2013 20th International Symposium on Temporal Representation and Reasoning

A New Approach To Abstract Reachability State Space of Time Petri Nets

Kais Klai
Institut TELECOM SudParis
CNRS UMR Samovar, France
kais.klai@lipn.univ-paris13.fr

Naim Aber
Université Paris 13, Sorbonne Paris Cité
LIPN, CNRS UMR, France
naim.aber@lipn.univ-paris13.fr

Laure Petrucci
Université Paris 13, Sorbonne Paris Cité
LIPN, CNRS UMR, France
laure.petrucci@lipn.univ-paris13.fr

***Abstract*—Time Petri nets (TPN model) allow the specification of real-time systems involving explicit timing constraints. The main challenge of the analysis of such systems is to construct, with few resources (time and space), a coarse abstraction preserving timed properties. In this paper, we propose a new finite graph, called Timed Aggregate Graph (TAG), abstracting the behaviour of bounded TPNs with strong time semantics. The main feature of this abstract representation compared to existing approaches is the encoding of the time information. This is done in a pure way within each node of the TAG allowing to compute the minimum and maximum elapsed time in every path of the graph. The TAG preserves runs and reachable states of the corresponding TPN and allows for verification of both event- and state-based properties.**

***Keywords*-Real Time Systems; Time Petri Nets; State Space Abstraction; Model Checking of Timed Properties;**

I. Introduction

Time Petri nets are one of the most used formal models for the specification and the verification of systems where the explicit consideration of time is essential, such as communication protocols, circuits, or real-time systems. The main extensions of Petri nets with time are *time Petri nets* [15] and *timed Petri nets* [19]. In the first, a transition can fire within a time interval whereas, in the second, time durations can be assigned to the transitions; tokens are meant to spend that time as reserved in the input places of the corresponding transitions. Several variants of timed Petri nets exist: time is either associated with places (p-timed Petri nets), with transitions (t-timed Petri nets) or with arcs (a-timed Petri nets) [20]. The same holds for time Petri nets [7]. In [18], the authors prove that p-timed Petri nets and t-timed Petri nets have the same expressive power and are less expressive than time Petri nets. Several semantics have been proposed for each variant of these models. Here we focus on t-time Petri nets, which we simply call TPNs. There are two ways of letting the time elapse in a TPN [18]. The first way, known as the *Strong Time Semantics* (STS), is defined in such a manner that time elapsing cannot disable a transition. Hence, when the upper bound of a firing interval is reached, the transition must be fired. In contrast, the *Weak Time Semantics* (WTS) does not make any restriction on the elapsing of time.

For real-time systems, the dense time model (where time is considered in the domain $\mathbb{R}_{\geq 0}$) is the only possible option, raising the problem of handling an infinite number of states. In fact, the set of reachable states of the TPN is generally infinite due to the infinite number of time successors a given state could have. Two main approaches are used to treat this state space: region graphs [1] and the state class approach [3]. The other methods [2], [21], [4], [9], [5], [14], [6], [10] are either refinements or improvements of these basic approaches. The objective of these representations is to yield a state-space partition that groups concrete states into sets of states with similar behaviour with respect to the properties to be verified. These sets of states must cover the entire state space and must be finite in order to ensure the termination of the verification process. In this work, we propose a new contribution for the abstraction and the verification of finite TPN-based timed systems.

This paper is organised as follows: In Section II, some preliminaries about TPNs and the corresponding semantics are recalled. In Section III, we define the Timed Aggregate Graph (TAG) associated with a TPN and we discuss the main preservation results of the TAG-based approach. Section IV relates our work to existing approaches. In Section V, we discuss the experimental results obtained with our implementation compared to two well-known tools, namely Romeo [8] and TINA [5]. Finally, a conclusion and some perspectives are given in Section VI.

II. Preliminaries and Basic Notations

A t-time Petri net (TPN) is a P/T Petri net [17] where a time interval $[t_{\min};t_{\max}]$ is associated with each transition t.

Definition 1: A *TPN* is a tuple $\mathcal{N} = \langle P, T, Pre, Post, I\rangle$ where:

- $\langle P, T, Pre, Post\rangle$ is a *P/T Petri net* where:
  - P is a finite set of places;
  - T is a finite set of transitions with $P \cap T = \emptyset$;
  - $Pre : T \longrightarrow \mathbb{N}^P$ is the backward incidence mapping;
  - $Post : T \longrightarrow \mathbb{N}^P$ is the forward incidence mapping;
- $I : T \longrightarrow \mathbb{N} \times (\mathbb{N} \cup \{+\infty\})$ is a *time interval function* s.t. $I(t) = (t_{\min}, t_{\max})$ $(t_{\min} \leq t_{\max})$, where $t_{\min}$ (resp. $t_{\max}$) is the earliest (resp. latest) firing time of transition t.

A *marking* of a TPN is a function $m : P \longrightarrow \mathbb{N}$ where $m(p)$, for a place p, denotes the number of tokens in p. A *marked TPN* is a pair $\mathcal{N} = \langle \mathcal{N}_1, m_0\rangle$ where $\mathcal{N}_1$ is a TPN and m_0 is

1530-1311/13 $26.00 © 2013 IEEE
DOI 10.1109/TIME.2013.22

a corresponding *initial marking*. A transition t is enabled by a marking m iff $m \geq Pre(t)$ and $Enable(m) = \{t \in T : m \geq Pre(t)\}$ denotes the set of enabled transitions in m. If a transition t_i is enabled by a marking m, then $\uparrow(m,t_i)$ denotes the set of newly enabled transitions [2]. Formally, $\uparrow(m,t_i) = \{t \in T \mid (m - Pre(t_i) + Post(t_i)) \geq Pre(t) \wedge (m - Pre(t_i)) < Pre(t)\}$. If a transition t is in $\uparrow(m,t_i)$, we say that t is newly enabled by the successor of m by firing t_i. Dually, $\downarrow(m,t_i) = \{t \in T \mid (m - Pre(t_i) + Post(t_i)) \geq Pre(t) \wedge (m - Pre(t_i)) \geq Pre(t)\}$ is the set of oldly enabled transitions. The possibly infinite set of reachable markings of $\mathcal{N}$ is denoted $Reach(\mathcal{N})$. If $Reach(\mathcal{N})$ is finite we say that $\mathcal{N}$ is bounded.

The semantics of TPNs can be given in terms of Timed Transition Systems (TTS) [12] which are usual transition systems with two types of labels: discrete labels for events (transitions) and positive real labels for time elapsing (delay). States of the TTS are pairs $s = (m,V)$ where m is a marking and $V : T \longrightarrow \mathbb{R}_{\geq 0} \cup \{\bot\}$ a time valuation. In the following, $s.m$ and $s.V$ denote the marking and the time valuation respectively of a state s. If a transition t is enabled in m then $V(t)$ is the elapsed time since t became enabled, otherwise $V(t) = \bot$. Given a state $s = (m,V)$ and a transition t, t is said to be firable in s iff $t \in Enable(m) \wedge V(t) \neq \bot \wedge t_{min} \leq V(t) \leq t_{\max}$.

