Partial Array Token Petri Net and *P* System

K. Sasikala[1], F. Sweety[1], T. Kalyani[2(✉)], and D. G. Thomas[3]

[1] Department of Mathematics, St. Joseph's college of Engineering, Chennai 600119, India
[2] Department of Mathematics, St. Joseph's Institute of Technology, Chennai 600119, India
[3] Department of Applied Mathematics, Saveetha School of Engineering, SIMATS, Chennai 602105, India

Abstract. The innovative model of partial array languages generated by basic puzzle partial array grammars is available in the literature. Here we define Partial array Token Petri Net Structure (PATPNS) to generate partial array languages. Further we introduce Partial Array Token Petri Net *P* System (PATPNPS) to generate partial array languages and compared with basic puzzle partial array grammars for generative power. PATPNS is also compared with local and recognizable partial array languages.

Keywords: Partial array · Basic puzzle partial array grammar · Partial Array Token Petri Net

1 Introduction

In the context of a syntactic approach to pattern recognition, there have been several studies in the last few decades on theoretical models for generating or recognizing two-dimensional objects, pictures and picture languages [2]. Picture languages generated by several array grammars, matrix grammars have been advocated since the seventies and they have been applied in practical problems such as character recognition, pattern recognition, kolam patterns and tiling systems.

Petri nets are the models in mathematics proposed to model dynamic systems. To simulate the activity of the dynamic system tokens are used, represented by black dots. The tokens move when the transition fires. Array token Petri nets [3–7,10] are proposed to generate array languages. The arrays over an alphabet are used as tokens over an alphabet not the black dots. The transitions are associated with catenation rules. Firing of transitions helps to catenate the arrays to build bigger arrays.

The class of grammars with array rewriting methods is a dynamic device to describe picture languages. The picture languages produced by such array

R. Freund et al. (Eds.): CMC 2020, LNCS 12687, pp. 135–152, 2021.
https://doi.org/10.1007/978-3-030-77102-7_8

grammars, matrix grammars have been applied in practical problems. Nivat et al. [8] proposed puzzle grammars to generate two-dimensional picture languages.

Partial words were introduced by Berstel and Boasson [1]. Later on, Partial array languages were introduced in [14] and the combinatorial properties of partial array languages were studied in [15]. Gh. Paun [9] introduced a computational model, called P system. The notion of P system related to arrays can be seen in [12]. Partial array grammars and partial array rewriting - P system were introduced in [11]. We have proposed Basic Puzzle Partial Array Grammar (BPPAG) to generate Partial array languages and studied the generative capacity of the resulting partial array P system with BPPAG [13]. Motivated by these studies, in this paper, we introduce Partial Array Token Petri Net Structure (PATPNS) and Partial Array Token Petri Net P System (PATPNPS) to generate partial array languages and we examine the generative capacity of both systems and give some comparison results. PATPNS is compared with local and recognizable partial array languages and we have proved that PATPNS has more generative power.

2 Preliminaries

The basic concepts and definitions of Partial Word, Partial Array, Basic Puzzle Partial Array Grammar and Petri Net are given here with examples.

Definition 1. *[1] A partial word u of length n over Σ, is a partial function $u : N \to \Sigma$. For $1 \leq i \leq n$, if $u(i)$ is defined, then we say that i belongs to the domain of u (denoted by $i \in D(u)$); Otherwise, we say that i belongs to the set of holes of u (denoted by $i \in H(u)$). A word over Σ is a partial word over Σ with an empty set of holes. $H(u)$ is the set of positions in which the 'do not know' symbol '$\diamondsuit$' appears in u.*

Definition 2. *[1] If u is a partial word of length n over Σ, then the companion of u (denoted by $u_\diamondsuit$) is the total function $u_\diamondsuit : N \to \Sigma \cup \{\diamondsuit\}$ defined by*

$$u_\diamondsuit(i) = \begin{cases} u(i), i \in D(u); \\ \diamondsuit, \quad otherwise. \end{cases} \quad where \quad \diamondsuit \notin \Sigma.$$

The symbol '$\diamondsuit$' is viewed as a 'do not know' symbol and not as a 'do not care' symbol as in pattern matching.

Definition 3. *[14] A partial array A of size $m \times n$ over Σ is a partial function $A : Z_+^2 \to \Sigma$, where Z is the set of all positive integers. For $1 \leq i \leq m$, $1 \leq j \leq n$, if $A(i, j)$ is defined then we say that (i, j) belongs to the domain of A (denoted by $(i, j) \in D(A)$); Otherwise, we say that (i,j) belongs to the set of holes of A (denoted by $(i, j) \in H(A)$). An array over Σ is partial array over Σ with an empty set of holes. $H(A)$ is the set of positions in which the 'do not know' symbol '$\diamondsuit$' appears in A.*

Definition 4. *[14] If A is a partial array of size $m \times n$ over Σ, then the companion of A (denoted by $A_\diamondsuit$) is the total function $A_\diamondsuit : Z_+^2 \to \Sigma \cup \{\diamondsuit\}$ defined by*

$$A_\diamondsuit(i,j) = \begin{cases} A(i,j), & (i,j) \in D(A); \\ \diamondsuit, & \text{otherwise.} \end{cases} \quad \text{where} \quad \diamondsuit \notin \Sigma.$$

Example 1. [14] The Partial array, $A_\diamondsuit = \begin{pmatrix} a & b & a \\ \diamondsuit & b & a \\ a & \diamondsuit & b \end{pmatrix}$ is a companion of a partial array A of size $(3,3)$ where $D(A) = \{(1,1),(1,2),(1,3),(2,2),(2,3),(3,1),(3,3)\}$ and $H(A) = \{(2,1),(3,2)\}$

Definition 5. *[14] If A and B are two partial arrays of equal size, then A is contained in B, denoted by $A \subset B$ if $D(A) \subseteq D(B)$ and $A(i,j) = B(i,j)$ for all $(i,j) \in D(A)$. The partial arrays A and B are said to be compatible, denoted by $A \uparrow B$ if there exists a partial array C such that $A \subset C$ and $B \subset C$.*

$A_\diamondsuit = \begin{pmatrix} a & b & a \\ \diamondsuit & b & a \\ a & \diamondsuit & b \end{pmatrix}$ *and* $B_\diamondsuit = \begin{pmatrix} \diamondsuit & b & \diamondsuit \\ a & b & a \\ a & \diamondsuit & b \end{pmatrix}$ *are the companions of two partial arrays A and B that are compatible.*

The set of all partial arrays over Σ is denoted by Σ_p^{**}, where $\Sigma_p = \Sigma \cup \{\diamondsuit\}$. We denote the empty array with no symbols by Λ and $\Sigma_p^{++} = \Sigma_p^{**} - \{\Lambda\}$. The set of all partial arrays over Σ of size (k,r), $k \le m, r \le n$ is denoted by $\Sigma_p^{(k,r)}$.

Definition 6. *[13] The structure of a Basic Puzzle Partial Array Grammar (BPPAG) is $BPG_p = (A, B \cup \{\diamondsuit\}, P, S)$ where A is a finite non empty set of non terminal symbols and B is a finite non empty set of terminal symbols. '$\diamondsuit$' is a 'do not know' symbol, where $\diamondsuit \notin A \cup B$, $S \in A$ is the axiom pattern and P is a set of rules of the following forms:*

(i) $X \to \textcircled{x}Y$ $\quad$ (ii) $X \to \textcircled{\diamondsuit}Y$ $\quad$ (iii) $X \to \textcircled{Y}x$

(iv) $X \to \textcircled{Y}\diamondsuit$ $\quad$ (v) $X \to Y\textcircled{x}$ $\quad$ (vi) $X \to Y\textcircled{\diamondsuit}$

(vii) $X \to x\textcircled{Y}$ $\quad$ $(viii)$ $X \to \diamondsuit\textcircled{Y}$ $\quad$ (ix) $X \to \begin{matrix}\textcircled{x}\\ Y\end{matrix}$

(x) $X \to \begin{matrix}\textcircled{\diamondsuit}\\ Y\end{matrix}$ $\quad$ (xi) $X \to \begin{matrix}\textcircled{Y}\\ x\end{matrix}$ $\quad$ (xii) $X \to \begin{matrix}\textcircled{Y}\\ \diamondsuit\end{matrix}$

$(xiii)$ $X \to \begin{matrix}Y\\ \textcircled{x}\end{matrix}$ $\quad$ (xiv) $X \to \begin{matrix}Y\\ \textcircled{\diamondsuit}\end{matrix}$ $\quad$ (xv) $X \to \begin{matrix}x\\ \textcircled{Y}\end{matrix}$

(xvi) $X \to \begin{matrix}\diamondsuit\\ \textcircled{Y}\end{matrix}$ $\quad$ $(xvii)$ $X \to \textcircled{x}$ $\quad$ $(xviii)$ $X \to \textcircled{\diamondsuit}$

where $X, Y \in A$ and $x, y \in B$.

