Parallel Splicing on ISO-Array Token Petri Nets

D.K. Shirley Gloria[1], K. Nirmala[2]

[1]Department of Mathematics

Dr. Ambedkar Govt. Arts College

Vyasarpadi, Chennai - 600 039, India

[2]Department of Mathematics

SRM University, Kattankulathur - 603203, India

Abstract

Iso-Array Token Petri net is introduced. Every iso-array splicing system can be generated by iso-array token petri net using parallel splicing. Also, some of the closure properties of iso-array token petri nets are studied.

Keywords: Iso-Array Token Petri net (IATPN), Iso-Array Splicing System (IASS), iso-arrays.

Mathematics Subject Classification: 68Q45.

1 Introduction

Petri net has its origin in Carl Adam Petri's dissertation submitted in the year 1962. It is a directed weighted bipartite graph consists of two nodes namely, places and transitions where arcs are either from a place to a transition or from a transition to a place. The dynamic behavior of this Petri net is simulated by the token game. 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., if 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 [7].

One of the modifications made on Petri net is colored Petri nets (CPN). In CPN, colors are associated with places, transitions and tokens. CPN is a modeling language developed for systems in which communication, synchronization and resource sharing play an important role. Also, in CPN, colours are associated with places, transitions and tokens. A transition can fire with respect to each of its colours.

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 token with a string label from the input places and deposits it in the output places of the transition after performing on the strings, the evolution rule indicated at the transition.

An extension of the string-token Petri net called array-token Petri net is formulated by labeling tokens by arrays and is used to generate picture languages. It has been shown in [2] that the class of languages generated by array-token Petri nets intersects certain classes of picture languages generated by 2D matrix grammars. It is proved that regular and linear families of languages that are generated by 2D grammars are generated by array-token Petri nets [3].

Splicing systems introduced on biological considerations [4] to model certain recombinant behaviour of DNA molecules are of current interest and study [5]. The splicing systems make use of a new operation called splicing on strings of symbols [8]. This splicing operation has been recently applied to rectangular arrays [6].

In [9], parallel splicing on iso arrays is introduced. Certain properties are studied.

Using [9], we introduce a new Petri net, called iso array token petri net. Some of its closure properties are obtained.

2 Basic Definitions

Definition 2.1. *Let* $\Sigma_I = \left\{ \begin{array}{c} S_1 \\ S_2 \end{array} \!\!\triangle\!\!\! \begin{array}{c} \\ \mathrm{X} \end{array} \!\!\! \begin{array}{c} S_3 \\ \end{array}, S_3 \begin{array}{c} S_2 \\ \end{array} \!\!\triangledown\!\!\! \begin{array}{c} \\ \mathrm{Y} \end{array} \!\!\! \begin{array}{c} S_1 \\ \end{array}, \begin{array}{c} S_1 \\ S_3 \end{array} \!\!\triangleleft\!\!\! \begin{array}{c} \\ \mathrm{Z} \end{array} \!\!\! \begin{array}{c} S_2 \\ \end{array}, S_2 \begin{array}{c} S_3 \\ \end{array} \!\!\triangleright\!\!\! \begin{array}{c} \\ \mathrm{W} \end{array} \!\!\! \begin{array}{c} S_1 \\ \end{array} \right\}$. *The sides of each tile in* Σ_I *are of length* $\frac{1}{\sqrt{2}}, 1, \frac{1}{\sqrt{2}}$.

An iso-array is an arrangement of isosceles right angled triangles of tiles from the set Σ_I.

Note 1. Iso-arrays of same-size can be catenated using the following catenation operations. There are four types of catenations of iso-arrays.
(i) Horizontal ($\ominus$) (ii) Vertical ($\oslash$) (iii) Right ($\oslash$) and (iv) Left ($\oslash$) catenations. A catenation can be defined between any two gluable iso-arrays of same size [9].

Note 2. Let L_1 and L_2 be two iso-picture languages over Σ_I.
1) Horizontal Catenation:
$L_1 \ominus L_2 = \{X \ominus Y \ / \ X \in L_1$ and $Y \in L_2$ where the sizes of X and Y are (n_1, m) and (n_2, m) respectively and they can be catenated only if they have gluable edges$\}$.
2) In a similar way Vertical Catenation, Right Catenation and Left Catenation are defined.

Notation. The family of iso-picture languages generated by H iso-array systems is denoted by $L(IASS)$.

