Chapter

The Possibilities of Modeling Petri Nets and Their Extensions

Goharik Petrosyan

Abstract

This chapter 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 an algorithm that describes the minimum sequence of possible transitions. The algorithm developed by us finds the shortest possible sequence for the network promotion state, which transfers the mentioned network state to the coverage state. The corresponding theorem is proven, which states that due to the describing algorithm, the number of transitions in the covering state is minimal. This chapter studies the interrelation of languages of colored Petri nets and traditional formal languages. The Venn diagram, modified by the author, is presented, which shows the relationship between the languages of the colored Petri nets and some traditional languages. As a result, it is shown that the language class of colored Petri nets includes an entire class of context-free languages and some other classes. The results obtained show that it is not possible to model the Patil problem using the well-known semaphores P and V or classical Petri nets, so the mentioned systems have limited properties.

Keywords: petri nets, colored petri nets, traditional languages, transition, position

1. Introduction

Modeling and designing systems cannot 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 lumped physical systems, 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.

Petri nets are a mathematical apparatus for modeling dynamic discrete systems. Their feature is the ability to display parallelism, asynchrony, and hierarchy. They were first described by Karl Petri in 1962.

The Petri net is a bipartite oriented graph consisting of two types of vertices—positions and transitions—connected by arcs between each other; vertices of the same type cannot be directly connected. Positions can be placed tags (markers) that can move around the network [1].

Petri net—a tool for modeling dynamic systems. The theory of Petri nets makes it possible to model a system with a mathematical representation of it in the form of

a Petri net, the analysis of which helps to obtain important information about the structure and dynamic behavior of the simulated system.

There are several ways of practical application of Petri nets in the design and analysis of systems. In one of the approaches, the Petri nets are considered as an auxiliary analysis tool. Here, to build the system, generally accepted design methods are used, then the constructed system is modeled by the Petri net, and the constructed model is analyzed.

In another approach, the entire process of design and characterization is carried out in terms of Petri nets. In this case, the task is to transform the representation of the Petri net into a real information system [2].

The undoubted advantage of Petri nets is a mathematically rigorous description of the model. This allows their analysis with the help of modern computing techniques (including those with a massively parallel architecture) [1].

In modern society, reliable transmission and protection of information are of wide use and are topical tasks. 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 equipment where one has to minimize resource costs and time and maximize speed.

Colored petri net (CPN) modeling mechanisms are a convenient graphic language for designing, modeling, and testing systems [3–7]. They are well suited for systems that discuss interaction issues and synchronize. The colored Petri nets are well suited for modeling distributed systems, automated production systems, and for the design of VLSI circuit chips [8–10].

Colored Petri nets are called if the chips are the values of some types of data, which are usually called color sets. Expressions are assigned to arcs in such a network. When transitions are triggered, the values of expressions on arcs are calculated. The results of the calculations are extracted from the markup of the input transition points and placed in the marking of the output points. Transitions may be assigned with security expressions. If the guard expression assumes the value "false," the transition is prohibited [3–6, 11, 12].

The language generated by CPN allows to represent a model that is a collection of modules, allowing you to hierarchically represent complex networks or systems.

In classical Petri nets, the tokens do not differ from each other; they are colorless. In colored Petri nets, a position can contain chips that are of arbitrary complexity, such as lists, etc., that allow you to simulate more reliable models [8–10, 13].

2. The algorithm description of the shortest possible sequence of transitions in petri nets

To build models of discrete systems, it needs various components of the system with abstract operations: switching the transition from one state to another; the action of a program operator, machine, or conveyor; interruptions in the operating system; phase completion in the project; etc. The same system can work differently under different conditions, generating many processes that will bring nondeterministic work. In real systems, cases occur at certain periods and last a certain time. In synchronous models of discrete systems, events are correctly associated with certain pauses, moments during which all components simultaneously change the state of the system, changing the state of the system.

The modeling approach has several drawbacks when dealing with large systems.

To make the model look impressive, first of all, with every change, the system must take into account all the components of its general condition.

Secondly, with the above approach, information in systems disappears between random links.

Thirdly, the so-called asynchronous systems can cause undefined events at time intervals.

Petri nets and the above types of models are called asynchronous.

Causal relationships make it possible to more clearly describe the structural features of the system.

Asynchronous models of nonformal 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, has 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 term has its capacity. The term is not fulfilled (capacity is equal to 0), the term is fulfilled (capacity is equal to 1), and the term is fulfilled in n times (capacity is equal to n, where n- is a positive integer).

Most systems are suitable as discrete structures that consist of two elements: type of events and terms. Cases and terms in Petri nets, sets that do not intersect with each other, respectively, are called positions and transitions. Transitions are vertical lines and places with circles in a graphical representation of Petri nets [1, 2].