Definition 2 (Semantics of a TPN): Let $\mathcal{N} = \langle P,T,Pre, Post,I,m_0\rangle$ be a marked TPN. The semantics of $\mathcal{N}$ is a TTS $\mathcal{S}_{\mathcal{N}} = \langle Q,s_0,\rightarrow\rangle$ where:

1) Q is a (possibly infinite) set of states
2) $s_0 = (m_0,V_0)$ is the initial state such that:

$$\forall t \in T,\ V_0(t) = \begin{cases} 0 & \text{if } t \in Enable(m_0) \\ \bot & \text{otherwise} \end{cases}$$

3) $\rightarrow \subseteq Q \times (T \cup \mathbb{R}_{\geq 0}) \times Q$ is the discrete and continuous transition relations:
   a) the discrete transition relation: $\forall t \in T :\ (m,V) \xrightarrow{t} (m',V')$ iff:

$$\begin{cases} t \in Enable(m) \wedge m' = m - Pre(t) + Post(t) \\ t_{min} \leq V(t) \leq t_{max} \\ \forall t' \in T : V'(t') = \begin{cases} 0 & \text{if } t' \in \uparrow(m,t) \\ V(t') & \text{if } t' \in \downarrow(m,t) \\ \bot & \text{otherwise} \end{cases} \end{cases}$$

   b) the continuous transition relation: $\forall d \in \mathbb{R}_{\geq 0}$, $(m,V) \xrightarrow{d} (m',V')$ iff:

$$\begin{cases} \forall t \in Enable(m),\ V(t) + d \leq t_{max} \\ m' = m \\ \forall t \in T : \\ V'(t) = \begin{cases} V(t) + d & \text{if } t \in Enable(m); \\ V(t) & \text{otherwise.} \end{cases} \end{cases}$$

The above definition requires some comments. First, the delay transitions respect the STS semantics: an enabled transition must fire within its firing interval unless disabled by the firing of others. Second, a state change occurs either by the firing of transitions or by time elapsing: The firing of a transition may change the current marking while the time elapsing may make some new transitions firable. When a deadlock marking is reached, then only elapsing time is possible and the system remains in the same state.

Given a TPN $\mathcal{N}$ and the corresponding TTS $\mathcal{S}_{\mathcal{N}}$, a path $\pi = s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} \ldots$, where $\alpha_i \in (T \cup \mathbb{R}_{\geq 0})$, is a run of $\mathcal{S}_{\mathcal{N}}$ iff $(s_i,\alpha_i,s_{i+1}) \in \rightarrow$ for each $i = 0,1,\ldots$. The length of a run π can be infinite and is denoted by $\mid \pi \mid$. The possibly infinite set of runs of $\mathcal{S}_{\mathcal{N}}$ is denoted $[\mathcal{S}_{\mathcal{N}}]$. Without loss of generality, we assume that for each non empty run $\pi = s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} \ldots$ of a STS corresponding to a TPN, there do not exist two successive labels α_i and α_{i+1} belonging both to $\mathbb{R}_{\geq 0}$. Then, π can be written, involving the reachable markings of $\mathcal{N}$, as $\pi = m_0 \xrightarrow{(d_1,t_1)} m_1 \xrightarrow{(d_2,t_2)} \ldots$ where d_i is the time elapsed at marking m_{i-1} before firing t_i. In order to associate a run π of $\mathcal{S}_{\mathcal{N}}$ with a run of $\mathcal{N}$, denoted $\mathcal{P}(\pi)$, we define the following projection function, where $.$ denotes the concatenation operator between paths and π^i, for $i = 0,1\ldots$, denotes the suffix of π starting at state s_i.

$$\mathcal{P}(\pi) = \begin{cases} s_0.m & \text{if } \mid \pi \mid = 0 \\ s_0.m \xrightarrow{(0,\alpha_1)} . \mathcal{P}(\pi^1) & \text{if } \alpha_1 \in T \\ s_0.m \xrightarrow{(\alpha_1,\alpha_2)} . \mathcal{P}(\pi^2) & \text{if } \alpha_1 \in \mathbb{R}_{\geq 0} \wedge \mid \pi \mid \geq 2 \\ s_0.m \xrightarrow{\alpha_1} . \mathcal{P}(\pi^1) & \text{if } \alpha_1 \in \mathbb{R}_{\geq 0} \wedge \mid \pi \mid = 1 \end{cases}$$

III. Abstraction of a TPN State Space

A. Timed Aggregate Graph

In this subsection, we propose to abstract the reachability state space of a TPN using a new graph called Timed Aggregate Graph (TAG) where nodes are called *aggregates* and are grouping sets of states of a TTS. The key idea behind TAGs is that time information is encoded inside aggregates. The time information includes the time the system is able to stay in the aggregate as well as a dynamic interval associated with each enabled transition. The first feature allows to encapsulate the delay transitions of the corresponding TTS (the arcs of a TAG are labeled with transitions of the corresponding TPN only), while the second allows to dynamically update the earliest and latest firing times of enabled transitions. It also allows to maintain the relative differences between the firing times of transitions (diagonal constraints).

Before we formally define the TAG and illustrate how the attributes of an aggregate are computed, let us first formally define aggregates.

Definition 3 (Timed Aggregate): A *timed aggregate* associated with a *TPN* $\mathcal{N} = \langle P,T,Pre,Post,I\rangle$ is a 4-tuple $a = (m,E,h,H)$, where:

- m is a marking;

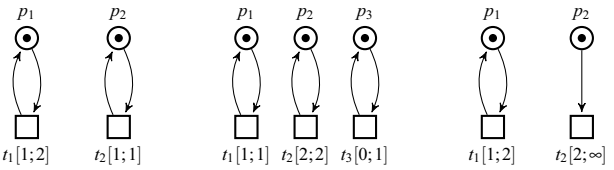


Figure 1. Three TPN Examples

- $E = \{\langle t, \alpha_t, \beta_t \rangle \mid t \in Enable(m),\ \alpha_t \in (\mathbb{Z} \cup \{-\infty\}) \wedge \beta_t \in \mathbb{N} \cup \{+\infty\}\}$ is a set of enabled transitions each asssociated with two time values;
- $h = \min_{\langle t,\alpha_t,\beta_t \rangle \in E}(max(0,\alpha_t))$: the minimum time the system can stay in the aggregate a;
- $H = \min_{\langle t,\alpha_t,\beta_t \rangle \in E}(\beta_t)$: the maximum time the system can stay in the aggregate a.