While processing the derivations in the production rule $X \rightarrow \textcircled{x}Y$, the non-terminal X is replaced by the right-hand member whose left-hand side is X.

The replacement is possible only if the noncircled symbol of the production rule consists of a blank symbol. The blank symbol is represented by the letter '#', which is an unoccupied place where any symbol can be occupied as per the derivation. The language generated by BPPAG is denoted by $\mathcal{L}(BPPAG)$.

Example 2. [13] Consider a BPPAG $BPG_{p_1} = (A, B \cup \{\diamondsuit\}, P, S)$ where $A = \{X, Q, R, S_1, T, U, V\}$, $B = \{z\}$, $S = X$ and P consists of the following rules:

(i) $X \rightarrow \textcircled{z}Q$ (ii) $Q \rightarrow \textcircled{z}Q$ (iii) $Q \rightarrow \overset{R}{\textcircled{z}}$

(iv) $R \rightarrow S_1\textcircled{z}$ (v) $S_1 \rightarrow \textcircled{z}$ (vi) $S_1 \rightarrow S_1\textcircled{\diamondsuit}$

(vii) $S_1 \rightarrow \overset{T}{\textcircled{z}}$ $(viii)$ $T \rightarrow \textcircled{z}T$ (ix) $T \rightarrow \textcircled{z}$

(x) $T \rightarrow \textcircled{\diamondsuit}T$ (xi) $T \rightarrow \overset{U}{\textcircled{z}}$ (xii) $U \rightarrow U\textcircled{z}$

$(xiii)$ $U \rightarrow \textcircled{z}$ (xiv) $T \rightarrow \overset{R}{\textcircled{z}}$

This grammar generates square partial arrays of size $(m \times m, m \geq 2)$ with $(m-2 \times m-2, m \geq 2)$ square partial array in the center consisting of only $\{\diamondsuit\}$ symbol bounded by the terminal alphabet 'z' on the boundary of the square, for $m = 2$, the grammar generates 2×2 square array $\begin{matrix} z & z \\ z & z \end{matrix}$.

The first three members of the language are given below:

$$\begin{matrix} z & z \\ z & z \end{matrix} \qquad \begin{matrix} z & z & z \\ z & \diamondsuit & z \\ z & z & z \end{matrix} \qquad \begin{matrix} z & z & z & z \\ z & \diamondsuit & \diamondsuit & z \\ z & \diamondsuit & \diamondsuit & z \\ z & z & z & z \end{matrix} \qquad \ldots$$

A Petri Net [10] is an abstract formal model of information flow. Petri nets have been used for analyzing systems that are concurrent, asynchronous, distributed, parallel, non-deterministic and/or stochastic. Tokens are used in Petri Nets to simulate dynamic and concurrent activities of the system. A language can be associated with the execution of a Petri Net. By defining a labeling function for transitions over an alphabet, the set of all firing sequences, starting from a specific initial marking leading to a finite set of terminal markings, generates a language over the alphabet. Petri Net structure to generate rectangular arrays are found in [3–5]. The two models have different firing rules and catenation rules. In [6], Column Row Catenation Petri Net Structure (CRCPNS) has been defined. Several input places having different arrays is associated with a catenation rules label. The label of the transition decides the order in which the arrays are joined (column wise or row wise) provided the condition for catenation is satisfied. In CRCPNS a transition with a catenation rule as label and different arrays in the input places is enabled to fire. In ATPNS [15] the catenation rule

involves an array language. All the input places of the transition with a catenation rule as label, should have the same array as token, for the transition to be enabled. The size of the array language to be joined to the array in the input place, depends on the size of the array in the input place.

Definition 7. *[5] A Petri Net structure is a four tuple $C = (P, T, I, O)$ where $P = \{p_1, p_2, \ldots, p_n\}$ is a finite set of places, $n > 0$, $T = \{t_1, t_2, \ldots, t_m\}$ is a finite set of transitions, $m > 0$, $P \cap T = \phi$, $I : T \to P^\infty$ is the input function from transitions to bags of places and $O : T \to P^\infty$ is the output function from transitions to bags of places, where P^∞ is the bags of places.*

Definition 8. *[5] A Petri Net marking is an assignment of tokens to the places of Petri Net. The tokens are used to define the execution of a Petri Net. The number and position of tokens may change during the execution of a Petri Net, arrays over an alphabet are used as tokens.*

3 Partial Array Token Petri Net Structure

In this section, we define Partial Array Token Petri Net Structure (PATPNS) with an example and compare it with basic puzzle partial array languages.

Definition 9. *If $C = (P, T, I, O)$ is a Petri Net structure with partial arrays over $(\Sigma \cup \{\diamondsuit\})^{**}$ as initial markings. $\mu_0 : P \to (\Sigma \cup \{\diamondsuit\})^{**}$ label of at least one transition being catenation rule and a finite set of final places $F \subset P$, then the Petri net structure C is defined as a Partial Array Token Petri Net Structure (PATPNS).*

Definition 10. *If C is a PATPNS, then the Partial array language generated by the Petri Net C is defined as*

$$PL(C) = \{A_\diamondsuit \in (\Sigma \cup \{\diamondsuit\})^{**} \ / \ A_\diamondsuit \text{ is in } p \text{ for some } p \text{ in } F\}$$

with partial arrays over $(\Sigma \cup \{\diamondsuit\})^{**}$ in some places as initial marking when all possible sequences of transitions are fired. The set of all partial arrays collected in the final places F is called the partial array language generated by C. Let $\mathcal{L}(PATPNS) = \{PL(C)/C \text{ is a } PATPNS\}$.

$(\Sigma \cup \{\diamondsuit\})^{**}$ denotes the partial arrays made up of elements of $\Sigma \cup \{\diamondsuit\}$. If A and B are two partial arrays having same number of rows then $A\textcircled{|}B$ is the column wise catenation of A and B. If two partial arrays have the same number of columns then $A\textcircled{-}B$ is the row wise catenation of A and B. $(x)^n$ denotes a horizontal sequence of n 'x' and $(x)_n$ denotes a vertical sequence of n 'x' where $x \in (\Sigma \cup \{\diamondsuit\})^{**}$, $(x)^{n+1} = (x)^n\textcircled{|}x$ and $(x)_{n+1} = (x)_n\textcircled{-}x$.

The Petri Net model defined here has places and transitions connected by directed arcs. Rectangular partial arrays over an alphabet are taken as tokens to be distributed in places. Variation in firing rules and labels of the transition are listed out below.

Firing Rules in PATPNS

We define three different types of enabled transition in PATPNS. The pre and post condition for firing the transition in all the three cases are given below:

1. When all the input places of t_1 (without label) have the same partial array as token.
   - Each input place should have at least the required number of partial arrays.
   - Firing t_1 removes partial array from all the input places and moves the partial array to all its output places.

   The graph in Fig. 1 shows the position of the partial array before the transition fires and Fig. 2 shows the position of the partial array after transition t_1 fires.

