South East Asian J. of Math. & Math. Sci. **ISSN : 0972-7752 (Print)**
Vol. 14, No. 3 (2018), pp. 133-142

GRAPH-TOKEN PETRI NETS

Enitha Dorothy G and Beulah Immanuel

UG Department of Mathematics,
Women's Christian College, Chennai, INDIA
E-mail: enithadorothy@hotmail.com, beulah.immanuel@gmail.com

(Received: May 8, 2018)

Abstract: Graph-token Petri net ($GTPN$), an extension of Tree-token Petri net ($TTPN$), is introduced by labelling the tokens with graphs. A study is done on the languages generated by $GTPN$. Some subclasses of graphs are generated by this Petri net model.

Keywords and Phrases: Petri net language, cycle, complete graph, bi-partite graph.

2010 Mathematics Subject Classification: 68Q45.

1. Introduction

Petri net is a particular kind of directed bi-partite graph consisting of two types of nodes called places and transitions connected by directed arcs [4]. Tree-token Petri net ($TTPN$) was introduced with tokens as trees and evolution rules at transitions [5]. Tree- token Petri net languages were studied and it was proved that the sets of derivation trees of regular, linear and context free grammars of Chomsky hierarchy are accepted by $TTPNs$ [2], [6]. Subclasses of trees, namely caterpillars and lobsters are generated by these Petri nets [7].

Motivated by the generating capacity of Petri net models, Graph-token Petri net ($GTPN$), an extension of $TTPN$ is defined. New evolution rules at transitions are introduced to perform the extended operations on graph tokens. In a $GTPN$, an enabled transition removes the tokens labelled by graphs from the input places and deposits it in the output places performing the evolution rule indicated at the transition. For the study of languages generated by $GTPN$, the net is reduced to a standard form. It is proved that the $GTPN$ languages are closed under finite union. Some subclasses of graphs namely, cycles, complete graphs and complete bi-partite graphs are generated by these Petri nets.

In this section, the definitions of the well-known types of graphs [1], [3] that are needed for our study are presented. Graph-token Petri net, a new type of Petri net, by labelling tokens with graphs is defined. The set of evolution rules at the transitions of a $TTPN$ [7], are revised by introducing new rules.

Definition 1.1. *A graph is called simple if it has no loops and no parallel edges.*

Definition 1.2. *A simple graph with n- vertices ($n \geq 3$) and n edges is called a cycle graph if all its edges form a cycle of length n, denoted by C_n.*

Definition 1.3. *A simple graph G is said to be complete if every pair of distinct vertices of G are adjacent in G. A complete graph each on a set of n vertices is denoted by K_n.*

Definition 1.4. *A graph G is bi-partite if its vertex set can be partitioned into two empty non- empty subsets X and Y such that each edge of G has one end in X and the other in Y. The partition $V = X \cup Y$ is called a partition of G.*

Definition 1.5. *A complete bi-partite graph is a simple bi-partite graph with bi-partition $V = X \cup Y$ in which every vertex in X is joined to every vertex of Y. If X has m vertices and Y has n vertices such a graph is denoted by $K_{m,n}$.*

Definition 1.6. *A multi-set over a non-empty set S, is a function b: $S \to N$, where 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 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 1.7. *A graph-token Petri net ($GTPN$) is a 7-tuple $N = (P, T, C, A, R(t), W(a), M_0)$, where P is a set of places; T is a set of transitions; C is a set of colours and C_{GR} is the set of all graphs associated with the colour set C (That is graphs with vertices labelled with colours from the set C); $A \subseteq (P \times T) \cup (T \times P)$ is a set of arcs; $R(t)$ is a set of evolution rules associated with each transition t of T; $W(a)$ is the weight associated with arc $a \in A$; M_0 the initial marking, is a function defined on P such that, for $p \in P$, $M_0(p) \in [C_{GR}]_{MS}$. If no weight is mentioned it is assumed that the weight is 1. It is further assumed that there are no isolated places/transitions.*

Definition 1.8. *Let $C = \{a, x, y, z, x_1, x_2, x_3, ...\}$. A graph evolution rule over C_{GR}, where C is a colour set, is one of the following:*
1. Identity, which keeps the graph unaltered.
2. $a \to l(x_1, x_2, x_3, ..., x_n)$ replaces the vertex labelled 'a' by a tree with vertices

labelled a, x_1, x_2, x_3,$\cdots$, x_n *and edges* ax_1, ax_2, ax_3,$\cdots$, ax_n *as shown in Fig.*1.

