Joint NCC & IIS Bull., Comp. Science, 7 (1997), 85–93
© 1997 NCC Publisher

Model checking of time Petri nets*

I.B. Virbitskaite, E.A. Pokozy

The intention of the paper is to develop a model checking algorithm for real time systems represented by time Petri nets [5], and a real time extension of the branching time temporal logic CTL, called TCTL. Since time Petri nets model real time systems over dense time domain, the number of states of any net is infinite. Using a notion of region [1], we construct a finite representation of the behaviour of a time Petri net – the region graph – to which model-checking algorithm can be applied. Some results about the complexity of the model checker proposed are given.

1. Introduction

Nowadays the importance of using temporal logic as an appropriate formalism for specification and verification of concurrent and distributed systems is widely accepted. Research in this area seems to emphasize two directions. The first one concentrates on the proof-theoretic paradigm of manual program verification [7], while the second concerns itself with algorithmically solving special cases, such the case where the system is finite [3]. In the latter strategy, model checker finds out whether a temporal logic formula is true or false in a state transition graph representing the behaviour of a concurrent system.

In reality, concurrent systems must meet hard time constraints. However, traditional logic methods for reasoning about these systems [3] abstract quantitative time away preserving only qualitative properties. R.Alur, C.Courcobetis and D.Dill in [1] have proposed a real time extension of the branching time temporal logic CTL [3], called TCTL, and have defined its semantics based on continuous computation trees. A TCTL model checker with respect to timed graphs (state transition graphs extended with constant bounds on the delays between the state transitions) has also been developed.

Petri nets constitute a powerful automata-theoretic formalism that is often employed to model concurrent and distributed systems. Effective and fairly powerful verification algorithms are known to be successfully adopted to net models. Deductive and algorithmic approaches based on CTL have been proposed in [7] so as to verify behavioural properties of Petri nets. A

*Partially supported by the Volkswagen Foundation under Grant I/70 564, the INTAS-RFBR under Grant 95-01-0378, and the Russian State Committee of High Education for Basic Research in Informatics (Graph-Theoretical Methods in Programming).

temporal logic based approach to analysis of fairness properties of labelled Petri nets has been suggested in [4]. A method of verifying the behaviour of high level nets by means of S-invariants written as temporal logic formulas has been elaborated in [6]. However, in the case of time Petri nets [5], the situation is less advanced. There are only a few reports on verification methods for real time systems represented by time Petri nets. In [8], a model checker, based on a partial order approach, for one-safe time Petri nets and a real time linear time temporal logic has been suggested.

In this paper, we aim at developing a TCTL model checking algorithm for one-safe time Petri nets in which the timing constrains are expressed by associating lower and upper bounds with each transition. As time Petri nets have a more complex structure in comparing with timed graphs, it is necessary to study a possibility of using TCTL and to evaluate the complexity of a model checking algorithm for net systems.

The rest of the paper is organized as follows. The basic definitions concerning time Petri nets are given in the next section. Section 3 recalls the syntax and semantics of TCTL. In section 4, for a time Petri net we define a notion of a region and construct the region graph. A model checking algorithm is then provided. Some remarks about the complexity of the algorithm are finally given.

2. Time Petri nets

'Time Petri Net' [5] is a Petri net with temporal constraints associated to its transitions and expressed as intervals of time.

Let $\mathbf{N}$ be the set of constants $\{0, 1, 2, \ldots\}$ denoting the natural numbers and $\mathbf{R}^+$ be the set of nonnegative real numbers.

Definition 1. A *time Petri net* is a tuple $\mathcal{N} = (P, T, F, Eft, Lft, m_0)$, where

- $P = \{p_1, p_2, \ldots, p_m\}$ is a finite set of places;
- $T = \{t_1, t_2, \ldots, t_n\}$ is a finite set of transitions ($P \cap T = \emptyset$);
- $F \subseteq (P \times T) \cup (T \times P)$ is the flow relation;
- $Eft, Lft : T \to \mathbf{N}$ are functions for the *earliest* and *latest firing times* of transitions satisfying $Eft(t) \leq Lft(t)$ for all $t \in T$;
- $m_0 \subseteq P$ is the initial marking.

