INTERRELATION OF LANGUAGES OF COLORED PETRI NETS, MODIFIED PETRI NETS AND SOME CLASSES OF FORMAL LANGUAGES.

Goharik Petrosyan
ARMSPU, Yerevan, Armenia
e-mail: petrosyan_gohar@list.ru

ABSTRACT

The article studies the interrelation of languages of Colored Petri Nets, Modified Petri Nets and some classes Formal languages. The author constructed the graph of Colored Petri Net, which generates Context-free (CF) language

$$L(C) = \{\omega\omega^R / \omega \in \Sigma^*, \Sigma = \{a,b\}\},$$

and it may not be modeled by Classical Petri Nets [1].

Comparing the graphs

$$L(C) = \{\omega\omega^R / \omega \in \Sigma^*, \Sigma = \{a,b\}\}$$

modeled by CPN (Colored Petri Net, is shown in fig.1) and MPN (Modified Petri Net, is shown in fig.2); taking account of complexities of CP and MP nets (a number of positions, transitions and arcs) from the viewpoint of optimization, also the diagram [3, p. 42], one comes to the conclusion, that it might create the interrelation among the CPNL, MPNL and some classes of Formal languages.

Keywords: Petri Nets (PN), Modification Petri Nets (MPN), Colored Petri Nets (CPN), Context-free language (CF), Bounded Context-free language (BCF), Regular Language (R).

1. INTRODUCTION

Petri Nets (PN) is a graphical tool for the formal description of the flow of activities in complex systems. With respect to other more popular techniques of graphical system representation (like block diagrams or logical trees), PN is particularly suited to represent logical interactions among parts or activities in a system in a natural way. PN used for modeling real systems is sometimes referred to as Condition/Events nets [1,2].

Definition. Petri Net $M(C,\mu)$pair, where $C = (P,T,I,O)$ is the network structure and μ is the network condition.

In structure C of a P-positions, T-transitions are finite sets. $I: T \to P^\infty, O: T \to P^\infty$ are the 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. We determine (in a known manner) the allowed transitions of Petri Nets and the transitions from one state to another, as well the set of reachable states [1,2].

The idea of Modified Petri Nets belongs to the author [3, 4]. Suggested MPN-s, where the idea of *restrained positions* is imported, can model all CF language-classes. In [3] equivalence –theorem has been proved: for each CF –Grammar one can construct MPN which is equivalent to the language generated by CF –Grammar. An algorithm for constructing a Modified Petri Net equivalent to a given CF-Grammar described in detail [4].

Definition. Modified Petri Net is a $C = (P_1, P_2, T, I, O)$. In structure C of a P_1 -basic positions, P_2- restrained positions, T -transitions are finite sets. $P_1 \cap P_2 = \emptyset, P = P_1 \cup P_2$. $I: T \to P^\infty, O: T \to P^\infty$ are the input and output functions, respectively[4].

Colored Petri Nets (CPN) are considered as modern extension of Classical Petri Nets which was created by K. Jensen [5].

Colored Petri Nets (CPN) is a graphical oriented language for design, specification, simulation and verification of systems [5,6]. 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. 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 Classical Petri Nets in Colored Petri Nets of a position can contain tokens of arbitrary complexity, example, lists, etc., that enables modeling more reliable models.

Definition. The mathematical definition of Colored Petri Net: CPN is a nine-tuple

$CPN = (\Sigma, P, T, A, N, C, G, E, I)$, where:

1. 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 [6]. A token is a value belonging to a type.
2. P is a finite set of places. In the associated CPN Tool these are depicted as ovals/circles.
3. T is a finite set of transitions. In the associated CPN Tool these are depicted as rectangles.
4. 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

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

5. 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.
6. C is a colour-function. $C : P \to \Sigma$.
7. G is a guard function. It is defined from T into expressions such that:

$$t \in T : [Type(G(t)) = B \,\&\, Type(Var(G(t))) \subseteq \Sigma]$$

8. 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 $C(p)$.

9. I is an initialization function. It is defined from P into closed expressions so that:

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

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.

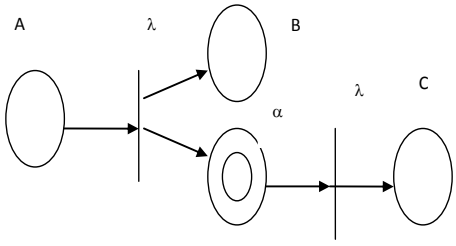
Unlike Regular languages, which are the languages of Petri Nets, there are Context-free languages, which are not languages of Petri Nets. Such examples of Context-free language we are noted the following: $\{\omega\omega^R / \omega \in \Sigma^*, \Sigma = \{a, b\}\}$ [1,7].

