Joint NCC & IIS Bull., Comp. Science, 15 (2001), 83–101

LTL model checking of coloured Petri nets based on net unfoldings

V. E. Kozura

In this paper, the model-checking algorithm from [22] is adapted to coloured Petri nets (CPN) [9, 10]. The state-based semantics of LTL for CPN is given and the correctness of the obtained approach is proven. It is also shown how to apply this model checking technique to interval-timed CPN.

1. Introduction

Model checking is a well-known and useful method for verifying the properties of distributed systems. Unfortunately, this method faces the state explosion problem. To avoid this problem, different approaches have been developed, such as the stubborn set method, symbolic binary decision diagrams (BDD), methods based on partial orders, methods using symmetry and equivalence properties of the state space, etc. [20].

In [18], McMillan has proposed an unfolding technique for PN analysis. In his works, instead of the reachability graph, a finite prefix of maximal branching process, large enough to describe a system, has been considered. The size of unfolding is exponential in the general case and there are few works which improve in some way the unfolding definitions and the algorithms of unfolding construction [7, 11].

Initially McMillan has proposed his method for the reachability and deadlock analysis (which has also been improved in the later work [17]). J. Esparza has proposed a model-checking approach to unfolding of 1-safe systems analysis [4] for the S_4 logic. In [1], the model-checking technique has been applied to timed PN. In [22], LTL-based model-checking on PN's unfolding has been developed. LTL model checking on PN unfoldings was further developed in [2, 5, 6]. Unfolding of coloured Petri nets has been considered in the general case in [19] for using it in the dependence analysis needed by the Stubborn Set method.

In papers [12, 13], the unfolding method, as it was developed for ordinary Petri nets, has been applied to coloured Petri nets (in the way they are described in [6, 7]). In [10], symmetry and equivalence specifications for CPN are introduced. It was also shown in [12, 13] how to use the unfolding technique taking into consideration symmetry or equivalence specifications.

In papers [14, 15], the unfolding technique was applied to the interval-timed CPN and to the CPN with the time structure presented in [10].

In this paper, the LTL model checking algorithm based on the net unfolding from [22] is adapted to coloured Petri nets. It is also proven that we can apply the same model-checking algorithm to the interval-timed CPN and their unfoldings presented in [14, 15].

2. Coloured Petri nets

In this section, we briefly remind the basic definitions related to coloured Petri nets and describe the subclass of colours we will use in the paper. A more detailed description of CPN can be found in [9].

A *multi-set* is a function m: $S \to N$, where S is a usual set and N is the set of natural numbers. In the natural way we can define operations, such as $m_1 + m_2$, $n \cdot m$, $m_1 - m_2$, and relations $m_1 \leq m_2$, $m_1 < m_2$. Also $|m|$ can be defined as $|m| = \sum_{s \in S} m(s)$. Let $Var(E)$ define the set of variables of an expression E, and $Type(E)$ define the type of an expression E. Notation $A_M S$ means a multiset over the set A.

A *coloured Petri net (CPN)* is the net $\mathbf{N} = (S, P, T, A, N, C, G, E, I)$, where S, P, T, A are the sets of colours, places, transitions, and arcs such that $P \cap T = P \cap A = T \cap A = \emptyset$; N is a mapping $N : A \to (P \times T) \cup (T \times P)$; C is a colour function $C : P \to S$; G is a guard function such that for all $t \in T$ $Type(G(t)) = bool$ and $Type(Var(G(t))) \subseteq S$; E is a function defined on arcs with $Type(E(a)) = C(p)_{MS}$, where p is the place from $N(a)$ and $Type(Var(E(a))) \subseteq S$ and I is an initial function defined on places such that for all $p \in P$ $Type(I(p)) = C(p)_{MS}$.

$A(t), Var(t), A(x, y) and E(x, y)$ can be defined in a natural way.

A *binding* b is a function from $Var(t)$ such that $b(v) \in Type(v)$ and $G(t)\langle b \rangle$. The set of bindings for t will be denoted by $B(t)$. A *token element* is a pair (p, c), where $p \in P$ and $c \in C(p)$. The set of all token elements is denoted by TE. The set of m token elements (p,c) is denoted by $m'(p, c)$. A *binding element* is a pair (t, b), where $t \in T$ and $b \in B(t)$. The set of all binding elements is denoted by BE. A *marking* M is a multi-set over TE. A *step* Y is a multi-set over BE. A step Y is *enabled* in the marking M, if for all $p \in P$ $\sum_{(t,b) \in Y} E(p, t)\langle b \rangle \leq M(p)$ and a new marking M_1 is given by $M_1(p) = M(p) - \sum_{(t,b) \in Y} E(p, t)\langle b \rangle + \sum_{(t,b) \in Y} E(t, p)\langle b \rangle$.

The subclass of coloured Petri nets presented in [12, 13] is large enough to describe many interesting systems and still allows us to build a finite prefix of its branching process. The detailed description can be found in [12]. The set of basic colour domains is obtained from the types of *Standard ML (SML)* [9] by considering only finite colour domains $s \in S$. All functions defined in [9]

which have the above described classes as their domains are allowed in our subclass. The CPN satisfying all the above-mentioned requirements is called *S-finite*.

A marking M of a CPN is *n-safe* if $|M(p)|\leq n$ for all $p\in P$. A CPN is called *n-safe* if all of its reachable markings are n-safe. 1-safe net is also called *safe*. A *preset* of an element $x\in P\cup T$ denoted by ${}^\bullet x$ is the set ${}^\bullet x = \{y\in P\cup T \mid \exists a : N(a) = (y, x)\}$. A *postset* of $x\in P\cup T$ denoted by $x^\bullet$ is the set $x^\bullet = \{y\in P\cup T \mid \exists a : N(a) = (x, y)\}$.

The CPN considered in this paper are the CPN satisfying three additional properties:

- *the number of places and transitions is finite,*
- *the CPN is n-safe,*
- *the CPN is S-finite.*

3. Branching process of coloured Petri nets

Let $\mathbf{N}$ be a Petri net. We will use the term *nodes* for both places and transitions. The nodes x_1 and x_2 are *in conflict*, denoted by $x_1\sharp x_2$, if there exist transitions t_1 and t_2 such that ${}^\bullet t_1\cap {}^\bullet t_2\neq\emptyset$ and (t_1,x_1) and (t_2,x_2) belong to the transitive closure of N (which we denote by R_t). The node x is in *self-conflict* if $x\sharp x$. We will write $x_1\leq x_2$ if $(x_1, x_2)\in R_t$ and $x_1 < x_2$ if $x_1\leq x_2$ and $x_1\neq x_2$. We say that x *co* y, or $x\| y$, if neither $x < y$, nor $x > y$, nor $x\sharp y$.

