

Tree-Token Petri Nets and Derivation Trees

*P. Usha*¹, *K.Thirusangu*² and *Beulah Immanuel*³

¹Department of Mathematics, D. G. Vaishnav College, Chennai – 600106
Chennai, India. E-mail: ushaprab@yahoo.co.in

²Department of Mathematics, S.I.V.E.T. College, Chennai – 601302
Chennai, India. E-mail: kthirusangu@gmail.com

³Department of Mathematics, Women's Christian College, Chennai – 600006
Chennai, India. E-mail: beulah.immanuel@gmail.com

Received 3 October 2014; accepted 21 November 2014

Abstract. String-token Petri net was introduced by labeling the tokens as strings over an alphabet. Languages in regular and linear families which are two basic classes in the Chomsky hierarchy are generated by these Petri nets. An extension of string-token Petri net called Tree-token Petri net (TTPN) was introduced by labeling the tokens with trees. It was proved that the set of derivation trees obtained by any regular language is accepted by a TTPN. In this paper, we prove that the set of derivation trees obtained by any linear language is accepted by a TTPN.

Keywords: Petri net; String-token Petri net; Tree-token Petri net; regular and linear languages; derivation tree.

AMS Mathematics Subject Classification (2010): 68Q45

1. Introduction

Petri net introduced by Carl Adam Petri in 1962 has served as a basic model of systems with concurrency and graphically depicts the structure of a distributed system as a directed bipartite graph. As such, a Petri net has place nodes, transition nodes and directed arcs connecting places with transitions but there are no arcs between places and no arcs between transitions. The place from which an arc enters a transition is called the input place of the transition, the place to which an arc enters from a transition is called the output place of the transition. Places may contain any number of tokens. A distribution of tokens over the places of a net is called a marking. Transitions act on input tokens by a process known as firing. A transition is enabled if it can fire, i.e., there are tokens in every input place of the transition and when a transition fires, tokens are removed from its input places and added at all of the output places of the transition [5].

A coloured Petri net (CPN) has the net structure of a Petri net, and colours are associated with places, transitions and tokens. A transition can fire with respect to each of its colours [3]. A different kind of CPN, called string-token Petri net was introduced in

[1] by labeling the tokens with strings of symbols and the transitions with evolution rules. Firing of a transition removes the token with a string label from the input place and deposits it in the output places of the transition after performing on the string the evolution rule indicated at the transition. This model was examined in [2] for generating regular and linear languages of the Chomsky hierarchy in the study of formal languages.

We defined Tree-token Petri Net (TTPN) [6], an extension of string-token Petri net by labeling the tokens with trees. We obtained the derivation trees generated by TTPN and showed that the set of derivation trees obtained by any regular language is accepted by a TTPN. In this paper, we prove that the set of derivation trees obtained by any linear language is accepted by a TTPN.

2. Basic notions

Definition 2.1. A Multi-set over a non-empty set S , is a function $b \in [S \rightarrow \mathbb{N}]$ where \mathbb{N} is the set of all non negative integers. A multi-set is a set which contains multiple occurrences of the same element. We deal only with finite multi-set, and each multi-set b over set S is represented as a formal sum $b = \sum b(s) s$, where the non negative integer $b(s)$ denotes the number of occurrences of the element s in the multi-set b . The set of all multi-sets over S is denoted by $[S]_{MS}$ or $Bag(S)$.

Definition 2.2. A String-token Petri net (STPN) is a 5-tuple $N = (P, T, C, R(t), M_0)$, where P is a set of places; T is a set of transitions; C is a set of colours and C_{SG} is the set of all strings over this colour set C , that are associated with the tokens; $R(t)$ is the set of evolution rules associated with a transition t ; M_0 , the initial marking, is a function defined on P such that, for $p \in P$, $M_0(p) \in [C_{SG}]_{MS}$. It is further assumed that there are no isolated places/transitions.

We recall the notion of grammar and derivation tree [4].

Definition 2.3. A grammar G is a quadruple $G = (V, U, S, R)$ where V is a finite set of variables, U is a finite set of terminal symbols, $S \in V$ is a special start variable and R is a finite set of production rules. G is said to be right-linear (in normal form) if all of its productions are of the form $A \rightarrow aB$, $A \rightarrow a$, $A \rightarrow \lambda$, where $A, B \in V$ and $a \in U$. G is said to be left-linear in normal form, if all of its productions are of the form $A \rightarrow Ba$, $A \rightarrow a$, $A \rightarrow \lambda$.