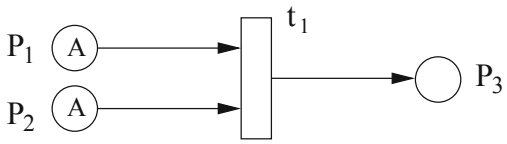

Fig. 1. Position of partial array before firing

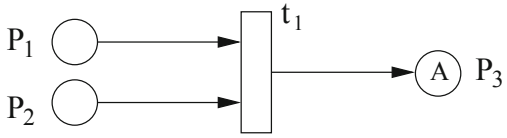

Fig. 2. Position of partial array after firing

2. When all the input places of t_1 have different partial arrays as token
   - The label of t_1 designates one of its input places.
   - The designated input place has sufficient number of partial arrays as tokens.
   - Firing t_1 removes partial array from all the input places and moves the partial array from the designated input place to all its output places.

   The graph in Fig. 3 shows the position of the partial array before the transition fires and Fig. 4 shows the position of the partial array after transition t_1 fires. Since the designated place is P_1, the partial array in P_1 is moved to the output place.
3. When all the input places of t_1 (with catenation rule as label) have the same partial array as token
   - Each input place should have at least the required number of partial arrays.

– The condition for catenation should be satisfied.
– The designated input place has sufficient number of partial arrays as tokens.
– Firing t_1 removes partial array from all the input places P and the catenation is carried out in all its output places.

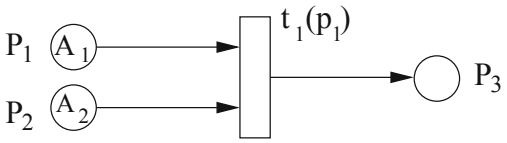

Fig. 3. Transition with label before firing

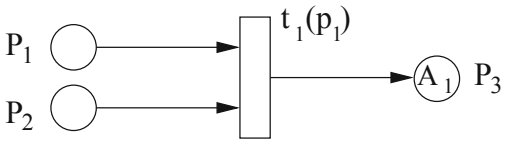

Fig. 4. Transition with label after firing

Catenation Rule as Label for Transitions
Column catenation rule is in the form $A(|)B$. Here the partial array A denotes the $m \times n$ partial array in the input place of the transition. B is a partial array whose number of rows will depend on 'm', the number of rows of A. The number of columns of B is fixed. For example $A(|)(x \quad x)_m$ adds two columns of x after the last column of the partial array A which is in the input place. But $(x \quad x)_m(|)A$ would add two columns of x before the first column of A. 'm' always denote the number of rows of the input partial array A. Row catenation rule is in the form $A(-)B$. Here again the partial array A denotes the $m \times n$ partial array in the input place of the transition. B is a partial array whose number of columns will depend on 'n', the number of columns of A. The number of rows of B is always fixed. For example $A(-)\begin{bmatrix} x \\ x \end{bmatrix}^n$ adds two rows of x after the last row of the array A which is in the input place. But $\begin{bmatrix} x \\ x \end{bmatrix}^n(-)A$ would add two rows of x before the first row of the partial array A. 'n' always denotes the number of columns of the input partial array A.

An example to explain row catenation rule is given below. The position of the partial array before the transition fires is shown in Fig. 5 and Fig. 6 shows

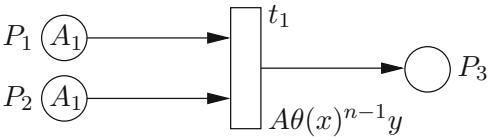

Fig. 5. Transition with catenation rule before firing

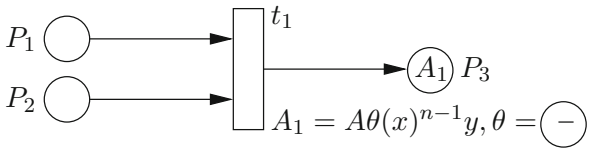

Fig. 6. Transition with catenation rule after firing

the position of the partial array after transition t_1 fires. Since the catenation rule is associated with the transition, catenation takes place in P_3.

In $A_\diamondsuit = \begin{matrix} a\ a\ a \\ a \diamondsuit a \\ a\ a\ a \end{matrix}$, the number of columns of A is 3, $n-1$ is 2, firing t_1 adds the row $x\ x\ y$ as the last row. Hence $A_{1\diamondsuit} = \begin{matrix} a\ a\ a \\ a \diamondsuit a \\ a\ a\ a \\ x\ x\ y \end{matrix}$

Example 3. Let $\Sigma = \{a\}, F = P_1$, where $S_\diamondsuit = \begin{matrix} a\ a\ a \\ a \diamondsuit a \\ a\ a\ a \end{matrix}$, $\quad Q_1 = (\diamondsuit)_m, Q_2 = (\diamondsuit)^n$

$Q_3 = (a)_m, \quad Q_4 = (a)^n$

S is the initial partial array placed in P_1. The PATPNS is shown in Fig. 7. Derivations in PATPNS is given in the following tabular column.

Input place	Transition	Output place
S	$A\,(\,\vert\,)\,Q_1$	$a\ a\ a \diamondsuit$ $a \diamondsuit a \diamondsuit$ $a\ a\ a \diamondsuit$
$a\ a\ a \diamondsuit$ $a \diamondsuit a \diamondsuit$ $a\ a\ a \diamondsuit$	$Q_1\,(\,\vert\,)\,A$	$\diamondsuit\ a\ a\ a \diamondsuit$ $\diamondsuit\ a \diamondsuit a \diamondsuit$ $\diamondsuit\ a\ a\ a \diamondsuit$
$\diamondsuit\ a\ a\ a \diamondsuit$ $\diamondsuit\ a \diamondsuit a \diamondsuit$ $\diamondsuit\ a\ a\ a \diamondsuit$	$A\,(-)\,Q_2$	$\diamondsuit\ a\ a\ a \diamondsuit$ $\diamondsuit\ a \diamondsuit a \diamondsuit$ $\diamondsuit\ a\ a\ a \diamondsuit$ $\diamondsuit \diamondsuit \diamondsuit \diamondsuit \diamondsuit$

Input place	Transition	Output place
$\diamondsuit\ a\ a\ a\ \diamondsuit$ $\diamondsuit\ a\ \diamondsuit\ a\ \diamondsuit$ $\diamondsuit\ a\ a\ a\ \diamondsuit$ $\diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit$	$Q_2(-)A$	$\diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit$ $\diamondsuit\ a\ a\ a\ \diamondsuit$ $\diamondsuit\ a\ \diamondsuit\ a\ \diamondsuit$ $\diamondsuit\ a\ a\ a\ \diamondsuit$ $\diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit$
$\diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit$ $\diamondsuit\ a\ a\ a\ \diamondsuit$ $\diamondsuit\ a\ \diamondsuit\ a\ \diamondsuit$ $\diamondsuit\ a\ a\ a\ \diamondsuit$ $\diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit$	$A(\mid)Q_3$	$\diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ a$ $\diamondsuit\ a\ a\ a\ \diamondsuit\ a$ $\diamondsuit\ a\ \diamondsuit\ a\ \diamondsuit\ a$ $\diamondsuit\ a\ a\ a\ \diamondsuit\ a$ $\diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ a$
$\diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ a$ $\diamondsuit\ a\ a\ a\ \diamondsuit\ a$ $\diamondsuit\ a\ \diamondsuit\ a\ \diamondsuit\ a$ $\diamondsuit\ a\ a\ a\ \diamondsuit\ a$ $\diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ a$	$Q_3(\mid)A$	$a\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ a$ $a\ \diamondsuit\ a\ a\ a\ \diamondsuit\ a$ $a\ \diamondsuit\ a\ \diamondsuit\ a\ \diamondsuit\ a$ $a\ \diamondsuit\ a\ a\ a\ \diamondsuit\ a$ $a\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ a$
$a\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ a$ $a\ \diamondsuit\ a\ a\ a\ \diamondsuit\ a$ $a\ \diamondsuit\ a\ \diamondsuit\ a\ \diamondsuit\ a$ $a\ \diamondsuit\ a\ a\ a\ \diamondsuit\ a$ $a\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ a$	$A(-)Q_4$	$a\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ a$ $a\ \diamondsuit\ a\ a\ a\ \diamondsuit\ a$ $a\ \diamondsuit\ a\ \diamondsuit\ a\ \diamondsuit\ a$ $a\ \diamondsuit\ a\ a\ a\ \diamondsuit\ a$ $a\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ a$ $a\ a\ a\ a\ a\ a\ a$
$a\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ a$ $a\ \diamondsuit\ a\ a\ a\ \diamondsuit\ a$ $a\ \diamondsuit\ a\ \diamondsuit\ a\ \diamondsuit\ a$ $a\ \diamondsuit\ a\ a\ a\ \diamondsuit\ a$ $a\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ a$ $a\ a\ a\ a\ a\ a\ a$	$Q_4(-)A$	$a\ a\ a\ a\ a\ a\ a$ $a\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ a$ $a\ \diamondsuit\ a\ a\ a\ \diamondsuit\ a$ $a\ \diamondsuit\ a\ \diamondsuit\ a\ \diamondsuit\ a$ $a\ \diamondsuit\ a\ a\ a\ \diamondsuit\ a$ $a\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ a$ $a\ a\ a\ a\ a\ a\ a$