2.1 The relationship of petri nets, reachable states, and reachable trees

Definition 1: Petri nets are $M(C, \mu)$, where $C = (P, T, I, O)$ is the network structure and μ is the network condition. P is positions and T is transitions, which are finite sets. $I : T \to P^{\infty}, O : T \to P^{\infty}$ are input and output functions, respectively, where P^{∞} are all possible multisets (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 [8]. 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 $0 \le \#(X,B) \le 1$, then we will reach the idea of the set. Since $\#(\mathrm{X},\mathrm{B})$ function determines the X element entering collection of B, it follows that $\#(\mathrm{X},\mathrm{B}) \ge 0$, the grouping of all the X and B. X element of the B collection, if $\#(\mathrm{X},\mathrm{B}) > 0$, i.e. $X \in B$,. Identically, if $\#(\mathrm{X},\mathrm{B}) = 0$, then $\mathrm{X} \notin \mathrm{B}$.

Let us set empty collection of ∅, which has members (i.e., all $\mathrm{X} : \#(\mathrm{X},B) = 0$). |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:

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

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

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

$$\mu(P_i) \ge \#\left(P_i, I\left(t_j\right)\right).$$

Suppose in μ state $\mathrm{t_j}$ 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:

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

Let us name $\mathrm{R}(\mathrm{C}, \mu_0)$ as the reachable state set:

1. $\mu_0 \in \mathrm{R}(\mathrm{C}, \mu_0)$,

2. If $\mu' \in \mathrm{R}(\mathrm{C}, \mu_0)$ and $\exists \mathrm{t_j} \in \mathrm{T}$ have transition in the way that $\delta\left(\mu', \mathrm{t_j}\right) = \mu"$, then $\mu'\prime \in R(C, \mu_0)$.

3. Other states do not belong to $\mathrm{R}(\mathrm{C}, \mu_0)$. The $\mathrm{R}(\mathrm{C}, \mu_0)$ can be infinite $\mu"$ marking covers μ′ marking if

$$\mu'\prime \ge \mu'.$$

First, build a reachability tree. Then you need to look for the peak as follows. If there is no such peak, then the marking is not covered by any achievable marking, if it is located inside and gives an accessible marking that covers [14–16].

We construct the reachability tree of Petri nets in **Figure 1**. The state of this network is (1101), which shows the presence of tokens in the network at this moment. Tokens shown in **Figure 1**, which are depicted with small dots, correspond to the availability of resources. The network state changes due to the movement of tokens.

Let the states correspond to the vertices and transitions to the sides. The root corresponds to the first state of the network.

Figure 1 corresponds to **Figure 2**, in which the reachable tree is infinite. To make the tree finite, we impose restrictions. If any peak is blocked, we will call it a terminal. If there is a state in any peak and there is another peak in the tree with the