Each aggregate is characterised by three attributes that are computed dynamically: a marking m, a set E of enabled transitions and two times h and H. Each enabled transition t is associated with two time values: α_t represents the minimum time the system should wait before firing t and β_t represents the maximum time the system can delay the firing of t. Note that α_t can be negative which means that t can be fired immediately. Otherwise, α_t represents the the earliest firing time of t. Starting from this aggregate, the firing of t can occur between $max(0,\alpha_t)$ and H. In the rest of this paper, α_t will be abusively called the *dynamic* earliest firing time of t and β_t its *dynamic* latest firing time. Finally, the h and H attributes represent the minimum and the maximum time, respectively, the system can spend at the current aggregate.

As for states of a TTS, the attributes of an aggregate a are denoted by $a.m$, $a.E$, $a.h$ and $a.H$.

Now, we are ready to formally define the TAG associated with a marked TPN $\mathcal{N}$. A TAG is a labeled transition system where nodes are timed aggregates. A TAG has an initial aggregate, a set of actions (the set of transitions of $\mathcal{N}$) and a transition relation. The initial aggregate is easily computed by considering static information of the TPN. In fact: (1) the marking attribute is the initial marking of $\mathcal{N}$, (2) the minimum (resp. maximum) time the system can stay in the initial aggregate is equal to the minimum of the earliest (resp. latest) firing time of the enabled transitions and, (3) each enabled transition t is firable between $t_{\min}$ and $t_{\max}$.

Definition 4 (Timed Aggregate Graph): A *TAG* associated with a TPN $\mathcal{N} = \langle P, T, Pre, Post, I, m_0 \rangle$ is a tuple $G = \langle \mathcal{A}, T, a_0, \delta \rangle$ where:

1) $\mathcal{A}$ is a set of timed aggregates;
2) $a_0 = \langle m_0, h_0, H_0, E_0 \rangle$ is the initial timed aggregate s.t.:
   a) m_0 is the initial marking of $\mathcal{N}$
   b) $h_0 = \min_{t \in Enable(m_0)}(t_{\min})$
   c) $H_0 = \min_{t \in Enable(m_0)}(t_{\max})$
   d) $E_0 = \{\langle t, t_{\min}, t_{\max} \rangle \mid t \in Enable(m_0)\}$
3) $\delta \subseteq \mathcal{A} \times T \times \mathcal{A}$ is the transition relation such that: for an aggregate $a = \langle m, h, H, E \rangle$, a transition t s.t. $\langle t, \alpha_t, \beta_t \rangle \in E$ and an aggregate $a' = \langle m', h', H', E' \rangle$, $(a, t, a') \in \delta$ iff the following holds:
   a) $m' = m - Pre(t) + Post(t)$
   b) $\alpha_t \leq H$
   c) $\forall \langle t', \alpha_{t'}, \beta_{t'} \rangle \in E$, $t_{min} > t'_{max} \Rightarrow (t_{min} - \alpha_t) - (t'_{min} - \alpha_{t'}) \geq (t_{min} - t'_{max})$
   d) $E' = E'_1 \cup E'_2$, where:
      - $E'_1 = \bigcup_{t' \in \uparrow(a,t)} \{\langle t', t'_{\min}, t'_{\max} \rangle\}$
      - $E'_2 = \bigcup_{t' \in \downarrow(a,t)} \{\langle t', \alpha_{t'} - H, \beta_{t'} - max(0, \alpha_t) \rangle\}$
   e) $h' = \min_{\langle t', \alpha_{t'}, \beta_{t'} \rangle \in E'}(max(0, \alpha_{t'}))$
   f) $H' = \min_{\langle t', \alpha_{t'}, \beta_{t'} \rangle \in E'}(\beta_{t'})$

Given an aggregate $a = \langle m, h, H, E \rangle$ and a transition $t \in T$, t is said to be enabled by a, denoted by $a \xrightarrow{t}$, iff: (1) $\exists (\alpha_t, \beta_t) \in (\mathbb{Z} \cup \{-\infty\}) \times (\mathbb{N} \cup \{+\infty\})$ s.t. $(t, \alpha_t, \beta_t) \in E$ and $\alpha_t \leq H$, and, (2) there is no other transition t', enabled by $a.m$, that should be fired before t. In fact, the first condition is sufficient if the static interval of t overlaps with any other transition's interval but is not sufficient when t_{min} is greater than t'_{max} for some other transition t'. The firing of t from a leads to a new aggregate $a' = \langle m', h', H', E' \rangle$ whose attributes are computed by analysing the sets $\uparrow(a,t)$ and $\downarrow(a,t)$. This is the key idea behind the TAG.

- The elements of E' are processed by taking the transitions that are enabled by m' and computing their earliest and latest firing times depending on their membership in $\uparrow(a,t)$ and $\downarrow(a,t)$. For each transition $t' \in Enable(a'.m)$, if t' is newly enabled, then its dynamic earliest and latest firing times are statically defined by t'_{min} and t'_{max} respectively. Otherwise, let $\langle t', \alpha_{t'}, \beta_{t'} \rangle \in a.E$ and $\langle t', \alpha'_{t'}, \beta'_{t'} \rangle \in a'.E$, then the maximum time elapsed by the system at $a.m$ (i.e., $a.H$) is subtracted from $\alpha_{t'}$ and the minium time is substracted from $\beta_{t'}$. Indeed, the more the system can stay in $a.m$ the less it can stay in $a'.m$ (and vice versa). Thus, the earliest firing time of t starting from a' is $max(0, \alpha_{t'} - a.H)$ while its latest firing time is $\beta_{t'} - max(0, \alpha_t)$.
- The computation of $a'.h$ (resp. $a'.H$) is ensured by taking the minimum of the dynamic earliest (resp. latest) firing time of enabled transitions.

B. Equivalence relation between aggregates

According to Definition 4, the dynamic earliest firing time of a transition can decrease infinitely which could lead to an infinite state space TAG. In the following, we define a relation allowing to identify equivalent aggregates. This equivalence relation will be used in the construction of a TAG so that each newly built aggregate is not explored as long as an already built equivalent aggregate has been.