For $t \in T$, ${}^\bullet t = \{p \in P \mid (p,t) \in F\}$ and $t^\bullet = \{p \in P \mid (t,p) \in F\}$ denote the *preset* and *postset* of t, respectively. To simplify the presentation, we assume that ${}^\bullet t \cap t^\bullet = \emptyset$ for each transition t. For the sake of convenience, we fix a time Petri net $\mathcal{N} = (P, T, F, Eft, Lft, m_0)$ and work with it throughout what follows.

Figure 1 shows an example of a time Petri net where a pair of numbers, corresponding to a transition, represents its earliest and latest firing times.

A *marking* m of $\mathcal{N}$ is any subset of P. A transition t is *enabled* in a marking m if ${}^\bullet t \subseteq m$ (all its input places have tokens in m), otherwise it is *disabled.* Let $enable(m)$ be the set of transitions enabled in m. Let $\Gamma = [T \to \mathbf{R}^+]$ be the set of *time assignments* for transitions from T. Assume $\nu \in \Gamma$ and $\delta \in \mathbf{R}^+$. Then $\nu + \delta$ denotes the time assignment of the value $\nu(t) + \delta$ to each t from T.

A *state* q of $\mathcal{N}$ is a pair $<m, \nu>$, where m is a marking and $\nu \in \Gamma$. The *initial state* of $\mathcal{N}$ is a pair $q_0 = <m_0, \nu_0>$, where $\nu_0(t) = 0$ for all $t \in T$. Let S denote the set of states of $\mathcal{N}$.

The states of time Petri nets change if time passes or if a transition fires. In a state $q = <m, \nu>$ of $\mathcal{N}$, time $\delta \in \mathbf{R}^+$ *can pass* if for all $t \in enable(m)$, $\nu(t) + \delta \leq Lft(t)$. In this case, the state $q' = <m', \nu'>$ of $\mathcal{N}$ is *obtained by passing* δ from q (written $q \overset{\delta}{\Rightarrow} q'$), if

- $m' = m$,
- $\nu'(t) = \nu(t) + \delta$ for all $t \in T$.

In a state $q = <m, \nu>$ of $\mathcal{N}$, a transition $t \in T$ is *fireable* if $t \in enable(m)$ and $\nu(t) \geq Eft(t)$. In this case, the state $q' = <m', \nu'>$ of $\mathcal{N}$ is *obtained by firing* t from q (written $q \overset{0}{\Rightarrow} q'$), if

- $m' = (m \setminus {}^\bullet t) \cup t^\bullet$,
- $\forall t' \in T \,.\, \nu'(t') = \begin{cases} 0, & \text{if} \quad t' \in enable(m') \setminus enable(m), \\ \nu(t'), & \text{otherwise.} \end{cases}$

A *q-run* r of $\mathcal{N}$ is an infinite sequence of states $q_i \in S$ and time values $\delta_i \in \mathbf{R}^+$ of the form

$$q = q_1 \overset{\delta_1}{\Rightarrow} q_2 \overset{\delta_2}{\Rightarrow} \dots \overset{\delta_{n-1}}{\Rightarrow} q_{n-1} \overset{\delta_n}{\Rightarrow} q_n \dots .$$

We define $time(r, q_n) = \sum_{1 \leq i \leq n} \delta_i$. A state q is *reachable*, if it belongs to some q_0-run. Let RS denote the set of all reachable states of $\mathcal{N}$.

