JCSE Journal of Computing Science and Engineering

Regular Paper

Journal of Computing Science and Engineering,
Vol. 9, No. 1, March 2015, pp. 9-19

Improved Region-Based TCTL Model Checking of Time Petri Nets

Mohammad Esmail Esmaili*, Reza Entezari-Maleki, and Ali Movaghar
Department of Computer Engineering, Sharif University of Technology, Tehran, Iran
mesmaili@ce.sharif.edu, entezari@ce.sharif.edu, movaghar@sharif.edu

Abstract
The most important challenge in the region-based abstraction method as an approach to compute the state space of time Petri Nets (TPNs) for model checking is that the method results in a huge number of regions, causing a state explosion problem. Thus, region-based abstraction methods are not appropriate for use in developing practical tools. To address this limitation, this paper applies a modification to the basic region abstraction method to be used specially for computing the state space of TPN models, so that the number of regions becomes smaller than that of the situations in which the current methods are applied. The proposed approach is based on the special features of TPN that helps us to construct suitable and small region graphs that preserve the time properties of TPN. To achieve this, we use TPN-TCTL as a timed extension of CTL for specifying a subset of properties in TPN models. Then, for model checking TPN-TCTL properties on TPN models, CTL model checking is used on TPN models by translating TPN-TCTL to the equivalent CTL. Finally, we compare our proposed method with the current region-based abstraction methods proposed for TPN models in terms of the size of the resulting region graph.

Category: Embedded computing

Keywords: Time Petri Net; Region abstraction; TCTL; State space; Timed automata; Model checking

I. INTRODUCTION

Formal methods are popular techniques for the modeling and verification of timed systems. Because of importance of timed systems in critical applications, verification of these systems has been addressed by many research papers in recent decades. One of the most successful formal methods is model checking, which seeks to automatically verify a property of a system expressed in temporal logics upon a model of the system showing the behavior of the system. Time-dependent models are generally exploited to model time features of timed systems. To specify the property of behavior of timed systems, timed temporal logics can be used. Several models have been proposed for modeling timed systems, such as Timed Automata (TA) [1, 2] and Time Petri Net (TPN) [3, 4]. In TA, as one of the most popular and applicable models which has been generated by extending ω-automata, time can be presented by a real variable called *clock* [1, 2]. TPN is an extension of Petri Nets (PNs) for modeling timed systems in which the time feature is demonstrated by a time interval. Two main timed extensions of PNs are TPN [3] and timed Petri net (TdPN) [4]. TPN, for any transition, assigns a time interval that a transition can fire in the specified interval, but in TdPN, each enabled transition fires upon passing the given duration as soon as

Open Access http://dx.doi.org/10.5626/JCSE.2015.9.1.9 http://jcse.kiise.org

Received 23 August 2014; **Revised** 29 November 2014; **Accepted** 28 February 2015
*Corresponding Author

 pISSN: 1976-4677 eISSN: 2093-8020

possible. Thus, TPN is a general concept, compared with TdPN, which makes it the most popular and compatible approach for modeling the behavior of timed systems. In both models, time can be assigned to *places* (P-TPN, P-TdPN), *transitions* (T-TPN, T-TdPN), or *arcs* (A-TPN, A-TdPN). Here, we consider the extension of TPN in which time is assigned to transitions (T-TPN).

Two main approaches can be found in the literature for model checking of TPNs. The first approach is translating a TPN model into a TA model, which shows that CTL, CTL*, and LTL model checking are decidable for bounded TPN [5-7]. The second is directly computing the abstract state space of the TPN model [8-14]. Because using dense-time firing interval for transitions in a TPN generates an infinite state space, abstraction methods are used to represent the state space of a TPN in a finite manner. To compute the state space of a TPN, there are three major abstraction methods: *state class*, *region-based*, and *zone-based* [8, 9, 12, 15]. The state class method defines a graph in which each node shows a state class containing a special marking and firing domain of transitions enabled in the state class [8, 9]. Although the basic state class technique preserves untimed linear properties, several abstraction methods have been proposed based on state class, such as geometry region [14], atomic state class [10, 14], and strong state class [10], which preserve untimed branching properties CTL* on TPN models. For more information on this concept, see [16]. Region and zone-based methods in the TPN context are the same methods defined for TA, where they use possible clock valuation of transitions for constructing a finite number of states [1, 2, 16, 17]. In [11], the state class based abstraction method was used for on-the-fly model checking of a subset of TCTL properties. In [15, 18], the region graph of the TPN model was exploited to TCTL model checking of TPN. To reduce the state space of TPN, a partial order reduction algorithm has been proposed in [18]. A zone-based approach has been used for computing the state space of TPN in [11, 12]. In [19], the proposed methods for model checking temporal properties on TPN and TA have been studied comprehensively.

The most important challenge in region-based abstraction methods is that these methods generate a very large number of regions, causing *state explosion*. Thus, region-based abstraction methods are not applicable in practice because of their lack of efficiency. Similar to TA, region-based abstraction methods for TPN also result in huge numbers of regions. To moderate this, this paper proposes a modified region-based abstraction method to compute the state space of TPN models, so that the number of regions is smaller than the existing methods, making it possible to be used efficiently in practice. The proposed approach is based on the distinguished feature of TPN against TA that enables us to construct a suitable and small region graph, which preserves the time properties of TPN. Furthermore, we use TPN-TCTL logic [11, 13] as an extension of TCTL [1] for specifying a subset of properties on TPN model.

Due to the nature of the TPN models, we try to define the region graph more effectively in which non-essential regions are integrated into a region, so that the graph obtained from the proposed method is much smaller than the graphs resulting from the current region-based approaches. The main idea is based on two specific features of TPN models: 1) at each marking, it suffices to consider the time value of the clock variables of only enabled transitions during their enabling interval, and 2) in the consecutive regions used for expressing continuous elapsing time of enabled transitions, where there is no possibility to change the marking (without any fireable transition), and in all consecutive regions in which just one single specific transition can fire, we can group these regions into a single region with a single time interval, obtained by combining the value of each clock in all its related regions. Thus, merging these regions, the resulting region graph will have a smaller number of regions, and consequently, the state space of a TPN model can be represented by a fewer number of states.

Using the proposed abstraction method, we can use existing efficient CTL model checking (especially symbolic CTL model checking) algorithms for model checking TCTL properties on TPN models. Moreover, CTL properties based on the proposed marking based logic (without considering the time constraint in logic) can be verified in the resulting transition systems. The proposed method is based on the marking of TPN models and preserves marking reachability of TPN models. Thus, we defined TPN-TCTL logic for specifying a subset of properties that express marking reachability properties. There are still some temporal properties that we cannot verify on the transition system obtained from the proposed method. For example, in [7], the authors proposed a TCTL logic in which a sequence of transitions firings is considered with regard to the bisimulation relation between a TPN and its state class TA, whereas these properties are not preserved by the proposed abstraction method. Due to the fact that marking reachability in the proposed abstraction method is considered, we use an appropriate class of TCTL logic to specify marking properties of TPN models. Given that in most applications, the marking properties are the most important properties required for model checking TPN models, as many research papers have considered, such as [11, 13].