Definition 2.2. *Evolution rules which are used in IATPN are as follows.* $A \to A$ *is the identity rule that keeps the iso array* A *unaltered.*

a) $\Lambda \to A\ominus$ *is the horizontal insertion rule,*

b) $\Lambda \to A\oslash$ *is the vertical insertion rule,*

$\Lambda \to A\oslash$ *is the right insertion rule,*

$\Lambda \to A\oslash$ *is the left insertion rule,*

c) $A\ominus \to \Lambda$ *is the horizontal deletion rule,*

$A\oslash \to \Lambda$ *is the vertical deletion rule,*

$A\oslash \to \Lambda$ *is the right deletion rule,*

$A\oslash \to \Lambda$ *is the left deletion rule,*

d) $A\ominus \to B\ominus$ *is the horizontal substitution rule provided they must have gluable sides,*

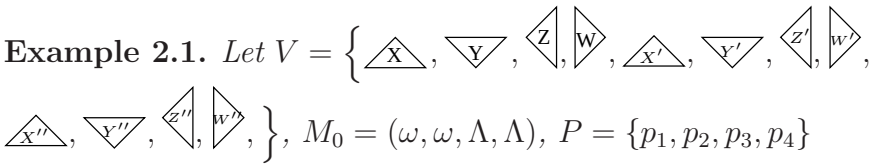
$A\oslash \to B\oslash$ *is the right substitution rule provided they must have gluable sides,*

$A\bigcirc \rightarrow B\bigcirc$ *is the vertical substitution rule provided they must have gluable sides,*

$A\bigcirc \rightarrow B\bigcirc$ *is the left substitution rule provided they must have gluable sides.*

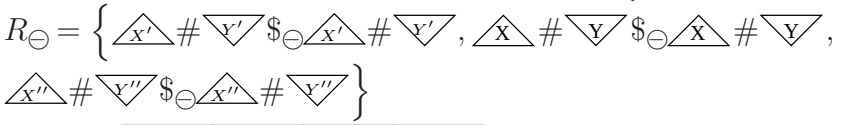
Definition 2.3. *An IATPN is a 7 tuple* $N = (P, T, V, S, R(t), F, M_0)$ *where* $P = \{p_1, p_2, \ldots, p_n\}$ *is a finite set of places,* $T = \{t_1, t_2, \ldots, t_m\}$ *is a finite set of transitions and each* t_i *is the label of evolution rules,* V *is a finite non-empty set of tiles over* Σ_I*,* $S \subseteq (P \times T) \cup (T \times P)$ *is a set of arcs (flow relation),* $R(t)$ *is the set of evolution rules associated with each transition* t *of* T*,* F *is the set of final places that is places without output arcs,* $M_0 : P \rightarrow$ *(Iso arrays over* V*) is the initial marking and* $P \cup T \neq \phi$*,* $P \cap T = \phi$*.*

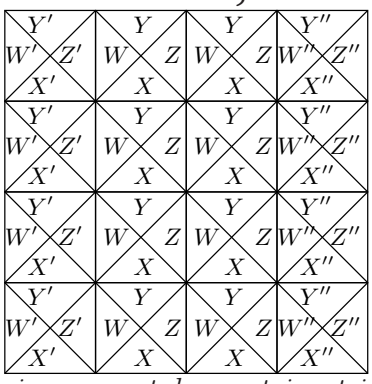
Note 3. In this paper, we use splicing rules in place of evolution rules.

Example 2.1. *Let* $V =$ 

$M_0 = (\omega, \omega, \Lambda, \Lambda)$*,* $P = \{p_1, p_2, p_3, p_4\}$

$T = \{t_1, t_2\}$*,* $R(t) = \{R_\bigcirc, R_\ominus\}$ *where* $R_\bigcirc =$ 

$R_\ominus =$ 

Let $\omega =$  *be an iso-array token on* p_1 *and* p_2*. The resulting iso-array token petri net is given in Figure 1.*
When t_1 *fires, we would get* ω_4 *on* p_4 *(see Figure 2) where*