An Occurrence Petri Net (OPN) is an ordinary Petri net $\mathbf{N} = (P, T, N)$, where

1. P, T are the sets of places and transitions,
2. $N\subseteq(P\times T)\cup(T\times P)$ gives us the incidence function,

satisfying the following properties:

1. for all $p\in P$ $|{}^\bullet p|\leq 1$;
2. N is acyclic, i. e., the (irreflexive) transitive closure of N is a partial order;
3. N is finitely preceded, i. e., for all $x\in P\cup T$ the set $\{y\in P\cup T \mid y\leq x\}$ is finite, which gives us the existence of $Min(\mathbf{N})$, the set of minimal elements of $\mathbf{N}$ with respect to R_t;
4. no transition is in self-conflict.

Let $\mathbf{N}_1 = (P_1, T_1, N_1)$ and $\mathbf{N}_2 = (P_2, T_2, N_2)$ be two Petri nets. A *homomorphism* h from $\mathbf{N}_2$ to $\mathbf{N}_1$ is a mapping $h : P_2\cup T_2 \to P_1\cup T_1$, such that

1. $h(P_2) \subseteq P_1$ and $h(T_2) \subseteq T_1$;
2. for all $t \in T_2$ $h|_{\bullet t} = {}^\bullet t \to {}^\bullet h(t)$, for all $t \in T_2$ $h|_{t^\bullet} = t^\bullet \to h(t)^\bullet$.

Now we give the main definition of the section. This is the first novelty of the paper, a formal definition of a branching process for coloured Petri nets. After the following definition, the existence result is given.

Definition 1. A *branching process* of a CPN

$$\mathbf{N}_1 = (S_1, P_1, T_1, A_1, N_1, C_1, G_1, E_1, I_1)$$

is a tuple $(\mathbf{N}_2, h, \varphi, \eta)$, where $\mathbf{N}_2 = (P_2, T_2, N_2)$ is an OPN, h is a homomorphism from $\mathbf{N}_2$ to $\mathbf{N}_1$, φ and η are the functions from P_2 and T_2, respectively, such that

1) $\varphi(p) \in C_1(h(p))$;
2) $\eta(t) \in B(h(t))$.

Other requirements are listed below:

3) for all $p_1 \in P_1$ $\sum_{p \in Min(\mathbf{N}_2) \mid h(p)=p_1} \varphi(p) = M_0(p_1)$;
4) $G_1(h(t))\langle \eta(t) \rangle$ for all $t \in T_2$;
5) $\forall t' \in T_2 \mid (\exists a \in A_1 : N_1(a) = (p,t) \text{ and } h(t') = t) \Longrightarrow$
   $E_1(a)\langle \eta(t') \rangle = \sum_{(p' \in {}^\bullet t' \mid h(p')=p)} \varphi(p')$,
   $\forall t' \in T_2 \mid (\exists a \in A_1 : N_1(a) = (t,p) \text{ and } h(t') = t) \Longrightarrow$
   $E_1(a)\langle \eta(t') \rangle = \sum_{(p' \in t'^\bullet \mid h(p')=p)} \varphi(p')$;
6) If $(h(t_1) = h(t_2))$ and $(\eta(t_1) = \eta(t_2))$ and $({}^\bullet t_1 = {}^\bullet t_2)$, then $t_1 = t_2$.

Using the first two properties, we can associate a token element (p,c) of $\mathbf{N}_1$ with every place in $\mathbf{N}_2$ and the binding element (t,b) of $\mathbf{N}_1$ with every transition in $\mathbf{N}_2$. So we can further consider the net $\mathbf{N}_2$ as containing the places, which we identify with token elements of $\mathbf{N}_1$, and transitions, which we identify with binding elements of $\mathbf{N}_1$. So we sometimes use them instead, like $h((t,b)) = t$ means that for some t' $h(t') = t$ and $\eta(t') = b$ or $p \in {}^\bullet(t,b)$ means $p \in {}^\bullet t'$ and $h(t') = t$ and $\eta(t') = b$. Analogously, we can consider $(p,c) \in P_2$ as $p' \in P_2$ and $h(p') = p$ and $\varphi(p) = c$. Also, $h(p,c) = p$ and $h(t,b) = t$.

It can be shown that any finite CPN has a maximal branching process (MBP) up to isomorphism (Theorem 1). We can declare existence of the maximal branching process when considering the algorithm of its generation. The algorithm is described in [12] and the following theorem is proven there.

Theorem 1. *For a given CPN* $\mathbf{N}$*, there exists a maximal branching process MBP(*$\mathbf{N}$*).*

This branching process can be infinite even for the finite nets if they are not acyclic. We are interested in finding a finite prefix of a branching process large enough to represent all the reachable markings of the initial CPN. This finite prefix will be called an unfolding of the initial CPN.

4. Unfoldings of CPN

A *configuration* C of an OPN $\mathbf{N} = (P, T, N)$ is a set of transitions such that $t \in C \Longrightarrow$ for all $t_0 \leq t$, where $t_0 \in C$ and for all $t_1, t_2 \in C$ $\neg(t_1 \sharp t_2)$. A set $X_0 \subseteq X$ of nodes is called a *co-set*, if for all $t_1, t_2 \in X_0$: (t_1 co t_2). A set $X_0 \subseteq X$ of nodes is called a *cut*, if it is a maximal co-set with respect to the set inclusion.

Finite configurations and cuts are closely related. Let C be a finite configuration of an occurrence net, then $Cut(C) = (Min(\mathbf{N}) \cup C^\bullet) \setminus {}^\bullet C$ is a cut.

Let $\mathbf{N}_1 = (S_1, P_1, T_1, A_1, N_1, C_1, G_1, E_1, I_1)$ be a CPN and MBP($\mathbf{N}_1$) = $(\mathbf{N}_2, h, \varphi, \eta)$, where $\mathbf{N}_2 = (P_2, T_2, N_2)$, be its maximal branching process. Let C be a configuration of $\mathbf{N}_2$. We define a marking *Mark(C)* which is a marking of $\mathbf{N}_1$ such that $Mark(C)(p) = \sum_{(p' \in Cut(C) \mid h(p')=p)} M_2(p')$.

Let $\mathbf{N}$ be an OPN. For all $t \in T$ the configuration $[t] = \{t' \in T \mid t' \leq t\}$ is called a *local configuration*. (The fact that [t] is a configuration can be easily checked).