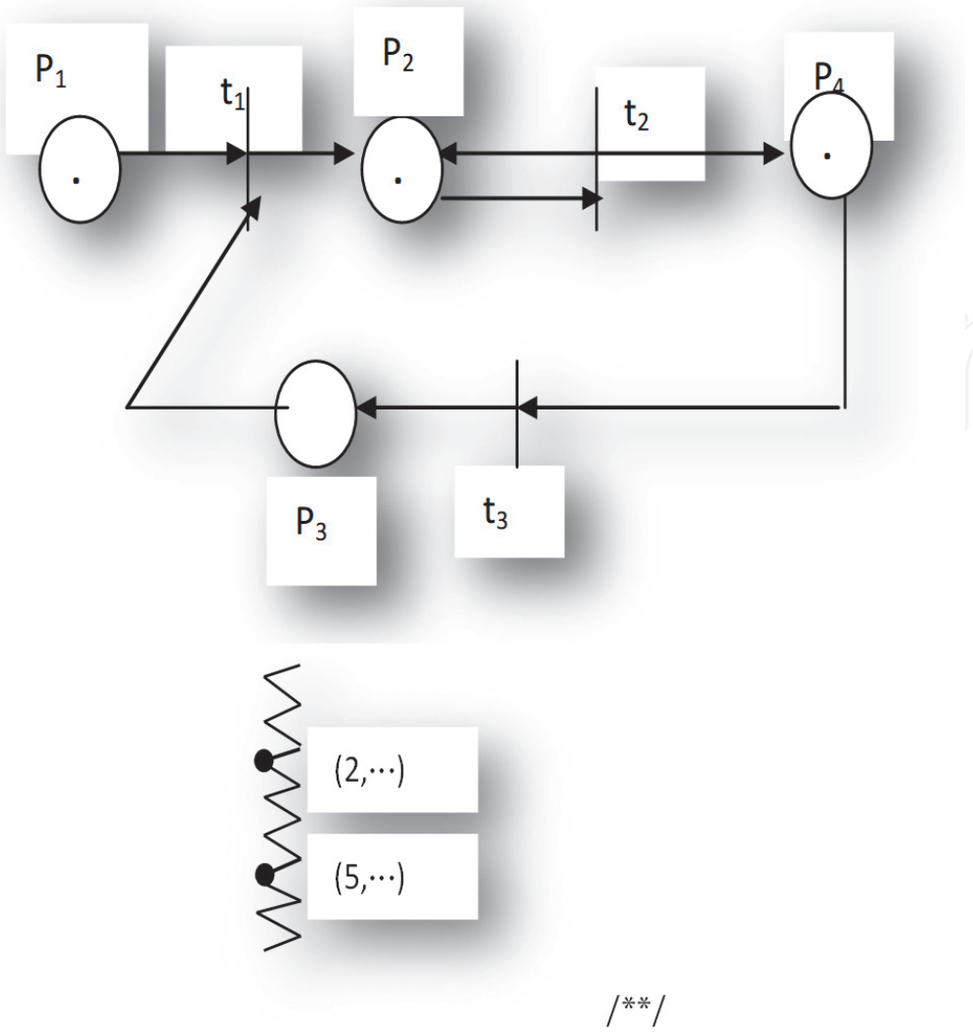

Figure 1.
An example of petri nets. The way in the tree.

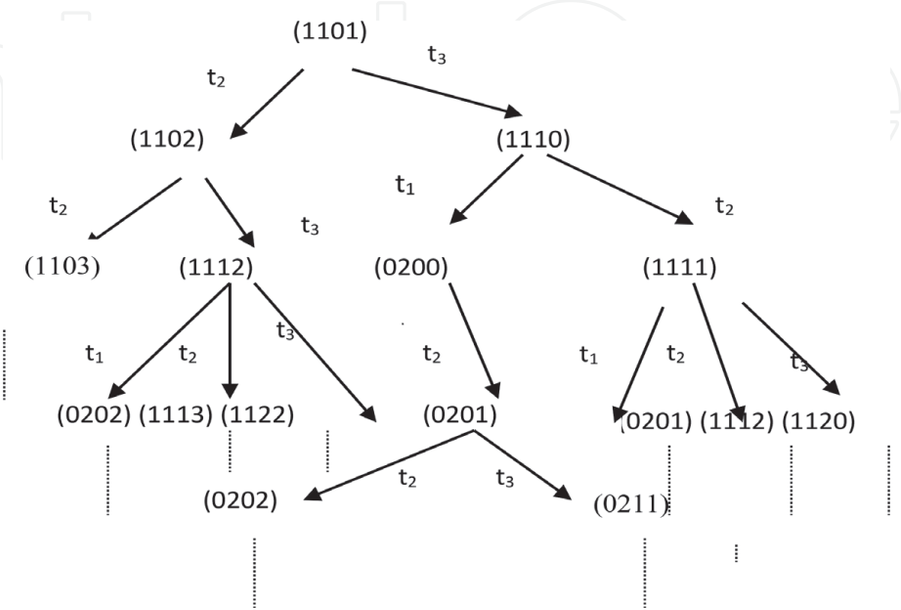

Figure 2.
The petri net reachable infinite tree.

same state that has already been developed, then we will call the new peak repeated and will not develop it.

If there is a path like / ** / in the tree, then the path through the second peak may repeat, and the states grow. Let us introduce the idea of infinite much as ω:

$\omega \geq \omega$, $\omega + a = \omega$, and $\omega - a = \omega$, where $a = \text{const}$: for example, instead of $(5,\cdots)$, we will write $(\omega, \cdots)$. In this case the tree will become as finite, and we will have loss of information [2].

Let us give several definitions, which will be used in the entire work.

Definition 2. A peak is called a boundary if it is in a processing state.

Definition 3. The peak is called a terminal if it does not contain a subtree.

Definition 4. The peak is called internal if it has already been processed.

Definition 5. The boundary peak is repeated if there is an internal peak with the same state.

A description of the structure of the reachability tree algorithm can be seen in an earlier published article [8].

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

2.2 The algorithm for finding the minimum number of transitions in a coverage state

Consider the Petri net in **Figure 1** and the corresponding reachable tree TT (see **Figure 3**).

We note the set of states in Petri nets with P. Let T^* denote the succession of transitions from the root TT to y, the transition sequence with G the succession of the peaks in T^*.

Consider $\mu[x] = (0, 1, 15, 13)$ state. Let us find the y peak of this reachable tree for which the following inequality holds: $\mu[y] \geq \mu[x]$.