The rest of the paper is organized as follows. In Section II, some preliminaries required to explain the proposed abstraction method are given. In Section III, the proposed approach for region-based state space abstraction of TPN models is introduced. In Section IV, TPN-TCTL which is based on TCTL is introduced for TPN models to specify the time properties, and then, its

semantic is defined by means of a transition system. The experimental results of the proposed method compared to the current region-based methods in term of the size of region graph are given in Section V. Finally, Section VI concludes the paper and presents some guidelines for future work.

II. PRELIMINARIES

In this section, some background information related to TA and TPN required to be able to explain the proposed idea are given. Because of the lack of the space, only important definitions and concepts are introduced here. For more comprehensive information, see [1-4, 16, 19].

A. Timed Automata

Let $\mathbb{N}$ and $\mathbb{R}_{\geq 0}$ denote the sets of natural and non-negative real numbers, respectively. Also, let C be a set of real value clocks. The clock constraint over clocks C, shown as $CC(C)$, is defined as $g ::= x<c \mid x>c \mid x\leq c \mid x\geq c \mid \neg g \mid g\wedge g$, where $c\in\mathbb{N}$ and $x\in C$.

DEFINITION 1. A *TA* is an 8-tuple (L, Act, C, $\rightarrow$, L_0, Inv, AP, $\mathcal{L}$) where L is a finite set of locations (states), Act is a finite set of actions and C is a finite set of real-valued clock variables. In the definition above, $\rightarrow \subseteq L\times CC(C)\times Act\times 2^{C}\times L$ is a finite set of transitions where (l, g, a, r, l')$\in\rightarrow$ is a transition from location l to location l' with the clock constraint (guard) g, action a, and a set of clock variables r that must be reset. Also, $L_0\subseteq L$ is a finite set of initial locations, and $Inv : L \rightarrow CC(C)$ is *invariant-assignment function* that for each location assigns a *clock constraint* over clocks C, AP is a finite set of atomic propositions, and $\mathcal{L} : Loc \rightarrow 2^{AP}$ is a labeling function for each location. Moreover, we can define *clock valuation function* as $V : C \rightarrow \mathbb{R}_{\geq 0}$ that for all clocks $x \in C$ assigns their current values $V(x)$.

Let $TA = (L, Act, C, \rightarrow, L_0, Inv, AP, \mathcal{L})$ is a TA, the semantics of TA is a timed transition system (TTS), $TTS = (S, s_0, \rightarrow)$, where $S = L \times V$ is a finite set of states and $s_0 = (l_0, 0)$ is a finite set of initial states. $\rightarrow\subseteq S(Act\cup\mathbb{R}_{\geq 0})\times S$ is a transition relation including *discrete* and *continuous transitions* defined as follows.

- ***Discrete Transition***: for all $a\in Act$, discrete transition $<l,v>\xrightarrow{\alpha}<l',v'>$ is defined by Eq. (1).

$$\exists(l,\ g,\ a,\ r,\ l')\in\rightarrow \textit{ in TA such as}:\quad \begin{cases} g(v)=true \\ v'=v[r\leftarrow 0] \\ Inv(l')(v')=true \end{cases} \tag{1}$$

- ***Continuous Transition***: for all $d\in\mathbb{R}_{\geq 0}$, continuous transition $<l,v>\xrightarrow{d}<l,v'>$ is defined by Eq. (2).

$$\begin{cases} v'=v+d \\ Inv(l)(v')=true \end{cases} \tag{2}$$

B. Region-Based TCTL Model Checking of TA

As mentioned earlier, there have been two proposed general abstraction methods for computing the state space of a TA: region-based and zone-based methods. The region-based method is based on the concept of *equivalence class*. The basic idea is constructing a finite transition system from a TA in which the states are equivalence classes of the states in the related TA [1, 2, 16].

DEFINITION 2. Let $\lfloor x\rfloor$ and *fract(x)* denote the integral and fractional parts of clock x, respectively, and c_x denote the largest constant that clock x is compared to in any clock constraint of TA. Clock valuations v and v' are *clock equivalent*, denoted by $v\cong v'$, if the following conditions hold:

1. For all $x \in C$, either $\lfloor v(x)\rfloor = \lfloor v'(x)\rfloor$ or $\lfloor v(x)\rfloor > c_x$ *and* $\lfloor v'(x)\rfloor > c_x$.
2. For all x,y $\in$ C, with $v(x) \leq c_x$ *and* $v(y) \leq c_y$, $fract(v(x)) \leq fract(v(y))$ iff $fract(v'(x)) \leq fract(v'(y))$.
3. For all $x \in C$ with $v(x) \leq c_x$, $fract(v(x)) = 0$ iff $fract(v'(x)) = 0$.

DEFINITION 3. After defining equivalence class, $\cong$, a *clock region* denoted by $[v]$ is defined by all clock valuations belonging to the same equivalence class, $[v] = \{v' \in V \mid v' \cong v\}$. Each region can be represented by specifying:

I. For each clock, $x \in C$, one clock constraint from the set :

$$\{x=c \mid c=0,\ 1,\ \ldots\ c_x\}\cup \{c-1<x<c \mid c=1,\ 2,\ \ldots\ c_x\}\cup \{x>c_x\} \tag{3}$$

II. For each pair of clocks, x and y, such that $c - 1 < x < c$ and $k - 1 < y < k$ in Eq. (3), fractional parts of clocks are considered as following sets:

1. $\{c - 1 < x < c \wedge k - 1 < y < k \wedge fract(x) > fract(y)\}$
2. $\{c - 1 < x < c \wedge k - 1 < y < k \wedge fract(x) < fract(y)\}$
3. $\{c - 1 < x < c \wedge k - 1 < y < k \wedge fract(x) = fract(y)\}$

To represent the behavior of a TA as a transition system, each state is shown as pair <s, v>, and *state region* is defined as [s] =< s, [v] >= {<s, v'> | $v' \in [v]$}. The passage of time over several regions implies the change from a region to another. So, the concept of *successor region* is defined for computing all possible regions. The *successor region* of r is r' which is denoted by r' = successor (r), if :

$$\textit{For all } v \in r\text{: } \exists d \in \mathbb{R}_{\geq 0}.\ v + d \in r' \textit{ and } \forall\ 0 \leq d' \leq d.v + d' \in (r\cup r') \tag{4}$$

Also, *successor state* can be defined as *successor*(< s, v >) = < s, *successor*(v) >. Now, we can define a *Region Transition System (RTS)* in which each state includes a state region. To check TCTL properties on RTS obtained from a TA, we need to translate TCTL to the corresponding CTL by eliminating timing constraints in TCTL in which all timed properties have been preserved [1, 16].