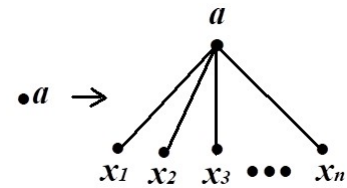

Figure 1: Evolution rule 2

3. (x, y) *introduces edge between the vertices labelled* x *and* y.
4. $x \to y$ *replaces the label of the vertex* x *by* y.
5. $x \to \lambda$ *deletes the label of the vertex labelled* x, *where* x, y, $\lambda \in C$ *and* λ *is an empty colour.*

Example 1.1. *The evolution rules* $x \to l(y)$, $x \to z$, $y \to l(x, y)$, $l(x) \to l(x, x)$ *generates trees as shown in Fig.*2.

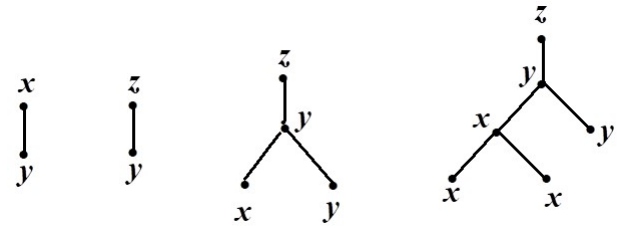

Figure 2: Trees generated using evolution rules

Example 1.2. *The evolution rules* $x \to l(y)$, $l(y) \to l(y)$ *and the new simple evolution rule* $(x, l(y))$ *generates graphs as shown in Fig.*3.

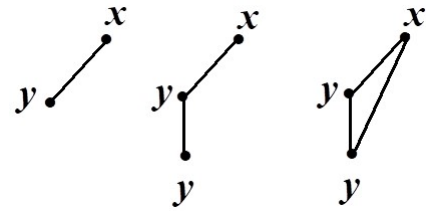

Figure 3: A cycle generated using evolution rules

2. Graph-token Petri net languages

The language generated by a graph-token Petri net is the set of graphs obtained as reachable markings in a selected set of places in the net. In this section we construct standard form for $GTPN$ generating a language. We use this result to prove that the set of $GTPN$ languages is closed under finite union.

Definition 2.1. *A L-type GTPN language L is defined as* $L = \{C_{GR}/C_{GR} \in M_(p) \in M$ *is a reachable marking of* N, $p \in P_F\}$, *if there exists a graph-token Petri net* $N = (P, T, C, A, R(t), W(a), M_0)$, *and* P_F *is a set of final places.*

In other words, L-type $GTPN$ language is defined in terms of graphs corresponding to some reachable markings in a specified set of final places P_F of a net N. The set of such reachable markings is called the language generated by N, denoted by $L(N)$.

Definition 2.2. *If L is a language generated by the* $GTPN$ $N = (P, T, C, A, R(t), W(a), M_0)$ *such that it has exactly one final place, then N is said to be in standard form.*

Definition 2.3. *If* L_1 *and* L_2 *are languages generated by* $GTPNN_1$ *and* N_2 *respectively then the union of two languages is defined by* $L_1 \cup L_2 = \{x/x \in L_1$ *or* $x \in L_2\}$

Theorem 2.1. *Every* $GTPN$ *can be reduced to a standard form.*

Proof. Let L be a language generated by $GTPN$, $N = (P, T, C, A, R(t), W(a), M_0)$, and P_F be set of final places.
If P_F has exactly one place then N is in standard form, and there is nothing to prove. Suppose P_F has more than one place, let $P_F = \{p_{fi} : i = 1, 2, , k\}$. Construct $GTPN$ N_1 as shown in $Fig.4$. In addition to the places in net N, introduce a new place p_f. Introduce new transitions t_{fi} for $p_{fi} \in P_F$, $i = 1, 2, , k$ as the input place respectively and p_f as outplace, labelled with identity rules.

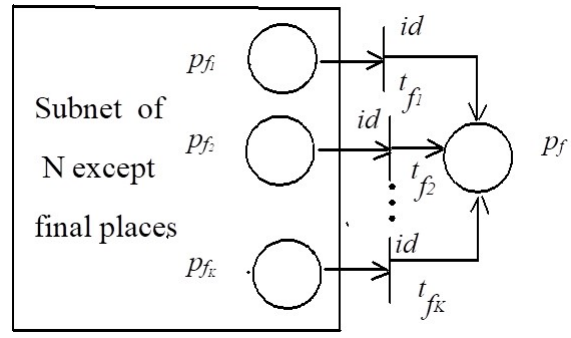

Figure 4: $GTPNN_1$ **standard form of** N

Considering the final place as p_f in the $GTPN$ N_1, $L(N_1) = L(N)$. Hence the theorem.