Definition 5: Let a and a' be two time aggregates. a is said to be equivalent to a', denoted by $a \equiv a'$, iff:

1) $a.h = a'.h$, $a.H = a'.H$ and $a.m = a'.m$
2) $\forall(\langle t, \alpha_t, \beta_t\rangle, \langle t, \alpha'_t, \beta'_t\rangle) \in (a.E \times a'.E)$, $\beta_t = \beta'_t$:
   a) $\alpha_t = \alpha'_t$ or,
   b) $max(0, \alpha_t) = max(0, \alpha'_t)$ and
      i) $\nexists t' \in T : (t'_{max} < t_{min}) \vee (t_{max} < t'_{min})$ or,
      ii) $\exists t' \in (T \setminus Enable(a.m)) : (t_{max} < t'_{min})$ or,
      iii) $\exists t' \in (T \setminus Enable(a.m)) : t'_{max} < t_{min} \wedge (\alpha_t \leq t'_{max}) \wedge (\alpha'_t \leq t'_{max})$ or,
      iv) $\exists t' \in Enable(a.m) : t'_{max} < t_{min} \wedge ((\alpha_t - \alpha_{t'}) \leq (t'_{max} - t'_{min}) \wedge (\alpha'_t - \alpha'_{t'}) \leq (t'_{max} - t'_{min})) \vee ((\alpha_{t'} - \alpha_t) = (\alpha'_{t'} - \alpha'_t))$.

Informally, the previous definition guarantees that two equivalent aggregates have the same behavior. First, point 1 guarantees that a and a' have the same marking and the same minimum and maximum stay time. Second, each enabled transition t should have the same latest firing time (point 2). Then, if the earliest firing time of a given enabled transition t in a is different from its earliest firing time in a', then this should not influence the future behavior starting from a and a'. First, the earliest firing times of t in a and a' must be negative so that the firing of t from a and a' do not lead to different aggregates. This condition is sufficient if there is no transition t' such that $t'_{max} < t_{min}$ (point 2(b)i), or if any transition t' s.t. $t_{max} < t'_{min}$ is not enabled by a (point 2(b)ii). In the opposite case, other conditions must be fulfilled: First, if there exists a transition t' such that $t'_{max} < t_{min}$ but t' is not enabled by a (point 2(b)iii), then α_t and α'_t should be both less or equal to t'_{max} (i.e. t is firable anyway in both aggregates). Second, if t' is enabled by a (point 2(b)iv), either t is firable by both aggregates a and a' or the difference $(t_{min} - \alpha_t) - (t'_{min} - \alpha_{t'})$ is equal to $(t_{min} - \alpha'_t) - (t'_{min} - \alpha'_{t'})$.

Figure 2 illustrates the TAGs corresponding to the first and the third TPNs of Figure 1. The TAG associated with the second TPN contains 18 aggregates and 38 edges, and is not presented here because of lack of space. In the first TAG, the marking associated with aggregates is omitted since it is always the same as the initial one. Although, the three models of Figure 1 are quite simple, they are representative enough to explain the TAG construction. Indeed, in the first one the transitions intervals overlap, while the case of disjoint intervals is considered through the second model. Finally, the third model illustrates the case of an infinite latest firing time. More significant examples are considered in Section V.

In the following, we establish that the TAG, built under the previous equivalence relation between aggregates, associated with a bounded TPN is finite. For this, we show that the number of its aggregates is bounded and compute this bound. The following two lemmas establish two preliminary results before proving the finiteness of a TAG in Theorem 1.

Lemma 1: Let $\mathcal{N}$ be a TPN and let $G = \langle \mathcal{A}, T, a_0, \delta\rangle$ be the corresponding TAG. Let a be an aggregate of G

aggregate	(h,H)	E
a_0	(1,1)	$\{\langle t_1,1,2\rangle, \langle t_2,1,1\rangle\}$
a_1	(0,0)	$\{\langle t_1,1,2\rangle, \langle t_2,-1,0\rangle\}$
a_2	(0,1)	$\{\langle t_1,0,1\rangle, \langle t_2,1,1\rangle\}$
a_3	(0,0)	$\{\langle t_1,1,2\rangle, \langle t_2,0,1\rangle\}$
a_4	(0,1)	$\{\langle t_1,0,2\rangle, \langle t_2,1,1\rangle\}$
a_5	(0,0)	$\{\langle t_1,-1,0\rangle, \langle t_2,1,1\rangle\}$

aggregate	marking	(h,H)	E
a_0	(1,1)	(1,2)	$\{\langle t_1,1,2\rangle, \langle t_2,2,\infty\rangle\}$
a_1	(1,0)	(0,0)	$\{\langle t_1,-1,0\rangle\}$
a_2	(1,0)	(1,2)	$\{\langle t_1,1,2\rangle\}$
a_3	(1,1)	(0,2)	$\{\langle t_1,1,2\rangle, \langle t_2,0,\infty\rangle\}$
a_4	(1,0)	(0,2)	$\{\langle t_1,-1,2\rangle\}$

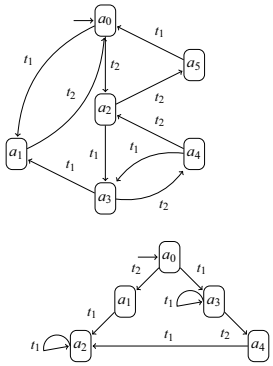


Figure 2. TAGs of Figure 1

and let $t \in Enable(a.m)$. Then, the number of possible dynamic intervals $[\alpha_t; \beta_t]$ associated with t in non equivalent aggregates is at most equal to $\mathbb{B}_t$ where

$$\mathbb{B}_t = \begin{cases} b_t & \text{if } \nexists t' \in T : t'_{max} < t_{min} \\ b_t * c_t & \text{otherwise} \end{cases}$$

where $b_t = (t_{min}+1)*(t_{max}+1) - (t_{min} + (t_{min}*(t_{min}+1)/2))$ and $c_t = Max_{t' \in T : t'_{max} < t_{min}}(t_{min} - t'_{max}) + 1$

Proof: The possible intervals are obtained by all the possible combinations of the earliest and latest firing times α_t and β_t, minus the invalid ones (i.e. those where $\alpha_t > \beta_t$ and those where $\alpha_t \neq t_{min} \wedge \alpha_t = \beta_t$). In the first case, we consider that there is no transition t' such that $t'_{max} < t_{min}$. Since negative values of α_t are equivalent to 0 the possible values of α_t are $\{t_{min}, t_{min}-1, \ldots, 0\}$ (i.e $t_{min}+1$ possible values). Moreover, $\beta_t \in \{t_{max}, t_{max}-1, \ldots, 0\}$ (i.e $t_{max}+1$ possible values) and there are $t_{min} + (t_{min} * (t_{min}+1)/2)$ invalid intervals. The first t_{min} intervals are invalid by construction of the TAG since β_t can never reach α_t by decrement ($\alpha < H$). Moreover, there are $(t_{min} * (t_{min}+1)/2)$ intervals where $\alpha_t > \beta_t$, and these are invalid as well. In the second case, the previous reasoning is also valid, but one has to consider the difference between t and t'. Following the definition of the equivalence relation between aggregates, this difference is ignored when it allows the firing of t. Thus, only $t_{min} - t'_{max} + 1$ values are to be distinguished (for t' having the minimal $t'_{max} < t_{min}$) and should be combined with the number of possibilities obtained in the first case. ■

Lemma 2: Let $\mathcal{N}$ be a TPN, let $G = \langle \mathcal{A}, T, a_0, \delta\rangle$ be the corresponding TAG and let $m \in Reach(\mathcal{N})$. The number of aggregates in G having m as marking is at most equal to $\sum_{s \in 2^{Enable(m)}} (\prod_{t \in s} \mathbb{B}_t)$.