C. Time Petri Nets

DEFINITION 4. A *TPN* is a 7-tuple (P, T, ${}^{\bullet}(.)$, $(.)^{\bullet}$, M_0, α, β), where P is a finite set of places, $T = \{t_1, ..., t_n\}$ is a finite set of transitions, ${}^{\bullet}(.) \in (N^P)^T$ and $(.)^{\bullet} \in (N^P)^T$ where N denotes natural numbers that are backward and forward incidence mappings, respectively. $M_0 \in (N^P)$ is the initial marking, $\alpha \in (\mathbb{R}_{\geq 0})^T$ is the earliest firing time of transitions (EFT), and finally $\beta \in (\mathbb{R}_{\geq 0} \cup \{\infty\})^T$ is the latest firing time of transitions (LFT). The semantics of a TPN is also defined by a TTS. Before defining the related TTS, we need to define several concepts. The marking M in TPN is defined by an element of N^P in which for each $p \in P$, $M(p)$ denotes the number of tokens in place p. The transition t_i is *enabled* in marking M ($t_i \in enabled(M)$), if $M \geq {}^{\bullet}t$; where the number of tokens in marking M satisfies the condition required for each place in ${}^{\bullet}t$. Also, *newly enabled* transition t_k after firing transition t_i in marking M which is denoted by $t_k \in \uparrow enabled(M, t_i)$ is defined as follows.

$$\begin{aligned} &\uparrow enabled(M, t_i) = \\ &\{t_k \in T \mid (M - {}^{\bullet}t_i + t_i{}^{\bullet} \geq {}^{\bullet}t_k) \wedge \\ &((M - {}^{\bullet}t_i < {}^{\bullet}t_k) \vee (t_i = t_k))\} \end{aligned} \quad (5)$$

To demonstrate the elapsing of time for transitions, *valuation function* $V \in (\mathbb{R}_{\geq 0})^T$ is defined in which V_k expresses the time elapsed since transition t_k was last enabled. The semantics of a TPN is defined as a $TTS = (S, s_0, \rightarrow)$, where $S = N^P \times (\mathbb{R}_{\geq 0})^T$ is a set of infinite states, $s_0 = (M_0, 0)$ is an initial state that includes initial marking and zero value of all enabled transitions in M_0, and $\rightarrow \in S \times (T \cup \mathbb{R}_{\geq 0}) \times S$ is transition relation including *discrete* and *continuous transitions* as follows.

• ***Discrete Transition***: for all $t_i \in T$, discrete transition $(M, V) \xrightarrow{t_i} (M', V')$ is defined if the following conditions hold:

$$\begin{cases} M \geq {}^{\bullet}t_i \\ M' = M - {}^{\bullet}t_i + t_i{}^{\bullet} \\ \alpha(t_i) \leq V_i \leq \beta(t_i) \\ V'_k = \begin{cases} 0 & ;if\ t_k \in \uparrow enabled(M, t_i) \\ V_k & ;otherwise \end{cases} \end{cases} \quad (6)$$

• ***Continuous Transition***: for all $d \in \mathbb{R}_{\geq 0}$, continuous transition $(M, V) \xrightarrow{d} (M, V')$ is defined as follows.

$$\begin{cases} V' = V + d \\ \forall\ k \in \{1, ..., n\},\ (M \geq {}^{\bullet}t_k \Rightarrow V'_k \leq \beta(t_k)) \end{cases} \quad (7)$$

III. PROPOSED ABSTRACTION METHOD

We propose an abstraction method to compute the state space of TPN models, based on region approach. We try to use the region method efficiently according to the special features of TPN models compared to TA models. To achieve this, several modifications are applied to the region-based abstraction method used in TA for decreasing the number of regions. We describe our proposed method step-by-step by defining some concepts.

DEFINITION 5. The state space of a bounded TPN can be defined as a TTS in which each state is shown as < M, V >, where M is a special marking among all finite markings obtained from the related bounded TPN, and V is the clock valuation of all enabled transitions in marking M. To demonstrate all possible states of a TPN model finitely, clock equivalence is used as described in Definition 2. It turns out that only the clock valuation of enabled transitions in marking M should be considered in definition of regions. Thus, the concept *state region* is defined as Eq. (8), by paying attention to the Definition 3.

$$\begin{aligned} &[s] = < M, [v = \{v_i \mid \forall\ t_i \in enabled(M)\}] > = \\ &\{< M, v' = \{v'_i \mid \forall\ t_i \in enabled(M)\} > \mid v' \in [v]\} \end{aligned} \quad (8)$$

In a sequence of consecutive regions in which changing the marking is not possible and in consecutive regions in which only a single specific transition can fire, we can merge these regions into a single *merged region*. A merged region is expressed by a time constraint obtained by merging clock valuations of all merged regions.

DEFINITION 6. Let r' denote the k^{th} successor region of region r, which is denoted by $r' = successor(r)^k$; for all $k \in \mathbb{N}$, if $k = 0$ then $r' = r$ and otherwise:

$$\begin{aligned} r' &= successor(r)^k = \\ &\underbrace{successor(successor(...successor(r)...))}_{for\ k\ times} \end{aligned} \quad (9)$$

Now, we can define the merged region concept using Definition 6. A *merged region* r_k with length k where the first region is r_i, is defined as $r_k = \cup_{j=0}^{k} successor(r_i)^j$. Hence, in a merged region with length k, the clock valuation of each transition can be represented by one of the following clock constraints:

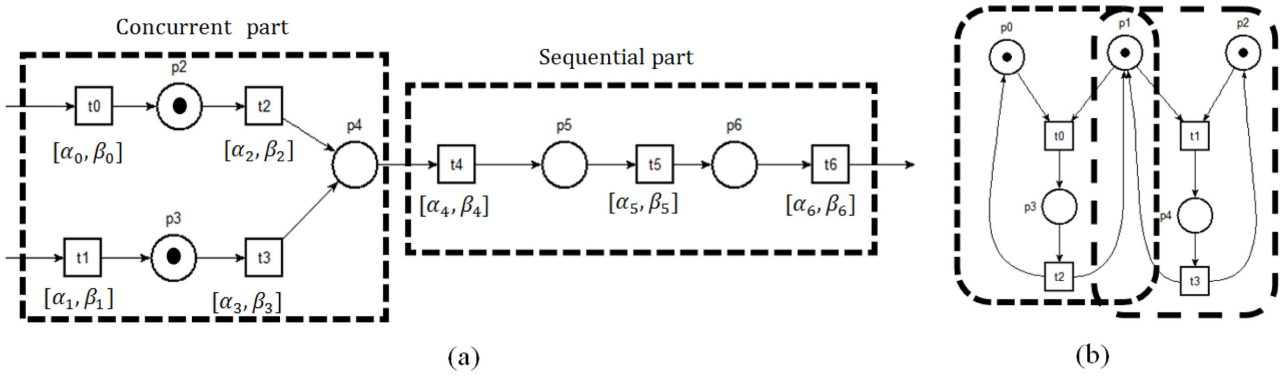

Fig. 1. (a, b) Two simple TPN models and their concurrent and sequential part.

