Available online www.ejaet.com

European Journal of Advances in Engineering and Technology, 2019, 6(3):1-11

Research Article **ISSN: 2394 - 658X**

Some Questions about the Possibilities of Modeling Petri Nets and their Extension

Goharik, Petrosyan

International Scientific-Educational Center of NAS RA, Armenian State Pedagogical University after Khachatur Abovyan, Yerevan, Armenia,
Email: petrosyan_gohar@list.ru, petrosyangoharik72@gmail.com

ABSTRACT
This paper is dedicated to several structure features of Petri Nets. There is detailed description of appropriate access in Petri Nets and reachable tree mechanism construction. There is algorithm description of the minimal sequence of possible transitions. The designed algorithm gets the shortest possible sequence for the net advance state, which brings the mentioned net state into covering state. There is theorem, which states that through the describing algorithm, the number of transitions in covering state is in minimal.
This chapter studies the interrelation of languages of Colored Petri Nets and Traditional formal languages. The Venn graph and diagram that the author modified show the interrelation between languages of Colored Petri Nets and some Traditional languages, where the class of languages of Colored Petri Nets is supposed to include an entire class of Context-free languages and some other classes. The work is done to show the modeling of Patil synchronization problem with the support of Colored Petri Net, which was impossible to present with well known P and V operations or through Classical Petri Net, as the limitation of the mentioned mathematical properties does not allow to model the mechanisms, where the process must be synchronized with optimal allocation of resources.

Key words: Petri Nets, Colored Petri Nets, Traditional Languages, Transition, Position

1. INTRODUCTION

Modeling and designing systems can’t be imagined without the use of computer technology. When creating automated systems and designing them, the problem of choosing a formal model for representing systems first arises. From the model through the algorithmic to the software - this is the way of modern modeling and system design. When considering physical systems with lumped, a convenient model is a linear graph, each vertex of which corresponds to a functional or constructive component, and an arc to a causal relationship.

In modern society, reliable transmission and protection of information are of wide use and are topical tasks. One of the most important applied fields is the theory of Petri Nets [1,2]. The main task of Petri Nets is the modeling of realistic systems from the point of view of optimization. Systematic study of the properties of Petri Nets and the possibility of using them for solving applied problems, mainly problems related to models and means of parallel processing of information.

The following Issues can serve as examples of those problems that often arise in the design and study of discrete systems:

- Does the system perform the functions for which it is intended?
- Does it function effectively?
- Can mistakes and emergencies occur in it?
- Does it have potential bottlenecks?
- Is it possible to simplify the system or replace its individual components and subsystems with more perfect ones, without disturbing its overall functioning?
- Is it possible to design more complex systems that meet the specified requirements from these systems, etc.?

These tasks are basically "qualitative" not quantitative.

The goal of in-depth study of various extensions of Petri nets (from the point of view of optimization) for modeling real-time systems, brings to the design of such technical equipments where one has to minimize resource costs, time and maximize speed.

Colored Petri Nets (CPN) is a graphical oriented language for design, specification, simulation and verification of systems [3-6, 8]. It is in particular well-suited for systems that consist of a number of processes which communicate and synchronize. Typical examples of application areas are communication protocols, distributed systems, automated production systems, work flow analysis and VLSI chips.
The CPN language allows the model to be represented as a set of modules, allowing complex nets (and systems) to be represented in a hierarchical manner.
In the classical or traditional Petri Net tokens do not differ from each other, we can say that they are colorless. Unlike standard Petri Nets in Colored Petri Net of a position can contain tokens of arbitrary complexity, for example, lists, etc., that enables modeling be more reliable.

2. THE ALGORITHM DESCRIPTION OF THE SHORTEST POSSIBLE SEQUENCE OF TRANSITIONS IN PETRI NETS

Construction of discrete systems models need system components, with its operations in the abstract, such as, the program operator action, trigger transition from one state to another, interruptions in the operating system, machine or conveyer action, project phase completion etc. In general, the same system can operate differently in different conditions, bringing a multitude of processes, which means operating not deterministically. The real system operates in certain time, cases occur in certain periods and last for certain time. In synchronic models of discrete systems, the events are clearly associated with certain moments or pauses, during which all the components make simultaneous change in the system state, which is interpreted as a change in the system state. State conditions change successively. Alongside, these large systems, modeling approach has several drawbacks.

- First of all, the system must take into account all the components of its overall condition of each change, so that model appears formidable.
- Second, in such an approach, information are disappeared between casual links in systems.
- Thirdly, the so-called asynchronous systems may occur uncertain events at intervals of time.

The above-mentioned types of models, including Petri Nets are called asynchronous.
Link replacements in time, with causal relationships, give chance to more clearly describe the structural features of the system.
Asynchronous models of non-formal description of the case, in particular, Petri Nets, must involve relationships of time (early, late, not at the same time, etc.), when it is convenient or accepted, but they represent a causal relationship. Great interplay of asynchronous systems, typically have a complex dynamic structure.
The relationship between the two, will be described more clearly if not immediate contacts are marked, or cases and situations in which the case can be realized. In this case, the conditions of implementation of the system of global situations are formed in the named local operations.
The terms has its capacity. The term is not fulfilled (capacity is equal to 0), the term is fulfilled (capacity is equal to 1), the term is fulfilled in n times (capacity is equal to n, where n- is a positive integer).
The condition is consistent with the existence of situations such as the operation of the system modeled data: any registry of computer equipment, parts availability on line.
Terms defined combinations allow to implement any of the cases (cases precondition), and the implementation of changes create certain conditions (cases post condition), which means the cases co-incident with the terms and conditions of the case.
Therefore, it is natural that many systems are suitable as discrete structures, consisting of two elements: the type of events and terms. The cases and terms in Petri Nets are disjoint sets with each other, respectively, called transitions and positions sets. Transitions are depicted in a graphical representation of Petri Nets (vertical lines), and places, with circles. [1,2].