Proof: Let m be a marking, the number of aggregates having m as a marking are different with respect to the E attribute. There are as many possibilities of partitionning enabled transitions into oldly and newly enabled as subset of $Enable(m)$. For a given subset s of $Enable(m)$ there are at most $\prod_{t \in s} \mathbb{B}_t$ (using Lemma 1) possible time intervals for

its transitions. ■

The two previous lemmas lead to the following theorem.

Theorem 1: Given a TPN $\mathcal{N}$, if $\mathcal{N}$ is bounded then the corresponding TAG is finite and its number of aggregates is at most equal to $\sum_{m\in Reach(\mathcal{N})}\sum_{s\in 2^{Enable(m)}}\prod_{t\in Enable(m)}\mathbb{B}_t$.

C. Preservation Results

In this section, we establish the main results of this paper: The TAG is an exact representation of the reachability state space of a TPN. For each path in the TAG (resp. in the TPN) it is possible to find a path in the TPN (resp. TAG) involving the same sequence of transitions and where the time elapsed within a given state is between the minimum and the maximum stay time of the corresponding aggregate.

Theorem 2: Let $\mathcal{N}$ be a TPN and let $G=\langle \mathcal{A},T,a_0,\delta\rangle$ be the TAG associated with $\mathcal{N}$. Then, for any path $\pi = a_0 \xrightarrow{t_1} a_1 \longrightarrow \dots \xrightarrow{t_n} a_n$ in the TAG, $\forall d\in\mathbb{R}_{\geq 0}$ s.t. $d\leq a_n.H$, there exists a run $\overline{\pi}=m_0 \xrightarrow{(d_1,t_1)} m_1 \longrightarrow \dots \xrightarrow{(d_n,t_n)} m_n \xrightarrow{d}$ in $\mathcal{N}$, s.t. $m_i=a_i.m$ and $a_{i-1}.h\leq d_i\leq a_{i-1}.H$, for $i=1\dots n$.

Proof: Let $\pi = a_0 \xrightarrow{t_1} a_1 \longrightarrow \dots \xrightarrow{t_n} a_n$ and let $d\in\mathbb{R}_{\geq 0}$ satisfying $d\leq a_n.H$. We denote by α_{i_t} (resp. β_{i_t}) the maximum between 0 and the dynamic earliest firing time (resp. the dynamic latest firing time) of a transition t at aggregate a_i. Let us prove that the path $m_0 \xrightarrow{(\alpha_{0_{t_1}},t_1)} m_1 \longrightarrow \dots \xrightarrow{(\alpha_{n-1_{t_n}},t_n)} m_n \xrightarrow{d}$ satisfies the required result. The particularity of this path is that the firing of each transition t_i is performed at its dynamic earliest firing time.

We proceed by induction on the length of π.

Note first that, $a_{i-1}.h\leq\alpha_{i-1_{t_i}}$ and $\alpha_{i-1_{t_i}}\leq a_{i-1}.H$ (by construction), for $i=1\dots n$.

- $|\pi|=0$: Since, $m_0=a_0.m$, $a_0.H$ is the minimum value of the latest firing time of all transitions enabled by m_0 and $a_0.h\leq d\leq a_0.H$. Thus $m_0\xrightarrow{d}$.
- Assume that for any path $a_0 \xrightarrow{t_1} a_1 \longrightarrow \dots \xrightarrow{t_n} a_n$ and for any $d\in\mathbb{R}_{\geq 0}$ satisfying $d\leq a_n.H$, the path $m_0 \xrightarrow{(\alpha_{0_{t_1}},t_1)} m_1 \longrightarrow \dots \xrightarrow{(\alpha_{n-1_{t_n}},t_n)} m_n \xrightarrow{d}$ is a run of $\mathcal{N}$. Let $\pi = a_0 \xrightarrow{t_1} a_1 \xrightarrow{t_2} \dots \xrightarrow{t_n} a_n \xrightarrow{t_{n+1}} a_{n+1}$ be a path of length $n+1$. Since $a_n.h\leq\alpha_{n_{t_{n+1}}}\leq a_n.H$, we can use the induction hypothesis to exhibit a path $m_0 \xrightarrow{(\alpha_{0_{t_1}},t_1)} m_1 \longrightarrow \dots \xrightarrow{(\alpha_{n-1_{t_n}},t_n)} m_n \xrightarrow{\alpha_{n_{t_{n+1}}}}$. We show now that it is possible to complete this path by firing t_{n+1} from m_n (after a delay $\alpha_{n_{t_{n+1}}}$ at this state). Note first that t_{n+1} is enabled by m_n (since $m_n=a_n.m$).
  Let l (with $l<n$) be the greatest integer, such that $t_{n+1}\in\uparrow(a_{l-1},t_l)$, then $\langle t_{n+1},t_{n+1_{min}},t_{n+1_{max}}\rangle\in a_l.E$. Moreover, $\forall i=l+1\dots n$, $\langle t_{n+1},\alpha_{i_{t_{n+1}}},\beta_{i_{t_{n+1}}}\rangle\in a_i.E$ with $\alpha_{i_{t_{n+1}}}=\max(0,\alpha_{i-1_{t_{n+1}}}-a_{i-1}.H)$ and $\beta_{i_{t_{n+1}}}=\beta_{i-1_{t_{n+1}}}-\alpha_{i-1_{t_i}}$. We know by construction that $\alpha_{i-1_{t_i}}\leq a_{i-1}.H$ (condition for firing t_i at a_{i-1}) (for all $i=l+1\dots n$). Using the fact that $\alpha_{l_{t_{n+1}}}\leq\beta_{l_{t_{n+1}}}$ and the fact that $\alpha_{l_{t_{l+1}}}\leq a_l.H\leq\beta_{l_{t_{n+1}}}$, we can deduce that $\alpha_{l_{t_{n+1}}}-a_l.H\leq\beta_{l+1_{t_{n+1}}}=(\beta_{l_{t_{n+1}}}-\alpha_{l_{t_{l+1}}})$ and that $\beta_{l+1_{t_{n+1}}}\geq 0$, thus $\alpha_{l+1_{t_{n+1}}}\leq\beta_{l+1_{t_{n+1}}}$. By iterating this result we establish that $\alpha_{n_{t_{n+1}}}\leq\beta_{n_{t_{n+1}}}$ and that $\beta_{n_{t_{n+1}}}\geq 0$. Moreover, it is clear that $\beta_{i-1_{t_{n+1}}}\geq\beta_{i_{t_{n+1}}}$ (for all $i=l+1\dots n$). In particular, we have $0\leq\alpha_{n_{t_{n+1}}}\leq\beta_{n_{t_{n+1}}}\leq\beta_{n-1_{t_{n+1}}}\leq\dots\leq\beta_{l_{t_{n+1}}}=t_{n+1_{max}}$. Thus, the system can stay at m_n for $\alpha_{n_{t_{n+1}}}$. Furthermore, the fact that t_{n+1} is firable at a_n implies that it is also firable in m_n leading to m_{n+1}, whose marking is the same as a_{n+1}. Let us show now that it is possible to stay at m_{n+1} for $a_{n+1}.H$ (or any $d\leq a_{n+1}.H$) time units. We can use the same reasoning as above to show that for any transiton t enabled by m_{n+1}, $\alpha_{n+1_t}\leq t_{max}$. Since $a_{n+1}.H=\min_{t\in Enable(m_{n+1})}\beta_t$ we can deduce that the system can stay at m_{n+1} $a_{n+1}.H$ (or any lower value) time units.