1. $c_1 < v_i < c_2$ shown as interval (c_1, c_2) where in the first region, v_i is represented by clock constraint $c_1 < v_i < c_1 + 1$, and in the last region, v_i is represented by clock constraint $c_2 - 1 < v_i < c_2$ and $k = 2 * (c_2 - c_1) - 1$.
2. $c_1 \leq v_i < c_2$ shown as interval $[c_1, c_2)$, where in the first region, v_i is represented by clock constraint $v_i = c_1$, and in the last region, v_i is represented by clock constraint $c_2 - 1 < v_i < c_2$ and $k = 2 * (c_2 - c_1)$.
3. $c_1 < v_i \leq c_2$ shown as interval $(c_1, c_2]$, where in the first region, v_i is represented by clock constraint $c_1 < v_i < c_1 + 1$, and in the last region, v_i is represented by clock constraint $v_i = c_2$ and $k = 2 * (c_2 - c_1)$.
4. $c_1 \leq v_i \leq c_2$ shown as interval $[c_1, c_2]$ where in the first region, v_i is represented by clock constraint $v_i = c_1$, and in the last region, v_i is represented by clock constraint $v_i = c_2$ and $k = 2 * (c_2 - c_1) + 1$.

The fractional parts of clocks are also considered in each merged region. Each TPN model can be considered as a combination of sequential and concurrent parts, where in a sequential part of a TPN, only a single transition is enabled in each marking, and in a concurrent part, several transitions can fire simultaneously at a specific marking. For example, Fig. 1(a) shows a segment of a TPN model containing one concurrent part and one sequential part. Moreover, Fig. 1(b) shows a simple TPN model for mutual exclusion problem of two processes that consists of two sequential parts.

The distinguished features of TPN models considered in our work for decreasing the number of regions are listed as follows.

1. In a sequential part of a TPN model, the marking does not change before the EFT of the enabled transition. For example, in Fig. 2 the marking m_i cannot be changed at interval [0, 2) and the clock valuation of corresponding transition at this interval can be expressed as a clock constraint $0 \leq x_k \leq 2$. This state is specified by defining a merged region. Also, for firing this transition, a merged region can be defined in which the clock valuation of a transition is represented by a clock constraint $(2 \leq x_k \leq 4)$ that is equal to the firing interval of the transition t_k. Thus,

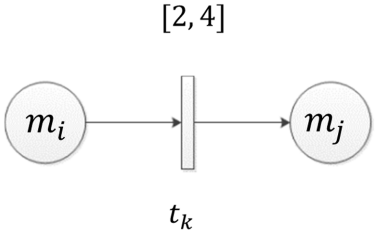

Fig. 2. A small part of a TPN model with a single enabled transition.

in each state of a TPN at which only a single enabled transition can fire in a specific interval, it is possible to express clock valuation of an enabled transition by a merged region.

2. In a concurrent part of TPN models, when several transitions are enabled in a special marking, before minimum EFT of these transitions, marking cannot change. This circumstance can be declared by a single merged region in computing the state space of the corresponding TPN. In a duration that several concurrent enabled transitions can fire simultaneously, a new marking is obtained by firing a transition, which can be shown by a new base region. For example, in Fig. 3, any of three transitions is enabled before it fires. The marking m_0 in time interval (0, 1) cannot change. In interval [1, 2], only transition t_0 can fire, which can be expressed by a single merged region, but in interval (2, 4) in which all transitions t_1 , t_2 and t_3 can fire at any moment, we should consider all possible base regions as Definition 3. Finally, in interval [4, 5] which can be represented by a single merged region, transition t_1 should fire.

The proposed abstraction method is based on the basic region graph approach, which has been proposed for computing the state space of TA models. The basic region can be used for model checking TPN models. Our method uses the basic region concept by considering the special features of the TPN models for decreasing the size of the required region graph to verify marking reachability properties. Our proposed method only uses a com-

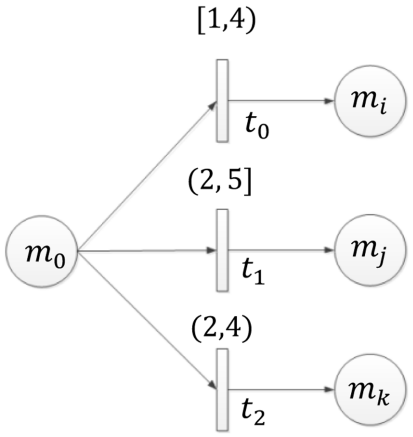

Fig. 3. A small part of a TPN model with three concurrent transitions.

pact representation of regions without any modification in the concept of region. So, we can prove the correctness of the proposed method based on the proved correctness of the basic region abstraction method. The significant issue in our method is using the merged regions that should be considered to demonstrate that our method is correct. By considering that each merged region shows a group of consecutive regions in corresponding basic region graph, all basic regions will be included in region graph, which results from the proposed method.

A. Region Transition System for TPN

Now, we can compute all regions of a TPN model as finitely in which all clock valuations of transitions in the TPN are represented by regions. There exist two types of regions: *base region* as in Definition 3, and *merged region* as in Definition 6. For a merged region, we can define *state merged region* as in Eq. (10) in which the clock valuation of transitions belongs to one of its regions.

$$[s] = <M,\ r> = \{< M,\ v' = \{v'_i | \forall t_i \in enabled(M)\} > | \ \exists r' \in r,\ v' \in r'\} \tag{10}$$

Subsequently, we compute the state space of a TPN model as an *RTS* in which each state contains a *state region* or *state merged region*.

DEFINITION 7. The state space of a $TPN = (P, T, {}^{\bullet}(.), (.)^{\bullet} \in (N^P)^T, M_0, (\alpha, \beta))$ can be defined as a *Region Transition System* $S = (S, s_0, \rightarrow)$, where:

- S is a finite set of states, each $s \in S$ is formed as $< M, r >$ where M is a marking and r is a region (base or merged region).
- $s_0 \in S$ is the initial state as $< M_0, [v_o] >$ where M_0 is an initial marking in TPN, and $[v_o]$ is a region where $v_o = \{v_i = 0 | \forall t_i \in enabled(M_0)\}$.
- $\rightarrow \in S \times (T \cup \mathbb{R}_{\geq 0}) \times S$ is a transition relation including *discrete* and *continuous transitions*, defined as follows.