A language L is said to be regular if and only if there exists a left-linear (or equivalently a right-linear) grammar G such that $L = L(G)$.

A linear grammar is a grammar in which at most one variable can occur on the right side of any production, without restriction on the position of this variable.

A language L is said to be linear if and only if there exists a linear grammar G such that $L = L(G)$.

Definition 2.4. A grammar G is a quadruple $G = (V, U, S, R)$ where V is a finite set of variables, U is a finite set of terminal symbols, $S \in V$ is a special start variable and R is a finite set of production rules. G is said to be context-free if all productions have the form $A \rightarrow x$, where $A \in V$ and $x \in (V \cup U)^*$.

A language L is said to be context-free if and only if there exist a context-free grammar G such that $L = L(G)$.

Tree-Token Petri Nets and Derivation Trees

A derivation tree is an ordered tree in which nodes are labeled with the left sides of productions and in which the children of a node represent its corresponding right sides.

Definition 2.5. Let $G = (V, U, S, R)$ be a context-free grammar. An ordered tree is a derivation tree for G if and only if it has the following properties,

- (i) The root is labeled S .
- (ii) Every leaf has a label from $U \cup \lambda$.
- (iii) Every interior vertex (a vertex which is not a leaf) has a label from V .
- (iv) If a vertex has label $A \in V$, and its children are labeled (from left to right) $a_1, a_2, a_3, \dots, a_n$ then R must contain a production of the form $A \rightarrow a_1 a_2 a_3 \dots a_n$.
- (v) A leaf labeled λ has no siblings, that is, a vertex with a child labeled λ can have no other children.

Definition 2.6. A Tree-token Petri net (TTPN) is a 6-tuple $N = (P, T, C, F, R(t), M_0)$, where P is a set of places; T is a set of transitions; C is a set of colours and C_{TR} is the set of all trees associated with the colour set C (That is trees with nodes labeled with colours from the set C); $F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs; $R(t)$ is a set of evolution rules associated with each transition t of T ; M_0 , the initial marking, is a function defined on P such that, for $p \in P$, $M_0(p) \in [C_{TR}]_{MS}$. It is further assumed that there are no isolated places/transitions.

Definition 2.7. Let $V = \{A, B, C, \dots\}$ = Set of Non-terminals. $U = \{a, b, c, \dots\}$ = Set of terminals, where $V, U \subseteq W$.

An evolution rule over W_{TR} , where W is an alphabet, is one of the following,

- Identity, which keeps the tree unaltered
- $A \rightarrow s(X, Y)$ replaces the node labeled A by a tree, s denoting a split of the node into two nodes as shown in Figure 1

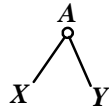


Figure 1:

$A \rightarrow l(X)$ replaces the node labeled A by a edge, l denoting the leaf obtained by this rule as shown in Figure 2 where $A \in V$ and $X, Y \in W$.



Figure 2:

Repeated use of these rules generates trees as shown below:

Example 2.1. The evolution rules $S \rightarrow s(A, B)$, $A \rightarrow l(x)$, $B \rightarrow s(x, a)$ generate the tree represented in Figure 3.

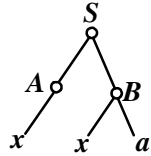


Figure 3:

Example 2.2. The rules $S \rightarrow s(A, c)$, $A \rightarrow s(X, b)$, $X \rightarrow s(x, a)$, generate the tree shown in Figure 4.

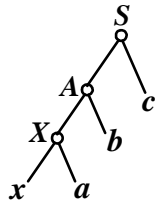


Figure 4:

We construct TTPNs that generate trees corresponding to above evolutionary rules as seen in the following examples:

Example 2.3. When t_1 fires, the TTPN N_1 generates a tree similar to the one in Figure 1.