As an illustration, we construct the following q_0-run r of the time Petri net $\mathcal{N}_1$ (see Figure 1):

$$\begin{aligned} r = {} & \langle \{p_1, p_2\}, \nu \equiv 0 \rangle \overset{\delta_1 = 0.7}{\Rightarrow} \langle \{p_1, p_2\}, \nu \equiv 0.7 \rangle \overset{\delta_2 = 0.3}{\Rightarrow} \\ & \overset{\delta_2 = 0.3}{\Rightarrow} \langle \{p_1, p_2\}, \nu \equiv 1 \rangle \overset{\delta_3 = 0}{\Rightarrow} \langle \{p_1, p_5\}, \nu \equiv 1 \rangle \overset{\delta_4 = 1}{\Rightarrow} \langle \{p_1, p_5\}, \nu \equiv 2 \rangle \overset{\delta_5 = 0}{\Rightarrow} \\ & \overset{\delta_5 = 0}{\Rightarrow} \langle \{p_4, p_5\}, \nu \equiv 2 \rangle \overset{\delta_6 = 0}{\Rightarrow} \langle \emptyset, \nu \equiv 2 \rangle \overset{\delta_7 = 3}{\Rightarrow} \langle \emptyset, \nu \equiv 5 \rangle \overset{\delta_8 = 10}{\Rightarrow} \dots . \end{aligned}$$

Then $time(r, \langle \emptyset, \nu \equiv 5\rangle) = \sum_{i=1}^{7} \delta_i = 5$.

$\mathcal{N}$ is *one-safe*, if for every $<m, \nu> \in RS$ and for every $t \in enable(m)$, $t^\bullet \cap m = \emptyset$ holds. To guarantee that in any $q-$run time is increasing beyond any bound, we need the following *progress condition*: for every set of transitions $\{t_1, t_2, \ldots, t_n\}$ such that $\forall\, 1 \le i < n \;\; t_i^\bullet \cap {}^\bullet t_{i+1} \neq \emptyset$ and $t_n^\bullet \cap {}^\bullet t_1 \neq \emptyset$, $\sum_{1 \le i \le n} Eft(t_i) > 0$ holds. Since in the time Petri net $\mathcal{N}_1$ (see Figure 1) $Eft(t_1) + Eft(t_4) = 1$, time is increasing beyond any bound in any q_0-run of $\mathcal{N}_1$.

In the sequel, $\mathcal{N}$ will always denote a one-safe time Petri net satisfying the progress condition.

3. TCTL: syntax and semantics

Timed Computation Tree Logic (TCTL) was introduced by R. Alur, C. Courcoubetis, D. Dill [1] as a specification language for real time systems. We now review the syntax and semantics of TCTL. Let AP be a set of atomic propositions.

For our purpose, it is convenient to take $AP = P$.

Definition 2. The *formula* ϕ of TCTL is inductively defined as follows:

$$\phi := p \mid \neg\phi_1 \mid \phi_1 \rightarrow \phi_2 \mid \forall\phi_1\mathcal{U}_{\sim c}\phi_2 \mid \exists\phi_1\mathcal{U}_{\sim c}\phi_2,$$

where $p \in AP$, $c \in \mathbf{N}$, ϕ_1 and ϕ_2 are formulas of TCTL, $\sim$ stands for one of the binary relations $\{<, \le, =, \ge, >\}$.

Informally, $\exists\phi_1\mathcal{U}_{<c}\phi_2$ means that for some computation path there exists an initial prefix of time length less then c such that ϕ_2 holds in the last state of the prefix, and ϕ_1 holds in all its intermediate states.

We define the derived connectives of the propositional connectives of the propositional calculus such as $\vee$ and $\wedge$ in terms of $\neg$ and $\rightarrow$ in the usual way. In addition, some of commonly used abbreviations are:

- $\forall\Diamond_{\sim c}\phi \equiv \forall true\ \mathcal{U}_{\sim c}\phi$,
- $\exists\Diamond_{\sim c}\phi \equiv \exists true\ \mathcal{U}_{\sim c}\phi$,
- $\forall\Box_{\sim c}\phi \equiv \neg\exists\Diamond_{\sim c}\neg\phi$,
- $\exists\Box_{\sim c}\phi \equiv \neg\forall\Diamond_{\sim c}\neg\phi$.

The unrestricted temporal operators correspond to TCTL-operators subscripted with "≥ 0". In TCTL we can also define temporal operators subscripted with time intervals. For instance, the formula $\exists\Diamond_{(a,b)}\phi$ which says that ϕ holds at least once during the time interval (a, b) along some computation path, can be written as $\exists\Diamond_{=a}\exists\Diamond_{<(b-a)}\phi$.