- ***Discrete Transition***: for all $t_i \in T$, discrete transition $< M, r > \xrightarrow{t_i} < M', r' >$ is defined if all the conditions listed in Eq. (11) are held.

$$\begin{cases} M \geq {}^{\bullet}t_i \\ M' = M - {}^{\bullet}t_i + t_i^{\bullet} \\ v'_j \in r' = \begin{cases} v'_j = 0 & ;if \quad t_j \in \uparrow enabled(M, t_i) \\ v'_j = v_j & ;Otherwise \end{cases} \\ \alpha(t_i) \leq v_i \leq \beta(t_i) \wedge v_i \in r \end{cases} \tag{11}$$

- ***Continuous Transition***: for action τ, in continuous transition $< M, r > \xrightarrow{\tau} < M, r' >$, region r' is a base or merged region, specified by considering aforementioned features described for all sequential and concurrent parts of TPN. Continuous transition is used to move from one state to another state when actually no transition fires in that state.

Fig. 4 shows a simple TPN model containing a single concurrent part with two transitions and its related region graph resulting from our proposed method.

B. Number of Regions in the Proposed Abstraction Method

The number of regions for a TPN model is computed by considering both the number of sequential and concurrent parts of the model and firing intervals of its transitions. To compute the number of regions, each TPN model can be considered as a combination of sets of sequential and concurrent parts. Thus, the number of regions in the TPN model can be computed by considering the following cases.

1) Sequential Transitions

When there is a set of transitions $T_{sequence} = \{t_i, t_{i+1}, ..., t_j\}$ in a sequential part of a TPN, where $\{t_i\} = enabled(m_i)$ and for each k, $i \leq k < j$, $\{t_{k+1}\} = \uparrow enabled(m_k, t_k)$, to represent the clock valuations of enabled transition t_i in marking m_i before it can fire, a single merged region r is used for expressing elapsed time from $x_i = 0$ until EFT of transition t_i. Also, for expressing firing interval of an enabled transition, a single merged region r' is used where clock valuation of the enabled transition is equal to its firing interval. Hence, for firing each transition belonging to a sequential part, two merged regions as r and r' are required. The enabled transition can fire in region r', which causes a new state to be obtained in which the clock valuation of each newly enabled transition in its related region is zero.

Thus, the number of all regions in a sequential part

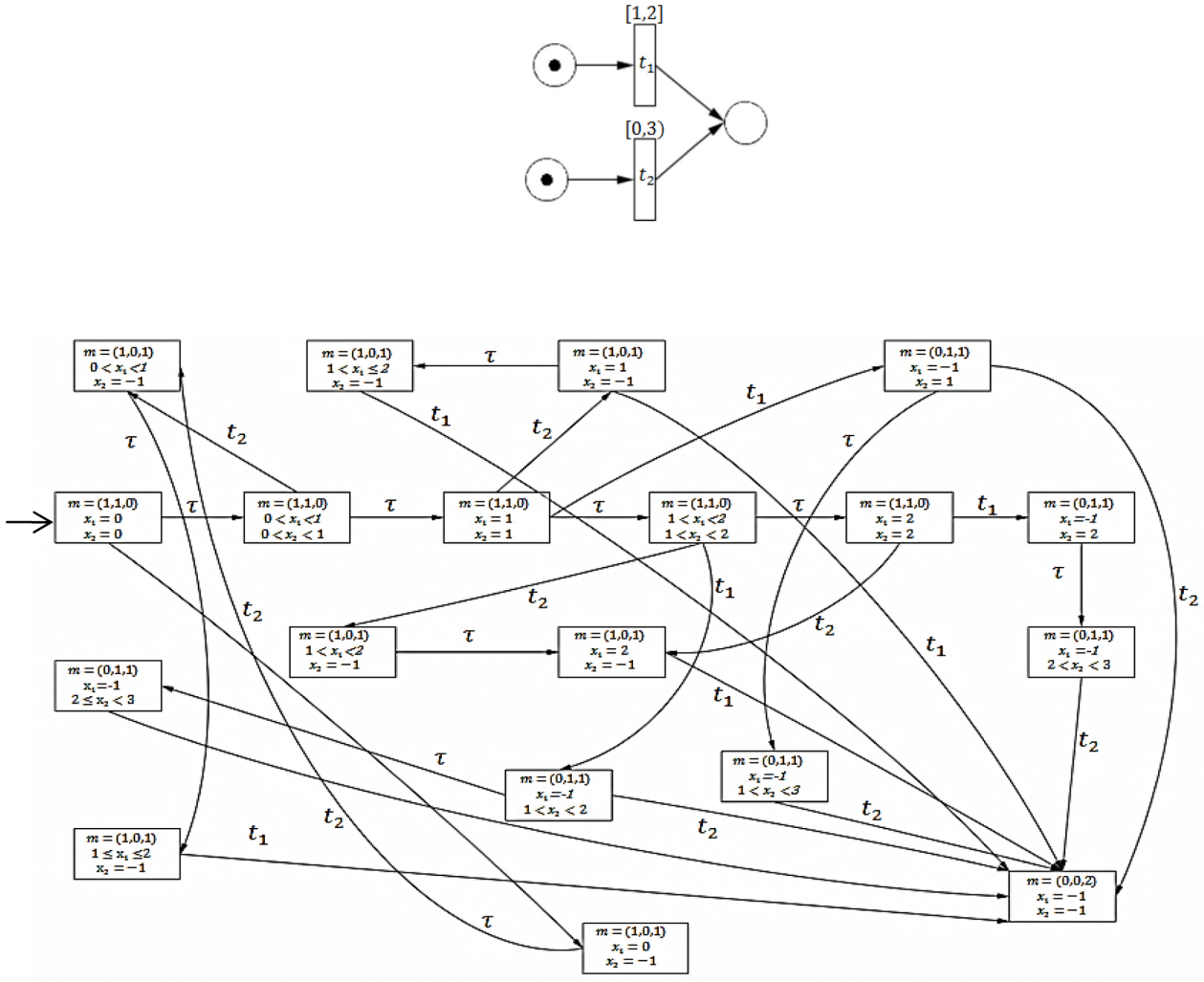

Fig. 4. A simple TPN model and resulted region graph from our proposed approach.

of TPN with transitions $T_{sequence} = \{t_i, t_{i+1}, .., t_j\}$ will be $2*|T_{sequence}|$ where $|T_{sequence}|$ is the number of transitions. According to the description above, the number of all regions for each sequential part of TPN model containing transitions $T_{sequence}$, can be computed linearly in the order of $O(|T_{sequence}|)$.