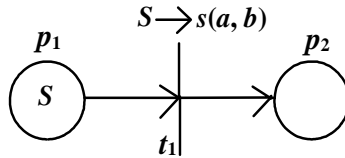



Figure 5: TTPN N_1

In N_1 , when t_1 fires token labeled S is removed from p_1 the evolution rule $S \rightarrow s(a, b)$ is

applied on it and the token labeled with the tree  is put in the place p_2 .

Example 2.4. The following TTPN N_2 generates a tree as in Figure 3.

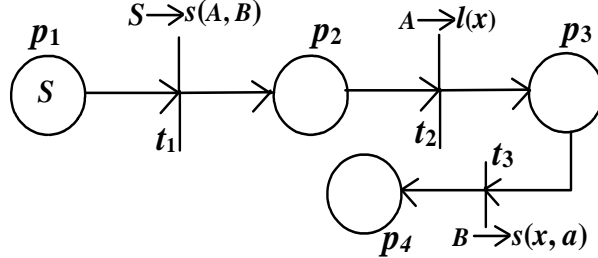


Figure 6: TTPN N_2

3. Tree-token petri net languages

We defined a new class of Petri nets where trees are associated with tokens [6]. Firing of a transition removes the token with a tree label from the input place and deposits it in the output places of the transition after performing on the tree the evolution rule indicated at the transition. This model is examined for generating derivation trees corresponding to regular and linear languages.

Definition 3.1. A L -type TTPN language L is defined as $L = \{C_{TR} / C_{TR} \in M(p) \in M \text{ is a reachable marking of } N, p \in P_F\}$, if there exists a TTPN $N = (P, T, C, F, R(t), M_0)$, and P_F is a set of final places.

In other words, L -type TTPN language is defined in terms of trees corresponding to some reachable markings in a specified set of final places P_F , where L is a set of all trees generated in the places for all reachable markings.

Theorem 3.1. Let $TR(G)$ be the set of derivation trees corresponding to a regular language generated by left-linear grammar $G = (V, U, S, R)$ then there exists a L -type language L accepted by TTPN, N such that $TR(G) = L$.

Proof: Let $G = (V, U, S, R)$, $V = \{A_1, A_2, A_3, \dots, A_n\}$, $U = \{a_1, a_2, a_3, \dots, a_n\}$, $S \in V$ and R is set of productions of the form $A_i \rightarrow A_j a_k$, $A_j \rightarrow a_k$. Construct a TTPN, $N = (P, T, C, F, R(t), M_0)$ as follows. P is the set of places labeled by the non-terminals $A_1, A_2, A_3, \dots, A_n$ and a final place p_f . T is the set of transitions. Let $C = V \cup U$ be the colour set and tokens in places be trees associated with C . Initially a tree with a single node labeled S is placed in the place with label S and all other places are empty. For every rule $A_i \rightarrow A_j a_k$, introduce a transition labeled $A_i \rightarrow s(A_j, a_k)$ with A_i as the input place and A_j as the output place. For every rule $A_j \rightarrow a_k$ introduce a transition labeled as $A_j \rightarrow l(a_k)$ with A_j as the input place and p_f as the output place.

By this construction we obtain a TTPN such that initially only the transitions with input place labeled S are enabled. On firing any one of these transitions, say the transition with evolution rule $S \rightarrow s(A_i, a_k)$, the rule is applied on the token and is deposited in the output place labeled A_i . Then all the transitions with input place A_i are enabled. On firing any one of these transitions, say transition with the rule $A_i \rightarrow s(A_j, a_k)$, the rule is applied on the token and deposited in the place A_j . This process is continued till the token reaches a place with input place A_j with an output transition labeled with the rule $A_j \rightarrow l(a_k)$. When this transition fires, the rule $A_j \rightarrow l(a_k)$ is applied and the token is deposited in the final place p_f . For any string accepted by the grammar G , we obtain a derivation tree C_{TR}

$\in M(p_f) \in M$, M is a reachable marking of N . It is clear that these are the only trees are accepted by the language L . Thus the set of derivation trees $TR(G) = L$.

Example 3.1. Let $G = (V, U, S, R)$ where $V = \{S, A\}$, is the set of non-terminals and $U = \{a, b, c\}$, is the set of terminals and R is a set of productions of the form $S \rightarrow Aa, A \rightarrow Ab, A \rightarrow c$.