■

Theorem 3: Let $\mathcal{N}$ be a TPN and let $G=\langle \mathcal{A},T,a_0,\delta\rangle$ be the TAG associated with $\mathcal{N}$. Then
$\forall\overline{\pi}=m_0 \xrightarrow{(d_1,t_1)} m_1 \xrightarrow{(d_2,t_2)} \dots \xrightarrow{(d_n,t_n)} m_n \xrightarrow{d_{n+1}}$, with $d_i\in\mathbb{R}_{\geq 0}$, for $i=1\dots n+1$, $\exists\pi=a_0 \xrightarrow{t_1} a_1 \longrightarrow \dots \xrightarrow{t_n} a_n$ s.t. $a_{i-1}.h\leq d_i\leq a_{i-1}.H$, $m_i=a_i.m$, for $i=0\dots n$ and $d_{n+1}\leq a_n.H$.

Proof: Let $\overline{\pi}=m_0 \xrightarrow{(d_1,t_1)} m_1 \xrightarrow{(d_2,t_2)} \dots \xrightarrow{(d_n,t_n)} m_n \xrightarrow{d_{n+1}}$ be a path of $\mathcal{N}$, with $d_i\in\mathbb{R}_{\geq 0}$, for $i=1\dots n+1$. Let us prove by induction on the length of $\overline{\pi}$ the existence of a path π in the TAG satisfying Theorem 3.

- $|\overline{\pi}|=0$: Obvious since $m_0=a_0.m$ (by construction) and since d_1 is less or equal to $\min_{t\in Enable(m_0)}t_{max}$ which is exactly the value of $a_0.H$.
- Assume Theorem 3 true for any path $\overline{\pi}$ s.t. $|\overline{\pi}|\leq n$.
  Let $\overline{\pi} = m_0 \xrightarrow{(d_1,t_1)} m_1 \xrightarrow{(d_2,t_2)} \dots \xrightarrow{(d_n,t_n)} m_n \xrightarrow{(d_{n+1},t_{n+1})} m_{n+1} \xrightarrow{d_{n+2}}$ be a path of length $n+1$. Let $\pi = a_0 \xrightarrow{t_1} a_1 \xrightarrow{t_2} \dots \xrightarrow{t_n} a_n$ be the path in the TAG associated with the n-length prefix of $\overline{\pi}$ (by the induction hypothesis). We denote by α_{i_t} (resp. β_{i_t}) the maximum between 0 and the dynamic earliest firing time (resp. the dynamic latest firing time) of a transition t at aggregate a_i.
  Since $a_n.m=m_n$ and $d_{n+1}\leq a_n.H$, t_{n+1} is also enabled by a_n (by construction) and can be fired leading to a_{n+1} whose marking is the same as m_{n+1}. Let us now show that $d_{n+2}\leq a_{n+1}.H$. If $Enable(m_{n+1})=\emptyset$ then m_{n+1} is a deadlock state and so is a_{n+1} and $d_{n+1}<a_{n+1}.H=+\infty$. Otherwise, let $t\in Enable(m_{n+1})$ let $1\leq k\leq n+1$ be the largest integer, satisfying $t\in\uparrow(m_k,t_{k+1})$. The fact that $m_{n+1}\xrightarrow{d_{n+2}}$ implies that $\sum_{i=k}^{n+1}d_i\leq t_{max}$, then that $d_{n+1}\leq t_{max}-\sum_{i=k}^{n}d_i$. By proving that $d_i\geq\alpha_{i_{t_{i+1}}}$ for all $i=k\dots n$, we can deduce that $d_{n+1}\leq t_{max}-\sum_{i=k}^{n}\alpha_{i_{t_{i+1}}}=\beta_{n+1_t}$. This result holds for any enabled transition t then $d_{n+1}\leq\min_{t\in Enable(m_{n+1})}(\beta_{n+1_t})=a_{n+1}.H$. Finally, let us prove that $d_i\geq\alpha_{i_{t_{i+1}}}$ for all $i=k\dots n$. Let $k\leq i\leq n$, if $\alpha_{i_{t_{i+1}}}=0$ then $d_i\geq\alpha_{i_{t_{i+1}}}$. Otherwise, let $1\leq l\leq i$ be the highest integer value, if any, satisfying $t_{i+1}\in\uparrow(m_l,t_l)$ (if such a value does not exist then $l=0$). Then $\sum_{j=l}^{i}d_j\geq t_{i+1_{min}}$ (since t_{i+1} is fired from

m_i) which means that $d_i \geq t_{i+1_{min}} - \sum_{j=l}^{i-1} d_j$. Using the induction hypothesis $d_j > a_{j-1}.H$ for all $j = l \ldots i-1$, then $d_i \geq max(0, (t_{i+1_{min}} - \sum_{j=l}^{i-1} a_{j-1}.H)) = \alpha_{i_{t_{i+1}}}$. ■

Using the above result one can use the TAG associated with a TPN in order to analyse both event and state based properties. In particular, we can check whether a given marking (resp. transition) is reachable (resp. is firable) before (or after) a some time.

IV. Related work

This section reviews the most well-known techniques, proposed in the literature, that abstract and analyse the state space of real-time systems described by means of TPN. Abstraction techniques aim at constructing, by removing some irrelevant details, a contraction of the state space of the model, which preserves properties of interest. The existing abstraction approaches mainly differ in the state agglomeration criteria, the characterisation of states and state classes (interval states or clock states) and the kind of preserved properties.