2) Concurrent Transitions

When several transitions in a TPN model can fire in a single marking, all possible clock valuations of transitions should be considered to compute the number of regions. To do this, we use the concept of *base region* according to the conditions mentioned earlier. The worst case occurs when the concurrent transitions have the most possible overlap in their firing intervals. In this case, all base regions for clock valuation of concurrent transitions should be taken into account. If the firing intervals of all concurrent transitions in this case are the same, then the number of required based regions to express time interval $[\alpha, \beta]$ is computed as $2(\beta - \alpha)+1$. In a state region where all |T| transitions can fire, new regions are created until either all transitions fire or some transitions do not fire, but the last possible region is reached (given the conditions mentioned above, in the last region, the clock valuation of transitions is equal to β). Thus, the number of all possible subset of transitions that can be selected to fire in each region can be computed by Eq. (12).

$$\binom{|T|}{0}+\binom{|T|}{1}+\cdots+\binom{|T|}{|T|}=2^{|T|} \tag{12}$$

With $2(\beta - \alpha) + 1$ number of regions, the number of all possible regions obtained after firing |T| number of transitions is estimated as $(2^{|T|})$. (*MaxDomain*) where *MaxDomain* $= 2(\max(\beta_i) - \min(\alpha_i)) + 1$ at which $\max(\beta_i)$ is the maximum LFT and $\min(\alpha_i)$ is the minimum EFT of transitions in a concurrent part of a TPN. Thus, the number of

regions required to express a concurrent part of TPN with |T| transitions is in order of $O(2^{|T|} \times MaxDomain)$.

Hence, to compute all number of regions considering the set of all sequential and concurrent parts of a TPN model, the formula presented in Eq. (13) can be used:

$$\sum_{i=1}^{|S|}(2*|T_i|)+\sum_{j=1}^{|C|}(2^{|T_j|}*MaxDomain_j) \quad (13)$$

where S is the set of all sequential parts of a TPN that contains $|T_i|$ sequential transitions in i^{th} part, and C is the set of all concurrent parts of TPN that $MaxDomain_j$ is *MaxDomain* of j^{th} part in set c. Thus, in the worst case, all transitions in a TPN model can be considered to be in a concurrent part, which results in the number of regions being bounded by $O(2^{|T|} \times MaxDomain)$. Without loss of generality, it is assumed that each transition is enabled only in a special single marking (i.e., for each $t_i \in T$, $|{}^\bullet t_i| = 1$). So, paying attention to the number of all possible markings (the size of the reachability graph), the number of regions resulting from the proposed method is bounded by $O(|RG| \times 2^{|T|} \times MaxDomain)$ where $|RG|$ is the size of the reachability graph of the TPN model.

C. Region Transition System with Real Clock

In the RTS resulting from a TPN model by applying the proposed algorithm, we use the clock variable for each transition in a certain period to show the passing of time. To be able to check time properties in a desired RTS, it is necessary to measure the actual time elapsed until each state. For this purpose, we add a new clock called *Real Clock* (RC) to the RTS to express the elapsing time till each state. On the other hand, we append an additional transition to the TPN model, which is enabled in all markings [18]. This transition fires never and its clock valuation keeps elapsed time because some properties are satisfied. The new clock independently increases without resetting until a certain property is satisfied [1, 16]. Hence, this clock is used to measure the actual time elapsed until each state that is expressed by a time interval. The largest value of the new clock (i.e., LFT of the newly added transition) is defined by considering the maximal constant in TCTL formula, which should be checked on TPN. The RTS with adding this clock variable is defined as $RTS \oplus RC = (S, s_0, Act, \rightarrow)$, where $S \subseteq M \times (X^{|T|} \cup RC)$, $s_0 \in S$ and $Act = T \cup \{\tau\}$. $\rightarrow \subseteq S \times Act \times S$ is the relation function, the same as the function defined for RTS in Definition 7.

Because the clock valuations of transitions are expressed by time intervals in merged regions against the base region where clock valuations are expressed as in Definition 3, the clock valuation of RC is expressed by an interval in each state. The lower and upper bounds of this interval are the minimum and maximum time required to elapse to reach the considered state, respectively.

IV. TCTL FOR TPN MODELS

To model check the time properties on TPN models, several logics based on TCTL [1] have been proposed [11, 13]. TCTL can be used for specifying branching time properties in TPNs, so we define an extension of TCTL logic for specifying a subclass of properties on TPN models. To do this, according to the proposed method where it preserves marking reachability by considering the firing time of transitions in each state, we should use a suitable logic for defining marking reachability properties to be checked in the transition system that is resulted from the proposed abstraction method.

Definition 8. According to a bounded TPN model in which LFT of transitions is not infinite, we define timed temporal logic *TPN-TCTL*, as follows:

$$\Phi ::= True \mid M(p_i) \sim c \mid \Phi \wedge \Phi \mid$$
$$\Phi \vee \Phi \mid \exists \Phi U_I \Phi \mid \forall \Phi U_I \Phi \mid \neg \Phi \quad (14)$$

where I is a time interval bounded to a natural number, $\sim = \{<, \leq, >, \geq, =\}$ is a set of relational operators, and $c \in \mathbb{N}$ is a natural constant. The notation $M(p_i)$ denotes the number of tokens in place p_i. So, to show a special marking, we can use conjunctions of $M(p_i) \sim c$ in which $\sim$ is $=$. For model checking of TPN-TCTL on TPN, we should translate TPN-TCTL to the equivalent CTL to be able to apply the standard CTL model checking on the desired RTS. For translating TPN-TCTL to the corresponding CTL, timing parameters should be eliminated from the TCTL formula. Timing parameters can be replaced by a clock constraint, based on the new clock added to CTL formula for specifying the time interval considered in TCTL formula [1, 16].

A. Translating TPN-TCTL to CTL

The basic idea for translating a TPN-TCTL formula is that we can express a timing parameter by a clock constraint of a new clock in the equivalent CTL formula. The new clock in CTL measures the elapsing time expressed by the timing parameter in TPN-TCTL, which is denoted by an interval. In fact, we need to express the actual time required to get each state. For this end, we use the concept of RC, defined for RTS. As stated earlier, RC represents the actual time required to reach each state as a time interval. Thus, to check the clock constraint of a new clock added to CTL formula, we can use the valuation of RC. In the TPN-TCTL formula, the timing parameter appears only as an *until* operator; thus, it is sufficient to translate the TCTL *until* operator to the equivalent CTL. Also, *modal* operators $\Diamond$ and $\Box$ can be obtained by the *until* operator. The CTL *until* operator after eliminating timing parameter from TPN-TCTL can be achieved, as follows [1, 16]:

- $s \models \exists(\Phi U_I \Psi)$ *iff*
  $s\{M, RC = 0\} \models \exists((\Phi \vee \Psi) U((RC \in I) \wedge \Psi))$
- $s \models \forall(\Phi U_I \Psi)$ *iff*
  $s\{M, RC = 0\} \models \forall((\Phi \vee \Psi) U((RC \in I) \wedge \Psi))$

where $s\{M, RC = 0\}$ denotes the clock valuation of RC in state s, which can be considered as zero. Therefore, we can use the difference existing between the clock valuation of RC in state s and the desired states to measure the actual elapsed time. Since the atomic propositions in each state of RTS and in TPN-TCTL are markings, we can use the resulting CTL for model checking of TPN-TCTL formula on the RTS.