Theorem 2.2. *If* L_1 *and* L_2 *are* $GTPN$ *languages then* $L_1 \cup L_2$ *is also a* $GTPN$ *language.*

Proof. Let N_1 and N_2 be $GTPNs$ in standard form generating the language L_1

and L_2 respectively.
Let $N_1 = (P_1, T_1, C_1, A_1, R_1(t), W_1(a), M_{01})$ and $N_2 = (P_2, T_2, C_2, A_2, R_2(t), W_2(a), M_{02})$. Combining N_1 and N_2, construct a $GTPN$ N' by introducing a new final place p'_f and new transitions t'_{f1}, t'_{f2} with identity rules having p_{f1} , p_{f2} as the input place respectively and p'_f as output place, as in Fig 5.
$GTPN$ $N' = (P_1 \cup P_2 \cup \{p'_f\}, T_1 \cup T_2 \cup \{t'_{f1}, t'_{f2}\}, C_1 \cup C_2$, $A_1 \cup A_2 \cup \{(p_{f1}, t'_{f1}), (p_{f2}, t'_{f2}),$ $(t'_{f1}, p'_f), (t'_{f2}, p'_f)\}$, $R_1(t) \cup R_2(t)$, $M'_0)$. The initial marking M'_0 has tokens as in N_1 and N_2 and p'_f is empty.

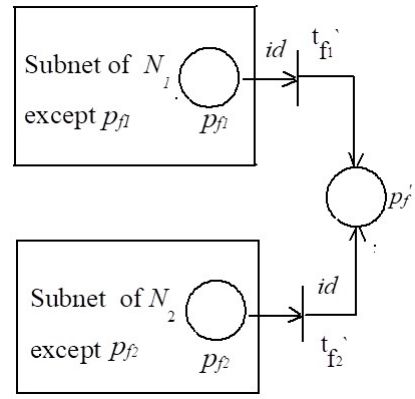

Figure 5: $GTPN N'$ **generates** $L_1 \cup L_2$

It is seen that the set of all reachable markings p_{f_1} in N_1 and p_{f_2} in N_2 are same as the reachable markings of p'_f in N' so that $L(N') = L_1 \cup L_2$. Hence the theorem.

Corollary. *The family of $GTPN$ languages is closed under finite union.*

Proof. Let N_1, N_2,$\cdots$, N_n be the $GTPNs$ in standard forms generating L_1, L_2,$\cdots$, L_n respectively. Let p_i , $i = 1, 2, \cdots, n$ be the final places of N_1, N_2,$\cdots$, N_n respectively. As in the case of the theorem, construct N by introducing a single final place p and transitions t_i with identity rules having p_i as input place and p as output place. The $GTPN$ N generates $\cup_{i=1}^{n} L_i$. Hence the family of $GTPN$ languages is closed under union.

3. Subclasses of graphs generated by $GTPNs$

In this section, we apply graph-token Petri nets to generate some types of graphs. We show that the subclasses of graphs, the cycle graphs C_n, the complete graphs K_n and the complete bi-partite graphs $K_{m,n}$ for arbitrary m, n are obtained as languages accepted by $GTPNs$.

Theorem 3.1. *Let G_1 be the set of all cycle graphs. Then there exists a graph-token Petri net N_1 such that the language generated by N_1, $L(N_1) = G_1$.*

Proof. $G_1 = \{C_n : C_n$ is the cycle of length n, $n \geq 3\}$. Construct a $GTPN$ $N_1 = (P, T, C, A, R(t), W(a), M_0)$ as follows:
$P = \{p_1, p_2, p_3, p_4\}$is the set of places;
$C = \{x, y, \lambda\}$is the colour set;
$M_0 = (\bullet x, 0, 0, 0)$ is the initial marking with a vertex labelled x as token in the place p_1 and all other places are empty. The set of transitions T and set of arcs A and the evolution rules $R(t)$, are as shown in $Fig.6$. Let $P_F = \{p_4\}$.

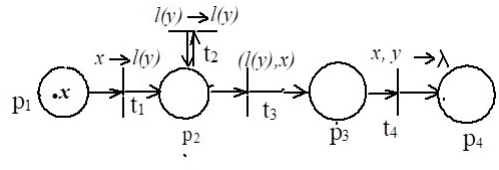

Figure 6: $GTPNN_1$ **generates cycles**

On firing the enabled transition t_1 the token $\bullet x$ is removed from p_1 and the edge shown in $Fig.6$ is deposited in p_2. By firing t_2 n times a path of length $n+1$ is obtained as a reachable marking in p_2, for $n = 1, 2, 3, \cdots$ as shown in $Fig.7$.