Definition 3. Given a time Petri net $\mathcal{N}$ and a state $q = <m, \nu> \in RS$, we define the satisfaction relation $q \models \phi$ inductively as follows:

$$\begin{array}{lll} q \models p & <=> & p \in m; \\ q \models \neg\phi_1 & <=> & q \not\models \phi_1; \\ q \models \phi_1 \rightarrow \phi_2 & <=> & q \not\models \phi_1 \text{ or } q \models \phi_2; \\ q \models \exists\phi_1\mathcal{U}_{\sim c}\phi_2 & <=> & \text{for some } q\text{-run } r \text{ of } \mathcal{N}\ r \models \phi_1\mathcal{U}_{\sim c}\phi_2; \\ q \models \forall\phi_1\mathcal{U}_{\sim c}\phi_2 & <=> & \text{for every } q\text{-run } r \text{ of } \mathcal{N}\ r \models \phi_1\mathcal{U}_{\sim c}\phi_2. \end{array}$$

For a q-run $r = (q = q_1 \overset{\delta_1}{\Rightarrow} q_2 \overset{\delta_2}{\Rightarrow} \ldots)$, the relation $r \models \phi_1\mathcal{U}_{\sim c}\phi_2$ holds iff there exists k and $\delta \leq \delta_k$ such that:

1) $(\delta + time(r, q_{k-1})) \sim c$;

2) $q_k \models \phi_2$;

3) $\forall 1 \leq i < k\ \forall 0 \leq \delta' < \delta_i\,.\, <m_i, \nu_i + \delta'> \models \phi_1$;

4) $\forall\ 1 \leq \delta' < \delta\,.\, <m_k, \nu_k + \delta'> \models \phi_1$.

$\mathcal{N}$ satisfies a TCTL-formula ϕ (written $\mathcal{N} \models \phi$) iff $q_0 \models \phi$. A TCTL–formula ϕ is satisfiable iff there is a time Petri net $\mathcal{N}$ such that $\mathcal{N} \models \phi$.

Theorem 1. [1] *The satisfiability question for TCTL is Σ_1^1-hard.*

4. Region graphs and model checking

In this section we will develop an algorithm for deciding whether a time Petri net meets its specification given as a TCTL–formula.

Since a time Petri net constitutes a dense time model, the number of its states is infinite. In order to get a finite representation of the behaviour of a time Petri net, we define a notion of a region [1]. Two states of a time Petri net in the same region are, in some sense, equivalent, i.e., the corresponding time assignment values agree on the integral parts and on the ordering of the fractional parts.

For any $\delta \in \mathbf{R}^+$, $fract(\delta)$ denotes the fractional part of δ, and $\lfloor\delta\rfloor$ denotes the integral part of δ.

Definition 4. Let ν, $\nu' \in \Gamma$. Then $\nu \simeq \nu'$ iff the following conditions are met:

- for each $t \in T$, either $\lfloor\nu(t)\rfloor = \lfloor\nu'(t)\rfloor$ or both $\nu(t)$ and $\nu'(t)$ are greater than $Lft(t)$;
- for each $t,\ t' \in T$ such that $\nu(t) \leq Lft(t)$ and $\nu(t') \leq Lft(t')$:
  - $fract(\nu(t)) \leq fract(\nu(t'))$ iff $fract(\nu'(t)) \leq fract(\nu'(t'))$;

– $fract(\nu(t)) = 0$ iff $fract(\nu'(t)) = 0$.

Let $[\nu]$ denote the equivalence class of Γ which ν belongs to. For $t \in T$, we define $[\nu(t)] = \{\nu(t) \mid \nu \in [\nu]\}$.

Lemma 1. *Let* $<m,\nu>, <m,\nu'> \in RS$ *with* $\nu \simeq \nu'$. *For every TCTL–formula* ϕ, $<m,\nu> \models \phi$ *iff* $<m,\nu'> \models \phi$.