Construct a TTPN $N_3 = (P, T, C, F, R(t), M_0)$ as specified by theorem 3.1, P is the set of places labeled by the non-terminals S, A and a final place p_f . Let $C = V \cup U$ be the colour set and tree tokens in places be trees over C . Initially a tree with a single node labeled S is placed in the place with label S and all other places are empty. For the production rule $S \rightarrow Aa$, initial place labeled as S is the input and A is the output place for the transition t_1 , labeled as $S \rightarrow s(A, a)$. For the production rule $A \rightarrow Ab$, the input and output place is A for the transition t_2 , labeled as $A \rightarrow s(A, b)$, (Here we can fire t_2 n times). For the rule $A \rightarrow c$ the input and output places are A and p_f respectively for the transition t_3 labeled as $A \rightarrow l(c)$. On firing the transitions in TTPN N_3 , generates the derivation tree as in Figure 8, corresponding to a regular language generated by left-linear grammar G is accepted by this TTPN such that $TR(G) = L$.

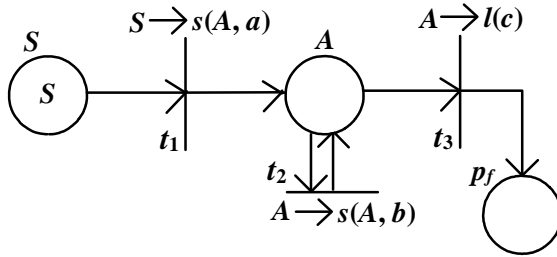


Figure 7: TTPN N_3

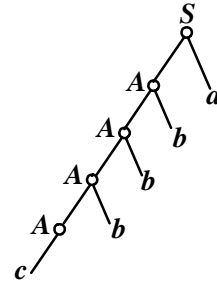


Figure 8: when t_2 fires 3 times

Theorem 3.2. Let $TR(G)$ be the set of derivation trees corresponding to linear language generated by linear grammar $G = (V, U, S, R)$ then there exists a L -type language L accepted by TTPN, N such that $TR(G) = L$.

Proof: Let $G = (V, U, S, R)$, $V = \{A_1, A_2, A_3, \dots, A_n\}$, $U = \{a_1, a_2, a_3, \dots, a_n\}$, $S \in V$ and R is set of productions of the form $A_i \rightarrow a_r A_j$, $A_j \rightarrow A_i a_k$, $A_i \rightarrow a_k / \lambda$. Construct a TTPN, $N = (P, T, C, F, R(t), M_0)$ as follows. P is the set of places labeled by the non-terminals $A_1, A_2, A_3, \dots, A_n$ and a final place p_f . T is the set of transitions. Let $C = V \cup U$ be the colour set and tokens in places be trees associated with C . Initially a tree with a single node labeled S is placed in the place with label S and all other places are empty. For every production rule $A_i \rightarrow a_r A_j$ introduce a transition labeled as $A_i \rightarrow s(a_r, A_j)$ with A_i as the input place and A_j as the output place. For the rule $A_j \rightarrow A_i a_k$ introduce a transition labeled as $A_j \rightarrow s(A_i, a_k)$ with A_j as the input place and A_i as the output place. For the rule $A_i \rightarrow a_k / \lambda$ introduce a transition with input place A_i and the output place p_f labeled as $A_i \rightarrow l(a_k) / \lambda$.

By this construction we obtain a TTPN such that initially only the transitions with input place labeled S are enabled. On firing any one of these transitions, say the transition

Tree-Token Petri Nets and Derivation Trees

with evolution rule $S \rightarrow l(A_i)$, the rule is applied on the token and is deposited in the output place labeled A_i . Then all the transitions with input place A_i are enabled.

On firing any one of these transitions, say transition with the rule $A_i \rightarrow s(a_r, A_j)$, the rule is applied on the token and is deposited in the output place A_j . Then all the transitions with input place A_j are enabled. On firing any one of these transitions, say transition with the rule $A_j \rightarrow s(A_i, a_k)$, the rule is applied on the token and is deposited in the output place A_i . The process is continued till the token reaches a place with input place A_i with an output transition labeled with the rule $A_i \rightarrow l(a_k)/\lambda$. When this transition fires, the rule $A_i \rightarrow l(a_k)/\lambda$ is applied and the token is deposited in the final place p_f .