Let us consider the maximal branching process for a given CPN. It can be noticed that MBP($\mathbf{N}$) satisfies the completeness property, i. e., for every reachable marking M of $\mathbf{N}$ there exists a configuration C of MBP($\mathbf{N}$) (i. e., C is the configuration of OPN), such that $Mark(C) = M$. Otherwise we could add a necessary path and generate a larger branching process. This would be a contradiction with the maximality of MBP($\mathbf{N}$).

Now we are ready to define three types of cutoffs used in the definition of an unfolding. The first two definitions for ordinary PNs can be found in [4, 18]. The last is the definition given in [11].

Definition 2. Let $\mathbf{N}$ be a coloured Petri net and MBP($\mathbf{N}$) be its maximal branching process. Then

1. a transition $t \in T$ of an OPN is a *GT_0-cutoff*, if there exists $t_0 \in T$ such that $Mark([t]) = Mark([t_0])$ and $[t_0] \subset [t]$;
2. a transition $t \in T$ of an OPN is a *GT-cutoff*, if there exists $t_0 \in T$ such that $Mark([t]) = Mark([t_0])$ and $|[t_0]| < |[t]|$;
3. a transition $t \in T$ of an OPN is a *EQ-cutoff*, if there exists $t_0 \in T$ such that
   (a) $Mark([t]) = Mark([t_0])$,
   (b) $|[t_0]| = |[t]|$,

(c) $\neg(t \| t_0)$,
(d) there are no EQ-cutoffs among t' such that $t' \| t_0$ and $|[t']| \leq |[t_0]|$.

Definition 3. For a coloured Petri net **N**, an *unfolding* is obtained from the maximal branching process by removing all the transitions t', such that there exists a cutoff t and $t < t'$, and all the places $p \in t'^{\bullet}$. If Cutoff = GT_0(GT)-cutoffs, then the resulted unfolding is called *GT_0(GT)-unfolding*. GT_0(GT)-unfolding is also called the *McMillan unfolding*. If Cutoff = GT-cutoffs ∪ EQ-cutoff, then the resulted unfolding is called *EQ-unfolding*.

It has been shown that the McMillan unfoldings are inefficient in some cases. The resulting finite prefix grows exponentially, when the minimal finite prefix has only a linear growth. The following proposition can be formulated for these three types of unfoldings ([12]).

Proposition 1.
size(EQ-unfolding) $\leq$ *size(GT-unfolding)* $\leq$ *size(GT_0-unfolding).*

The following theorem presents the main result of this section ([12]).

Theorem 2. *Let* **N** *be a CPN. Then for its unfoldings we have:*

1. *EQ-unfolding, GT-unfolding and GT_0-unfolding are finite.*
2. *EQ-unfolding, GT-unfolding and GT_0-unfolding are safe, i. e., if C and C' are configurations, then* $C \subseteq C' \implies Mark(C') \in [Mark(C)\rangle$.
3. *EQ-unfolding, GT-unfolding and GT_0-unfolding are complete, i. e.,*
   $M \in [M_0\rangle \implies$ *there exists a configuration C such that Mark(C) = M.*

In the general case, the algorithm of an unfolding construction proposed in [18] and applied to coloured Petri nets in [12] has an exponential complexity. The algorithm from [11] is rather efficient in the speed of unfolding generation. In the case of an ordinary PN, it gives the overall complexity $O(N_P \cdot N_T)$, where N_P and N_T are the numbers of places and transitions in EQ-unfolding. This algorithm was also transferred to coloured Petri nets [12] and a close estimation holds if we do not take into consideration the calculation complexity of arc and guard functions. In this case, we obtain $O(N_P \cdot N_T \cdot B)$, where $B = max\{|B(t)| \ : \ t \in T_{CPN}\}$.

5. Model checking of CPN

5.1. LTL logic

Let us briefly describe here the basic ideas concerned with the LTL logic. Let R be the set of atomic propositions.

Definition 4. The set of LTL-formulae is defined inductively as follows. If $\phi = a \in R$, then ϕ is a formula. If ϕ and φ are formulae, then $\phi \vee \varphi$, $\phi \wedge \varphi$, $\neg\phi$, $X\phi$, and $\phi U \varphi$ are formulae. Another operators are defined in the following way.

$\Diamond\phi = trueU\phi$

$\Box\phi = \neg\Diamond\neg\phi$

The set of propositions from ϕ is denoted by $\langle\phi\rangle$. We interpret the formula ϕ on *w-words* over the alphabet 2^R. A w-word ξ over the 2^R is an infinite sequence $\xi = a_0a_1...$, where $a_i \in 2^R$ for all i. The proposition $a \in R$ holds at a_i iff $a \in a_i$. We denote by $\xi \models \phi$ the fact that the w-word $\xi = a_0a_1...$ satisfies ϕ. This is defined inductively as follows.

Definition 5.

$\xi \models \pi$ iff $\pi \in \xi(0)$

$\xi \models \neg\phi$ iff not $\xi \models \phi$

$\xi \models \phi \wedge \varphi$ iff $\xi \models \phi$ and $\xi \models \varphi$

$\xi \models X\phi$ iff $\xi(1) \models \phi$

$\xi \models \phi U \varphi$ iff $\exists i \geq 0\ \xi(i) \models \varphi$ and $\xi(j) \models \phi$ for all $j < i$.

By L_ϕ we denote the set of w-words satisfying ϕ. According to the approach of the automata theory, we consider the Buechi automaton as representation of a given LTL formula.

Definition 6. A Buechi automaton over the alphabet 2^R is a quadruple $A = (Q, q_0, \delta, F)$, where Q is a finite set of states, q_0 is an initial state, $\delta \subseteq Q \times 2^R \times Q$ is the transition relation, and $F \subseteq Q$ is a set of *accepting states*.

Definition 7. A *run* of A on the w-word ξ is an infinite sequence $\sigma = q_0q_1...$ such that for all $i \geq 0$ $(q_i, \xi(i), q_{i+1}) \in \delta$. A run is called *accepting* if an accepting state occurs in δ infinitely often. Automaton A *accepts* the word ξ iff there exists an accepting run of A on ξ.

Let L(A) denote the set of all w-words accepted by A.

Theorem 3. *([21]). Let ϕ be an LTL formula. There exists a Buechi automaton A_ϕ, such that $L(A_\phi) = L_\phi$.*

There are some efficient methods for constructing the automaton A_ϕ from a given formula ϕ, for example, the method proposed in [8]. Having the system as an automaton A over the alphabet 2^R and the formula ϕ, we construct, according to the automata-theoretic approach, the product of A and the Buechi automaton $A_{\neg\phi}$. The basic idea is that we make a transition $(\langle a, b\rangle, \langle a', b'\rangle)$, if (a, T, a') belongs to the transition relation of the automata A and (b, T, b') belongs to the transition relation of the automata $A_{\neg\phi}$.