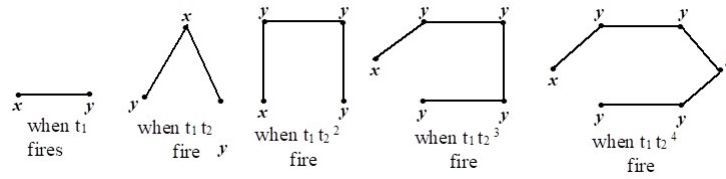

Figure 7: Markings at place p_2

Then firing t_3, a cycle of length $n+2$ is obtained as a marking in p_3, $n = 1, 2, 3, \cdots$. When the transition t_4 is fired unlabelled cycle graph C_n is deposited in the final state p_4 for $n = 3, 4, \cdots$ as shown in $Fig.8$.

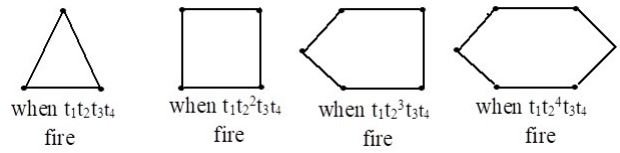

Figure 8: Markings at final place p_4

Hence $L(N_1) = G_1$.

Theorem 3.2. *If G_2 is the set of all complete graphs, then there exists a $GTPN$ N_2 such that the language generated by N_2, $L(N_2) = G_2$.*

Proof. Construct a $GTPN$ $N_2 = (P, T, C, A, R(t), W(a), M_0)$ as follows:
$P = \{p_1, p_2, p_3, p_4, p_5, p_6\}$the set of places;
$M_0 = (\bullet x, 0, 0, \bullet y, 0, 0)$ a node labelled x in the place p_1 and another labelled y at p_4. T, A, $R(t)$ and $W(a)$ are as shown in $Fig.9$. $P_F = \{p_6\}$.

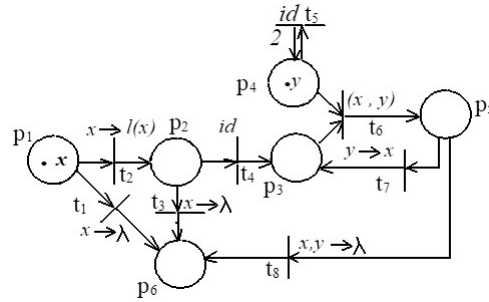

Figure 9: $GTPNN_2$ **generates complete graphs**

Initially t_1 and t_2 are enabled. Firing t_1, the graph K_1 is obtained as a marking at p_6. Instead if t_2 is fired, a path of length 1 is deposited in p_2. Then t_3 and t_4 are enabled. By firing t_3 the path of length 1 without labels is deposited at the final place p_6. When t_4 fires the same path with labels is deposited in p_3. Since the tokens are present in both places p_3 and p_4, transition t_6 can be fired and an edge is introduced between x and y. Fire t_5 before t_6 each time so that the sequence of firings can continue. On firing t_6, the vertex labelled y is joined to every vertex x in the graph token at the place p_3. On firing t_7 the vertex labelled y is replaced by the label x.
Hence the markings at p_3 are all complete graphs with vertices labelled x and the markings at p_5 are complete graphs with one vertex labelled y and all others labelled x. After firing t_8, the markings at the final place p_6 are unlabelled complete graphs. Hence $L(N_2) = G_2$.

Example 3.1. *The complete graphs K_1, K_2, K_3 and K_4 are obtained as reachable markings in N_2 as shown in $Fig.10$.*

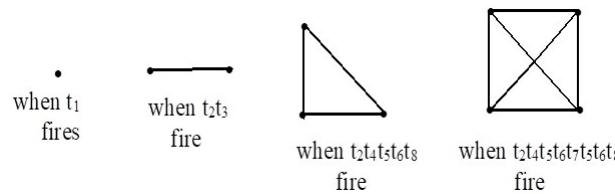

Figure 10: Markings at final place p_6

Theorem 3.3. *If G_3 is the set of all complete bi-partite graphs then there exists a $GTPN$ N_3 such that the language generated by N_3, $L(N_3) = G_3$.*

Proof. Construct a $TTPN$ $N_3 = (P, T, C, A, R(t), W(a), M_0)$ as follows:
$P = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9\}$, the set of places;
$C = \{x, y, a, b, \lambda\}$, the colour set;
$M_0 = (\bullet a, 0, 0, \bullet b, 0, 0, \bullet y, 0, 0)$, vertices labelled a, b and y is placed in the places p_1, p_4 and p_7. T, A, $R(t)$ and $W(a)$ are constructed as shown in $Fig.11$. $P_F = \{p_9\}$.

