that ensures the system stays in the set X^*. ■

It immediatly follows that :

*Corollary 1 (*GMECs *control problem):* Let N be a Time Petri Net and $(W, \overrightarrow{k})$ a set of GMECs defining a finite set of markings, the GMECs *control problem* is decidable.

*Corollary 2 (*GMECs *control problem):* Let N be a bounded Time Petri Net and $(W, \overrightarrow{k})$ a set of GMECs, the "GMECs control problem" is decidable.

Corollary 3 (k-boundedness): Given a Time Petri Net N and an integer k, the existence of a controller S such that S/N is k-bounded is decidable.

Proof: It suffices to consider the GMEC that bounds by k all the markings and to use the corollary 1. ■

It has to be noted that a control requirement expressed using *forbidden* markings (*i.e.* states to avoid) is also solved by this method. In this case, $\mathcal{A}_s$ is generally a semi-algorithm since backward reachability does not necessarily end. However if the Time Petri Net is bounded, it falls back into the scope of theorem 4.

IV. CONTROL SYNTHESIS

This section is dedicated to the synthesis of a *state-based controller*. Under the hypothesis of the existence of a controller given by $\mathcal{A}_s$, the control function is extracted from the set of states computed by the algorithm.

A. Synthesis of a feedback controller

Theorem 5 (Synthesis of a controller): If $\mathcal{A}_s$ converges, there exists a *state-based controller* S that ensures the control specification.

Proof: Let X^* be the solution of $\mathcal{A}_s$. We note $X^*_{|M}$ the set of states associated with the marking M: $X^*_{|M} = \{q' = (M', v') \in Q \mid M' = M\}$.

For each marking of the solution X^* we compute the control function as follows:

for all $M \in X^*$ **do**
  for all $M' \in X^* \mid M \xrightarrow{t} M'\ t \in T_c$ **do**
    $D = X^*_{|M} \cap Pred_\delta(Pred_t(X^*_{|M'}))$
    $\partial D = \{v \in D \mid \exists t > 0,\ v + t \in D\}$
    $S(D \setminus \partial D) = S(D \setminus \partial D) \cup \{t, \lambda\}$
    $S(\partial D) = S(\partial D) \cup \{t\}$

This algorihtm computes for each marking the subset of the set of controllable predecessors for which controllable transitions can be safely fired. ■

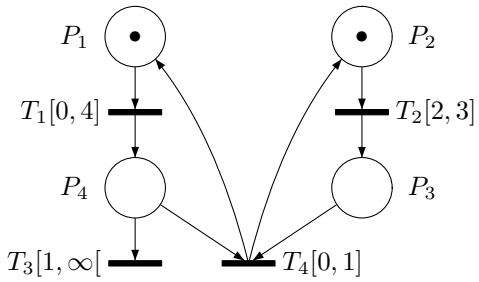

Fig. 3. Example

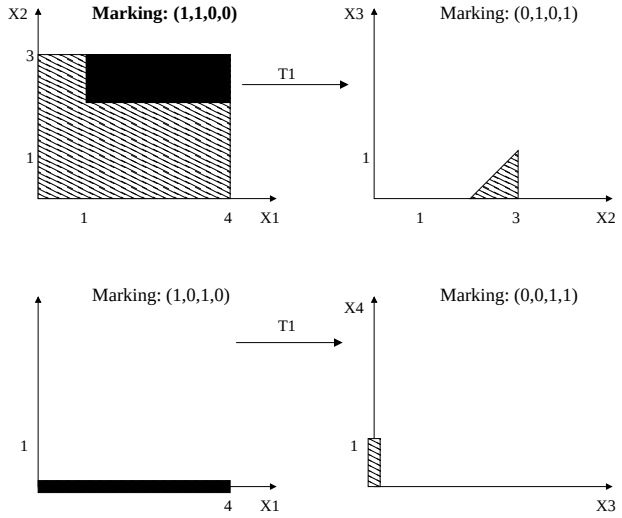

Fig. 4. Control function

B. Example

We consider the TPN of the figure 3 for which only the transition T_1 is controllable. The property want to ensure is that the transition T_3 is never fired. One can see that this property can be expressed as a safety property on markings : if the transition T_3 is fired, the token moving from P_1 to P_4 disappeared. If the transition T_3 is never fired, there are always two tokens in the net. Then, the corresponding safe set of markings is: $\{(1,1,0,0), (1,0,1,0), (0,0,1,1), (0,1,0,1)\}$. The controller must ensure this specification and control the firing of T_1 to prevent the firing of T_3