Assume that such peaks are $y_1, \ldots, y_m$. Let us 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 example, $\mu[x]$, we will cover the following peak:

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

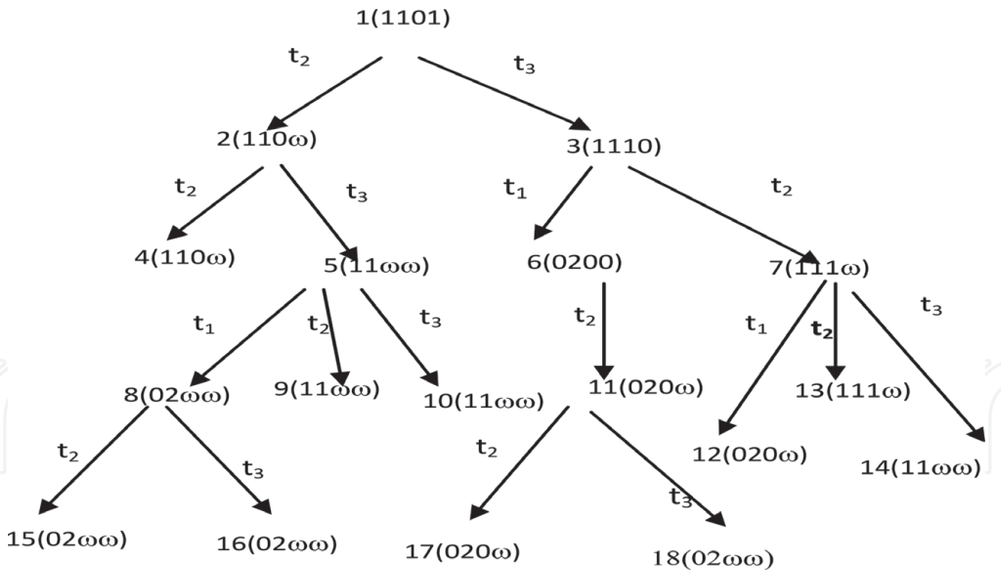

Figure 3.
The petri net reachable tree.

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

We found out that in minimum number, $S(y_1) = S(y_3)$, $|T^*(y_1)| = 2$, and $|T^*(y_3)| = 3 \Rightarrow$ need to take a y_1 peak. Choosing the appropriate peak coverage, we use the algorithm. Let it be a covering peak.

1. We choose the path connecting the root of the tree with$y$ and T^*; for our example t_2, t_3 let us mark $t'_i = t_j, 1 \le i \le |T^*|$, and $t_j \in T^*$. We get $t'_1 = t_2$, and $t'_2 = t_3$.

2. For each chosen transitions, t'_i is corresponded with a_i numbers in the following way:

   - If for t'_i transition $\exists 1 \le j \le |P|$ 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 $\exists 1 \le k \ne j \le |P|$ in the way that $\delta(\mu[y'], t'_i)_k = \omega$, then $a_i = \max\left\{\mu[x]_j, \mu[x]_k\right\}$.

Moreover, for t'_i transition we will correspond P_j and P_k positions. If instead of $t'_i\ \sigma = t'_{i_1} \ldots t'_{i_k}\left(t'_{i_k} = t'_i\right)$ for $\exists 1 \le j \le |P|$ 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 $1 \le j \le |P|$ 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 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 $n = \mu_0[P_1]$, and $m = \#\left(P_j, O\left(t'_i\right)\right)$.

If there is no corresponding position for t'_i transition, then we will leave a_i to remain the same.

For example:

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

Let us mark $b_i^1 = a_i$. For example:

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

2.2.1 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 does not change in the next moves.

We consider all T^* items from the right to left, starting from t_α.

Suppose t'_k is the considered transition or the transition succession and 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 $\exists 1 \leq j \leq l$ 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, $b_i^{j+1} = b_i^j$.

After that, we fix t_α the previous transition action and denote it as t_α.

We denote the new t_α corresponding to b_i^j as α and go to the second step again.

It follows that for each transition or sequence of transitions, there will be a correspondingly fixed number b_i^j, which will mark the number of implementation of the transition or sequence of transitions. For example:

t_1'	t_2'
12	15
27	15

The above shows that the t_1' transition must be implemented for 27 times and t_2' transition for 15 times.

Given our denote, we get that state $\mu[x] = (0, 1, 15, 13)$ covers state $\mu[y] = (1, 1, \omega, \omega)$. To achieve the goal, we need to implement the t_2 transition 27 times and the t_3 transition for 15 times.

Let us assign $t_s(y) = \sum_{i=1}^{l} b_i^j$, $l = |T^*(y)|$. 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 $\mu(y_1) \geq \mu[x] \& \mu(y_2) \geq \mu[x]$. In that case $t_s(y_1) \leq t_s(y_2)$.

Proof: We have.

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

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

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 is 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'' is equal to 1. Moreover $d'' \leq d'$, as $S(y_1) < S(y_2)$,

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

The lemma is proven.

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 proven.

Theorem. Through the abovementioned number of covering state, transition algorithm is in its minimal state.

Proof: Let y be the covering peak in our algorithm and $t_1', \cdots, t_k'$ the succession of transitions. It must be shown that the number of $t_1', \cdots, t_k'$ move is in minimal state. For this we need to show that $\nexists y'$ covering peak has less number of transitions than the number of $t_1', \cdots, t_k'$.

Let us consider two cases:

1.$y \neq y'$

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: $t_s(y) \leq t_s(y')$. We've come into a controversy.