This fact illustrates the limitation of Petri Net as a tool, that generates the languages. 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 (CF) languages. The interrelation of languages of Petri Nets with other classes of languages investigated Ven [1].

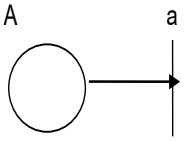
An algorithm for constructing a Modified Petri Net equivalent to a given CF-Grammar.

1. Given CF-Grammar convert the Binary normal form (Chomsk normal form) [7]. Denote the resulting a grammar by $G = (N, \Sigma, P, S)$, where N is the set of non-terminal symbols, Σ is the set of terminal symbols, P is the set of rules, S is the initial symbol.

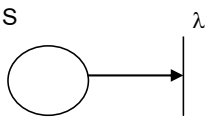
2. Construct a $C = (P_1, P_2, T, I, O)$ MPN follows:
i) If the rule from P has the form $A \to BC$, where $A, B, C \in N$, then build the following fragment of Petri Net:



Let us write the fragment with the the extended input and output functions: $O(A) = \{t_j\}, \sigma\{t_j\} = \lambda, A \in P_1$, where P_1 is the set of basic positions. $I(B) = \{t_j\}, B \in P_1$. $I(\alpha) = \{t_j\}, \alpha \in P_2$, where P_2 is the set of restrained positions. $O(\alpha) = \{t_m\}, \sigma(t_m) = \lambda$.
ii) If the rule from P has the form $A \to a$, where $a \in \Sigma$, then build the following fragment of Petri Net:



$O(A) = \{t_j\}, \sigma(t_j) = a, A \in P_1$.
iii) If the rule from P has the form $S \to e$, where $a \in \Sigma$, then build the following fragment of Petri Net:



$O(S) = t_k, \sigma(t_k) = \lambda, S \in P_1$.

According to the above algorithm may be construct MPN equivalent to a given CF-grammar(fig. 2).

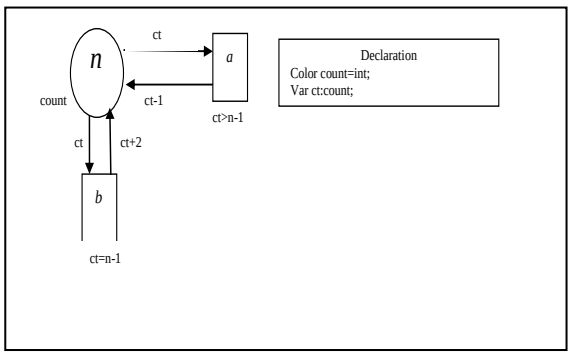


Fig. 1. Modeling $\{\omega\omega^R / \omega \in \Sigma^*, \Sigma = \{a,b\}\}$ CF language by Colored Petri Net.

In fig.2 it is constructed MPN modeled $\{\omega\omega^R / \omega \in \Sigma^*, \Sigma = \{a,b\}\}$ language, compared with fig.1 CPN it has complicated structure.

The fig.1 shows a Colored Petri Net, which generates the $\{\omega\omega^R / \omega \in \Sigma^*, \Sigma = \{a,b\}\}$ 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.

One of the output ranges of $\{\omega\omega^R / \omega \in \Sigma^*, \Sigma = \{a,b\}\}$ language is following: *abaaba* or *babbab*. Fig.1 models the first range, where n is a integer ($n \geq 1$). If transition-names are changed into each other, it will be generated by the second one. To understand types of data which are used in a figure, it is necessary to give a declaration. In the figure introduced a position of count of type and it has an initial value n. In the figure, a transition marked with the symbol **a** that is 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.

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): $ct > n-1$, $ct = (n-1)$, if the logical expression is true, then the transition is allowed, and if false, then the transition is not allowed.

Let $\mathrm{ct} = \mathrm{n}$, then is fired marked with **a** transition, generates the **a** symbol and $\mathrm{ct} = \mathrm{n}\text{-}1$, then is fired marked with **b** transition, are generated by **ab** symbols, in this case position of *count* of type value of token is equal to $\mathrm{n}+1$: $\mathrm{ct} = \mathrm{n}+1$, and twice is fired marked with **a** transition, are generated by **abaa** symbols, when the value of position of *count* of type is equal to $\mathrm{n}-1$: $\mathrm{ct} = \mathrm{n}-1$, then is fired marked with **b** transition, are generated by **abaab** symbols, $\mathrm{ct} = \mathrm{n}+1$, and is fired marked with **a** transition are generated by **abaaba** symbols, in this case position of *count* of type value of token is equal to n, the network comes to the initial state, as necessary, can be repeated the cycle.