B. Semantics of TPN-TCTL

Let $RTS' = RTS \oplus RC = (S, s_0, ACT, \rightarrow)$ denote a region transition system after adding the new clock RC obtained from a bounded TPN model, $TPN = (P, T, {}^{\bullet}(.), (.)^{\bullet} \in (N^P)^T, M_0, \alpha, \beta)$. In RTS', each state is denoted by $s = < M, r >$ where M is a marking and r is a region together with the clock valuation of RC. The semantics of TPN-TCTL formula is defined as follows:

- $s \models true$,
- $s \models M(p_i) \sim c$, iff $M_s \models M(p_i) \sim c$,
- $s \models \neg\Phi$ iff $s \not\models \Phi$,
- $s \models \Phi \wedge \Psi$ iff $s \models \Phi$ and $s \models \Psi$,
- $s \models \Phi \vee \Psi$ iff $s \models \Phi$ or $s \models \Psi$,
- $s \models \exists\Phi U_I \Psi$ that equivalent CTL formula is $\exists((\Phi \vee \Psi) U((RC \in I) \wedge \Psi))$. Therefore, $s \models \exists\Phi U_I \Psi$ iff for some $\pi \in Paths(s) : \exists\, j \geq 0.(\pi[j], \models ((RC \in I) \wedge \Psi) \wedge (\forall\ 0 \leq k < j.\pi[k] \models (\Phi \vee \Psi))$, and
- $s \models \forall\Phi U_I \Psi$ that equivalent CTL formula is $\forall((\Phi \vee \Psi) U((RC \in I \wedge \Psi)$. Therefore, $s \models \forall\Phi U_I \Psi$ iff for all $\pi \in Paths(s) : \exists\, j \geq 0.(\pi[j], \models ((RC \in I) \wedge \Psi) \wedge (\forall\ 0 \leq k < j.\pi[k] \models (\Phi \vee \Psi))$.

C. Complexity of TPN-TCTL Model Checking of TPN

Although the number of regions in a transition system of a TA model grows exponentially in the length of clock constraints, by observing only required paths of the considered TCTL formula, TCTL model checking on TA is PSPACE-Complete [1, 2]. Our approach is also based on the region method that was proposed for TA [1, 2]. Although the number of regions, and consequently, the number of states of RTS resulting from a bounded TPN by applying the proposed abstraction method depend on the structure of the model, in the worst case, it is exponential in the number of transitions. With translating the TPN-TCTL formula to the equivalent CTL formula, the problem of TPN-TCTL model checking is reduced to CTL model checking on the resulted RTS. Complexity of the CTL model checking on a transition system is $O(|S| + |E| + |\Phi|)$, where $|S|$ is the number of states, $|E|$ is the number of transitions in a given transition system, and $|\Phi|$ is the length of the CTL formula. Thus, TPN-TCTL model checking on the RTS of a TPN model can be reduced to CTL model checking. Hence, TPN-TCTL model checking is PSPACE-Complete.

V. EXPERIMENTAL RESULTS

To compute the state space of TPN models for TCTL model checking, several methods have been proposed [11, 13, 18]. The proposed approach in [11] is a zone-based abstraction that preserves marking reachability and traces of the TPN models. In [13], by using a state class method, an on-the-fly TCTL model-checking technique for a subclass of TCTL properties was proposed. Although this approach is an efficient model-checking approach, it is defined over a special TPN (called Alarm-clock) and cannot be used for general TPN models. We are especially concerned with region-based abstraction for model checking TCTL properties. The most important challenge in the region-based abstraction method of TPN models is the number of regions. Generally, according to the explanations in Subsection III-B, the number of regions in the proposed method for any given TPN models is bounded by Eq. (15), as follows:

$$|RG|.2^{|T|}.(Max(\beta_i)_{1 \leq i \leq |T|} - Min(\alpha_i)_{1 \leq i \leq |T|}) \qquad (15)$$

whereas using the basic region abstraction method, the number of resulting regions for a k-bounded TPN is bounded by $(k+1)^p.|T|!.2^{|T|}\prod_{x_i \in X}(2.c_{x_i}+2)$ [11, 18], where $|T|$ is the number of transitions, and x_i is a clock associated with the transition t_i, and $c_{x_i} = \beta_i$.

A partial order reduction method was proposed in [18] on a region graph for a TPN model to reduce the size of the region graph. Because our approach and this method were applied on TPN models for reducing the number of regions, we compared our approach with this method according to the example provided in [18] in terms of the size of the resulting region graphs (i.e., RTS). This example demonstrates TPN models for concurrent n-buffer which is extended with time constraints on transitions.

Table 1 shows the numbers of vertexes and edges of the region transition system resulting from this example verifying the property explained in [18] for the capacity of the buffer (parameter n). The column Proposed reports

Table 1. Experimental results of comparing our proposed approach with the two region-based methods

n	Proposed		Partial		Region	
	|V|	|E|	|V|	|E|	|V|	|E|
2	29	44	33	39	36	43
3	74	141	117	153	130	168
4	323	673	923	1,187	1,368	1,813
5	981	2,414	7,341	10,015	10,928	14,632
6	3,225	8,554	85,136	99,814	117,745	151,250
7	11,072	31,790	506,138	679,254	-	-

the results of our proposed abstraction method, column Partial refers to the partial order reduction method proposed in [18], and column Region shows the results of region graph approach given in [1, 2] for TPN models. The results obtained from all three approaches confirm the improvement of our proposed approach in term of the size of region graph. However, the penalty in our method is the increased time required to construct the region graph because of finding suitable merged regions and comparing them in the construction of the region graph. Due to the definition of the merged region, it is necessary to compute all possible basic regions for each merged region by using the original region-based abstraction method. Thus, all basic regions will be computed during using the proposed method. So, this is a penalty of our method compared with the method in [18] not for basic region approach. Finally, with attention to this that the complexity of TCTL model-checking of TPN models in region-based approach determined by resulted region graph (i.e., region transition system), and by considering the size of the resulted transition system, the penalty in our method is negligible and experimental results show a preference for our method compared with the two other region-based methods.