Proof. Immediately follows from Lemma in [1, Section 5]. □

Lemma 2. *The number of equivalence classes of* Γ *induced by* $\simeq$ *is bounded by*

$$| T |! \cdot 2^{2|T|} \cdot \prod_{t \in T} (Lft(t) + 1).$$

Proof. Immediately follows from Definition 3 and Lemma in [1, Section 5]. □

A *region of* $\mathcal{N}$ is a pair $<m, [\nu]>$, where $<m,\nu> \in RS$.

Let $<m, [\nu]>$, $<m', [\nu']>$ be regions of $\mathcal{N}$. Then $<m', [\nu']> = succ(<m, [\nu]>)$ iff for some positive $\delta \in \mathbf{R}^+$ $m = m'$, $\nu + \delta \in [\nu']$ and $\{\nu + \delta' \mid 0 \leq \delta' \leq \delta\} \subseteq [\nu] \cup [\nu']$.

As an example, we consider the time Petri net $\mathcal{N}_2$ (see Figure 2) and its state $<m,\nu>$ with $m = \{p_2, p_3\}$, $\nu(t_1) = \nu(t_2) = 0.7$ and $\nu(t_3) = 0$. Then $<m, [\nu]> = <m, \{\nu' \mid \nu'(t_1) = \nu'(t_2) \in (0,1),\ \nu'(t_3) = 0\}>$ and $succ(<m, [\nu]>) = <m, \{\nu' \mid \nu'(t_1) = \nu'(t_2) \in (0,1),\ \nu'(t_3) \in (0,1),$ $fract(\nu'(t_3)) < fract(\nu'(t_1))\}>$, $succ(succ(<m, [\nu]>)) = <m, \{\nu' \mid \nu'(t_1) = \nu(t_2) = 1,\ \nu(t_3) \in (0,1)\}>$, etc.

Definition 5. The *region graph* $G(\mathcal{N})$ is defined to be a graph (V, E), where:

- a vertex set V is the set of all regions of $\mathcal{N}$;
- an edge set E consists of two types of edges:
  - edges representing the passage of time:

    $(<m, [\nu]>, succ(<m, [\nu]>)) \in E$, if $<m, [\nu]>$, $succ(<m, [\nu]>) \in V$;

  - edges representing the firings of transitions:

    $(<m, [\nu]>, <m', [\nu']>) \in E$, if $<m,\nu>$, $<m',\nu'> \in V$, and $<m',\nu'>$

  is obtained by firing some $t \in T$ from $<m,\nu>$.

The region graph of $\mathcal{N}_1$ is shown in Figure 3.

Lemma 3. *The number of regions of $\mathcal{N}$ is bounded by*

$$\mid T \mid ! \cdot 2^{2|T|+|P|} \cdot \prod_{t \in T} (Lft(t) + 1).$$

Proof. Assume $<m, [\nu]>$ to be a region of $\mathcal{N}$. Since m is an array of the length $\mid P \mid$ consisting of '0' and '1', then the number of different markings of $\mathcal{N}$ is $2^{|P|}$. Moreover, the number of equivalence classes of time assignments is $\mid T \mid ! \cdot 2^{2|T|} \cdot \prod_{t \in T}(Lft(t) + 1)$, by Lemma 2. So, the number of regions of $\mathcal{N}$ is $\mid T \mid ! \cdot 2^{2|T|+|P|} \cdot \prod_{t \in T}(Lft(t) + 1)$. □

There is a simple correspondence between runs of $\mathcal{N}$ and paths through $G(\mathcal{N})$. Let $<m_1, \nu_1> \overset{\delta_1}{\Rightarrow} <m_2, \nu_2> \overset{\delta_2}{\Rightarrow} <m_3, \nu_3> \overset{\delta_3}{\Rightarrow} \ldots$ be a $<m_1, \nu_1>$-run r of $\mathcal{N}$. For each $i \geq 1$, we can find a finite path S_i through $G(\mathcal{N})$ as follows. If $<m_{i+i}, \nu_{i+1}>$ is obtained by passing δ_i from $<m_i, \nu_i>$, then $S_i = \{<m_i, [\nu_i]> = v_i^1, v_i^2, \ldots, v_i^k = <m_{i+1}, [\nu_{i+1}]>\}$, where $v_i^{j+1} = succ(v_i^j)$ for all $1 \leq j < k$. Otherwise, $S_i = \{<m_i, [\nu_i]>, <m_{i+1}, [\nu_{i+1}]>\}$. The path through $G(\mathcal{N})$ obtained by concatenation of sequences S_i corresponds to r in a natural way. Conversely, given path