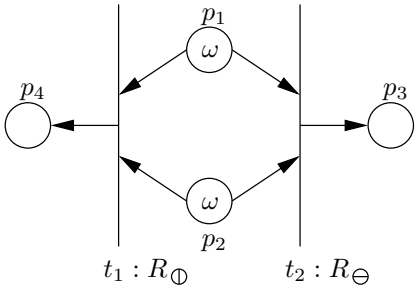


Figure 1

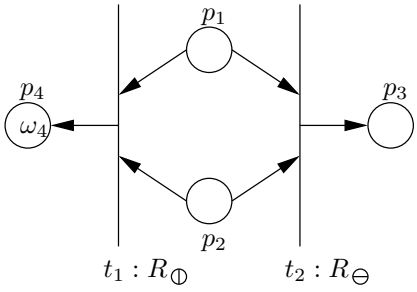
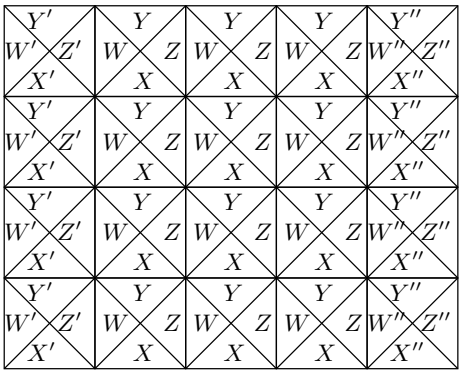


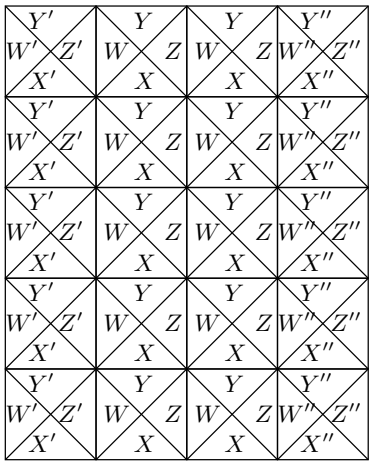
Figure 2

$\omega_4 =$



When t_2 fires from Figure 1 of IATPN, we would get ω_3 on p_3 (see Figure 3)

where $\omega_3 =$



Theorem 2.1. *If L is an iso-array splicing language, then $\exists$ an IATPN N $\ni$ $L = L(N)$, the language generated by N.*

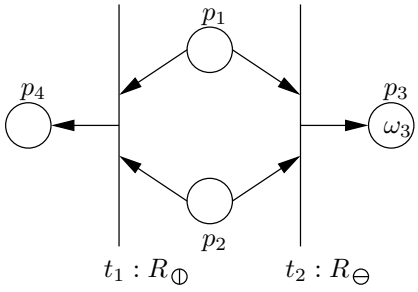


Figure 3

Proof. Let L be the language generated by H iso array system $S = (\sigma, \mathcal{A})$, where $\mathcal{A}$ is a finite subset of Σ_I^{**} (axiom set), $\sigma = (\Sigma_I, R_{\ominus}, R_{\textcircled{|}}, R_{\textcircled{/}}, R_{\textcircled{\backslash}})$ where Σ_I is an alphabet (given tiles). $R_{\ominus}, R_{\textcircled{|}}, R_{\textcircled{/}}, R_{\textcircled{\backslash}}$ are finite sets of horizontal, vertical, right and left splicing rules respectively.

We construct IATPN N as follows:

Let $V = \Sigma_I$. For each of rule in σ, construct a place. For example, if $\mathcal{A} = \{w_1, w_2, w_3\}$, then construct places p_1, p_2, p_3 respectively for w_1, w_2, w_3 (see Figure 4). i.e., if $(q_1, q_2) \vdash q_3$ where $q_1 = \omega_1$, $q_2 = \omega_2$, $q_3 = \omega_3$, then construct p_1, p_2, p_3 places. Let $R(t)$ be the collection of rules like $R_{\ominus}, R_{\textcircled{|}}, R_{\textcircled{/}}, R_{\textcircled{\backslash}}$ i.e., Let $R(t) = \{R_{\ominus}, R_{\textcircled{|}}, R_{\textcircled{/}}, R_{\textcircled{\backslash}}\}$.

For each of rules in $R(t)$, connect places $p_1, p_2, p_3, \ldots$ that we have constructed to the transition with a evolution rule of $R(t)$.

Figure 4

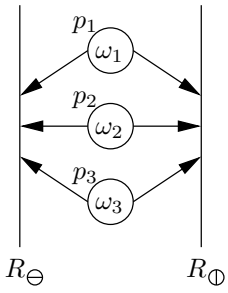


Figure 5

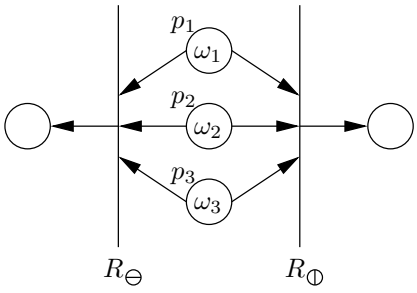


Figure 6

For example, if $R(t) = \{R_{\textcircled{|}}, R_{\ominus}\}$, then connect p_1, p_2, p_3 to transitions with a rule $R_{\textcircled{|}}$, $R_{\ominus}$ (see Figure 5)

Connect these transitions to its output places (see Figure 6). Now, the resulting iso arrays would be deposited in the output places of $t_1, t_2, \ldots$.