2.1 Petri Nets, Reachable States, Reachable Trees

Definition 1: Petri Nets are in $M(C,\mu)$pair, where $C=(P,T,I,O)$ is the net structure and μ is the net condition. In structure C of P-positions, T-transitions are finite sets. $I: T \to P^{\infty}, O: T \to P^{\infty}$ are input and output functions, respectively, where P^{∞} are all possible collections (repetitive elements) of P. $\mu: P \to N_0$ is the function of condition, where $N_0 = \{0,1,\cdots\}$ is the set of integers and included 0.
Now, we will define a function that determines the number of elements in their entering numbers in the collection. X element enters into collection of B, which we will appoint as: $\#(X,B)$, (called: X number in B). If we limit the number of elements in the collection so that $0 \leq \#(X,B) \leq 1$, then we will reach the idea of the set. Since $\#(X,B)$ function determines the X element entering collection of B, it follows that $\#(X,B) \geq 0$, the grouping of all the X and B. X element of the B collection, if $\#(X,B) > 0$, i.e. $X \in B$, identically, if $\#(X,B) = 0$, then $X \notin B$.
Let's set empty collection of $\emptyset$, which has members (i.e. all $X : \#(X,B) = 0$). $|B|$ is the capacity of the entire number of elements entering B collection:

$$|B| = \sum_X \#(X, B).$$

Saying net state, we will understand the following:

$$(\mu(P_1), \mu(P_2), \ldots, \mu(P_n)),\ n = |P|,\ P = \{P_1, \ldots, P_n\}$$

Suppose we have $M = (C, \mu)$.

We will say that in μ state $t_j \in T$ transition is allowed to implement if for $\forall P_i \in I(t_j)$ there is:

$$\mu(P_i) \geq \#(P_i, I(t_j)).$$

Suppose in μ state t_j transition is allowed to implement and it is actually acted. In this case the net will appear in its new state, μ', which is solved in the following way:

$$\forall P_i \in P, \mu'(P_i) = \mu(P_i) - \#(P_i, I(t_j)) +$$
$$+ \#(P_i, O(t_j))$$

Let's name $R(C, \mu_0)$ as the reachable state set:

1. $\mu_0 \in R(C, \mu_0)$,
2. If $\mu' \in R(C, \mu_0)$ and $\exists t_j \in T$ has transition in the way that $\delta(\mu', t_j) = \mu''$, then $\mu'' \in R(C, \mu_0)$.
3. Other states don't belong to $R(C, \mu_0)$. The $R(C, \mu_0)$ can be infinite.

μ'' marking covers μ' marking if

$$\mu'' \geq \mu'.$$

The meaning of coverage problem is for μ' decide whether it is reachable to $\mu'' \geq \mu'$. The coverage problem can be solved through the reachable tree. At first for μ we will build the reachable tree. Then we will search x peak in the way that $\mu[x] \geq \mu'$. If there is no such a peak, then μ' marking isn't covered with any reachable marking, if it is located in $\mu[x]$ and gives a reachable marking which covers μ' [11-13].

Let's build Petri Nets reachable tree in Figure 1. The natural state of this net is (1101), which shows the presence of tokens in the net at that moment. The tokens that are in Figure 1 with little dots, correspond with the presence of resources in the net. The state of the net is due to the move of the tokens.

Let's correspond states in the edges of the peaks, and transitions in the sides. The root is corresponded with the first state of the net.

Figure 1 is corresponded with Figure 2, in which the reachable tree is infinite. Let's put limitations, for the tree to be finite. If any peak is locked, then we will name it as terminal. If there is a state in any peak and there is another peak in the tree with the same state which is already developed, then we will name the new peak as repeated and will not develop it.

If there is /*/ type way in the tree, then the way through the second peak can be repeated and the states will grow.

Let's introduce the idea of infinite much as ω: $\omega \geq \omega, \omega + a = \omega,\ \omega - a = \omega$, where $a = const$: For example, instead of (5,⋯), we will write (ω, ⋯). In this case the tree will become as finite, and we will have loss of information. [1]

Let's give several definitions, which will be used in entire work.

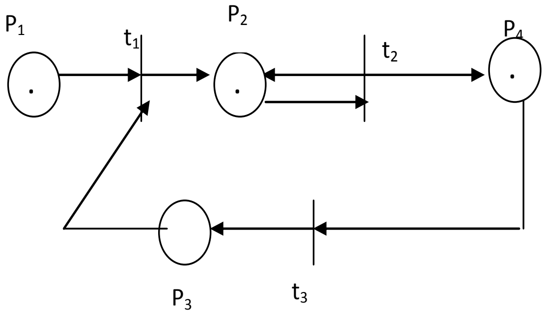
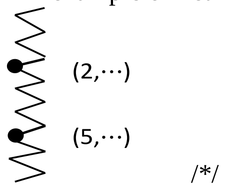


Fig. 1 An example of Petri Nets



A way in the tree.

Definition 2. The peak is called as boundary if it is a subject of processing.
Definition 3. The peak is called as terminal if it doesn't content a sub tree.
Definition 4. The peak is called internal, if it is already processed.
Definition 5. The Boundary Peak is repeated if there is an internal peak with the same state.
Let's describe the structure of the algorithm of the reachable tree.

Suppose x peak is the next Boundary Peak. The state relating to it, will be marked as $\mu[x]$. Let's mark with $\mu[x]_i$ the i – state condition vector.

1. If x is terminal or repeated peak, then when processing, it will become as second peak and go to the second step.
2. For $\forall t_j$ transition $t_j \in T$, so that $\delta(\mu[x], t_j)$ is solved, then do the following: Add the tree a new side coming from x and mark the side as t_j and name the peak as z. The $\mu[x]$ will be solved with the following way.