$$\{<m_1, [\nu_1]>, <m_2, [\nu_2]>, <m_3, [\nu_3]> \ldots\}$$

through $G(\mathcal{N})$, we can construct a corresponding $<m_1, \nu_1>$-run of $\mathcal{N}$ as follows. For each $i \geq 1$, if $<m_{i+1}, [\nu_{i+1}]> = succ(<m_i, [\nu_i]>)$, then there exists a positive time value δ such that $<m_i, \nu_i> \overset{\delta}{\Rightarrow} <m_{i+1}, \nu_i + \delta>$ and $\nu_i + \delta \in [\nu_{i+1}]$. Otherwise, $<m_i, \nu_i> \overset{0}{\Rightarrow} <m_{i+1}, \nu_i>$. So, the given path through $G(\mathcal{N})$ corresponds to a constructed $<m_1, \nu_1>$-run of $\mathcal{N}$.

Given any region $<m, [\nu]>$ of $\mathcal{N}$, satisfaction $<m, [\nu]> \vdash \phi$ is defined in an obvious way.

Labelling algorithm

Suppose we want to determine whether $v = <m, [\nu]> \vdash \phi$. We label the regions with subformulas of ϕ starting from the subformulas of length 1, then of length 2, and so on.

The region graph has information only about reachability, and not about "time" reachability. The time reachability analysis can be performed by introducing an auxiliary transition t^* which is never enabled in any marking of $\mathcal{N}$. $\Gamma^* = [T \cup \{t^*\} \to \mathbf{R}^+]$ is the new set of time assignments. We extend the equivalence relation $\sim$ to t^* supposing that $Lft(t^*)$ is equal to the maximal constant appearing in the formula ϕ. For a time assignment ν and $x \in \mathbf{R}^+$, let $[[x]\nu]$ denote the time assignment from Γ^* that assigns

x to $\nu(t^*)$ and agrees with ν on the values of the remaining transitions. To compute the value of ϕ at v, we consider $<m, [[0]\nu]>$ as an initial region of $\mathcal{N}$.

Let $p_{\sim c}$ be a new proposition corresponding to every subscript $\sim c$ appearing in ϕ and $v' = <m', [\nu']>$ be a region of $\mathcal{N}$ such that $<m', \nu'>$ belong to some $<m, \nu>$-run r. Then $v' \vdash p_{\sim c}$, if $<m', \nu'> \models \nu'(t^*) \sim c$, else $v' \vdash \neg p_{\sim c}$.

Let ψ be a subformula of ϕ. Assume that the regions are already labelled with each subformula of ψ.

Case $\psi \in P$. If $\psi \in m$, then $v \vdash \psi$, else $v \vdash \neg\psi$.

Case $\psi = \neg\phi$. If $v \vdash \phi$, then $v \vdash \neg\psi$, else $v \vdash \psi$.

Case $\psi = \phi_1 \to \phi_2$. If $v \vdash \neg\phi_1$ or $v \vdash \phi_2$, then $v \vdash \psi$, else $v \vdash \neg\psi$.

Case $\psi = Q\phi_1\mathcal{U}_{\sim c}\phi_2$, where Q is either an existential or universal quantifier. The vertices of $G(\mathcal{N})$ are labelled with ϕ_1 or its negation, and with ϕ_2 or its negation. v should be labelled with ψ, if some (or, every, depending upon Q) path through $G(\mathcal{N}, \phi)$ starting at v, has a prefix $v_1, v_2, \ldots, v_n$, such that for each $1 \leq i < n$ $v \vdash \phi_1$, $v_n \vdash \phi_2$, and $v_n \vdash p_{\sim c}$. □

As an example, we consider an application of the labelling algorithm to the time Petri net $\mathcal{N}_1$ (see Figure 1) and the TCTL–formula

$$\phi = \forall true\mathcal{U}_{>1}(p_4 \wedge p_5).$$

Figure 3 shows truth labellings of the regions of $\mathcal{N}_1$ with ϕ. So, using the labelling algorithm, we obtain $\mathcal{N}_1 \models \phi$.