The resulting net would generate the given language L.
Hence $L = L(N)$. □

Theorem 2.2. *The class of languages generated by IATPN is closed under union.*

Proof. Let L_1 and L_2 be the languages generated by IATPN's N_1 and N_2 respectively.

Connect the initial places of N_1 and N_2 to transitions t' and t'' respectively. Further connect t' and t'' to the place called p_s with some empty iso array Λ. Attach p_s to t' with the evolution rule $\Lambda \to \Lambda$. Also attach p_s to t'' with the evolution rule $\Lambda \to \Lambda$. Rest of the operations are done as in N_1 and N_2. Then the resulting IATPN would generate $L_1 \cup L_2$ (see Figure 7). If there are n languages, then attach n transitions to the initial

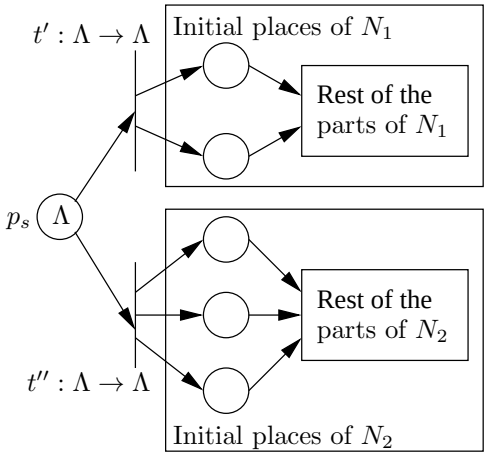


Figure 7

places of $N_1, N_2, N_3, \ldots$ and connect all these transitions to a place p_s, in a similar way with the evolution rule $\Lambda \to \Lambda$. Then the resulting IATPN would generate $L_1 \cup L_2 \cup L_3 \cup \cdots \cup L_n$. □

Note 4: The following results can also be proved. (i) The class of languages generated by IATPN is closed under horizontal and vertical catenation. (ii) The class of languages generated by IATPN is closed under left and right catenation.

Conclusion

Every Iso-Array splicing system could be generated by Iso-Array Token Petri Nets which is introduced in this paper using parallel splicing. Also, it could be seen that the class of languages generated by IATPN is closed under union, horizontal and vertical catenation, left and right catenation. This paper can be extended to study the behaviour of IATPN.

References

[1] Beulah Immanuel, K. Rangarajan, K.G. Subramanian, String-token petri nets, *Proceedings of european conference on artificial intelligence: one day workshop on symbolic networks at valencia*, Spain (2004).

[2] Beulah Immanuel, K.G. Subramanian, P. Usha, Array token petri nets and character generation, *Proceedings of National Conference on Computational Mathematics and Soft Computing* (2009).

[3] Beulah Immanuel, P. Usha, Array token petri nets and 2D grammars, *International Conference on Mathematical Computer Engineering - ICMCE 2013* (2013), 722–727.

[4] T. Head, Formal language theory and DNA: an analysis of the generative capacity of specific recombinant behaviours, Bull. Math. Biol., 49 (1987), 735–759.

[5] T. Head, Gh. Păun and D. Pixton, *Language theory and molecular genetics: generative mechanisms suggested by DNA recomination*, In Handbook of Formal Languages, G. Rozenberg and A. Salomaa, Editors, Springer-Verlag, 2(7), 1997, 296–358.

[6] P. Helen Chandra, K.G. Subramanian, D.G. Thomas and D.L. Van, A note on parallel splicing on images, Electronic Notes in Theoretical Computer Science, 46 (2001), 255–268.

[7] James L. Peterson, *Petrinet theory and modeling of systems*, Prentice Hall, USA (1997).

[8] K. Krithivasan, V.T. Chakaravarthy and R. Rama, Array splicing systems, in *New Trends in Formal Languages, Control Cooperation and*

Combinatorics, Gh. Păun and A. Salomaa, Editors, Lecture Notes in Computer Science, Springer-Verlag, 1218 (1997), 346–365.

[9] V. Masilamani, D.K. Sheena Christy, D.G. Thomas and T. Kalyani, Parallel splicing on iso-arrays, *Proceedings of Fifth International Conference on Bio-Inspired Computing: Theories and Applications* (2010), 1535–1542.