It holds for this construction that the product contains no cycle including an accepted state iff the system automaton A satisfies ϕ.

Below we define the state-based LTL semantics for coloured Petri nets. Let us remind that we consider only n-safe CPN. The set of atomic propositions is identified with the set $n \cdot TE = TE \times TE \times ... \times TE$, where n is the number from the n-safeness condition and TE is the set of token elements. A proposition $m'(p,c)$ holds at the state M iff $m'(p,c) \in M(p)$.

The sequence of steps $M_0[t_0, b_0\rangle M_1[t_1, b_1\rangle M_2...$ satisfies ϕ iff the w-word $M_0M_1M_2...$ belongs to L_ϕ. We say that the marking M satisfies ϕ if any sequence of steps $M[t, b\rangle M_1[t_1, b_1\rangle M_2...$ satisfies ϕ. In this case we write $M \models \phi$. The CPN N satisfies ϕ if M_0 satisfies ϕ. We write this as $N \models \phi$. If $(p,c) \in \langle\phi\rangle$, then sometimes we will say for convenience that $p \in \langle\phi\rangle$.

5.2. Model checking technique

The model checking problem requires the detection of cycles that contain accepting transitions. To find such a cycle, we use the algorithm proposed in [22]. We suppose the CPN **N** to be "deadlock free". The effective method of finding the deadlocks using the unfoldings of CPN is described in [12]. If the obtained deadlocks have some system meaning, for example, the final state of getting all the sent messages in communicational protocols, we can easily modify the system CPN by adding some transitions and eliminate the deadlock states.

The problem is solved in two steps. First, the direct graph $G = (V, Edg)$ is constructed, where $V = Off$ is the set of cut-off events of the prefix, and Edg is the set of edges. An edge $e \to e'$ means that the state $[e']$ is reachable from the state $[e]$. Some of the edges will be labelled by a. If the edge $e \to e'$ is labelled by a, then this means that on the partial computation from $[e]$ to $[e']$ an accepting transition occurs.

The second step is to apply standard algorithms on G for detecting a strongly connected component or a cycle containing an a-labelled edge.

We will show how to adapt the automata-theoretic approach and the approach proposed in [22] to the LTL model checking of coloured Petri nets.

Let us have a CPN $\mathbf{N} = (S, P, T, A, N, C, G, E, I)$ and an automaton $A_{\neg\phi} = (Q, q_0, \delta, F)$.

We consider the automaton $A_{\neg\phi}$ to be a CPN in the following way. The set of states is considered to be the set of places. The colour set consists only of one colour element */color H=unit with e/*. Each edge of the obtained automaton net is considered to be a transition without any guard condition. Initial marking A_0 has only one token at the initial state q_0. Each transition is considered to be labelled with n-safe marking consisting of token elements $(p,c) \in \langle\phi\rangle$.

Definition 8. Let us have a CPN $\mathbf{N} = (S, P, T, A, N, C, G, E, I)$ and an automaton $A_{\neg\phi} = (Q, q_0, \delta, F)$ considered to be a CPN. The *Prod* is constructed so that the automaton net and the system alternate their moves.

1. First we add a complementary place p' for each place $p \in P$ of the system CPN, so that $(p, c) \in \langle\phi\rangle$ for some $c \in C(p)$. This is done to test the marking M when the transition of the automaton net labelled by M is fired. For the complementary place we propose $M(p') = \sum_{\{c|(p,c)\in\langle\phi\rangle\}}(n'(p, c)\backslash M(p))$. The set of such complementary places is denoted by P'. The set $Obs(phi) = \{p, p' \mid (p, c) \in \langle\phi\rangle\}$ is called the set of *observable places*.
2. We add the places s_f and s_s to organize the alternation of the moves. The colour set for these places is $C(s_f) = C(s_s) =$ */color H=unit with e/*. This models the simple (uncoloured) places. The whole set of places is $P \cup P' \cup Q \cup \{s_f, s_s\}$.
3. For each transition t of the automaton net labeled by M, we add the arcs $a_{p,t}$ and $a_{t,p}$ as described below:

   $N(a_{p,t}) = (p, t)\ \forall p \in Obs(phi)$, $N(a_{t,p}) = (t, p)\ \forall p \in Obs(phi)$

   $E(a_{p,t}) = E(a_{t,p}) = m'(p, c)\ \forall p$ such that $(p, c) \in \langle\phi\rangle$, where m is the index of (p, c) in M.

   For complementary places $E(a_{p,t}) = E(a_{t,p}) = (n - m)'(p, c)$.
4. For each transition t of the automaton net we add the arcs $a_{t,s}$ and $a_{t,f}$ as described below:

   $N(a_{t,s}) = (t, s_s)$, $N(a_{t,f}) = (s_f, t)$, $E(a_{t,s}) = E(a_{t,f}) = e$.

   For each transition T of the system CPN, we add the arcs $a_{T,s}$ and $a_{T,f}$ as described below:

   $N(a_{T,s}) = (s_s, T)$, $N(a_{T,f}) = (T, s_f)$, $E(a_{T,s}) = E(a_{T,f}) = e$.
5. For each transition T of the system CPN such that for some $(p, c) \in \langle\phi\rangle$ $p \in {}^\bullet T$ or $p \in T^\bullet$, we add the arc $a_{N,p}$ for each such place p. $N(a_{T,p}) = (T, p)$ if $p \in {}^\bullet T$, and $N(a_{T,p}) = (p', T)$ otherwise. If $p \in {}^\bullet T$ and $p \in T^\bullet$, then we add both arcs. The arc expressions are $E(a_{T,p}) = E(p, T)$ in the case $N(a_{T,p}) = (T, p)$, and $E(a_{T,p}) = E(T, p)$ otherwise.
6. The initial marking is $M_0 \cup M'_0 \cup \{q_0\} \cup \{s_f\}$, where M_0 is the initial marking of the system CPN, $M'_0(p') = \sum_{\{c|(p,c)\in\langle\phi\rangle\}}(n'(p, c)\backslash M_0(p))$, $\{q_0\} \cup \{s_f\}$ means that we put initially one token in the places q_0 and s_f.

Let $\gamma = M_0[t_0, b_0\rangle M_1[t_1, b_1\rangle ...$ be the sequence of steps in *Prod*. For each state M_j of γ, let $P_j = M_j \cap P$ be the restriction of M_j onto system

places. The projection of γ onto system places is defined as $Proj(\gamma) = P_{i0}[t_{i0}, b_{i0}\rangle P_{i1}[t_{i1}, b_{i1}\rangle ....$ The sequence $P_{i0}P_{i1}P_{i2}...$ is denoted by $Proj_M(\gamma)$.