The firing of sequence $(t_1t_2t_3t_4t_5t_6t_7t_8)^k$, $k \geq 0$ puts a square partial arrays of size $4k+3$ in P_1, where the boundaries of the squares are alternatively $\diamondsuit$'s and a's. The partial array language generated by the PATPNS is a square partial array of size $4k+3$, $k \geq 0$ where the boundaries are alternatively $\diamondsuit$'s on the odd numbered boundaries and a's on the even numbered boundaries.

Theorem 1. *The family of languages generated by PATPNS is properly contained in the family of languages generated by Basic Puzzle Partial Array Grammars.*

Proof. The row catenation in PATPNS can be handled by the following Basic Puzzle Partial Array Grammar rules:

$(i)\quad X \to \overset{x}{Y} \qquad (ii)\quad X \to \overset{\diamondsuit}{Y} \qquad (iii)\quad X \to \overset{Y}{\diamondsuit} \qquad (iv)\quad X \to \overset{Y}{x}$

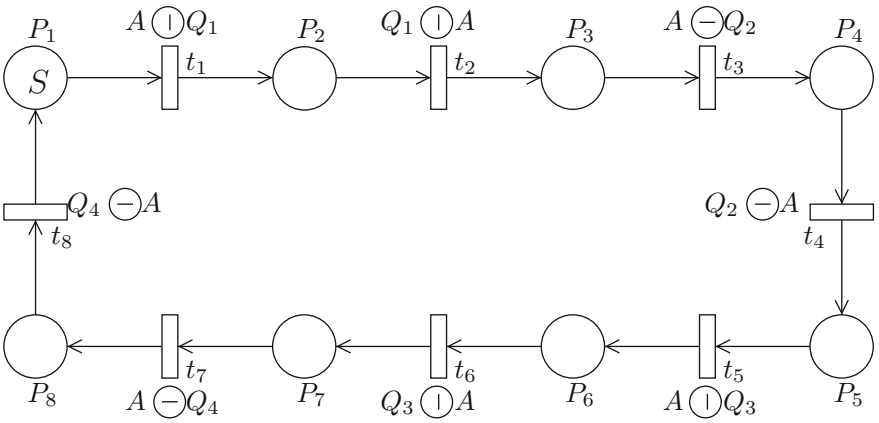

Fig. 7. PATPNS generating square partial arrays of size $4k + 3$, $k \geq 0$

(v) $X \to \overset{Y}{(x)}$ (vi) $X \to \overset{Y}{(\diamondsuit)}$ (vii) $X \to \overset{x}{(Y)}$ $(viii)$ $X \to \overset{\diamondsuit}{(Y)}$

(ix) $X \to (x)$ (x) $X \to (\diamondsuit)$

The column catenation in PATPNS can be handled by the following Basic Puzzle Partial Array Grammar rules:

(i) $X \to (x)\, Y$ (ii) $X \to (\diamondsuit)\, Y$ (iii) $X \to (Y)\, x$

(iv) $X \to (Y)\, \diamondsuit$ (v) $X \to Y\, (x)$ (vi) $X \to Y\, (\diamondsuit)$

(vii) $X \to \diamondsuit (Y)$ $(viii)$ $X \to x\, (Y)$ (ix) $X \to (x)$ (x) $X \to (\diamondsuit)$

Hence $\mathcal{L}(PATPNS)$ is a subset of $\mathcal{L}(BPPAG)$, this is also evident from the following example.

Consider a partial array language of square partial arrays of size $4k + 3$, $k \geq 0$ whose boundaries are alternatively $\diamondsuit$'s on the odd numbered boundaries and a's on the even numbered boundaries given in Example 3. This partial array language is generated by both systems PATPNS and BPPAG.

Now let us consider a BPPAG generating this partial array language.

$$BPG_{P_2} = (A, B \cup \{\diamondsuit\}, P, S)$$

where $A = \{X, Q_1, Q_2, Q_3, Q_4, Q_5, Q_6\}$, $B = \{a\}$, $S = X$ and P consists of the following rules:

(i) $X \to (a)Q$ (ii) $Q \to (a)Q$ (iii) $Q \to \overset{Q_1}{(a)}$

(iv) $Q_1 \to Q_2(a)$ (v) $Q_2 \to Q_3(\diamondsuit)$ (vi) $Q_3 \to \overset{Q_4}{(a)}$

(vii) $Q_4 \to \textcircled{a} Q_5$ $(viii)$ $Q_5 \to \textcircled{a} Q_5$ (ix) $Q_5 \to \textcircled{a}$

(x) $Q_3 \to \textcircled{\diamondsuit} Q_3$ (xi) $Q_5 \to \textcircled{\diamondsuit} Q_6$ (xii) $Q_6 \to \textcircled{a} Q_6$

$(xiii)$ $Q_6 \to \textcircled{\diamondsuit} Q$ (xiv) $Q_3 \to Q_2 \textcircled{a}$ (xv) $Q_3 \to Q_3 \textcircled{\diamondsuit}$

(xvi) $Q \to \textcircled{\diamondsuit} Q$

The first member of the language generated is shown below:

$$X \xrightarrow{(i)} \textcircled{a}\, Q \xrightarrow{(ii)} a\, \textcircled{a}\, Q \xrightarrow{(iii)} \begin{matrix} & Q_1 & \\ a & a & \textcircled{a} \end{matrix} \xrightarrow{(iv)} \begin{matrix} & Q_2 & \textcircled{a} \\ a & a & a \end{matrix}$$

$$\xrightarrow{(v)} \begin{matrix} & Q_3 & \textcircled{\diamondsuit} & a \\ a & a & a & \end{matrix} \xrightarrow{(vi)} \begin{matrix} & Q_4 & & \\ a & \diamondsuit & a & \\ a & a & a & \end{matrix} \xrightarrow{(vii)} \begin{matrix} & \textcircled{a} & Q_5 & \\ a & \diamondsuit & a & \\ a & a & a & \end{matrix} \xrightarrow{(viii)} \begin{matrix} a & \textcircled{a} & Q_5 & \\ a & \diamondsuit & a & \\ a & a & a & \end{matrix} \xrightarrow{(ix)} \begin{matrix} a & a & a \\ a & \diamondsuit & a \\ a & a & a \end{matrix}$$

The partial array language given in Example 2 generated by BPPAG cannot be generated by PATPNS, since the axiom array $\begin{matrix} z & z \\ z & z \end{matrix}$ can only be concatenated either row wise or column wise, but $\diamondsuit$ cannot be inserted, which proves a proper containment.