In a first time the set of controllable predecessors of the set of safe markings is computed and illustrated in the figure 4 (in grey) (table I). Let us recall that these sets of states represent the states of controllable predecessors solution of the fix-point algoirthm, that is, states of the Time Petri Nets for which there exist a solution for a controller to enfore the system in the safe set of markings. A additional step is needed to extract from these sets of controllable predecessors a control function ensuring the control requirement. Indeed, at marking $(1,1,0,0)$ the transition T_1 can be fired at time 0 whereas its firing would unavoidably lead to the firing of T_3.

Marking	Clocks valuations
$(1,1,0,0)$	$0 \leq X_1 \leq 4 \wedge 0 \leq X_2 \leq 3$
$(0,1,0,1)$	$X2 \leq 3 \wedge X_3 \geq 0 \wedge X_2 > X_3 + 2$
$(1,0,1,0)$	$0 \leq X_1 \leq 4$
$(0,0,1,1)$	$X_4 = 0 \wedge X_3 \geq 0 \wedge X_3 < 1$

TABLE I

CONTROLLABLE PREDECESSORS

Marking	Clocks valuations	Control Function
$(1,1,0,0)$	$2 < X_2 < 3 \wedge 1 \leq X_1 < 4$	$\{\lambda, T_1\}$
$(1,1,0,0)$	$(X_1 = 4 \wedge 2 < X_2 \leq 3) \vee (X_2 = 3 \wedge 1 \leq X_1 \leq 4)$	$\{T_1\}$
$(1,0,1,0)$	$0 \leq X_1 < 4$	$\{T_1, \lambda\}$
$(1,0,1,0)$	$X_1 = 4$	$\{T_1\}$

TABLE II

CONTROL FUNCTION

The control function is extracted from this set of states using the algorithm of theorem 5 : it extracts from the set of controllable predecessors, states that effectively lead to a safe state by the firing of a controllable transition. For instance, firing the transition T_1 in the marking $(1,1,0,0)$ at time $X_1 = 0$ is not allowed since it wouldn't generate a state belonging to the set of controllable predecessors computed for the marking $(0,1,0,1)$.

The regions in black in the figure 4 (table II) represents the set of states in which the transition T_1 is enabled by the controller.

V. IMPLEMENTATION

A. Zenoness

In a time framework, time blocking (*zenoness*) is an important issue. A Time Petri Net is *zeno* if there exists an infinite sequence of transitions (in a null time for instance) that prevents time from diverging.

Since no hypothesis were made on the Time Petri Net used to model the studied plant, it may contain zeno runs. Consequently, the control function may also generate zeno runs. This is all the more problematic that zeno runs may be the only solution for the system to be controllable.

Let us consider the Time Petri Net of the figure 5. T_1 is controllable and T_2 is not.

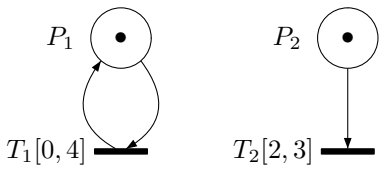

Fig. 5. T-TPN with zenon runs

The control specification is that the system always remain in the marking $(1,1)$ *i.e.* T_2 is never fired.

To ensure the property, the controller will constraint the firing of T_1 to $[0,2[$. Then, there exists a set of zeno runs that prevent the clock of T_2 from reaching 2.

If a non-null earliest firing time Δ is provided for T_1, then no controller exists: at each firing of T_1, the clock T_2 grows at least of Δ and unavoidably reach 2 after a finite number of T_1 firings.

Detecting zenoness is then important to study if the control function is implementable on a plant.

Theorem 6 (Sufficient Condition for non-zenoness (1)): If the Time Petri Net is such that all earliest firing dates are non-null then, if a controller exists, it is non-zeno.