If from the tree root, on its way to bring z, there is y peak, that

$$\mu[y] \le \delta(\mu[x], t_j)\ \&\ \mu[y]_i < \delta(\mu[x], t_j),$$

then $\mu[z]_i = \omega$. If $\mu[x]_k = \omega$, then $\mu[z]_k = \omega$. In the opposite case:
$\mu[z]_r = \delta(\mu[x], t_j)_r$. The second step is repeated for t_j.

3. If the number of the peaks are more, then the algorithm finishes its work.

With the help of this algorithm, we will build (as in Figure 1) the Petri Net reachable tree (see Figure 3).

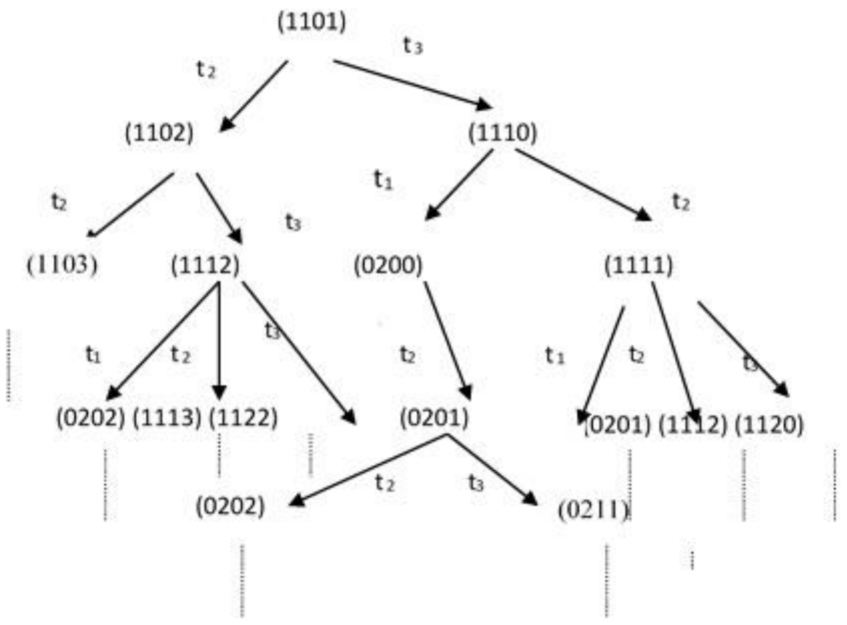


Fig. 2 The Petri Net Reachable Infinite Tree.

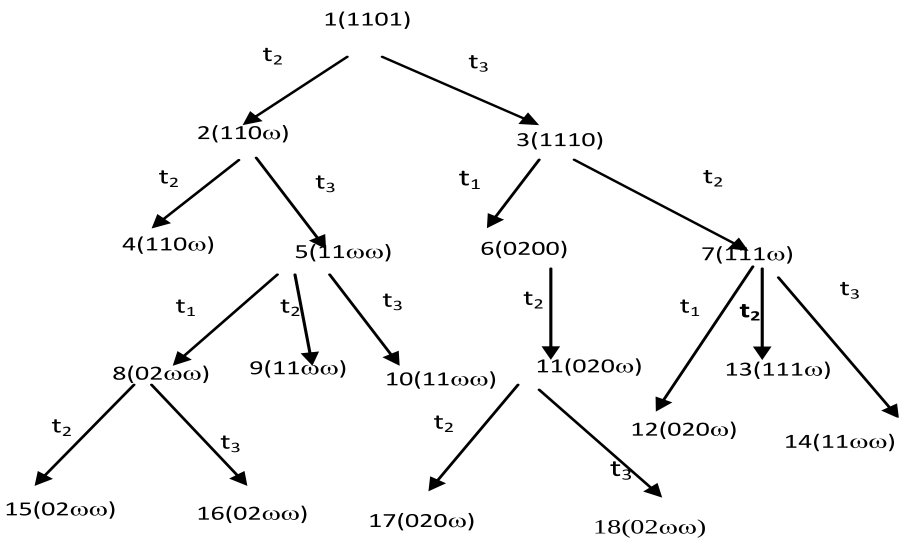


Fig. 3 The Petri Net Reachable Tree

2.2 Description of the Algorithm for Finding the Minimum Number of Transitions in the State of Coverage
Suppose we have Petri Nets in Figure 1, and the corresponding TT (see Figure 3) reachable tree.
Let's mark as P , and the set of states in Petri Nets as $T*$ from TT root till y, transition succession with, G the succession of the peaks in $T*$.
Suppose we have $\mu[x]=(0,1,15,13)$ state. Let's find a y peak in the reachable tree that $\mu[y]\geq\mu[x]$.
Suppose such peaks are $y_1,\ldots,y_m$. Let's choose one peak among the peaks on which we will use the algorithm.
For every y_i peak, we profile $\mu[y_i]$.

Suppose in $\mu[y_i]$ there is ω in $\mu[y_i]_{i1},\ldots,\mu[y_i]_{ik}$. For each $\mu[y_i]$ we count $S=\sum_{j=i_1}^{i_k} z_j(t)$, where

$$z_j(t)=\#(P_j, I(t_k)), \forall t_k\in T*:$$

We take the y_i for which the S is the minimum. If for any peak, these numbers are equal, then we take the y_i in which $T*$ height is the minimum.
For our example $\mu[x]$ we will cover the following peak:

y_1 $\mu[y_1]=(1,1,\omega,\omega)$, $T^*=\{t_2,t_3\}$
y_2 $\mu[y_2]=(0,2,\omega,\omega)$, $T^*=\{t_2,t_3,t_1\}$
y_3 $\mu[y_3]=(1,1,\omega,\omega)$, $T^*=\{t_2,t_3,t_2\}$
y_4 $\mu[y_4]=(1,1,\omega,\omega)$, $T^*=\{t_2,t_3,t_3\}$
y_5 $\mu[y_5]=(1,1,\omega,\omega)$, $T^*=\{t_3,t_2,t_3\}$
y_6 $\mu[y_6]=(0,2,\omega,\omega)$, $T^*=\{t_3,t_1,t_2,t_3\}$