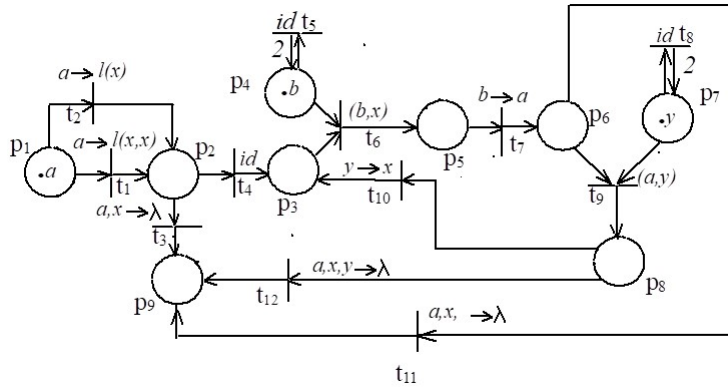

Figure 11: $GTPNN_3$ **generates complete bi-partite graphs**

On firing the sequence t_2t_3, $K_{1,1}$ is obtained as the marking at p_9. Instead if t_1 is fired, a tree with vertex a and two leaves with label x is deposited in place p_2. The transitions t_3 and t_4 are enabled. If t_3 fires then the marking $M(p_9)$ is $K_{1,2}$. If not, on firing t_4 the graph token at p_2 is deposited in p_3. The transitions t_6 and t_9 are enabled only if there are tokens in pair of places (p_3, p_4)and(p_6, p_7) respectively. Firing t_5 and t_8 repeatedly multi-sets of vertices with labels $\bullet b$, $\bullet y$ are obtained as markings in p_4 and p_7 respectively. Since b is replaced by a and y is replaced by x, at t_7 and t_{10} respectively, any arbitrary number of vertices in the bi-partition is obtained. The transitions t_6 and t_9 ensure the join of any two pairs of vertices in the bi-partition.

Thus we obtain all isomorphic complete bi-partite graphs with labels at p_6 and p_8.The graphs $K_{m,n}$ without labels are obtained as markings in the final state p_9 on firing t_{11} and t_{12}. Hence $L(N_3) = G_3$

Example 3.2. *The complete bi-partite graphs $K_{1,2}$, $K_{1,1}$, $K_{2,2}$, $K_{3,2}$, $K_{2,3}$ are obtained as markings in p_9 as shown in Fig.12.*

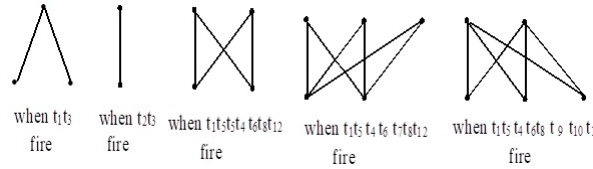

Figure 12: Markings at final place p_9

3. Conclusion

Graph-token Petri net ($GTPN$), a new class of Petri nets, is introduced. Some closure properties of the $GTPN$ languages are explored. It is found that this model is useful to generate some subclasses of graphs. This is illustrated by examples. It is of interest to note that these graphs are generated by applying simple evolution rules at the transitions of the $GTPN$. The properties of $GTPN$ languages and generation of other classes of graphs is considered for future study.

References

[1] R. Balakrishnan,K. Ranganathan, *A Textbook of Graph Theory (second edition),* Springer Science and Business Media, (2012).

[2] G. Enitha Dorothy and B. Immanuel, Tree-token Petri net for Derivation Trees of Context-Free Grammars, *International Journal of Pure and Applied Mathematics*, Volume 109 No.8(2016), 33-40.

[3] John Clark, Derek Allan Holton, *A First Look at Graph Theory,* World Scientific, (1991)330.

[4] J. I. Peterson, *Petri net Theory and The Modeling of systems,* Prentice Hall, Englewood Cliffs, N.J., (1981).

[5] 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), 125-127.

[6] P. Usha, K. Thirusangu and B. Immanuel, Tree-token Petri nets and derivation trees, *Annals of Pure and Applied Mathematics*, Volume 8 No.2(2014), 219-226.

[7] P. Usha, B. Immanuel and Enitha Dorothy, Caterpillars and Lobsters generated by Tree-token Petri nets,*Proceedings of the Centenary International Conference on Viable Synergies in Mathematical and Natural Sciences,* Women's Christian College, Chennai, India, (2015), 147-154.