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

2.$y = y'$

Suppose succession transitions of y' is $s_1, \cdots, s_r$. As the tree does not contain any cycle, $y = y' \Rightarrow t'_1, \cdots, t'_k$ and $s_1, \cdots, s_r$ are the same $\Rightarrow t_s(y) \leq t_s(y')$. The theorem is proven.

2.3 Conclusion

The proven theorem and research reveal some important features of Petri nets from the point of view of optimization, that is, if the idea of Petri nets is used in technical devices, then the idea of sequential transitions save resources and time.

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:

$\sum$ is a finite set of non-empty types called color sets [17].
P is a finite set of places which are depicted as ovals/circles.
T is a finite set of transitions which are depicted as rectangles.
A is a finite set of arches which are depicted as directed edges; moreover.

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

N is a node function, $A \to P \times T \cup T \times P$.
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, which is defined as follows:

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

I is an initialization function [3–6, 9, 10],

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

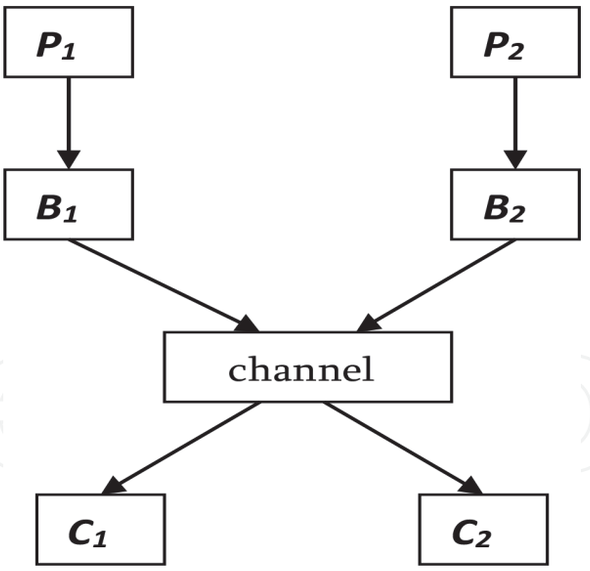

Figure 4.
The consumers' process with the common usage and buffer is an action.

The distribution of tokens, called marking, determines the state of the simulated system. The dynamic behavior of CPN is due to the triggering of a transition that transfers the system from one state to another. A transition is enabled if the associated arc expressions of all input arches can be evaluated as a multi-set, which is compatible with the current tokens in corresponding input places, and its guard is satisfied. After the transition is triggered, tokens are removed from the input places, respectively, by the specified expression of the arches of all incoming arches, and tokens are placed in the output places, respectively, by the specified expressions of the outgoing arches [3–6, 17].

3.1 The example of modeling consumers' process with CPN

Let us suppose that there are two processes of producers and consumers [1, 9].

The following picture shows the process diagram (**Figure 4**).

There is a distribution problem in the described system. To use the channel, the pair(P_1, C_1) must have priority toward (P_2, C_2) in the sense of using the channel. This is described as follows: while the buffer is not empty, the channel cannot report data from the buffer to the consumer. It is impossible to solve this problem with the help of classical Petri nets, since it is permissible in nature. The proof of this fact is described in the literature [1].

To solve that problem, it is needed to extend Petri net's several properties in such a manner that the proposed properties are headed toward the opportunity of checking the zero in Petri nets [13].

3.2 Declaration

```
Color E = {e};
Color Control = {0;1};
Color S = product E*Control;
Var ct:Control;
```

The CPN (**Figure 5**) is the model of the solved problem of priority usage [17, 19].

Declaration:
Color E={e};
Color
Control={0;1};
Color S=product
E*Control;
Var ct:Control;

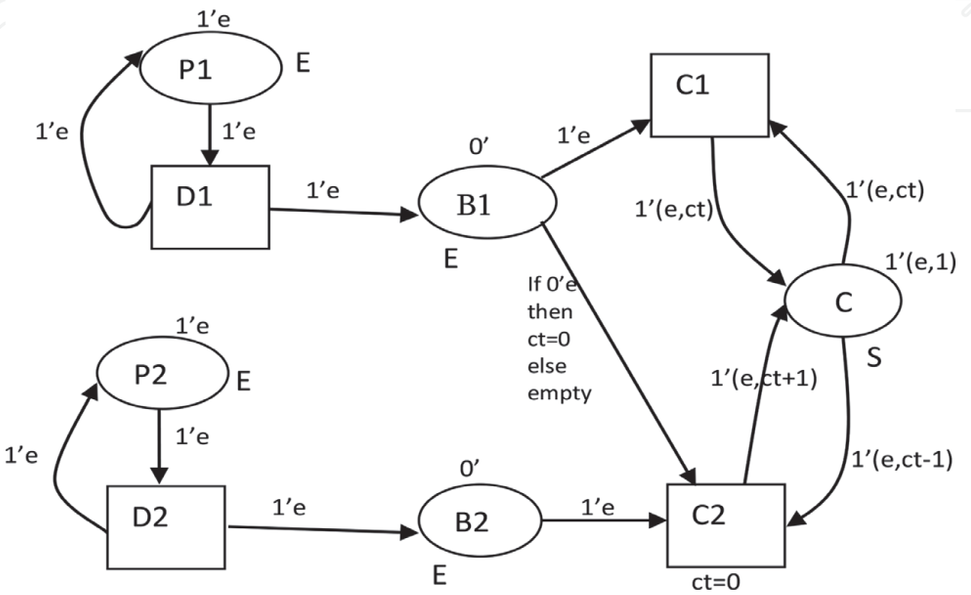

Figure 5.
The modeling of consumer problem with colored petri net.