VI. CONCLUSIONS AND FUTURE WORK

For TCTL model checking of TPN models, state class, region-based, and zone-based abstraction methods have been proposed to compute the state space of TPNs. Because of the enormous number of regions generated by region-based methods, these methods are not appropriate for use in practical tools. In this paper, we considered two special features of TPN models for computing the state space of TPN models based on region abstraction method with fewer number of regions to model check a subclass of TCTL properties. To do this, all consecutive regions in which changing the marking is not possible in them are first merged into a single region where clock valuations of transitions are represented by a time interval. Next, all consecutive regions in which just a single specific transition can fire on them are represented by a single region where the value of its clock constraint is specified by a time interval that represents actual time duration for firing the given transition. Then, we exploit TPN-TCTL for specifying a subset of time properties of TPN models. Next, by translating TPN-TCTL to the equivalent CTL, standard CTL model checking is used for model checking of TPN models. Thus, by decreasing the number of regions, the proposed method can be used efficiently to be applied to the large subset of TPN models.

As future work, other temporal properties (e.g., CTL* and MITL) and efficient on-the-fly model checking techniques can be studied to be used in the proposed abstraction method for decreasing the complexity of model checking algorithms in both time and space aspects. Also, the proposed method can be improved by using symbolic methods to store the merged regions in a suitable structure. We are also working on developing a tool for analysis and direct TCTL model checking of TPN models based on the proposed abstraction method.

REFERENCES

1. R. Alur, C. Courcoubetis, and D. Dill, "Model-checking in dense real-time," *Information and Computation*, vol. 104, no. 1, pp. 2-34, 1993.
2. R. Alur and D. Dill, "The theory of timed automata," in *Real-Time: Theory in Practice, Lecture Notes in Computer Science vol. 600*, Heidelberg: Springer, pp. 45-73, 1992.
3. P. M. Merlin and D. J. Farber, "Recoverability of communication protocols: implications of a theoretical study," *IEEE Transactions on Communications*, vol. 24, no. 9, pp. 1036-1043, 1976.
4. C. Ramchandani, "Analysis of asynchronous concurrent systems by timed Petri nets," Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, MA, USA, 1974.
5. F. Cassez and O. H. Roux, "Structural translation from time Petri nets to timed automata," *Journal of Systems and Software*, vol. 79, no. 10, pp. 1456-1468, 2006.
6. D. D'Aprile, S. Donatelli, A. Sangnier, and J. Sproston, "From time petri nets to timed automata: an untimed approach," in *Tools and Algorithms for the Construction and Analysis of Systems, Lecture Notes in Computer Science vol. 4424*, Heidelberg: Springer, pp. 216-230, 2007.
7. D. Lime and O. H. Roux, "State class timed automaton of a time Petri net," in *Proceedings of the 10th International Workshop on Petri Nets and Performance Models (PNPM'03)*, Urbana, États-Unis, 2003, pp. 124-133.
8. 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, 1991.
9. B. Berthomieu and M. Menasche, "An enumerative approach for analyzing time Petri nets," in *Proceedings of the IFIP Congress*, Paris, France, 1983, pp. 41-46.
10. B. Berthomieu and F. Vernadat, "State class constructions for

branching analysis of time Petri nets," in *Tools and Algorithms for the Construction and Analysis of Systems, Lecture Notes in Computer Science vol. 2619,* Heidelberg: Springer, pp. 442-457, 2003.
11. H. Boucheneb, G. Gardey, and O. H. Roux, "TCTL model checking of time Petri nets," *Journal of Logic and Computation*, vol. 19, no. 6, pp. 1509-1540, 2009.
12. 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, Lecture Notes in Computer Science vol. 2791*, Heidelberg: Springer, pp. 246-259, 2004.
13. R. Hadjidj and H. Boucheneb, "On-the-fly TCTL model checking for time Petri nets using state class graphs," in Proceedings of the 6th *International Conference on Application of Concurrency to System Design*, Turku, Finland, 2006, pp. 111-122.
14. T. Yoneda and H. Ryuba, "CTL model checking of time Petri nets using geometric regions," *IEICE Transactions on Information and Systems*, vol. 81, no. 3, pp. 297-306, 1998.
15. Y. Okawa and T. Yoneda, "Symbolic computation tree logic model checking of time Petri nets," *Electronics and Communications in Japan (Part III: Fundamental Electronic Science)*, vol. 80, no. 4, pp. 11-20, 1997.
16. C. Baier and J. P. Katoen, *Principles of Model Checking*, Cambridge, MA: MIT Press, 2008.
17. T. A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine, "Symbolic model checking for real-time systems," *Information and Computation*, vol. 111, no. 2, pp. 193-244, 1994.
18. I. Virbitskaite and E. Pokozy, "A partial order method for the verification of time Petri nets," in *Fundamentals of Computation Theory, Lecture Notes in Computer Science vol. 1684*, Heidelberg: Springer, pp. 547-558, 1999.
19. W. Penczek and A. Pólrola, "Specification and model checking of temporal properties in time Petri nets and timed automata," in *Applications and Theory of Petri Nets 2004, Lecture Notes in Computer Science vol. 3099*, Heidelberg: Springer, pp. 37-76, 2004.

Mohammad Esmail Esmaili

Mohammad Esmail Esmaili received his M.Sc. in Computer Engineering (Software Engineering) from the Sharif University of Technology, Tehran, Iran, in 2014 and B.Sc. in Software Engineering from the University of Science and Culture, Iran, Tehran in 2011. Currently, he is working on cloud computing verification techniques. His research interests include formal verification and testing, model checking of embedded and real-time systems, automata, logics, and system modeling.

Reza Entezari-Maleki

Reza Entezari-Maleki is currently a Postdoctoral Researcher in the School of Computer Science at Institute for Research in Fundamental Sciences (IPM) in Tehran, Iran. He received his Ph.D. degree in Computer Engineering (Software discipline) from the Sharif University of Technology, Tehran, Iran, in 2014, and B.S. and M.S. degrees in Computer Engineering (Software discipline) from the Iran University of Science and Technology, Tehran, Iran in 2007 and 2009, respectively. His main research interests are in performance/dependability modeling and evaluation, grid and cloud computing, and task scheduling algorithms.

Ali Movaghar

Ali Movaghar is a Professor in the Department of Computer Engineering at Sharif University of Technology in Tehran, Iran, and has been on the Sharif faculty since 1993. He received his B.S. degree in Electrical Engineering from the University of Tehran in 1977, and M.S. and Ph.D. degrees in Computer, Information, and Control Engineering from the University of Michigan, Ann Arbor, in 1979 and 1985, respectively. He visited the Institut National de Recherche en Informatique et en Automatique in Paris, France, and the Department of Electrical Engineering and Computer Science at the University of California, Irvine, in 1984 and 2011, respectively, worked at AT&T Information Systems in Naperville, IL in 1985-1986, and taught at the University of Michigan, Ann Arbor, in 1987-1989. His research interests include performance/dependability modeling and formal verification of wireless networks and distributed real-time systems. He is a senior member of the IEEE and the ACM.