For any string accepted by the grammar G , we obtain a derivation tree $C_{TR} \in M(p_f) \in M$, M is a reachable marking of N . It is clear that these are the only trees are accepted by the language L . Thus the set of derivation trees $TR(G) = L$.

Example 3.2. Let $G = (V, U, S, R)$ where $V = \{S, A, B\}$, is the set of non-terminals and $U = \{a, b\}$, is the set of terminals and R is a set of productions of the form $S \rightarrow A, A \rightarrow aB/\lambda, B \rightarrow Ab$.

Construct a TTPN $N_4 = (P, T, C, F, R(t), M_0)$ as specified by theorem 3.2, P is the set of places labeled by the non-terminals S, A, B and a final place p_f . Let $C = V \cup U$ be the colour set and tree tokens in places be trees over C . Initially a tree with a single node labeled S is placed in the place with label S and all other places are empty. For the production rule $S \rightarrow A$, S as the input and A as the output place for the transition t_1 , labeled as $S \rightarrow l(A)$. For the rule $A \rightarrow aB$, A as the input and B as the output place for the transition t_2 , labeled as $A \rightarrow s(a, B)$. For the rule $B \rightarrow Ab$, the input and output places are B and A respectively for the transition t_3 , labeled as $B \rightarrow s(A, b)$ (Here we can fire t_2 and t_3 n times). The transition t_4 , labeled as $A \rightarrow \lambda$ with A and p_f as the input and output places respectively. On firing the transitions in TTPN N_4 , generates the derivation tree as in Figure 10(a) and (b), corresponding to a linear language generated by linear grammar G is accepted by this TTPN such that $TR(G) = L$.

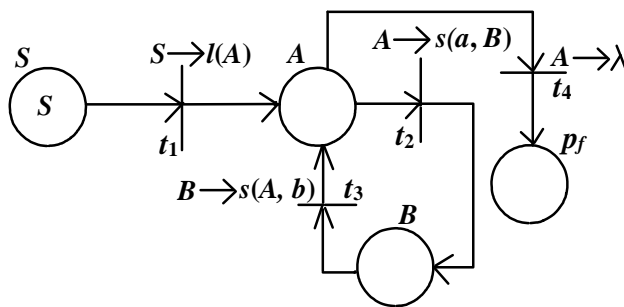


Figure 9: TTPN N_4

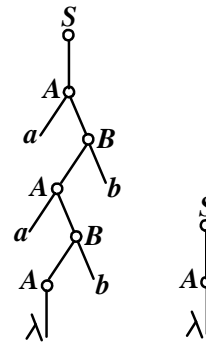


Figure 10: (a) when t_2 and t_3 fire 2 times.
(b) when t_1 and t_4 only fire.

4. Conclusion

We have examined the languages generated by TTPNs and proved that TTPNs can be constructed to generate the set of derivation trees corresponding to two classes of

P.Usha, K.Thirusangu and Beulah Immanuel

languages of the Chomsky hierarchy namely regular and linear languages. Extending this concept to other classes of languages is considered for future work.

REFERENCES

1. B.Immanuel, K.Rangarajan and K.G.Subramanian, String-token petri nets, Proceedings of the *European Conference on Artificial Intelligence*, One-day Workshop on Symbolic Networks, at Valencia, Spain, 2004.
2. B. Immanuel, K.G. Subramanian and A. Roslin Sagaya Mary, Petri nets with String-labeled Tokens, Proceedings of the *2nd International Conference on Cybernetics and Information Technologies Systems and Applications*, Orlando, Florida, USA, (2005) 245-249.
3. K.Jensen, Coloured petri nets, *Lecture Notes in Computer Science*, 254 (1987) 248-299.
4. P.Linz, *An introduction to Formal Languages and Automata*, Jones and Barletts Publishers, (2004).
5. J.I.Peterson, *Petri Net Theory and The Modeling of systems*, Prentice Hall, Englewood Cliffs, N.J., (1981)
6. P.Usha, K.Thirusangu and B.Immanuel, Tree token Petri net, Proceedings of the *International conference on Mathematics – A Global Scenario*, D. G. Vaishnav College, Chennai, India, (2012) 13-14.