4 Partial Array Token Petri Net *P* System

In this section, Partial Array Token Petri Net P System (PATPNPS) is introduced and it is compared with PATPNS and BPPAG.

Definition 11. *A Partial Array Token Petri Net P System (PATPNPS)* $\pi = (V, T \cup \{\diamondsuit\}, \#, \mu, F_1, F_2, \ldots, F_m, R_1, R_2, \ldots, R_m, i_0)$ *where* V *is a finite set of column partial arrays and row partial arrays of the form* $Q_1 = (a)^n$ *and* $Q_2 = (a)_m$ *where* $a \in T \cup \{\diamondsuit\}$. T *is a finite set of terminal alphabets. '#' is a blank symbol not in* $T \cup \{\diamondsuit\}$. μ *is a membrane structure with 'm' membranes,* $F_1, F_2, \ldots, F_m$ *are finite set of partial arrays over* $T \cup \{\diamondsuit\}$ *associated with the 'm' regions.* $R_1, R_2, \ldots, R_m$ *are rules associated with the m regions of the form* $\left(\{A\textcircled{-}Q, A\textcircled{|}Q\}, tar, P\right)$, *where* $tar \in \{here, in, out\}$ *and* P *is the output obtained after the catenation rule is applied.*

If the target indication is 'here', the output partial array 'P' remains in the same region, if the target indication is '*in*', 'P' goes to the immediate inner region and if the target indication is 'out' it goes to the outer membrane, i_0 is the elementary membrane of μ.

A computation in a partial array token petri net P system is defined in the same way as in array rewriting P system. The set of all partial arrays computed by π with 'm' membranes is denoted by $PATPNPL_m(\pi)$.

Example 4. Consider the Partial Array Token Petri Net P System PATPNPS

$$\pi_1 = (\{Q_1, Q_2, Q_3, Q_4\}, \{a, \diamondsuit\}, \#, [_1[_2[_3]_2]_1, F_{1\diamondsuit}, F_{2\diamondsuit}, F_{3\diamondsuit}, R_1, R_2, R_3, 3)$$

where $Q_1 = (\diamondsuit)_m$, $Q_2 = (\diamondsuit)^n$, $Q_3 = (a)_m$, $Q_4 = (a)^n$, $F_{1\diamondsuit} = \begin{matrix} a\ a\ a \\ a\ \diamondsuit\ a \\ a\ a\ a \end{matrix}$, $F_{2\diamondsuit} = F_{3\diamondsuit} = \phi$,

$$R_1 = \left\{ \begin{matrix} (F_1 \bigcirc\!\!\!| Q_1, here, P_1), (Q_1 \bigcirc\!\!\!| P_1, here, P_1), (P_1 \bigcirc\!\!\!- Q_2, here, P_1), \\ (Q_2 \bigcirc\!\!\!- P_1, in, P_1), (P_2 \bigcirc\!\!\!| Q_1, here, P_1) \end{matrix} \right\};$$

$$R_2 = \left\{ \begin{matrix} (P_1 \bigcirc\!\!\!| Q_3, here, P_2), (Q_3 \bigcirc\!\!\!| P_2, here, P_2), (P_2 \bigcirc\!\!\!- Q_4, here, P_2), \\ (Q_4 \bigcirc\!\!\!- P_2, in, P_2), (Q_4 \bigcirc\!\!\!- P_2, out, P_2) \end{matrix} \right\}$$

$R_3 = \phi$.

The content of region 1 is $F_{1\diamondsuit} = \begin{matrix} a\ a\ a \\ a\ \diamondsuit\ a \\ a\ a\ a \end{matrix}$. The derivations in PATPNPS is given in the following tabular column:

Region(i)	Content($F_{i\diamondsuit}$)	Rule(R_i)	Target	Resultant partial array (P_i)
1	$\begin{matrix} a\ a\ a \\ a\ \diamondsuit\ a \\ a\ a\ a \end{matrix}$	$F_1 \bigcirc\!\!\!| Q_1$	here	$\begin{matrix} a\ a\ a\ \diamondsuit \\ a\ \diamondsuit\ a\ \diamondsuit \\ a\ a\ a\ \diamondsuit \end{matrix}$
1	$\begin{matrix} a\ a\ a\ \diamondsuit \\ a\ \diamondsuit\ a\ \diamondsuit \\ a\ a\ a\ \diamondsuit \end{matrix}$	$Q_1 \bigcirc\!\!\!| P_1$	here	$\begin{matrix} \diamondsuit\ a\ a\ a\ \diamondsuit \\ \diamondsuit\ a\ \diamondsuit\ a\ \diamondsuit \\ \diamondsuit\ a\ a\ a\ \diamondsuit \end{matrix}$
1	$\begin{matrix} \diamondsuit\ a\ a\ a\ \diamondsuit \\ \diamondsuit\ a\ \diamondsuit\ a\ \diamondsuit \\ \diamondsuit\ a\ a\ a\ \diamondsuit \end{matrix}$	$P_1 \bigcirc\!\!\!- Q_2$	here	$\begin{matrix} \diamondsuit\ a\ a\ a\ \diamondsuit \\ \diamondsuit\ a\ \diamondsuit\ a\ \diamondsuit \\ \diamondsuit\ a\ a\ a\ \diamondsuit \\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit \end{matrix}$
1	$\begin{matrix} \diamondsuit\ a\ a\ a\ \diamondsuit \\ \diamondsuit\ a\ \diamondsuit\ a\ \diamondsuit \\ \diamondsuit\ a\ a\ a\ \diamondsuit \\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit \end{matrix}$	$Q_2 \bigcirc\!\!\!- P_1$	in	$\begin{matrix} \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit \\ \diamondsuit\ a\ a\ a\ \diamondsuit \\ \diamondsuit\ a\ \diamondsuit\ a\ \diamondsuit \\ \diamondsuit\ a\ a\ a\ \diamondsuit \\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit \end{matrix}$
2	$\begin{matrix} \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit \\ \diamondsuit\ a\ a\ a\ \diamondsuit \\ \diamondsuit\ a\ \diamondsuit\ a\ \diamondsuit \\ \diamondsuit\ a\ a\ a\ \diamondsuit \\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit \end{matrix}$	$P_1 \bigcirc\!\!\!| Q_3$	here	$\begin{matrix} \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ a \\ \diamondsuit\ a\ a\ a\ \diamondsuit\ a \\ \diamondsuit\ a\ \diamondsuit\ a\ \diamondsuit\ a \\ \diamondsuit\ a\ a\ a\ \diamondsuit\ a \\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ a \end{matrix}$
2	$\begin{matrix} \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ a \\ \diamondsuit\ a\ a\ a\ \diamondsuit\ a \\ \diamondsuit\ a\ \diamondsuit\ a\ \diamondsuit\ a \\ \diamondsuit\ a\ a\ a\ \diamondsuit\ a \\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ a \end{matrix}$	$Q_3 \bigcirc\!\!\!| P_2$	here	$\begin{matrix} a\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ a \\ a\ \diamondsuit\ a\ a\ a\ \diamondsuit\ a \\ a\ \diamondsuit\ a\ \diamondsuit\ a\ \diamondsuit\ a \\ a\ \diamondsuit\ a\ a\ a\ \diamondsuit\ a \\ a\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ a \end{matrix}$
2	$\begin{matrix} a\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ a \\ a\ \diamondsuit\ a\ a\ a\ \diamondsuit\ a \\ a\ \diamondsuit\ a\ \diamondsuit\ a\ \diamondsuit\ a \\ a\ \diamondsuit\ a\ a\ a\ \diamondsuit\ a \\ a\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ a \end{matrix}$	$P_2 \bigcirc\!\!\!- Q_4$	here	$\begin{matrix} a\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ a \\ a\ \diamondsuit\ a\ a\ a\ \diamondsuit\ a \\ a\ \diamondsuit\ a\ \diamondsuit\ a\ \diamondsuit\ a \\ a\ \diamondsuit\ a\ a\ a\ \diamondsuit\ a \\ a\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ a \\ a\ a\ a\ a\ a\ a\ a \end{matrix}$