The proof is straightforward since the set of runs of S/N is included in the one of N.

As zenoness is decidable for bounded Time Petri Net [22], the following less restrictive sufficient condition holds:

Theorem 7 (Sufficient Condition for non-zenoness (2)): If a bounded Time Petri Net is non-zeno then, if a controller exists, it is non-zenon.

Under the hypothesis of a zeno Time Petri Net, the controller may be studied to detect the existence of zenon path. If S/N is non-zeno the controller may be implemented.

B. Sampling Control

The control function built in this paper is state-based and non-deterministic: given the current state of the plant, the controller chooses a control action among the possible ones suggested by the control function.

Although a Time Petri Net models a real-time plant, the control function is generally not implemented by a continuous function: clocks variables and the state of the system is not monitored continuously. The plant is sampled at a fix rate and the controller takes a decision at those sample times according to the current inputs. We suppose the controller response to be instantaneous (relatively to the time constant of the plant).

Given a value of the sample rate Δ, the problem is then to state about the existence of a controller that must take a decision each Δ time units. Unlike a full sampled approach by time discretization to the control synthesis that suffers of a states number explosion, the symbolic method developed in this paper allows to state about the existence of a dense-time controller. A discrete method can then be used to determine a new controller or extract a sampled controller from the dense-time controller.

Since the control function is non-deterministic, two main approaches may be followed to implement the control on the plant:

- **Synchronous**. The control implementation enables the firing of all control actions that can occur in a null time.
- **Sampled Asynchronous**. The control implementation enables only a control action at each sample time.

All behaviors between synchronous and asynchronous are also realizable and dependent of the plant: a set of synchronous transitions can be fired in a null time way whereas a strict positive amount of time must separate two transitions of a sampled asynchronous set of transitions.

This method can be the used in a design process by refinement. The plant is modeled by a dense-time model (Time Petri Net) and its controllability with respect to a safety property is decidable at a low cost in comparison to model that would analyze the sampled system.

We can sum-up our results in the following theorem:

Theorem 8 (Sampled Controller): Given a plant P, Δ a sampled rate and a behavior B of P, the existence of a Δ-sampled controller that ensures a safety property is decidable. The Δ-sampled state-based controller can be computed and is implementable on the plant.

CONCLUSIONS

We present a method for the control synthesis of state-based controllers for Time Petri Nets that ensures a safety property on the set of markings of the net. Unlike existing methods, the proposed one is applicable to unbounded Time Petri Net and we prove that given an integer k and a Time Petri Net, it is possible to build a state-based controller that k-bounds it. In the case of an unbounded specification (for instance using a requirement expressed by forbidden states), our method provides a semi-algorithm for control synthesis.

Some solutions were proposed to solve the problems raised by the implementation of the control function on a real plant and we show how this method may be integrated in a top-bottom analysis of the system. A prototype was implemented to test the method and will be soon integrated into our Time Petri Net tool Romeo [23].

Control requirements expressed as a safe set of markings (or states) of the Time Petri Net may be not sufficient for a system designer: some language requirements are sometimes needed to express the properties of a system. We think the method is scalable to properties expressed as a Time Petri Net and are involved in this research. Besides, the question about the existence of a Δ-sampled controller (Δ not known *a priori*) is still pending. Finally, an extension of the work to control synthesis with partial observability should be explored.

REFERENCES

[1] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM J. Control Optim.*, vol. 25, no. 1, pp. 206–230, 1987.

[2] P. M. Merlin, "A study of the recoverability of computing systems," Ph.D. dissertation, Department of Information and Computer Science, University of California, Irvine, CA, 1974.

[3] R. Alur and D. L. Dill, "A theory of timed automata," *Theoretical Computer Science*, vol. 126, no. 2, pp. 183–235, 1994.

[4] B. Berthomieu and M. Diaz, "Modeling and verification of time dependent systems using time petri nets," *IEEE transactions on software engineering*, vol. 17, no. 3, pp. 259–273, March 1991.