3.3 The modeling of context-free languages with colored Petri nets: the diagram of interrelation of colored Petri nets and traditional languages

It is known that the class of regular languages is one of the most studied simple classes of formal languages and any regular language is the language of Petri nets [2, 18].

There are context-free languages that are not languages of Petri nets. Such examples of the context-free languages are $\{\omega\omega^R/\omega\in\Sigma^*\}, L^* = L\cup LL\cup LLL\ldots$ (in particular, $\{a^n b^n/n>1\}$).

The noted fact shows the limitation of Petri nets as a mechanism that generates languages [2].

In Petri nets one can only remember a sequence of limited length (similar to finite automata) [2].

It is clear that Petri nets do not possess the "capacity of pushdown memory" necessary for generating context-free languages. The relationship of the languages of Petri nets with other classes of languages (Venn diagram) is shown in **Figure 6** [2, 10].

3.4 Results

A model of the $L^* = L\cup LL\cup LLL\ldots$ language (Klins' star) is constructed using colored Petri nets, in particular $\{L = a^n b^n/n\geq 1\}$.

Colored Petri net (**Figure 7**) generates such a language, which proves that the colored Petri net is a more powerful tool than the classical Petri nets. The following declaration is for the concept of data types.

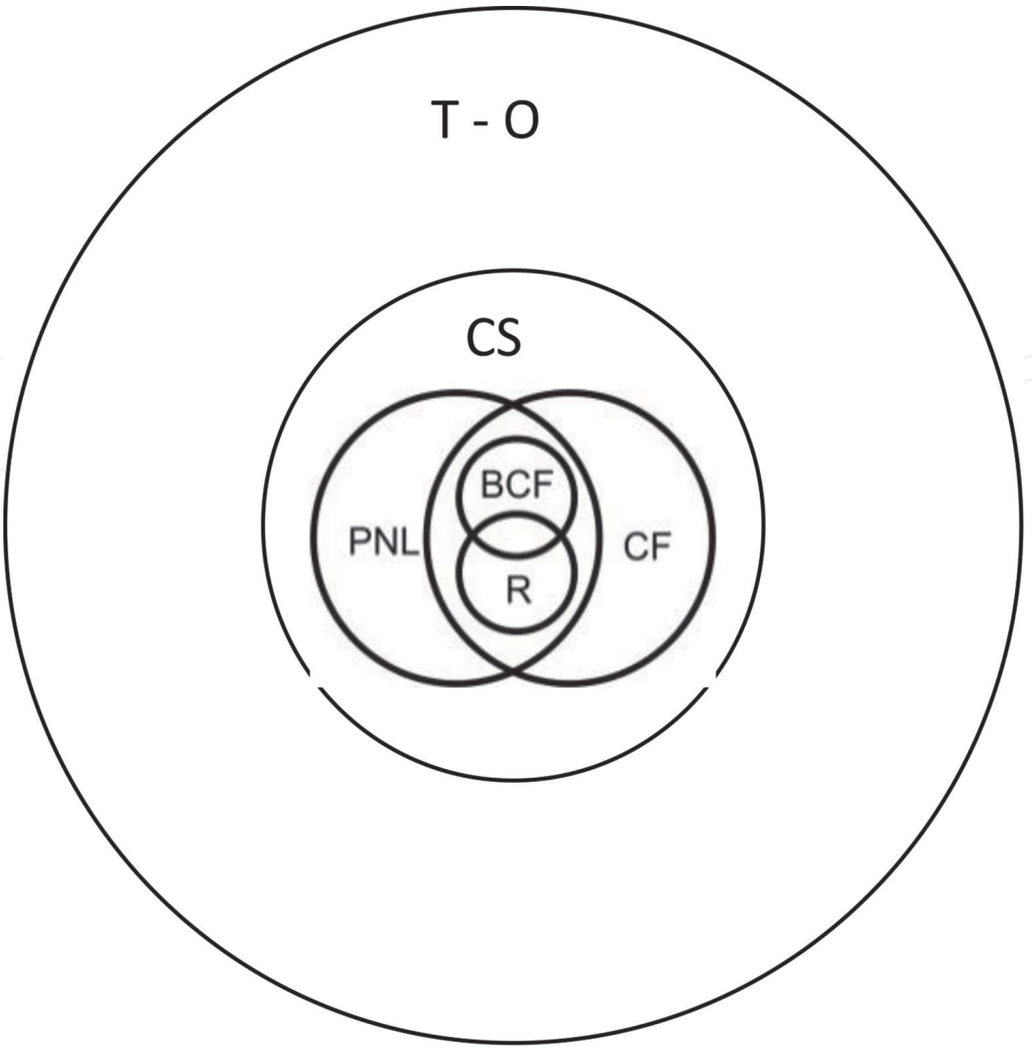

Figure 6.
Interrelation of petri nets and traditional languages (T-0, the general type of languages; CS, context-sensitive languages; PNL; petri net languages; CF, context-free languages; BCF, bonded context-free languages; R, regular languages).

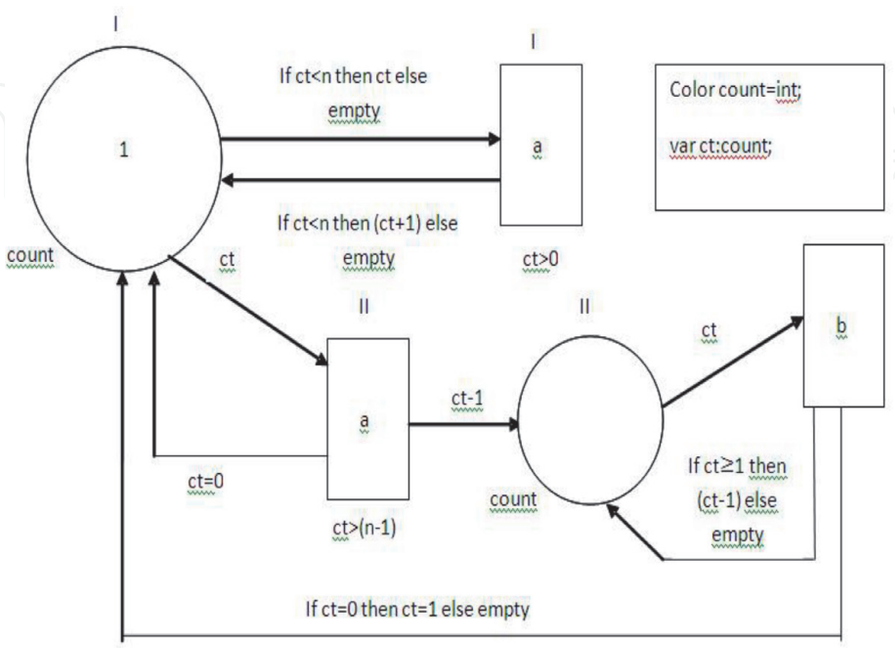

Figure 7.
Modeling $L^* = L \cup LL \cup LLL \ldots$ language by colored petri net.

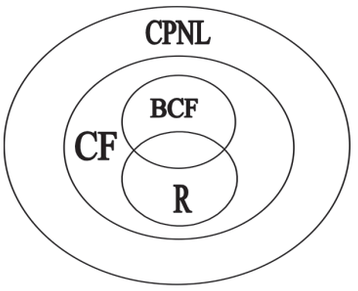

Figure 8.
Interrelation of colored petri nets and traditional languages. (CPNL, language of colored petri net).