Region(i)	Content($F_{i\diamondsuit}$)	Rule(R_i)	Target	Resultant partial array (P_i)
2	$a\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ a$ $a\ \diamondsuit\ a\ a\ a\ \diamondsuit\ a$ $a\ \diamondsuit\ a\ \diamondsuit\ a\ \diamondsuit\ a$ $a\ \diamondsuit\ a\ a\ a\ \diamondsuit\ a$ $a\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ a$ $a\ a\ a\ a\ a\ a\ a$	$Q_4 \ominus P_2$	in or out	$a\ a\ a\ a\ a\ a\ a$ $a\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ a$ $a\ \diamondsuit\ a\ a\ a\ \diamondsuit\ a$ $a\ \diamondsuit\ a\ \diamondsuit\ a\ \diamondsuit\ a$ $a\ \diamondsuit\ a\ a\ a\ \diamondsuit\ a$ $a\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ a$ $a\ a\ a\ a\ a\ a\ a$
3 (If tar=in)	$a\ a\ a\ a\ a\ a\ a$ $a\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ a$ $a\ \diamondsuit\ a\ a\ a\ \diamondsuit\ a$ $a\ \diamondsuit\ a\ \diamondsuit\ a\ \diamondsuit\ a$ $a\ \diamondsuit\ a\ a\ a\ \diamondsuit\ a$ $a\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ a$ $a\ a\ a\ a\ a\ a\ a$	ϕ	-	$a\ a\ a\ a\ a\ a\ a$ $a\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ a$ $a\ \diamondsuit\ a\ a\ a\ \diamondsuit\ a$ $a\ \diamondsuit\ a\ \diamondsuit\ a\ \diamondsuit\ a$ $a\ \diamondsuit\ a\ a\ a\ \diamondsuit\ a$ $a\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ a$ $a\ a\ a\ a\ a\ a\ a$ The output is collected in the elementary membrane
1 (If tar=out)	$a\ a\ a\ a\ a\ a\ a$ $a\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ a$ $a\ \diamondsuit\ a\ a\ a\ \diamondsuit\ a$ $a\ \diamondsuit\ a\ \diamondsuit\ a\ \diamondsuit\ a$ $a\ \diamondsuit\ a\ a\ a\ \diamondsuit\ a$ $a\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ \diamondsuit\ a$ $a\ a\ a\ a\ a\ a\ a$	$P_2 \mid Q_1$	here	Procedure continues

Thus the partial array language of square partial arrays of size $4k+3$, $k \geq 0$, where the boundaries are alternatively $\diamondsuit$'s on the odd numbered boundaries and a's on the even numbered boundaries is generated by this $PATPNPS$ π_1.

Theorem 2. $\mathcal{L}(PATPNPL_m(\pi)) \cap \mathcal{L}(PATPNS) \neq \phi$.

Proof. The partial array language of square partial arrays of size $4k+3$, $k \geq 0$ is generated by both systems. It is evident from Examples 3 and 4.

Theorem 3. $\mathcal{L}(PATPNPL_m(\pi)) \cap \mathcal{L}(BPPAG) \neq \phi$.

Proof. $\mathcal{L}(PATPNS)$ is a subclass of the family of Basic Puzzle Partial Array Languages by Theorem 1. By Theorem 2, we get that $\mathcal{L}(PATPNPL_m(\pi))$ intersects $\mathcal{L}(PATPNS)$. Thus the two families intersects.

5 Comparative Study with Local and Recognizable Partial Array Languages

In this section, we recall Local and Recognizable Partial Array Languages and compare with PATPNS.

Definition 12. *[14] Let $\Gamma_p = \Gamma \cup \{\diamondsuit\}$ be a finite alphabet. A two dimensional partial array language $PL \subseteq \Gamma_p^{**}$ is local if there exists a finite set θ of tiles over the alphabet $\Gamma_p \cup \{\#\}$ such that $PL = \{A \in \Gamma_p^{**} / B_{2,2}(\hat{A}) \subseteq \theta\}$, where $\hat{A}$ is a partial array surrounded by a special boundary symbol $\# \notin \Gamma$.*

The partial array language PL is local if given such a set θ, we can exactly retrieve the language PL. We call the set θ a representation by tiles for the local language PL and write $PL = L(\theta)$. The family of all local partial array languages is denoted by PAL-LOC.

Example 5. [14] Let $\Gamma_p = \{a, b\} \cup \{\diamondsuit\}$, and θ be the following set of tiles over $\Gamma_p \cup \{\#\}$.

$$\theta = \left\{ \begin{array}{|c|c|}\hline \# & \# \\ \hline \# & b \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline b & b \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline b & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & b \\ \hline \# & a \\ \hline \end{array}, \begin{array}{|c|c|}\hline b & b \\ \hline a & \diamondsuit \\ \hline \end{array}, \begin{array}{|c|c|}\hline b & b \\ \hline \diamondsuit & b \\ \hline \end{array}, \begin{array}{|c|c|}\hline b & \# \\ \hline b & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & a \\ \hline \# & a \\ \hline \end{array}, \begin{array}{|c|c|}\hline a & \diamondsuit \\ \hline a & a \\ \hline \end{array}, \begin{array}{|c|c|}\hline \diamondsuit & b \\ \hline a & b \\ \hline \end{array}, \begin{array}{|c|c|}\hline a & a \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline a & b \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & a \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline b & \# \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline b & b \\ \hline \diamondsuit & \diamondsuit \\ \hline \end{array}, \begin{array}{|c|c|}\hline a & \diamondsuit \\ \hline a & \diamondsuit \\ \hline \end{array}, \begin{array}{|c|c|}\hline \diamondsuit & \diamondsuit \\ \hline \diamondsuit & \diamondsuit \\ \hline \end{array}, \begin{array}{|c|c|}\hline \diamondsuit & b \\ \hline \diamondsuit & b \\ \hline \end{array}, \begin{array}{|c|c|}\hline \diamondsuit & \diamondsuit \\ \hline a & a \\ \hline \end{array} \right\}$$

Then $L(\theta)$ is a partial array language over Γ_p with equal sides of length, the symbols along the top row (the last row) and right most column (the last column are b's, the symbols along the first row (the bottom row) and the left most column (the first column) except the first and last elements of the principal diagonals are a's. The remaining elements of the array are holes.

The first two members of this language are given below:

$$\begin{array}{ccc} b & b & b \\ a & \diamondsuit & b, \\ a & a & b \end{array} \quad \begin{array}{cccc} b & b & b & b \\ a & \diamondsuit & \diamondsuit & b \\ a & \diamondsuit & \diamondsuit & b, \\ a & a & a & b \end{array} \quad \cdots$$

Definition 13. *[14] Let Σ be a finite alphabet. A partial array language $PL \subseteq \Sigma_p^{**}$ is called recognizable if there exists a local partial array language PL' over Γ_p and a mapping $\pi : \Gamma_p \to \Sigma_p$ such that $PL = \pi(PL')$, where $\Sigma_p = \Sigma \cup \{\diamondsuit\}$. The family of all recognizable partial array languages is denoted by PAL-REC.*

Example 6. [14] The set of all partial array languages over one letter alphabet 'a' with all sides of equal length and the symbols along the first row, first column, the last row and last column are holes is not a local partial array language, but it is a recognizable partial array language. This language is obtained from Example 5 by taking a mapping $\pi : \Gamma_p \to \Sigma_p$ where $\Gamma = \{a, b\}$, $\Sigma = \{a\}$ such that $\pi(b) = \pi(a) = \diamondsuit$ and $\pi(\diamondsuit) = a$.

Theorem 4. *$PAL - LOC \subsetneq PATPNS$.*

Every local partial language can be easily generated by some PATPNS. Let PL be a partial array language over Γ_p in PAL-LOC with a finite set of tiles θ such that $PL = L(\theta)$.