[5] G. Gardey, O. H. Roux, and O. F.Roux, "Using zone graph method for computing the state space of a time petri net," in *Formal Modeling and Analysis of Timed Systems (FORMATS'2003)*, ser. LNCS. Marseille, France: Springer–Verlag, september 2003.

[6] O. Maler, A. Pnueli, and J. Sifakis, "On the synthesis of discrete controllers for timed systems," in *Proc. STACS '95*, ser. LNCS, E. Mayr and C. Puech, Eds., no. 900. Springer–Verlag, 1995, pp. 229–242.

[7] N. D. Jones, L. H. Landweber, and Y. E. Lien, "Complexity of some problems in petri nets," ser. Theoretical Computer Science, vol. 4, no. 3, 1977, pp. 277–299.

[8] A. Giua, F. DiCesare, and M. Silva, "Petri net supervisors for generalized mutual exclusio constraints," in *1993*, vol. 1, Sidney, Australia, jul, pp. 267–270.

[9] ——, "Generalized mutual exclusion constraints on nets with uncontrollable transitions," in *IEEE Int. Conf. on Systems, Man, and Cybernetics*, Chicago, Illinois, oct 1992, pp. 974–979.

[10] L. E. Holloway and B. H. Krogh, "Synthesis of feedback control logic for a class of controlled petri nets," in *IEEE Trans. on Automatic Control*, vol. 35, no. 5, may 1990, pp. 514–523.

[11] B. H. Krogh and L. E. Holloway, "Synthesis of feedback control logic for discrete manufacturing systems," in *Automatica*, vol. 27, no. 4, jul 1991, pp. 641–651.

[12] Y. Li and W. Wonham, "Control of vector discrete event systems i-the base model," *IEEE Transactions on Automatic Control*, vol. 38, no. 8, pp. 1214–1227, 1993.

[13] ——, "Control of vector discrete event systems ii- controller synthesis," *IEEE Transactions on Automatic Control*, vol. 39, no. 3, pp. 512–531, 1994.

[14] J. O. Moody and P. J. Antsaklis, "Petri net supervisors for des in the presence of uncontrollable and unobservable transitions," in *Proc. 33rd Annaul Allerton Conference*, oct 1995.

[15] K. Yamalidou, J. O. Moody, M. D. Lemmon, and P. J. Antsaklis, "Feedback control of petri nets based on place invariants," in *Automatica*, vol. 32, no. 1, 1996.

[16] H. Wong-Toi and G. Hoffmann, "The control of dense real-time dicrete event systems," in *Proc. 30th IEEE Conf. Decision and Control*, 1527–1528, 1991.

[17] A. S. Sathaye and B. H. Krogh, "Logical analysis and control of time petri nets," in *Proc. of the 31st Conf. on Decision and Control*, Tucson, Arizona, dec 1992, pp. 1198–1203.

[18] A. S. Sathaye, "Synthesis of real-time supervisors for controlled time petri nets," in *Proc. 32nd Conf. on Decision and Control*, San Antonio, Texas, dec 1993, pp. 235–236.

[19] H. Chen and H.-M. Hanisch, "Control synthesis of timed discrete event systems based on predicate invariance," in *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 30, no. 5, oct 2000, pp. 713–724.

[20] B. A. Brandin and W. M. Wonham, "Supervisory control of timed discret-event systems," in *IEEE Tras. Automat. Control.*, vol. 39, feb 1994, pp. 329–341.

[21] E. Asarin, O. Maler, and A.Pnueli, "Symbolic controller synthesis for discrete and timed systems," in *Hybrid Systems II*, ser. LNCS, Springer-Verlag, Ed., vol. 999, 1995, pp. 1–20.

[22] F. Cassez and O. H. Roux, "Structural translation from time Petri nets to timed automata," in *Fourth International Workshop on Automated Verification of Critical Systems (AVoCS'04)*, London (UK), SEP 2004.

[23] G. Gardey, D. Lime, M. Magnin, and O. H. Roux, "Romeo: a tool for analyzing time petri nets," in *Proc. 17th International Conference on Computer Aided Verfication (CAV'05)*, ser. Lecture Notes in Computer Science, vol. 3576. Edinburgh, Scotland, UK: Springer–Verlag, July 2005, pp. 418–423.