The operation of the colored Petri net shown in **Figure 7** is described in more detail in the literature [3, 10].

The colored Petri net (**Figure 7**), which is built for $L^* = L \cup LL \cup LLL \ldots$ language, suggests the following relationship between the languages of the colored Petri nets with some classes of traditional languages (see **Figure 8**) [10].

The Venn diagram, modified by the author (**Figure 8**), shows the relationship between the languages of the colored Petri nets and some traditional languages. This fact illustrates that the language class of colored Petri nets includes an entire class of languages without context.

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 [9].

The actions of the smokers without the coordination are as follows.

Let X be the smoker with tobacco, Y the smoker with paper, Z the smoker with matches, and A the agent (see **Table 1**).

It is proven that the problem of smokers has no solution using semaphores [9].

Patil showed that there is no sequence of P and V actions to correctly solve the problem [1, 2]. Modeling the problem using the classical Petri net, we get an inactive network. Since all tokens in classical Petri nets are of the same type, the ingredients will not differ from each other.

The author simulated a problem with the colored Petri net (see **Figure 9**) [3–6, 9, 18, 19].

The operation of the colored Petri net shown in **Figure 9** is described in more detail in the literature [9].

If we were to represent this problem using the classical Petri net, then we need to use three transitions instead of one **T** transition. It also means that minimization of the network is ensured, which implies a reduction in costs due to the reduction of arches in positions and transitions.

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

Table 1.
The actions of the smokers.