Proposition 2. *Let Prod be the product of a CPN* $\mathbf{N}$ *and* $A_{\neg\phi}$. *N satisfies* ϕ *iff Prod contains no cyclic sequences of steps including the places from* $A_{\neg\phi}$ *corresponding to accepting states.*

Proof. Follows from the construction of *Prod* and the above-described state-based semantics of LTL for CPN.

For the marking M of the system CPN we consider the submarking M^{obs} consisting of places from $\langle\phi\rangle$. From the LTL-semantics, $M^{obs} \models \phi \Leftrightarrow M \models \phi$. By construction of *Prod*, the submarkings M^{obs} are controlled by the automaton net. If we have a cycle containing some accepting state in γ, then the w-word $M^{obs}_{i0}M^{obs}_{i1}M^{obs}_{i2}...$ satisfies $\neg\phi$. This means that the sequence of steps $M_{i0}M_{i1}M_{i2}...$ also satisfies $\neg\phi$.

Otherwise, if the system net $\mathbf{N}$ doesn't satisfy ϕ, then there exists a sequence of steps $M_{i0}M_{i1}M_{i2}...$ satisfying $\neg\phi$. By construction of the automaton net, the accepting w-word $M^{obs}_{i0}M^{obs}_{i1}M^{obs}_{i2}...$ satisfying $\neg\phi$ corresponds to some sequence of steps of the automaton net. While we consider only n-safe and S-finite CPN, the infinite sequence of steps must contain at least one cycle. While the considered sequence of steps of the automaton net contains infinitely many accepting states, there must be an accepting place in the cycle. □

As it was noticed in [22], strictly sequential behaviour of the obtained product ruins the benefits of any partial order representation. It was proposed to restrict the considered properties to *stutter-invariant properties*. In this case it is sufficient to observe only all the visible transitions. In this paper we also restrict ourselves to stutter-invariant properties. The purpose of this article is to show the possibility of applying the model-checking procedure to coloured Petri nets.

In [16] it has been shown that stutter-invariant properties are expressed by the "next free" fragment of LTL. This is LTL-logic without the next step operator X. In this fragment it is impossible to distinguish between the word $\xi = x_0x_1...$ and the word ξ' similar to ξ except that any of x_i can repeat finitely often. Such two words will be called *stutter-equivalent*.

Lemma 1. *([16]) If* ϕ *is a next-free formula and* ξ *and* ξ' *are stutter-equivalent w-words, then* $\xi \models \phi \approx \xi' \models \phi$.

In the construction of the product, only the visible transitions of the system, i. e., the transitions which have a common arc with an observable place, are synchronized with the transitions of the automaton net. As noticed in [22],

in this construction it is possible that some run ξ satisfies $\neg\phi$ but is not accepted. This is the case when finitely many visible transitions occur in ξ.

However, as also in [22], we can prove the following theorem, which allows us to apply the model-checking procedure to coloured Petri nets.

Proposition 3. *If γ is a sequence of steps of $Prod$ containing infinitely many visible transitions, then the projection $Proj(\gamma)$ of γ satisfies $\neg\phi$ iff γ is accepting.*

Proof. While we have infinitely many visible transitions, we can apply the same considerations, as in the case of the fully synchronized $Prod$. □

In this paper, we apply the model-checking approach from the paper [22] to CPN. We apply the 2-phase model-checking algorithm to unfoldings of CPN.

Phase 1.

Having the product of a given CPN and the automaton CPN $A_{\neg\phi}$, we build a graph T as described above. Additionally some of the edges will be labelled by b. $e \rightarrow^b e'$ means that, on the partial computation from $[e]$ to $[e']$, some transition from the automaton $A_{\neg\phi}$ occurs. The edges of the graph can be labelled by either a or b or a, b.

Now we can apply any algorithm of finding the maximal strongly connected components (scc). If we have at least one scc containing a-labelled transition, then we have an accepting run and can construct the sequence of steps satisfying $\neg\phi$.

Phase 2.

We delete each scc containing a- or b-labelled edges. In the remaining cycles for each cutoff e we can obtain the last reached automaton state $q_e = Mark([e]) \cap Q$ and the last reached observable submarking $P_e = Mark([e]) \cap Obs(\phi)$. Now, for each such cutoff e, we can, using only the automaton net, define if the marking P_e allows an accepting cycle at state q_e. If such a cycle is found, then we can reconstruct the sequence of steps satisfying $\neg\phi$.

6. Model checking of timed CPN

6.1. Unfolding of interval-timed CPN

In this section, we use the technique of unfolding of ITCPN described above. We show the possibility of applying the above-described model-checking approach to ITCPN. We only require from a CPN to be finite, n-safe and S-finite.

Definition 9. An *interval-timed CPN (ITCPN)* is a pair $N_{IT} = (N, \chi)$, where N is a CPN and χ is a transition inscription $\chi : T \to t = Nat \times Nat$, where Nat is the set of natural numbers (t consists of nonnegative integer intervals).

For $\chi(t) = (eft(t), lft(t))$, $eft(t)$ and $lft(t)$ are called the earliest firing time and the latest firing time of t, respectively.

Definition 10. A *state* of an interval-timed CPN N is a pair (M, I), where M is a marking of N and I is a clock vector $I : T \to (N \cup \{\$\})$, such that either $I(t) = \$$ or $I(t) < lft(t)$ for all $t \in T$. The symbol \$ indicates that the corresponding transition is not enabled. A state is called *consistent*, if for all $t \in T$ $I(t) \neq \$ \Longleftrightarrow t \in Enabled(M)$. Only the consistent states will be considered in this paper. For an integer $q > 0$ and for all $t \in T$, $(I + q)$ is defined by $(I + q)(t) = I(t) + q$ if $t \in Enabled(M)$ and \$ otherwise.

Definition 11. The *initial state* (M_0, I_0) is defined by the initial marking M_0 and the initial clock vector I_0, so that $I_0 = 0$ if $t \in Enabled(M_0)$ and \$ otherwise.