$S(y_1)=1$	$S(y_4)=2$
$S(y_2)=2$	$S(y_5)=2$
$S(y_3)=1$	$S(y_6)=3$

We found out that in minimum number: $S(y_1)=S(y_3)$, $|T^*(y_1)|=2$, $|T^*(y_3)|=3$ $\Rightarrow$ we take the y_1 peak. After choosing the covering peak, we go to the usage of the algorithm. Suppose the y is the covering peak.
1. We take the way, which connects the tree root with y and T^* for our example t_2,t_3 let's mark $t_i^{'}=t_j, 1\leq i\leq|T^*|$, $t_j\in T^*$. In this case: $t_1^{'}=t_2$, $t_2^{'}=t_3$.
2. For each chosen transitions, the $t_i^{'}$, is corresponded with a_i numbers in the following way:

- If for $t_i^{'}$ transition $\exists 1\leq j\leq|P|$ in the way that $\delta(\mu[y^{'}], t_i^{'})_j=\omega$, $y^{'}\in G$ then $a_i=\mu[x]_j$ in which case $t_i^{'}$ transition corresponds with P_j position.
- If for the same $t_i^{'}$ transition $\exists 1\leq k\neq j\leq|P|$ in the way that $\delta(\mu[y^{'}], t_i^{'})_k=\omega$, then $a_i=\max\{\mu[x]_j,\mu[x]_k\}$.

Moreover, for $t_i^{'}$ transition we will correspond P_j and P_k positions. If instead of $t_i^{'}$ $\sigma=t_{i_1}^{'}\ldots t_{i_k}^{'}$ $(t_{i_k}^{'}=t_i^{'})$ for $\exists 1\leq j\leq|P|$ in the way that $\delta(\mu[y^{'}],\sigma)_j=\omega, y^{'}\in G$, then we will correspond a_i with σ and $a_i=\mu(x)_j$.
In this case, we will correspond σ with P_j position. In the opposite case, if there is no $t_i^{'}$ transition for $1\leq j\leq|P|$ in the way that $\delta(\mu[y^{'}], t_i^{'})_j=\omega$, then $a_i=1$, in which case there is no related position for $t_i^{'}$.
For our example:
$a_1=13$ $a_2=15$
$t_1^{'}\sim P_4$ $t_2^{'}\sim P_3$.
Now, we will define the following action for a_i:

$$a_i = \begin{cases} \dfrac{a_i - n}{m}, & \text{if } (a_i - n)\text{modm} = 0 \\ \left[\dfrac{a_i - n}{m}\right] + 1, & if\,(a_i - n)\text{modm} \neq 0 \end{cases}$$

Where n to $t_i^{'}$ or in σ corresponding P_j position, the number of tokens are in their first position, and m from $t_i^{'}$ or σ to the number of the arrows in the state: $\#\left(P_j, O\left(t_i^{'}\right)\right)$.

If for $t_i^{'}$ transition $P_1,\cdots,P_k$ positions correspond, then we will take the P_1 position for which: $\mu[x]_1 = a_i$. In this case: $n = \mu_0[P_1]$, $m = \#\left(P_j, O\left(t_i^{'}\right)\right)$.

If there is no corresponding position for $t_i^{'}$ transition, then we will leave a_i to remain the same.
For our example:

$a_1 = (a_1 - 1)/1 = (13-1)/1 = 12$
$a_2 = (a_2 - 0)/1 = (15-0)/1 = 15.$

Let's mark $b_i^1 = a_i$. For our example:

$b_1^1 = a_1 = 12$
$b_2^1 = a_2 = 15.$

Cumulative move.
We will take T^* last transition or the succession of transition, fix it and mark as t_α.
t_α corresponding b_i^1 is marked as α which we also fix. The fixed b_i^j doesn't change in the next moves.
We consider all T^* items from right to left, starting from t_α.
Suppose the $t_k^{'}$ is the considered transition or the transition succession and the P_1 is the corresponding position of $t_k^{'}$.
If $P_1 \in I(t_\alpha)$, then $t_k^{'}$ corresponding b_i^j in the next move will get the following value: $b_i^{j+1} = b_i^j + \alpha \cdot l$, where l from P_1 position t_α is the number of arrows.
Suppose $t_k^{'}$ corresponds with $P_1,\cdots,P_l$ positions. If $\exists\ 1 \le j \le l$ in the way that $P_j \in I(t_\alpha)$, then, $b_i^{j+1} = b_i^j + \alpha \cdot l$, where $l = \#\left(P_j, I(t_\alpha)\right)$. In the opposite case $b_i^{j+1} = b_i^j$.
Now we will fix t_α the previous action of transition and mark it as t_α.
The new t_α corresponding b_i^j, we will mark as α and move again to the second step. The algorithm will implement its work if T^* fixes its first item.
So, for every $t_i^{'}$ transition or transition succession, there will be a corresponding fixed b_i^j number, which will mark for $t_i^{'}$ transition or transition succession implementation number. For our example:

$t_1^{'}$	$t_2^{'}$
12	15
27	15.

We got that $t_1^{'}$ transition must be implemented for 27 times, and $t_2^{'}$, 15 times.
Returning to our appointment, we will get that $\mu[x] = (0,1,15,13)$ for covering the state $\mu[y] = (1,1,\omega,\omega)$. On the way to reach the state we need to implement t_2 transition 27 times, and the t_3 transition for 15 times.

Let's assign $t_s(y) = \sum_{i=1}^{l} b_i^j$, $l = \left|T^*(y)\right|$. Which means $t_s(y)$ is y the number of enabled transitions.