The *States Class Graph* (SCG) [3] was the first method of state space representation adapted to TPNs. A class (m,D) is associated with a marking m and a time domain D represented by a set of inequalities over variables. The variables represented in the SCG are the firing time intervals of enabled transitions. The SCG allows for the verification of some TPN properties like reachability and boundness. However, it preserves the linear time properties only. To address this limitation, a refinement of the method was proposed in [21], in the form of a graph called *Atomic States Class Graph* (ASCG). The authors use a cutting of state class by adding linear constraints so that each state of an atomic class has a successor in all the following classes. With this improvement, they are able to verify CTL* properties on TPN, but with the limitation that the time intervals of transitions are bounded. A new approach for the construction of atomic classes was proposed in [4] and allows the verification of CTL* without restriction on time intervals. The state class approach is implemented in a software tool called TINA [5].

The *Zones Based Graph* (ZBG) [9] is another approach allowing to abstract the TPN state space. This approach is inspired by the *Region Graph* (RG) [1] technique, initially introduced for *timed automata*. In practice, the number of regions is too large for an effective analysis, thus, the regions are grouped into a set of *zones*. A zone is a convex union of regions and can be represented by a DBM (Difference Bound Matrix). In [9], the clocks of transitions are directly encoded within the zones. This allows to verify temporal and quantitative properties but not CTL* properties. As for timed automata, a disadvantage of the method is the necessary recourse to approximation methods (k-approximation or kx-approximation) in the case where the infinity is used in the bounds of time intervals. Lime and Roux also used TPNs to model system behavior [14]. They used the state class approach to build a *timed automaton* that preserves the behaviour of the TPN using as few clock variables as possible. The resulting model is then verified using the UPPAAL tool [13]. However, even though UPPAAL can answer about quantitative temporal properties, it can only verify a subset of TCTL. Adding a new transition to measure elapsed time was proposed in [6] to perform TCTL model-checking in TPNs. Using this transition, TCTL formulae are translated into CTL formulae. Then a ZBG for TPN is refined leading to a graph called *Atomic Zone Based Graph* (AZBG) that preserves CTL properties.

Unlike the TAG, in all existing approaches, the time information does not appear explicitly in nodes. This limitation, in turn leads to additional and complex calculations such as: the manipulation of DBM to encode the zones (for zones-based approaches) and the classes (for state-class based approaches), the approximations to counter the problem of unbounded transitions, conversion of graphs to timed automata (using UPPAAL) to model check properties, etc. In our work the time information is encoded within the aggregates allowing to check time properties just by browsing the graph, which has a significant impact on the construction complexity. The encoding of the timing information in the aggregates is such that the minimum and maximum elapsed time in every path of the TAG can be computed. Another difference with these approaches is that every TAG preserves the runs of the corresponding TPNs as well as the reachable states, which allows the preservation of both event- and state-based (linear and branching) logics.

V. Experimental results

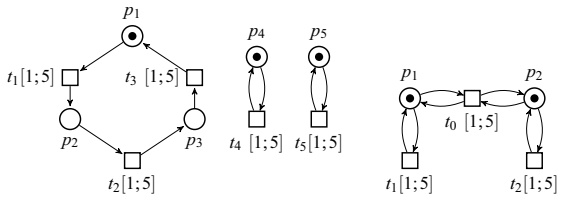
Our approach for building TAG-TPN was implemented in a prototype tool (written in C++), and used for experiments in order to validate the sizes of the graphs generated by the approach. Note that the prototype was not optimised for time efficiency yet, therefore no timing figures are given in this section. All results reported in this section have been obtained on 2.8 gigahertz Intel with four gigabytes of RAM. The implemented prototype allowed us to have first comparison with existing approaches with respect to the size of obtained graphs. We used the TINA tool to build the SCGs, ROMEO tool for the ZBGs and our tool for the TAGs. We tested our approach on several TPN models and we report here the obtained results. The considered models are representative of the characteristics that may occur in a TPN, such as: concurrency, synchronisation, disjoint firing intervals and infinite firing bounds. The first two models (Figure 3(a) and Figure 3(b)) are two parametric models where the number of processes can be increased. In Figure 3(a), the number of self loops ($p_n \rightarrow t_n \rightarrow p_n$) is increased while in Figure 3(b) the number of processes,

Parameters	SCG (with Tina) (nodes / arcs)	ZBG (with Romeo) (nodes / arcs)	TAG-TPN (nodes / arcs)
Nb. prod/cons	TPN model of producer/consumer		
1	34 / 56	34 / 56	34 / 56
2	748 / 2460	593 / 1 922	407 / 1 255
3	4 604 / 21891	3 240 / 15 200	1 618 / 6 892
4	14 086 / 83 375	9 504 / 56 038	3 972 / 20 500
5	31 657 / 217 423	20 877 / 145 037	8 175 / 48 351
6	61 162 / 471 254	39 306 / 311 304	15 157 / 99 539
7	107 236 / 907 708	67 224 / 594 795	26 113 / 186 363
8	175 075 / 1 604 319	107 156 / 1 044 066	42 503 / 324 600
9	270 632 / 2 655 794	161 874 / 1 718 104	66 103 / 534 055
10	400 648 / 4 175 413	234 398 / 2 687 147	99 036 / 839 011
Nb. self-loops	TPN example with concurrency (Figure 3(a))		
1	39 / 72	40 / 74	39 / 72
2	471 / 1 296	472 / 1 299	354 / 963
3	6 735 / 25 056	6 736 / 25 060	2 745 / 9 888
4	119 343 / 563 040	119 344 / 563 045	19 488 / 87 375
5	2 546 679 / 14 564 016	? / ?	130 911 / 701 748
Nb. processes	TPN example with synchronization (Figure 3(b))		
1	1 / 2	2 / 4	1 / 2
2	13 / 35	14 / 38	13 / 35
3	157 / 553	158 / 557	118 / 409
4	2 245 / 10 043	2 246 / 10 048	915 / 3 909
5	3 9781 / 21 7681	39 782 / 217 687	6 496 / 33 071
6	848 893 / 5 495 603	848 894 / 5 495 610	43 637 / 258 051
7	? / ?	? / ?	282 514 / 1.90282e+06
Nb. processes	Fischer protocol		
1	4 / 4	4 / 4	4 / 4
2	18 / 29	19 / 32	20 / 32
3	65 / 146	66 / 153	80 / 171
4	220 / 623	221 / 652	308 / 808
5	727 / 2 536	728 / 2 615	1 162 / 3 645
6	2 378 / 9 154	2 379 / 10 098	4 274 / 15 828
7	7 737 / 24 744	7 738 / 37 961	15 304 / 66 031
8	25 080 / 102 242	25 081 / 139 768	53 480 / 265 040

Table I
EXPERIMENTATION RESULTS

whose behavior is either local, by transition t_i, or global, by transition t_0, is increased.