Definition 12. Two types of *events* are considered:

1. *Tic-event*: tic is firable at the state (M, I), if for all $t \in T$ $I(t) < lft(t)$. In this case, the successor state (M_1, I_1) is given by $M_1 = M$ and $I_1 = (I + 1)$. The tick-event is denoted by $(M, I) \to^{tic} (M_1, I_1)$.
2. *Occur-event*: An occur event is fireable at the state (M, I), if some transition t may occur with the binding element b, i. e., if $(t, b) \in Enabled(M)$ and $eft(t) \leq I(t) \leq lft(t)$. In this case, the successor state (M_1, I_1) is given by $M_1(p) = M(p) - E(p, t)\langle b \rangle + E(t, p)\langle b \rangle$ and $I_1(t')$ is

$$\begin{array}{ll} \$, & \text{if } t \in Enabled(M_1), \\ 0, & \text{if } t' = t \text{ and } t' \in Enabled(M_1), \\ 0, & \text{if } t' \neq t \text{ and } t' \in Enabled(M_1) \text{ and } t' \in Enabled(M'), \\ & \qquad \text{where } M'(p) = M(p) - E(p, t) < b >, \\ I(t') & \text{otherwise.} \end{array}$$

An occur event is denoted by $(M, I) \to^{(t,b)} (M_1, I_1)$.

Let us notice that the initial state is consistent and both occur- and tic-events preserve the consistency property. Now we define the time expansion of an interval-timed CPN — X(ITCPN) which captures the behaviour of the initial ITCPN and is an ordinary (untimed) CPN. In general, the size of X(ITCPN) may be exponential in the size of the initial ITCPN, but the unfolding of ITCPN can be generated without constructing X(ITCPN). However, we need the definition of a time expansion of an interval-timed CPN to prove existence of ITCPN's unfolding.

Definition 13. A *time expansion* of an interval-timed CPN $N_IT = (N, \chi)$ is defined in the following way.

1. For every place $p \in P$, a place p^c (complementary place) is introduced so that $C(p) = C(p^c)$. The set of all complementary places is denoted by P^c.
2. For each transition $t \in T$, a new place p^t is introduced so that $C(p^t) = int$ with $-1...lft(t)$, where the symbol \$ is denoted by -1. The set of such places is denoted by P^t.
3. The marking $PL(I)$ is defined in the following way: $PL(I)(p) = I(t)$, if $p = p^t$, and empty otherwise. The state (M, I) of the initial ITCPN is represented by the state $X(M, I) = M \cup M^c \cup PL(I)$, where for all $p^c \in P^c$ $M^c(p^c) = n'C(p) \backslash M(p)$ and n is the constant from the n-safety condition of the initial CPN.
4. For each marking M of ITCPN, a new transition $tic(M)$ is introduced so that the preset and postset of tic(M) are the set $M \cup P^c \cup P^t$. (This means that ${}^\bullet tic(M) \cap P = \{p | M(p) \neq empty\}$. It is denoted by ${}^\bullet tic(M) \cap P = M$). The arc expressions are:

   $\forall p(M(p) \neq empty) : N(a) = (p, tic(M, I)) \Rightarrow E(a) = M(p)$;

   $\forall p^c \in P^c N(a) = (p^c, tic(M, I)) \Rightarrow E(a) = n'C(p) \backslash M(p)$;

   $\forall p^t \in P^t N(a) = (p^t, tic(M, I)) \Rightarrow E(a) = i_t$, if $t \in Enabled(M)$, and empty otherwise;

   $\forall p(M(p) \neq empty) : N(a) = (tic(M, I), p) \Rightarrow E(a) = M(p)$;

   $\forall p^c \in P^c N(a) = (tic(M, I), p^c) \Rightarrow E(a) = n'C(p) \backslash M(p)$;

   $\forall p^t \in P^t N(a) = (p^t, tic(M, I)) \Rightarrow E(a) = i_t + 1$, if $t \in Enabled(M)$, and empty otherwise;

   $guard[tic(M)] = \forall i_t (i_t < lft(t))$.

   The set of these transitions is denoted by Tic.
5. For each marking M and each $(t, b) \in BE$ we define a transition $T_{(t,b),M}$. The arcs are described below.

   For all $p \in P$, if $a \in A_{IT} | N_{IT}(a) = (p, t)$, we define $a^p, a^{pc} \in A$ such that $N(a^p) = (p, T_{(t,b),M})$, $N(a^{pc}) = (T_{(t,b),M}, p^c)$;

   $E(a^p) = E_{IT}(a) < b >$, $E(a^{pc}) = E_{IT}(a) < b >$.

   For all $p \in P$, if $a \in A_{IT} | N_{IT}(a) = (t, p)$, we define $a^p, a^{pc} \in A$ such that $N(a^p) = (T_{(t,b),M}, p)$, $N(a^{pc}) = (p^c, T_{(t,b),M})$;

   $E(a^p) = E_{IT}(a)\langle b \rangle$, $E(a^{pc}) = E_{IT}(a)\langle b \rangle$.

   For all $t' \in T_{IT}$ we define $a_{1,t'}, a_{2,t'} \in A$ such that

$$
\begin{aligned}
N(a_{1,t'}) &= (p^{t'}, T_{(t,b),M});\\
N(a_{2,t}) &= (T_{(t,b),M}, p^{t'});\\
E(a_{1,t'}) &= i_{t'};\\
E(a_{2,t'}) &= \begin{cases}
-1, & \text{if } t'\in Enabled(M_1),\\
 & \quad \text{where } M_1(p) = M(p) - E(p,t)\langle b\rangle + E(t,p)\langle b\rangle,\\
0, & \text{if } t' = t \text{ and } t'\in Enabled(M_1),\\
0, & \text{if } t'\neq t \text{ and } t'\in Enabled(M_1) \text{ and } t'\in Enabled(M'),\\
 & \quad \text{where } M'(p) = M(p) - E(p,t)\langle b\rangle,\\
i_{1,t'}, & \text{otherwise.}
\end{cases}
\end{aligned}
$$

The set of these transitions is denoted by *Fire*.

The whole CPN so constructed is

$$N_{X(ITCPN)} = S_X, P_X, T_X, A_X, N_X, C_X, G_X, E_X, I_X),$$

where

$S_X = S_{IT} \cup C(P^t)$,
$P_X = P \cup P^c \cup P^t$,
$T_X = Tic \cup Fire$,
$C_X(p) = C_X(p^c) = C(p)$,
$C_X(p^t) = int$ with $-1..lft(t)$,
the sets A_X,N_X,G_X,E_X are described in the definition, the initial marking $M_{X_0} = M_0 \cup M_0^c \cup PL(I_0)$.

Now let us write some comments to each of these five points.

1. As shown in [1], we have some problems when modelling the clock events during the time expansion construction. First, if we introduce a tic transition for each state (M, I) when tic is possible, we can come to a situation when, instead of this tic transition, the tic transition for (M', I') fires, where $M' \subset M$. This is the reason for introducing the complementary places.