Lemma 1. Suppose there are y_1 and y_2 peaks in the way that $\mu(y_1) \ge \mu[x]\,\&\,\mu(y_2) \ge \mu[x]$. In that case $t_s(y_1) \le t_s(y_2)$.
Proof: We have:

$S(y) = \sum_{j=i_1}^{i_k} z_j(t)$, where $z_j(t) = \#\left(P_j, I(t_k)\right), \forall t_k \in T^*$.

$t_s(y) = \sum_{i=1}^{k} b_i^{'}$, where $k = \left|T^*(y_1)\right|$.

In this number some $b_i^{'}$s are equal to 1. Without breaking the sense we will suppose that the first $d^{'}$ number of $b_i^{'}$s are equal to 1. We will get:

$$t_s(y_1) = d^{'} + \sum_{i=d^{'}+1}^{k} b_i^{'} = d^{'} + \sum_{i=d^{'}+1}^{k} \left(\mu[x]_l + m^{'} \cdot b^{'}\right)$$

We have: $t_s(y_2) = \sum_{i=1}^{k^{'}} b_i^{''}$, where $k^{'} = \left|T^*(y_2)\right|$.

Suppose for $b_i^{''}$ s, number of $d^{''}$ are equal to 1. Moreover: $d^{''} \le d^{'}$, as $S(y_1) < S(y_2)$,

$$t_s(y_2) = d^{''} + \sum_{i=d^{''}+1}^{k^{'}} \left(\mu[x]_l + n_i^{''} \cdot b^{''}\right) \ge d^{'} + \sum_{i=d^{'}+1}^{k} \left(\mu[x]_l + n_i^{'} \cdot b^{'}\right) = t_s(y_1) \Rightarrow t_s(y_1) \le t_s(y_2)$$

The lemma is proved.

Lemma 2. Suppose y_1 and y_2 are covering peaks. There is $S(y_1) = S(y_2) \& \left|T_1^*\right| < \left|T_2^*\right|$: in this case $t_s(y_1) < t_s(y_2)$.

Proof:

$$t_s(y_1) = d^{'} + \sum_{i=d^{'}+1}^{k} b_i^{'} = d +$$

$$^{'} + \sum_{i=d^{'}+1}^{k} \left(\mu[x]_l + n_i^{'} \cdot b^{'}\right) \underset{d^{'} < d^{''}}{<} d^{''} +$$

$$+ \sum_{i=d^{''}+1}^{k^{'}} \left(\mu[x]_l + n_i^{''} \cdot b^{''}\right) = t_s(y_2).$$

The lemma is proved.

Theorem. Through the above mentioned number of covering state transition algorithm is in its minimal state.

Proof: Suppose y is the covering peak in our algorithm and $t_1^{'}, \cdots, t_k^{'}$ is the succession of transitions. We will show that the number of $t_1^{'}, \cdots, t_k^{'}$ move is in minimal state. For this reason, we need to show that $\nexists y'$ covering peak has less number of transitions than the number of $t_1^{'}, \cdots, t_k^{'}$.

Let's consider two cases:

1. $y \ne y'$.

Suppose the transition number of y' is less than $t_1^{'}, \cdots, t_k^{'}$ implementation number. According to the algorithm:

$S(y) < S(y')$ or

$S(y) = S(y') \& \left|T_1^*\right| < \left|T_2^*\right|$.

If $S(y) < S(y') \Rightarrow$ according to Lemma 1: $t_s(y) \le t_s(y')$. We've come into a controversy.

If $S(y) = S(y') \& \left|T_1^*\right| < \left|T_2^*\right| \Rightarrow$ according to Lemma 2: $t_s(y) \le t_s(y')$. We've come into a controversy.

2. $y = y'$.

Suppose succession transitions of y' is $s_1, \cdots, s_r$. As the tree doesn't contain cycle: $y = y' \Rightarrow t_1^{'}, \cdots, t_k^{'}$ and $s_1, \cdots, s_r$ are the same $\Rightarrow t_s(y) \le t_s(y')$. The theorem is proved.

3. INTERRELATION OF LANGUAGES OF COLORED PETRI NETS AND SOME TRADITIONAL LANGUAGES

Definition. The mathematical definition of Colored Petri Net: CPN is a nine-tuple $CPN = (\Sigma, P, T, A, N, C, G, E, I)$, where:

Σ is a finite set of non-empty types, also called color sets. In the associated CPN Tool, these are described using the language CPN-ML [9]. A token is a value belonging to a type.

P is a finite set of places. In the associated CPN Tool these are depicted as ovals/circles.

T is a finite set of transitions. In the associated CPN Tool these are depicted as rectangles.

A is a finite set of arcs. In the associated CPN Tool these are depicted as directed edges. The sets of places, transitions, and arcs are pairwise disjoint, that is

$$P \cap T = P \cap A = T \cap A = \emptyset.$$

N is a node function. It is defined from A into $P \times T \cup T \times P$. In the associated CPN Tool this depicts the source and sink of the directed edge.
C is a color function, $C : P \to \Sigma$.
G is a guard function. It is defined from T into expressions such that:
$t \in T : [Type(G(t)) = B \& Type(Var(G(t))) \subseteq \Sigma]$
E is an arc expression function. It is defined from A into expressions such that:

$$\forall a \in A : [Type(E(a)) = C(p)_{MS} \ \& \ Type(Var(E(a))) \subseteq \Sigma],$$

where p is the place of $N(A)$ and $C(p)_{MS}$ denotes the multi-set type over the base type $C(p)$.
I is an initialization function. It is defined from P into closed expressions so that:

$$\forall p \in P : [Type(I(p)) = C(p)_{MS}].$$

In the CPN Tool this is represented as initial marking next to the associated place.
The distribution of tokens, called marking, in the places of a CPN determines the state of a system being modeled. The dynamic behavior of a CPN is described in terms of the firing of transitions. The firing of a transition takes the system from one state to another. A transition is enabled if the associated arc expressions of all incoming arcs can be evaluated to a multi-set, compatible with the current tokens in their respective input places, and its guard is satisfied. An enabled transition may fire by removing tokens from input places specified by the arc expression of all the incoming arcs and depositing tokens in output places specified by the arc expressions of outgoing arcs. [3-6, 9]
One of the most studied simple class of formal languages is the class of Regular languages. It is known that any Regular language is the language of Petri Nets. [1,10] It's possible to construct Petri Net, which generates $\{a^n b^n / n > 1\}$ a Context-free language, which is not Regular. [1] Not all the languages of Petri Nets are Context-free, built a network that generates $\{a^n b^n c^n / n > 0\}$ a Context-sensitive language, which is not Context-free language. [1] Unlike Regular languages, which are the languages of Petri Nets, there are Context-free languages, which are not languages of Petri Nets. Such examples of Context-free language we are noted the following: $\{\omega\omega^R / \omega \in \Sigma^*\}$, $L^* = L \cup LL \cup LLL...$ (in particular, $\{a^n b^n / n > 1\}$). This fact illustrates the limitation of Petri Net as a tool, that generates the languages. [1]
In Petri Nets is not possible to remember arbitrarily long sequence of arbitrary characters. In Petri Nets the sequence of limited length can be remembered (this is also possible in finite automata). [1]
However, Petri Nets do not have the "capacity of pushdown memory" which is necessary for the generation of Context-free languages. The interrelation of languages of Petri Nets with other classes of languages investigated Venn, this is shown in Figure 4 in the form of a diagram [1].

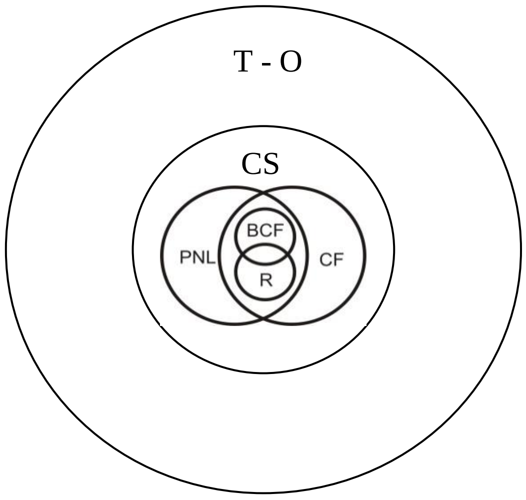


Fig. 4 Interrelation of Petri Nets and Traditional Languages (T-0-the General type of languages, CS-Context- sensitive languages, PNL-Petri nets languages, CF-Context- free languages, BCF-bonded Context- free languages, R-Regular languages).

Results:
We modeled $L^* = L \cup LL \cup LLL...$language (star Klin) by CPN, in particular $\{L = a^n b^n / n \geq 1\}$.

The Figure 5 shows a Colored Petri Net, which generates the L^* language that is, Colored Petri Net is a more powerful tool than the classical Petri Net. To understand types of data which are used in a figure, it is necessary to give a declaration.

In the Figure 5 introduced two positions of count of type and marked as first and second. In the figure, two transitions marked with the symbol **a** that are generating symbol **a**, and a transition marked with the symbol **b**, which generates the symbol **b**.
In the figure position of count of type remembers the number of transitions are fired and regulates, so the number of appearances a symbol **b** was equal to the number of appearances a symbol **a**.
In fact, when the marked with a transition is fired, generates the symbol **a**, if the marked with b transition is fired, generates the symbol **b**. To the transitions are attached logical expressions (guards): $ct > 0$, $ct > (n-1)$, if the logical expression is true, then the transition is allowed, and if false, then the transition is not allowed. If the first position of count of type value of token is equal to one, the marked with a first transition is fired. The value of **n** must be fixed advance.
Let $n = 2$, then is fired marked with **a** the first transition, and $ct = 2$, then is fired marked with **a** the second transition, are generated by **aa** symbols, in this case second position of count of type value of token is equal to two: count= 2, and twice is fired marked with **b** the transition, are generated by **bb**, when the value of the first counter is equal to one, cycle will be repeated, etc.
Many properties of Colored Petri nets, as logical expressions, types of tokens, the expression of the arcs, etc., which are used to control the transition firing [3].
In Figure 5 Colored Petri Net is constructed for the given language, which supposes following interrelation of languages of Colored Petri Net with some of traditional languages classes (see Figure 6).

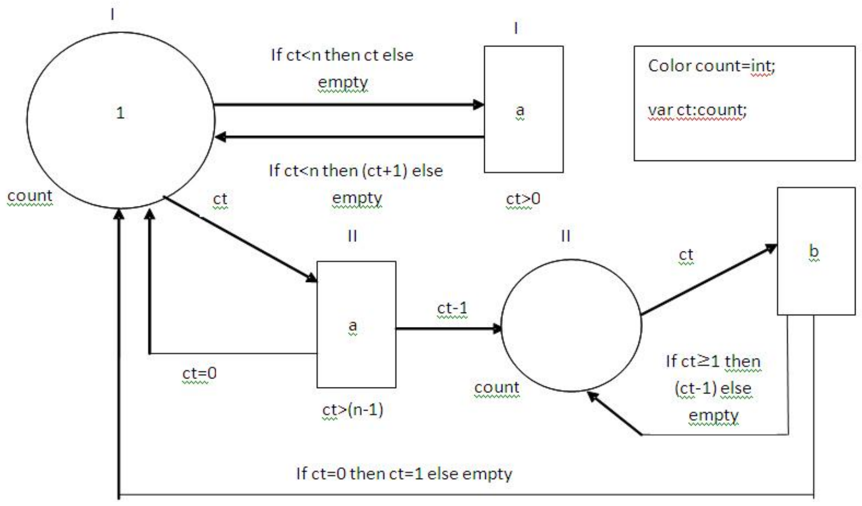

I
1
count
If ct<n then ct else empty
If ct<n then (ct+1) else empty
I
a
ct>0
Color count=int;
var ct:count;
ct
II
a
ct>(n-1)
ct=0
ct-1
II
count
ct
b
If ct≥1 then (ct-1) else empty
If ct=0 then ct=1 else empty

Fig. 5 Modeling $L^{*} = L \cup LL \cup LLL\ldots$language by Colored Petri Net.