In addition to these two illustrative examples, we used two other well known parametric TPN models. The first one [11] represents a composition of producer/consumer models by fusion of a single buffer (of size 5). The second (adapted from [16]) is Fischer's protocol for mutual exclusion.



(a) concurrency (b) synchronisation

Figure 3. TPN models used in the experiments

Table I reports the results obtained with the SCG, the ZBG and the TAG-TPN approaches, in terms of graph size (number of nodes/number of edges). The obtained results for the producers/consumers models show that the TAG yields better abstraction (linear order) than the SCG and the ZBG approaches. Each time a new module of producer/consumer is introduced, the size of graphs increases for all three approaches. However, the TAG achieves a better performance than the two other approaches. For the TPN of Figure 3(a), the obtained results show that the size of the TAG exponentially increases when the parallelism occurs in the structure of TPN. This is also the case also for the ZBG and the SCG methods, and we can see that our method behaves better when we increment the self-loop structures in the model. The ZBG's and the SCG's executions have aborted due to a lack of memory when the number of self-loops was equal to 5. The number of edges of the obtained graphs follows the same proportion. In the synchronisation pattern exmaple, our approach also behaves. Indeed, with, 1, 2 and 3 processes, the sizes of the obtained graphs are almost similar with the three approaches. But, from 4 synchronised processes, the size of the SCGs and the ZBGs increase exponentially, leading to a state explosion with 7 processes, whereas the TAGs have been computed

successfully with 7 processes (and even more). The Fischer protocol model is the only model where our approach leads to relatively bad results (although the difference with the two other approaches is linear). Our first explanation is that, in case of disjoint firing intervals, the abstraction can be weak in some cases. In fact, when a transition t is enabled by an aggregate t and there exists a transition t', not enabled by a, s.t. $t_{min} > t'_{max}$, a is considered non-equivalent (while it could be) to all aggregates where the earliest firing time of t is not the same. However, it could be that t and t' are never enabled simultanously, in which case the difference does not play any role. This issue should be investigated in order to refine our abstraction (e.g. by taking into account the structure of the TPN model).

To conclude, the experimental results show (in most cases) an important gain in performance in terms of graph size (nodes/arcs) compared to the SCG and the ZBG approaches for the tested examples. Although we are aware that our approach needs to be confronted to more significant applications, the obtained preliminary results are promising.

VI. Conclusion

We proposed a new symbolic graph for the abstraction of the TPN state space. The proposed graph, called TAG, produces a finite representation of the bounded TPN behaviour and allows for analysing timed reachability properties. Unlike the existing approaches, our abstraction can be directly used to check both state and event-based logic properties. Our ultimate goal is to use the TAG traversal algorithm for the verification of timed reachability properties expressed in the *TCTL* logic. Several issues have to be explored in the future: We first have to carry out additional experimentation (using more significant use cases) to better understand the limits of our approach and to better compare the TAG technique to the existing approaches. Second, we believe that the size of the TAG can be further reduced while preserving time properties without necessarily preserving all the paths of the underlying TPN. We also plan to design and implement model checking algorithms for full *TCTL* logic.

References

[1] R. Alur and D. L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126(2):183–235, 1994.

[2] B. Berthomieu and M. Diaz. Modeling and Verification of Time Dependent Systems Using Time Petri Nets. *IEEE Trans. Software Eng.*, 17(3):259–273, 1991.

[3] B. Berthomieu and M. Menasche. An Enumerative Approach for Analyzing Time Petri Nets. In *IFIP Congress*, pages 41–46, 1983.

[4] B. Berthomieu and F. Vernadat. State Class Constructions for Branching Analysis of Time Petri Nets. In *TACAS 2003*, volume 2619 of *LNCS*, pages 442–457. Springer, 2003.

[5] B. Berthomieu and F. Vernadat. Time Petri Nets Analysis with TINA. In *QEST*, pages 123–124, 2006.

[6] H. Boucheneb, G. Gardey, and O. H. Roux. TCTL Model Checking of Time Petri Nets. *J. Log. Comput.*, 19(6):1509–1540, 2009.

[7] M. Boyer and O. H. Roux. Comparison of the Expressiveness of Arc, Place and Transition Time Petri Nets. In *ICATPN 2007*, volume 4546 of *LNCS*, pages 63–82. Springer, 2007.

[8] G. Gardey, D. Lime, M. Magnin, and O. (h. Roux. Romeo: A Tool for Analyzing time Petri nets. In *In Proc. CAV05, vol. 3576 of LNCS*, pages 418–423. Springer, 2005.

[9] G. Gardey, O. H. Roux, and O. F. Roux. Using Zone Graph Method for Computing the State Space of a Time Petri Net. In *FORMATS 2003*, volume 2791 of *LNCS*, pages 246–259. Springer, 2003.

[10] R. Hadjidj and H. Boucheneb. Improving state class constructions for CTL* model checking of time Petri nets. *STTT*, 10(2):167–184, 2008.

[11] R. Hadjidj and H. Boucheneb. On-the-fly TCTL model checking for time Petri nets. *Theor. Comput. Sci.*, 410(42):4241–4261, 2009.

[12] K. G. Larsen, P. Pettersson, and W. Yi. Model-checking for real-time systems. In *FCT '95*, volume 965 of *LNCS*, pages 62–88. Springer, 1995.

[13] K. G. Larsen, P. Pettersson, and W. Yi. UPPAAL: Status and Developments. In *CAV*, pages 456–459, 1997.

[14] D. Lime and O. H. Roux. Model Checking of Time Petri Nets Using the State Class Timed Automaton. *Discrete Event Dynamic Systems*, 16(2):179–205, 2006.

[15] P. M. Merlin and D. J. Farber. Recoverability of modular systems. *Operating Systems Review*, 9(3):51–56, 1975.

[16] W. Penczek, A. Pólrola, and A. Zbrzezny. SAT-Based (Parametric) Reachability for a Class of Distributed Time Petri Nets. *T. Petri Nets and Other Models of Concurrency*, 4:72–97, 2010.

[17] C. A. Petri. Concepts of net theory. In *MFCS'73*, pages 137–146. Mathematical Institute of the Slovak Academy of Sciences, 1973.

[18] M. Pezzè and M. Young. Time Petri Nets: A Primer Introduction. In *Tutorial at the Multi-Workshop on Formal Methods in Performance Evaluation and Applications*, 1999.

[19] C. Ramchandani. Analysis of asynchronous concurrent systems by timed Petri nets. Technical report, Cambridge, MA, USA, 1974.

[20] J. Sifakis. Use of Petri nets for performance evaluation. *Acta Cybern.*, 4:185–202, 1980.

[21] T. Yoneda and H. Ryuba. CTL model checking of time Petri nets using geometric regions. 1998.