Consider the PATPNS, $C = (P, T, I, O)$ with partial arrays over Γ_p^{**}, $S_\diamondsuit = \begin{array}{|c|c|}\hline \# & \# \\ \hline \# & a \\ \hline \end{array}$, $a \in \Gamma_p$. T, the set of all transitions, they can be either row or column catenations.

(i) For all $\begin{array}{|c|c|}\hline \# & \# \\ \hline \# & a \\ \hline \end{array} \in \theta$, where $a \in \Gamma_p$ we define $t_1 = A\textcircled{|}Q_1$, where $Q_1 = \begin{pmatrix} \# \\ b \end{pmatrix}$, $b \in \Gamma_p$.

(ii) For all $\begin{array}{|c|c|}\hline \# & \# \\ \hline a & b \\ \hline \end{array} \in \theta$, $a, b \in \Gamma_p$, we define $t_2 = A\textcircled{|}Q_1$, where $Q_1 = \begin{pmatrix} \# \\ b \end{pmatrix}$, $b \in \Gamma_p$. This transition is repeated till the tile of the form $\begin{array}{|c|c|}\hline \# & \# \\ \hline b & \# \\ \hline \end{array} \in \theta$ is reached and let this process be repeated 'r' $r \geq 0$, no. of times.

(iii) For all $\begin{array}{|c|c|}\hline \# & \# \\ \hline b & \# \\ \hline \end{array} \in \theta$, $b \in \Gamma_p$, we define $t_3 = A\textcircled{|}Q_2$, where $Q_2 = \begin{pmatrix} \# \\ \# \end{pmatrix}$.

(iv) For all $\begin{array}{|c|c|}\hline \# & a \\ \hline \# & b \\ \hline \end{array}, \begin{array}{|c|c|}\hline a & c \\ \hline b & d \\ \hline \end{array}, \begin{array}{|c|c|}\hline c & \# \\ \hline d & \# \\ \hline \end{array} \in \theta$, $a, b, c, d \in \Gamma_p$, we define $t_4 = A\textcircled{-}Q_3$, where $Q_3 = (\#\ B\ \#)$, $B \in \Gamma_p^{(1\times n)}$, where $\Gamma_p^{(1\times n)}$ is a partial array of size $1 \times n$.

(v) For all tiles of the form $\begin{array}{|c|c|}\hline \# & a \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline a & b \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline b & \# \\ \hline \# & \# \\ \hline \end{array} \in \theta$, $a, b \in \Gamma_p$, we define $t_5 = A\textcircled{-}Q_4$, $Q_4 = (\#)_m$, $m \geq 2$. Here, A represents the partial array collected in the output place by the previous transition.

$$P = \{P_1, \underbrace{P_2, P_3, \ldots P_k}_{t_2}, \underbrace{P_{k+1}}_{t_3}, \underbrace{P_{k+2}, P_{k+3}, \ldots P_{k+s+1}}_{t_4}, \underbrace{P_{k+s+2}}_{t_5}\},$$

where P is the set of places, $S_\diamondsuit$ is placed in P_1 initially and then transition t_1 is applied, the resultant partial array is stored in P_2, and then transition t_2 is applied 'r' no. of times. After applying 'r' times the transition t_2, the partial array reaches the place P_k, where $k = r + 2$. After transition t_3 is applied the partial array reaches P_{k+1}. The transition t_4 is repeated 's' $s \geq 0$ number of times until the tile $\begin{array}{|c|c|}\hline \# & a \\ \hline \# & \# \\ \hline \end{array}$ is reached. After this transition, the partial array reaches the place P_{k+s+1}. After transition t_5 is applied the partial array reaches the final place $F = P_{k+s+2}$.

The PATPNS generating the local language PL is given in Fig. 8.

Clearly PATPNS can generate any partial array language in PAL-LOC and hence $PAL - LOC \subseteq PATPNS$.

Now, to prove the proper inclusion, we consider the partial array language given in Example 3, the square partial arrays of size $4k+3$, $k \geq 0$. This language is not local, since the θ set of this language can also generate any array over one letter alphabet 'a' of size $1 \times n$, $n \geq 1$, where θ is given as follows.

$$\theta = \left\{ \begin{array}{|c|c|}\hline \# & \# \\ \hline \# & a \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline a & a \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline a & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & a \\ \hline \# & a \\ \hline \end{array}, \begin{array}{|c|c|}\hline a & a \\ \hline a & \diamondsuit \\ \hline \end{array}, \begin{array}{|c|c|}\hline a & a \\ \hline \diamondsuit & \diamondsuit \\ \hline \end{array}, \begin{array}{|c|c|}\hline a & a \\ \hline \diamondsuit & a \\ \hline \end{array}, \begin{array}{|c|c|}\hline a & \# \\ \hline a & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline a & \diamondsuit \\ \hline a & \diamondsuit \\ \hline \end{array}, \begin{array}{|c|c|}\hline \diamondsuit & \diamondsuit \\ \hline \diamondsuit & a \\ \hline \end{array}, \right.$$
$$\left. \begin{array}{|c|c|}\hline \diamondsuit & \diamondsuit \\ \hline a & a \\ \hline \end{array}, \begin{array}{|c|c|}\hline \diamondsuit & \diamondsuit \\ \hline a & \diamondsuit \\ \hline \end{array}, \begin{array}{|c|c|}\hline \diamondsuit & a \\ \hline \diamondsuit & a \\ \hline \end{array}, \begin{array}{|c|c|}\hline a & \diamondsuit \\ \hline a & a \\ \hline \end{array}, \begin{array}{|c|c|}\hline \diamondsuit & a \\ \hline a & a \\ \hline \end{array}, \begin{array}{|c|c|}\hline a & \diamondsuit \\ \hline \diamondsuit & \diamondsuit \\ \hline \end{array}, \begin{array}{|c|c|}\hline a & a \\ \hline \diamondsuit & \diamondsuit \\ \hline \end{array}, \begin{array}{|c|c|}\hline a & \diamondsuit \\ \hline \diamondsuit & \diamondsuit \\ \hline \end{array}, \begin{array}{|c|c|}\hline a & a \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline a & \# \\ \hline \# & \# \\ \hline \end{array} \right\}$$

Hence PAL-LOC is properly contained in PATPNS.

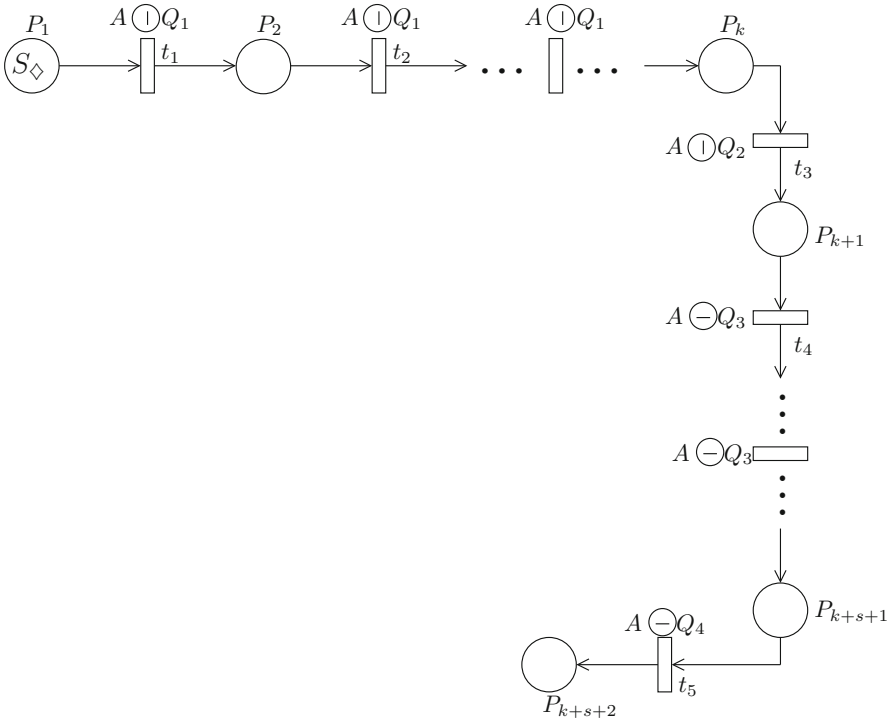

Fig. 8. PATPNS generating any PAL-LOC

Example 7. Consider a PATPNS $C = (P, T, I, O)$, generating a local partial array language given in Example 5.