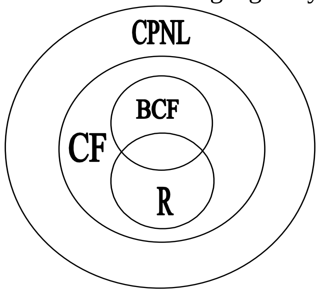


Fig. 6 Interrelation of Colored Petri Nets and Traditional Languages. CPNL (Language of Colored Petri Net)

4. ON A SOLUTION TO THE CIGARETTE SMOKER'S PROBLEM WITH COLORED PETRI NETS

In 1971 Patil proved that P- and V- actions have insufficient capacity for resolving synchronization issues. His proposed solution to model problem is called smoking a cigarette. The problem consists of four processes, which model the states and actions of the agent and 3 smokers. Each smoker continuously makes a cigarette and smokes it. To make and smoke the cigarette we need three ingredients: tobacco, paper and matches. One of the smokers always has the paper, the second smoker has tobacco, the third has the matches. The agent has unlimited number of the ingredients. The agent puts the two ingredients on the table, the smoker who has the third ingredient, can make the cigarette and then he signals the agent. In this case, the agent offers the other two ingredients, which are necessary for the smoker, and the action is repeated.
The actions of the smokers without the coordination are as follows.
Let X - the smoker with tobacco, Y - the smoker with paper, Z - the smoker with matches, A-agent (see Table 1).

Table 1: The actions of the smokers

Processes A_X	**Processes A_Y**	**Processes A_Z**
pick up the paper	pick up the tobacco	pick up the tobacco
pick up the match	pick up the match	pick up the paper
roll the cigarette	roll the cigarette	roll the cigarette
light the cigarette	light the cigarette	light the cigarette
smoke the cigarette	smoke the cigarette	smoke the cigarette
return to A_X	return to A_Y	return to A_Z

The smokers' problem is, then, to define some additional semaphores and processes, if necessary, and to introduce necessary P and V statements in these processes so as to attain the necessary cooperation among themselves required ensuring continued smoking of cigarettes without reaching a deadlock. There is, however, a restriction that the process which supplies the ingredients cannot be changed and that no conditional statements may be used. The first restriction is placed because the smokers are seeking cooperation among themselves and therefore should not change the supplier, and the second restriction is justified because P and V primitives were introduced to avoid having to coordinate processes by repeatedly testing a variable until it changes its value, and because the operation of making and smoking a cigarette has no conditional actions. It will be seen that the cigarette smokers' problem has no solution.

Patil showed that there is no sequence of P- and V- actions and can not solve the problem correctly [1,2]. If we try to model the problem with the Classical Petri Net, we will get a non-active net. Also, since all tokens in Classical Petri Nets have the same type, then ingredients will not be differed from each other.

The author modeled the problem with Colored Petri Net (see Figure 7), which gives the possibility of number of modelings for solving similar problems due to a number of properties, such as the existance of the attached data types, functions, transitions [3-10].

Position A presents the agent status, which has 3 types of tokens: (p, m) - (p-paper, m-match), (t, m) - (t-tobacco, m-match), (p, t) - (p-paper, t- tobacco), with the help of an organized net of cycle, it is possible to constantly offer 2 different ingredients to the smokers.

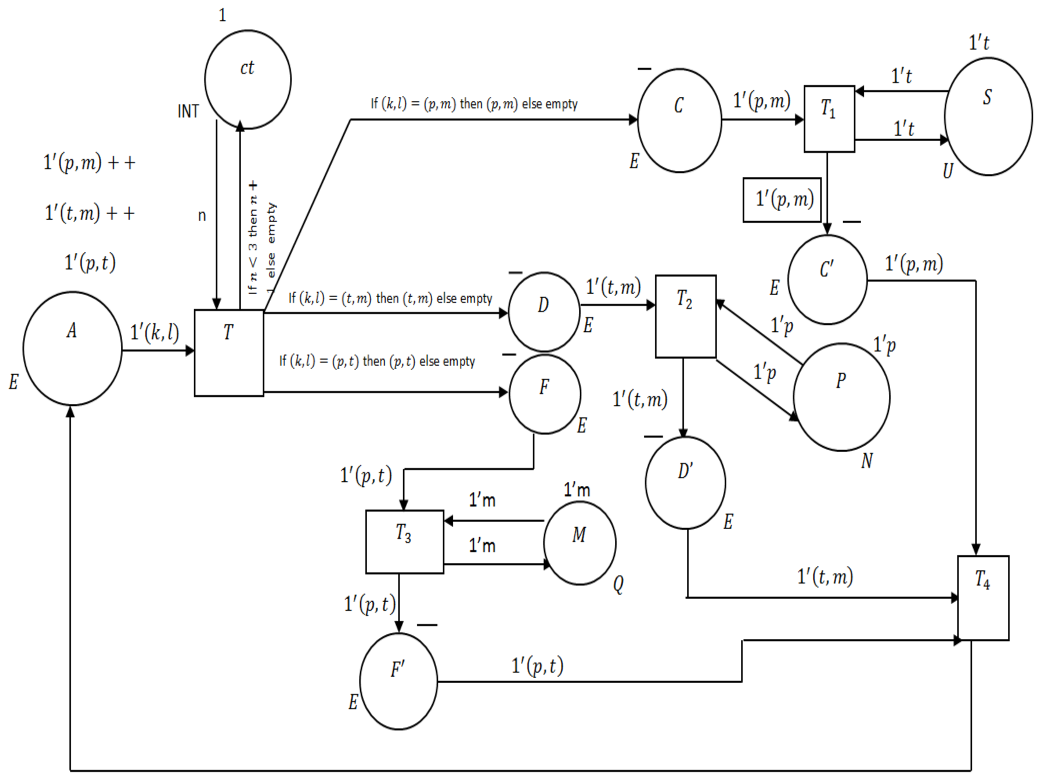
In position A and also in number of positions in the net, E type is attached, that is a combination of 3 types, which means the combination can be any one of these 3 types, and each of which is Cartesian product: either with N and Q , U and Q, or N and U. (k, l) the pair of variables is a Cartesian product of E-type, that is either (p, m), or (t, m) or (p, t). C', D', F', C, D, F positions have E type, where C, D, F transport the relevant ingredients to the relevant smokers. C ', D', F' positions give ingredients to the agent again, so the cycle is repeated. On the top of the position we put the first mark, which means the net is initialized. If on the top of the position there is the "-" sign, it means that the position is empty or it contains no token. The type is marked in italic letters below the position. The **ct** position is connected to **T** transition. It allows to make **T** transition 3 times during per cycle. S, P, M positions respectively have U, N, Q types. They present the state of the 3 smokers who respectively own tobacco, paper and matches. T4 transition will be occured only if all 3 smokers finished smoking.