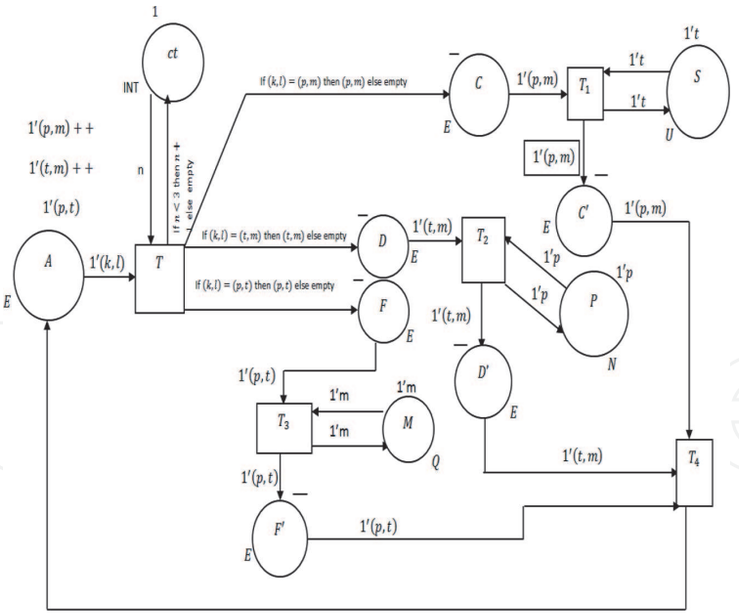

Figure 9.
The modeling of cigarette smoker's problem with colored petri nets.

4.1 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;

4.2 Conclusion

In the problem, 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.

Author details

Goharik Petrosyan
Plekhanov Russian University of Economics Yerevan Branch, Member of Armenian Mathematical Union, ASPU, ISEC of NAS RA, Yerevan, Armenia

*Address all correspondence to: petrosyan_gohar@list.ru; petrosyangoharik72@gmail.com

References

[1] Tadao M. Petri nets: Properties, analysis and applications. Proceedings of the IEEE. 1989;**77**(4)

[2] Peterson J. Petri Net Theory and the Modelling of Systems. Prentice Hall; 1981. ISBN 0-13-661983-5

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

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

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

[6] Jensen K. Colored petri nets: Basic concepts, analysis methods and practical use. In: Basic concepts. Monographs in Theoretical Computer Science. Vol. 1–3. Berlin, Germany: Springer–Verlag; 1997

[7] Raising W, Rosenberg G, editors. Lecture notes on petri nets. Parts I and II. In: Lecture Notes in Computer Sciences. Vol. 1491–1492. Springer-Verlag; 1998

[8] Petrosyan GR. Description of the algorithm for finding the shortest sequence of transitions in a Petri net. Multidisciplinary Scientific Edition International Academy Journal Web of Scholar. 2017;**5**(14):20-25. Available at: http://webofscholar.com/ ISSN 2518-167X Founder – RS Global Media LLC, Kiev, Ukraine

[9] Petrosyan GR. The modelling of the synchronization problem with Colored petri nets. International Journal of Electronic Engineering and Computer Science. 2016;**1**(2):56-60 Available at: http://www.aiscience.org/journal/ijeecs

[10] Petrosyan GR, Avetisyan AM, Ter-Vardanyan LA. Interrelation of languages of colored petri nets and some traditional languages. Open Journal of Modelling and Simulation. 2013;**1**:27-29. DOI: 10.4236/ojmsi.2013.13005. Published Online July 2013 (http://www.scirp.org/journal/ojmsi)

[11] Westergaard M, Kristiansen L. The access/CPN framework: A tool for interacting with the CPN tools simulator. In: Proc. of 30th International Conference on Applications and Theory of Petri Nets (Petri Nets 2009). Lecture Notes in Computer Science 5606. Berlin: Springer-Verlag; 2009. pp. 313-322

[12] Jensen K, Kristiansen L, Wells L. Colored petri nets and CPN tools for modelling and validation of concurrent systems. International Journal on Software Tools for Technology Transfer (STTT). 2007;**9**(3–4):213-254

[13] Petrosyan GR, Ter-Vardanyan LA, Gaboutchian AV. Modeling of biometric identification system using the colored petri nets. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. 2015; **XL-5/W6** Photogrammetric techniques for video surveillance, biometrics and biomedicine, 25–27 May 2015, Moscow, Russia

[14] Knut D. The Art of Programming. Vol. 1–3. Moscow: Mir; 1976

[15] Orlov S. Technology of Software Development, Textbook for Universities. Petersburg; 2002

[16] Gordeev A, Molchanov A. System Software, Textbook. St. Petersburg; 2002

[17] Jensen K, Kristiansen L. Coloured Petri Nets—Modeling and Validation of Concurrent Systems. Berlin: Springer-Verlag; 2009

[18] Alfred A, Jeffrey U. Theory of Parsing, Translation, & Compiling. Vol. 1-2. Prentice Hall; 1973

[19] Ullman J. Elements of ML Programming. Upper Saddle River: Prentice-Hall; 1998