2. For every transition t we introduce the place p^t, where the clock position for t will be stored.

3. In this point, we define a marking $X(M, I)$ of a time expansion using complementary and clock places.

4. These transitions model the time-events. The arc expressions in the definition could be written using the variables whose evaluations could be moved to the guard functions. Such a definition would be more in the style of a CPN description in [9, 10]. However, we leave the arc functions as they are to make the definition more observable. Let us notice that we also could make the set of tic transitions based on the subsets $T' \subseteq T$ (tic(T')) instead

of basing them on the set of markings. In this case, the descriptions of M will be transferred to the guard functions.

5. These transitions model the occur-events and additionally update clock vectors. As shown in [1], the clock updating is needed to model firing of transitions. Since we represent the clock by the unique place for each transition, we need not the set of transitions to be parameterised by the clock positions. We do not require our CPN to satisfy DT-property as in [1]. This means that we have to store in some way the "intermediate" markings. This is needed when some place p loses its token at the time t and at the same time some token arrives at p. We elaborate such an "intermediate" marking in $E(a_{1,t'})$ and $E(a_{2,t'})$.

From the definition we conclude that X(ITCPN)'s unfolding exists. Since the time expansion is finite, n-safe and S-finite, we obtain, by Theorem 2, finiteness, safety and completeness of the generated unfolding. We can also consider a part of the unfolding of X(ITCPN) to be an unfolding of initial ITCPN (see below). The adequacy of this approach is shown by the theorem below. Let us first give the following definition.

Definition 14. A marking M of X(ITCPN) is called *consistent* iff

1. $|M(p^t)| = 1\ \forall t{\in}T_{IT}$,
2. $M(p){\cup}M(p^c) = n'C(p)$,
3. $M(p^t) = -1{\Leftrightarrow}t{\in}Enabled(M{\cap}P)\ \forall t{\in}T$.

Let us notice that the initial state is consistent and, by the time expansion definition, any state reached from the consistent state is consistent.

Theorem 4. *Let $N_{IT} = (N, \chi)$ be an ITCPN and its time expansion N_X be constructed. Then we have the following:*

1. *A tic-event can occur at (M,I) and $(M,I){\rightarrow}^{tic}(M,I') \Leftrightarrow tic{\in}Tic$ is enabled in $M{\cup}M^c{\cup}PL(I)$ and $M{\cup}M^c{\cup}PL(I){\rightarrow}^{tic}M{\cup}M^c{\cup}PL(I')$.*
2. *(t,b) can occur at (M,I) and $(M,I){\rightarrow}^{(t,b)}(M,I') \Leftrightarrow T{\in}Fire$ is enabled in $M{\cup}M^c{\cup}PL(I)$ and $M{\cup}M^c{\cup}PL(I){\rightarrow}^{T}M'{\cup}M'^c{\cup}PL(I')$.*
3. *The (consistent) state (M,I) is reachable in $N_{IT} \Leftrightarrow$ the (consistent) state $M{\cup}M^c{\cup}PL(I)$ is reachable in N_X. In particular, M is reachable in $N_{IT} \Leftrightarrow M = M'{\cap}P$ for some reachable marking M' of N_{IT}.*

As mentioned earlier, the time expansion of a CPN is used only to prove the existence of a finite, safe and complete unfolding of ITCPN. Below we give the definition of a reduced unfolding which is obtained from the unfolding of X(ITCPN) by removing the parts with unnecessary information and can

be constructed directly from the ITCPN. Although the exact algorithm description is out of the scope of the paper, the basic idea of how to construct a reduced unfolding directly from ITCPN will be given.

Definition 15. Let N be an ITCPN and the finite unfolding $Unf(N_X)$ of its time expansion be constructed. Then a *reduced unfolding* is obtained from $Unf(N_X)$ in the next two steps:

1. remove all the places p^c and p^t from the unfolding and all the incidental arcs;
2. add the names (t,b) and tic to $T_{(t,b),M}$ and tic(M), respectively.

The reduced unfolding is denoted by $R(Unf(N_X))$.

The configuration $C = (t_1...t_n)$ of $Unf(N_X)$ has a corresponding configuration $C' = (t_{1'}...t_{n'})$ of $R(Unf(N_X))$ such that if $t_i = T_{M,(t,b)}$, then $t_{i'} = (t, b)$, and if $t_i = tic(M)$, then $t_{i'} = tic$ and vice versa.

It also follows from the reduced unfolding definition that

$$Mark_{Unf(N_x)}(C) \cap P = Mark_{R(Unf(N_x))}(C').$$

6.2. Model checking of ITCPN

Semantics of LTL can be defined on interval-timed coloured Petri nets. Let us remind that we consider only n-safe CPN. As it was made for standard CPN, the set of atomic propositions for ITCPN is identified with the set $n \cdot TE = TE \times TE \times ... \times TE$, where n is the number from the n-safeness condition and TE is the set of token elements. A proposition $m'(p, c)$ holds at the state M iff $m'(p, c) \in M(p)$.

The sequence of steps $(M_0, I_0) \rightarrow^{T_0} (M_1, I_1) \rightarrow^{T_1} (M_2, I_2)...$, where each T_i is either an occur-event given by the binding element $[t_i, b_i\rangle$ or a tic-event, satisfies ϕ iff the w-word $M_0M_1M_2...$ belongs to L_ϕ. We say that the state (M, I) satisfies ϕ if any sequence of steps $(M, I) \rightarrow^{T} (M_1, I_1) \rightarrow^{T_1} (M_2, I_2)...$ satisfies ϕ. The ITCPN N_{IT} satisfies ϕ if the state (M_0, I_0) satisfies ϕ. We write this as $N_{IT} \models \phi$.

The time expansion X(ITCPN) has all the necessary information about the future behaviour of ITCPN. The following results give us the possibility of applying the model-checking procedure to interval-timed coloured Petri nets. Although this is out of the scope of this paper, it is also possible to apply the same approach to TCPN [9, 10]. The unfolding of TCPN was described in [14, 15].

Theorem 5. *Let N_{IT} be an interval-timed CPN, and ϕ be an LTL-formula over N_{IT}. Then $N_{IT} \models_t \phi \iff X(N_{IT}) \models \phi$.*

Proof. According to the theorem describing a behavior of the time expansion, for each sequence of steps $(M_0, I_0) \rightarrow^{T_0} (M_1, I_1) \rightarrow^{T_1} (M_2, I_2)...$ of ITCPN there exists a corresponding sequence of steps

$$X(M_0, T_0) T_{0X} X(M_1, T_1) T_{1X} X(M_2, T_2)...$$

of its time expansion, where T_{0X} is either $T_{M,(t,b)}$ or $tic(M)$.