Lemma 4. *Let ψ be a subformula of ϕ or the negation of a subformula of ϕ. If the above labelling algorithm labels $<m, [\nu]>$ with ψ, then $<m, \nu> \models \psi$.*

Proof. Notice that for each q-run of $\mathcal{N}$ there exists a path through $G(\mathcal{N})$ and vice versa. We will prove $<m, [\nu]> \vdash \psi \Rightarrow <m, \nu> \models \psi$ by induction on the structure of ψ. According to the induction hypothesis, for each subformula ψ' of ψ $<m, [\nu]> \vdash \psi' \Rightarrow <m, \nu> \models \psi'$. Then by the above labelling algorithm and Definition 3, we have $<m, [\nu]> \vdash \psi \Rightarrow <m, \nu> \models \psi$. □

Theorem 2. *Let ϕ be a TCTL–formula. There is a decision procedure for checking whether or not $\mathcal{N} \models \phi$ which runs in time*

$$O\Big[|\phi| \cdot |T|! \cdot 2^{|P|} \cdot \prod_{t \in T} Lft(t)\Big].$$

Proof. From Lemma 3, it follows that $|V| = O[|T|! \cdot 2^{|P|} \cdot \prod_{t \in T} Lft(t)]$. For a vertex v of $G(\mathcal{N})$, there are at most $|T|$ output edges representing the firings of transitions and one output edge representing the passage of time. Hence, $| E |= O[| T |! \cdot 2^{|P|} \cdot \prod_{t \in T} Lft(t)]$. Moreover, $G(\mathcal{N})$ can be constructed in the time $O[| V | + | E |]$. So, the labelling algorithm takes the time $O[| \phi | \cdot (| V | + | E |)] = O[| \phi | \cdot | T |! \cdot 2^{|P|} \cdot \prod_{t \in T} Lft(t)]$. □

The model checking algorithm proposed has been implemented as a part of the system PEP (Programming Environment based on Petri Nets) [2].

References

[1] R. Alur, C. Courcoubetis, D. Dill, *Model-checking for real-time systems*, Proc. of 5th IEEE LICS, 1990.

[2] E. Bes, B. Grahlmann, *PEP – more than a Petri net tool*, Proc. of 2nd International Workshop on Tools and Algorithms for the Construction and Analysis of Systems, Lect. Notes in Comp. Sci., **1055**, 1996, 397–401.

[3] E.M. Clarke, E.A. Emerson, A.P. Sistla, *Automatic verification of finite-state systems using temporal logic specifications*, ACM TOPLAS, **8**, No. 2, 1996, 244–263.

[4] R.R. Howell, L.E. Rosier, H.-C. Yen, *A taxonomy of fairness and temporal logic problems for Petri nets*, Lect. Notes in Comp. Sci., **324**, 1988, 351–359.

[5] P. Merlin, D.J. Faber, *Recoverability of communication protocols*, IEEE Trans. on Communication, COM-24(9), 1976.

[6] A. Moslemie, *Automated interpretation of S-invariants of predicate/transition nets: An application of non-classical logics*, Helsinki University of Technology, Digital Systems Laboratory, Tech. Rep. No. 15, 1991.

[7] H. Tuominen, *Logic in Petri net analysis*, Helsinki Univ. of Technology, Digital Systems Laboratory, Res. rep. No. 5, 1988.

[8] T. Yoneda, A. Shibayaama, B.H. Schlingloff, E.M. Clarke, *Efficient verification of parallel real-time systems*, Proc. of 5th International Conference on Computer Aided Verification, Lect. Notes in Comp. Sci., **697**, 1993, 321–333.