Conclusion.

Due to its important properties CPN, are more comfortable for system-modeling mentioned above.

Taking account of the fact that all CF languages are modeled by both MPN and CPN, but CPN are more comfortable for the solution of mentioned problems from the viewpoint of optimization of net-complexity, we can represent the interrelation among the CPNL, MPNL and some classes of Formal languages by means of following diagrams (fig.3).

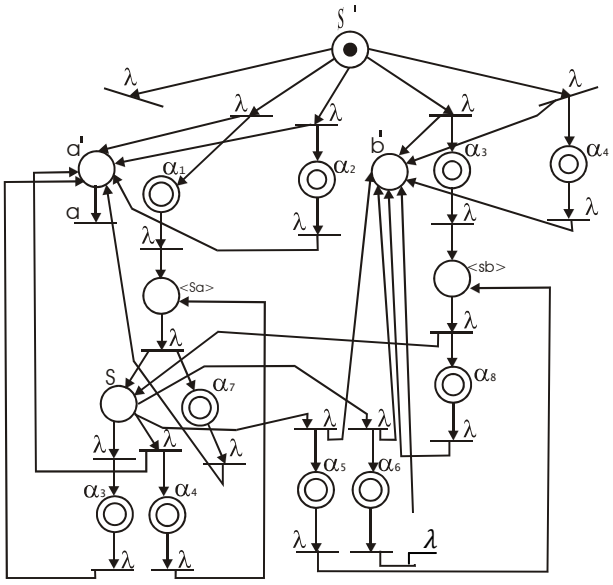


Fig. 2. Modeling $\{\omega\omega^R / \omega \in \Sigma^*, \Sigma = \{a,b\}\}$ CF language by Modified Petri Net.

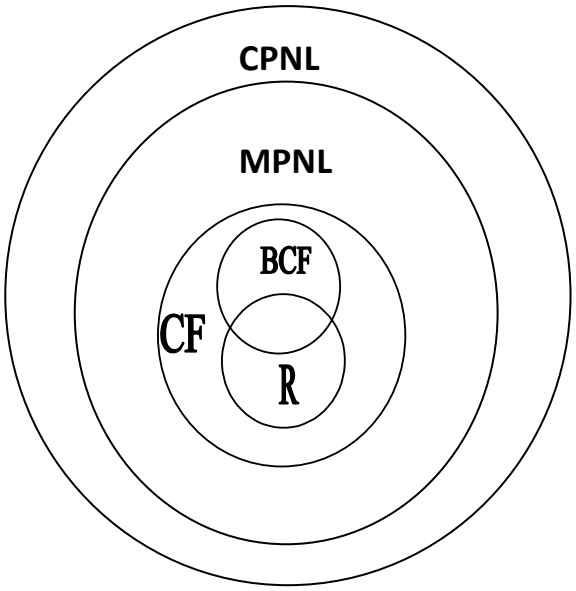


Fig. 3. Interrelation of Languages of Colored Petri Nets, Modified Petri Nets and some classes Formal Languages (Modified Petri Nets Language (MPNL), Colored Petri Nets Language (CPNL), Context-free Language (CF), Bounded Context-free Language (BCF), Regular Language (R)).

REFERENCES

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

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

[3] Г. Р. Петросян, Взаимосвязь языков модифицированных сетей Петри с некоторыми классами формальных языков, *Математические вопросы кибернетики и вычислительной техники*, XXV, стр. 39-44, Ереван 2006.

[4] Г. Р. Петросян, Модифицированные сети Петри, описание поведения с помощью формальных языков, *Математические вопросы кибернетики и вычислительной техники*, XXVI, стр. 48-53, Ереван 2006.

[5] Jensen K. (1992). Coloured Petri Nets: Basic Concepts, *Analysis Methods and Practical Use*. Springer – Verlag, Berlin, Germany.

[6] Jensen K. Coloured Petri Nets: A High – level Language for System Design and Analysis. In: G. Rozenberg (ed.): Advances in Petri Nets 1990, Lecture Notes in Computer Science, v. 483, Springer – Verlag 1991, 342 – 416.

[7] Aho Alfred V., Ullman Jeffrey D. Theory of Parsing, Translation&Compiling. *Prentice Hall* , January 1, v. 1,2, 1973.