The formula ϕ is true on the net N_{IT} iff for each sequence of steps

$$(M_0, I_0) \rightarrow^{T_0} (M_1, I_1) \rightarrow^{T_1} (M_2, I_2)...$$

of N_{IT} the w-word $M_0 M_1 M_2...$ belongs to L_ϕ. From the construction of a time expansion, it follows that the projections of the state $X(M, I)$ and the marking M on the set of observable places are equal.

From the fact that for each sequence of steps of ITCPN there exists the corresponding sequence of steps of X(ITCPN), $N_{IT} \models_t \phi \iff X(N_{IT}) \models \phi$ follows. □

As described above, we consider only a part of the unfolding of time expansion to be an unfolding of the initial ITCPN. This part of the unfolding was called a reduced unfolding and was constructed from the unfolding of the time expansion of ITCPN by removing all the unnecessary information. The following proposition gives us the possibility of applying the LTL model-checking procedure to the unfolding of ITCPN.

Proposition 4. *Let N_{IT} be an interval-timed CPN, ϕ be an LTL-formula over N_{IT}, and $\gamma = M_0[t_0, b_0\rangle M_1[t_1, b_1\rangle...$ be the sequence of steps of $X(N_{IT})$. Then $\gamma \models \phi$ iff $R(\gamma) \models \phi$.*

Proof. The first step of constructing the reduced unfolding from an unfolding of X(ITCPN) consists in deleting the places p^t and places p^c from the unfolding of X(ITCPN). By construction of the time expansion, the situation of having the transition t in the unfolding with pre- or postset consisting only of the places p^t or p^c is not possible. While the places p^t and p^c do not belong to $\langle\phi\rangle$, we obtain that after this step $\gamma \models \phi \iff R(\gamma) \models \phi$.

The second step of constructing the reduced unfolding from the unfolding of X(ITCPN) consists in renaming the transitions and has not any influence on whether the sequence of steps satisfies ϕ or not. □

7. Conclusion

In this paper, the model-checking technique based on net unfolding proposed in [22] is applied to coloured Petri nets as they are described in [9, 10]. The

technique is formally transferred; LTL semantics for CPN and construction of the product of two CPN are considered. We require a CPN to be finite, n-safe and to contain only finite sets of colours.

The unfolding is a finite prefix of the maximal branching process. The size of unfolding is often much smaller than the size of the reachability graph of a CPN. The use of an unfolding for the model-checking procedure instead of the whole reachability graph is a step forward in fighting with a famous state explosion problem.

The papers [14, 15] give us the possibility to construct the unfolding of interval-timed CPN. In this paper, it is also shown that we can apply the model-checking procedure to the interval-timed CPN satisfying the above mentioned requirements of finiteness, n-safeness and finiteness of the considered colour sets.

Acknowledgements. I would like to thank Valery Nepomniaschy for drawing my attention to this problem and for valuable remarks.

References

[1] Bieber B., Fleischhack H. Model checking of time Petri nets based on partial order semantics // Lect. Notes Comput. Sci. — 1999. — Vol. 1664. — P. 210–225.

[2] Couvreur J.-M., Grivet S., Poitrenaud D. Designing an LTL model-checker based on unfolding graphs // Lect. Notes Comput. Sci. — 2000. — Vol. 1825. — P. 123–145.

[3] Engelfriet J. Branching processes of Petri nets // Acta Informatica. — 1991. — Vol. 28. — P. 575–591.

[4] Esparza J. Model-checking using net unfoldings // Lect. Notes Comput. Sci. — 1993. — Vol. 668. — P. 613–628.

[5] Esparza J., Heljanko K. A new unfolding approach to LTL model-checking // Lect. Notes Comput. Sci. — 2000. — Vol. 1853. — P. 475–486.

[6] Esparza J., Heljanko K. Implementing LTL model checking with net unfoldings // Lect. Notes Comput. Sci. — 2001. — Vol. 2057. — P. 37–56.

[7] Esparza J., Romer S., Vogler W. An improvement of McMillan's unfolding algorithm // Lect. Notes Comput. Sci. — 1996. — Vol. 1055. — P. 87–106.

[8] Gerth R., Peled D., Vardi M., Wolper P. Simple on-the-fly automatic verification of linear temporal logic // Proc. Intern. Conf. on Protocol Specification, Testing and Verification, PSTV'95, Warsaw, Poland, 1995. — P. 3–18.

[9] Jensen K. Coloured Petri Nets. — Berlin a.o.: Springer, 1995. — Vol. 1.

[10] Jensen K. Coloured Petri Nets. — Berlin a.o.: Springer, 1995. — Vol. 2.

[11] Kondratyev A., Kishinevsky M., Taubin A., Ten S. A Structural approach for the analysis of Petri nets by reduced unfoldings // Proc. 17th Intern. Conf. on Application and Theory of Petri Nets, Osaka, June 1996. — P. 346–365.

[12] Kozura V. E. Unfoldings of Coloured Petri Nets. — Novosibirsk, 2000. — (Prepr. / SB RAS. IIS; № 80).

[13] Kozura V. E. Unfoldings of coloured Petri nets // Lect. Notes Comput. Sci. — 2001. — Vol. 2244. — P. 270–280.

[14] Kozura V. E. Unfoldings of timed coloured Petri nets. — Novosibirsk, 2001. — (Prepr. / SB RAS. IIS; № 82).

[15] Kozura V. E. Unfoldings of timed coloured Petri nets // Proc. CS&P'2001 Workshop, Warsaw, 3–5 October 2001. — P. 128–139.

[16] Lamport L. What good is temporal logic? // Information Processing '83. — 1983. — P. 657–668.

[17] Melzer S., Romer S. Deadlock checking using net unfoldings // Proc. Conf. on Computer Aided Verification, CAV'97, Haifa, 1997. — P. 352–363.

[18] McMillan K. L. Using unfolding to avoid the state explosion problem in the verification of asynchronous circuits // Lect. Notes Comput. Sci. — 1992. — Vol. 663. — P. 164–174.

[19] Valmari A. Stubborn sets of coloured Petri nets // Proc. 12th Intern. Conf. On Application and Theory of Petri Nets, Gjern, 1991. — P. 102–121.

[20] Valmari A. The state explosion problem // Lect. Notes Comput. Sci. — 1998. — Vol. 1491. — P. 429–528.

[21] Vardy M. Y., Wolper P. An automata-theoretic approach to automatic program verification // Proc. 1st IEEE Symp. on Logic in Computer Science, Cambridge, Mass., 1986. — P. 322–331.

[22] Wallner F. Model-checking LTL using net unfoldings // Lect. Notes Comput. Sci. — 1998. — Vol. 939. — P. 207–218.