Let $S_\diamondsuit = \begin{array}{|c|c|}\hline \# & \# \\ \hline \# & b \\ \hline \end{array}$, $Q_1 = \begin{bmatrix} \# \\ b \end{bmatrix}$, $Q_2 = \begin{bmatrix} \# \\ \# \end{bmatrix}$, $Q_3 = [\#\ B\ \#]$, $B \in \Gamma_p^{1\times n}$, where $B = a\ (\diamondsuit)_r\ b$, $Q_4 = (\#)_m$.

PATPNS generating the 2^{nd} member of the partial array language namely

$\# \# \# \# \# \#$
$\# \ b \ b \ b \ b \ \#$
$\# \ a \ \diamondsuit \ \diamondsuit \ b \ \#$
$\# \ a \ \diamondsuit \ \diamondsuit \ b \ \#$
$\# \ a \ a \ a \ b \ \#$
$\# \# \# \# \# \#$

is given in Fig. 9 as an example.

Theorem 5. *PATPNS is closed under projection.*

We consider a partial array token Petri Net structure $C = (P, T, I, O)$ generating the partial array language PL. Let $\pi : \Gamma_p \to \Sigma_p$ be a projetion such that $\pi(a) = \alpha$, $a \in \Gamma_p$, $\alpha \in \Sigma_p$. Without loss of generality $\Gamma_p \cap \Sigma_p = \phi$. We can construct a PATPNS $C' = (P', T', I', O')$ such that $PL(C') = PL$, where $T' = \{t'_1, t'_2, \ldots t'_k\}$, $t'_i = \{A'(|)Q'_i, A'(-)Q'_i / A'_{ij} = (\pi(A))_{ij}, (Q'_i)_{rs} = (\pi(Q_i))_{rs}\}$,

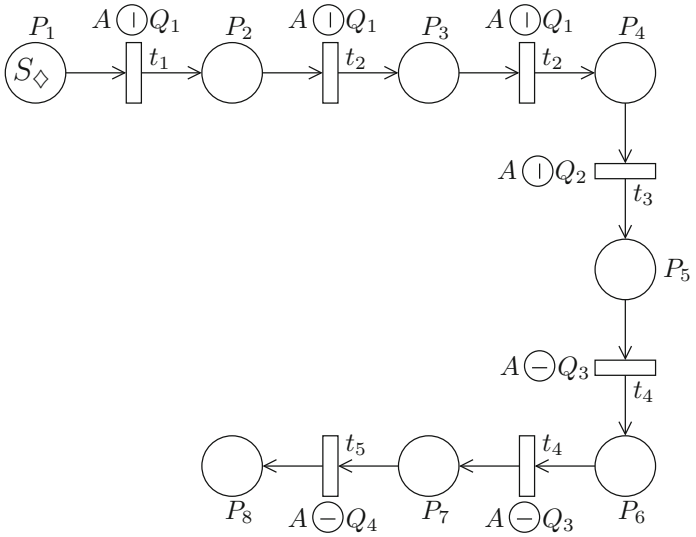

Fig. 9. PATPNS generating PAL-LOC given in Example 5

$1 \leq i \leq k$, $A, Q_i \in \Gamma_p^{**}$, $A', Q_i' \in \Sigma_p^{**}$. I' and O' are input and outplaces of the transitions. P' is the set of places.

Hence we can clearly say that PATPNS is closed under projection.

Theorem 6. *$PAL - REC \subsetneq PATPNS$.*

Proof. *$PAL - REC \subseteq PATPNS$* follows from Theorems 4 and 5, since every recognizable partial array language is a projection of a local partial array language.

Now the proper inclusion can be proved easily by giving an example of a partial array language which is not in PAL-REC but in PATPNS.

6 Conclusion

In this paper we have proposed PATPNS and PATPNPS. PATPNS is compared with PAL-LOC, PAL-REC and BPPAG. It is also compared with PATPNPS. The properties of PATPNS and PATPNPS can be studied further by introducing inhibitor arc to increase the generative capacity of PATPNS. This is our future work.

References

1. Berstel, J., Boasson, L.: Partial words and a theorem of Fine and Wilf. Theor. Comput. Sci. **218**(1), 135–141 (1999)

2. Giammarresi, D., Restivo, A.: Two-dimensional languages. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, pp. 215–267. Springer, Heidelberg (1997). https://doi.org/10.1007/978-3-642-59126-6_4
3. Lalitha, D.: Rectangular array languages generated by a Colored Petri Net. In: IEEE International Conference on Electrical Computer and Communication Technologies, pp. 1–5 (2015)
4. D., L.: Rectangular array languages generated by a Petri net. In: Sethi, I.K. (ed.) Computational Vision and Robotics. AISC, vol. 332, pp. 17–27. Springer, New Delhi (2015). https://doi.org/10.1007/978-81-322-2196-8_3
5. Lalitha, D., Rangarajan, K., Thomas, D.G.: Rectangular arrays and Petri nets. In: Barneva, R.P., Brimkov, V.E., Aggarwal, J.K. (eds.) IWCIA 2012. LNCS, vol. 7655, pp. 166–180. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34732-0_13
6. Mary Metilda, M.I., Lalitha, D.: Kolam generated by color Petri nets. In: Tuba, M., Akashe, S., Joshi, A. (eds.) Information and Communication Technology for Sustainable Development. AISC, vol. 933, pp. 675–681. Springer, Singapore (2020). https://doi.org/10.1007/978-981-13-7166-0_68
7. Mary Metilda, M.I., Lalitha, D.: Petri nets for pasting tiles. In: Solanki, V.K., Hoang, M.K., Lu, Z.J., Pattnaik, P.K. (eds.) Intelligent Computing in Engineering. AISC, vol. 1125, pp. 701–708. Springer, Singapore (2020). https://doi.org/10.1007/978-981-15-2780-7_76
8. Nivat, M., Saoudi, A., Subramanian, K.G., Siromoney, R., Dare, V.R.: Puzzle grammar and context-free array grammars. Int. J. Pattern Recogn. Artif. Intell. **05**(05), 663–676 (1991)
9. Paun, Gh.: Computing with membranes. J. Comput. Syst. Sci. **61**(1), 108–143 (2000)
10. Peterson, J.L.: Petri Net Theory and Modeling of Systems. Prentice Hall Inc., Englewood Cliffs (1981)
11. Sasikala, K., Kalyani, T., Thomas, D.G.: Partial array grammars and partial array-rewriting P systems. Math. Eng. Sci. Aerosp. **11**(1), 227–236 (2020)
12. Subramanian, K.G., Saravanan, R., Geethalakshmi, M., Helen Chandra, P., Margenstern, M.: P systems with array object and array rewriting rules. Prog. Nat. Sci. **17**(4), 479–485 (2007)
13. Sweety, F., Sasikala, K., Kalyani, T., Thomas, D.G.: Partial array-rewriting P systems and basic puzzle partial array grammar. In: AIP Conference Proceedings, vol. 2277, p. 030003 (2020)
14. Sweety, F., Thomas, D.G., Dare, V.R., Kalyani, T.: Recoginizability of partial array languages. J. Comb. Math. Comb. Comput. **69**, 237–249 (2009)
15. Vijaya Chitra, S., Sasikala, K.: Squares in partial arrays. In: AIP Conference Proceedings, vol. 2112, pp. 20–34 (2019)