Now we will describe the net performance. **T** will occur the first transition, as there is the corresponding marks in **A** and **ct** positions. After **T** firing transition we check if $(k, l) = (p, m)$, token is for C position and the other two (D, F) positions do not get any marks. If $(k, l) = (p, m)$, then T1 transition can be occured, which describes the process of the first smoker, and (p, m) ingredients give C' position. It may occur **T** transition in this condition, which will move (t, m), or (p, t) tokens and T2, or T3 transitions will be occured. If T1, T2, T3 transitions are occured, or the agent has proposed the 3 types of ingredients and the 3 smokers have finished smoking, then the T4 transition will be occured, the agent will offer 3 types of ingredients again and the cycle will be repeated.

If we try to present this problem with Classical Petri Net, then we need to use 3 transitions instead of **T** transition. That also means the minimization problem of the net is also solved, which implies cost reduction due to the reduction of arcs in positions and transitions.

Declaration

```
Color INT = integer;
Color  U = t;
Color  N = P;
Color  Q = m;
Color  E = {Product N*Q OR Product U*Q OR Product N*U};
Var(K, l): E;
 n: INT;
```



ct
INT
n
1
If n < 3 then n + 1 else empty
1'(p,m) ++
1'(t,m) ++
1'(p,t)
A
E
1'(k,l)
T
If (k,l) = (p,m) then (p,m) else empty
If (k,l) = (t,m) then (t,m) else empty
If (k,l) = (p,t) then (p,t) else empty
C
D
F
E
1'(p,m)
T1
1't
S
U
1'(p,m)
C'
1'(t,m)
T2
1'p
P
N
1'(t,m)
D'
1'(p,t)
T3
1'm
M
Q
1'(p,t)
F'
1'(t,m)
T4
1'(p,t)

Fig. 7 The modeling of Cigarette smoker's problem with Colored Petri Nets

5. CONCLUSION

In the studies and the proved theorem of the section "The Algorithm Description of the Shortest Possible Sequence of Transitions in Petri Nets" brings out several important features of Petri Nets in optimization perspective, according which, if Petri Nets are used in technical devices, then the idea of succession transition passages brings resources and saves time.

In the section of Interrelation of Languages of Colored Petri Nets and Some Traditional Languages The Venn graph and diagram that the author modified, show the interrelation between languages of Colored Petri Nets and some Traditional languages. Thus the class of languages of Colored Petri Nets is supposed to include an entire class of Context-free languages.

In the problem of "On a Solution to the Cigarette Smoker's Problem with Colored Petri Nets" we identify certain advantages of Colored Petri Net to P and V operations and Classical Petri Net with the synchronization problem. The mentioned studies allow identification of synchronization modeling opportunities with the help of Colored Petri Net.

REFERENCES

[1]. Peterson J., Petri Net Theory and the Modelling of Systems. Prentice Hall. ISBN 0-13-661983-5 (1981).

[2]. Tadao M., "Petri nets: Properties, Analysis and Applications." Proc. of the IEEE, 77(4), 1989.

[3]. Jensen K. and Rozenberg G., Eds., High-Level Petri Nets. Theory and Application, Springer-Verlag, Berlin, 1991, pp. 44-122.

[4]. Jensen K., Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use. Springer - Verlag, Berlin, 1992.

[5]. Jensen K. Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use. Springer, 1996. Vol. 1–3.

[6]. Jensen K., "Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use", Basic Concepts. Monographs in Theoretical Computer Science, Springer – Verlag, Berlin, Germany, V.1,2, 3, 1997.

[7]. Ullman J., "Elements of ML Programming," Prentice- Hall, Upper Saddle River, 1998.

[8]. Reising W., Rozenberg G. (eds), Lecture Notes on Petri Nets. Parts I and II // Lecture Notes in Computer Sciences. V. 1491 – 1492. Springer – Verlag, 1998.

[9]. Jensen K. and Kristensen L., Coloured Petri Nets - Modeling and Validation of Concurrent Systems. Springer-Verlag Berlin, 2009.

[10]. Alfred A., Jeffrey U., Theory of Parsing, Translation, & Compiling Prentice Hall, January 1, 1973, v. 1,2.

[11]. Knut D., The Art of Programming, v. 1-3, Moscow, Mir 1976.

[12]. Orlov S., Technology of Software Development, textbook for Universities, Petersburg, 2002.

[13]. Gordeev A., Molchanov A., System Software, textbook, St. Petersburg, 2002.