Abstract: Shuffle operation on trajectories is useful in modeling parallel composition of words and languages. In this work, a new class of P systems with shuffle operation and catalytic-like rules is presented. Such a system has a membrane structure, where language-objects and shuffle-operation rules are placed in its regions. It can be used as a language generator. In this study, we propose a variant P system with shuffle operation on string-language objects. Some comparison results are obtained, which show that the power of shuffle operation is enlarged in the framework of P systems. Moreover, string-language objects are extended to array-language objects, and another variant P system with shuffle operation on picture-language objects is introduced. We also illustrate how to generate picture languages by using this kind of devices.

Key Words: membrane computing, P system, shuffle on trajectories, picture language

Category: F.1.1, F.4.3, I.7.0

1 Introduction

Membrane computing was introduced by Păun in 1998, which is inspired from the structure and the functioning of living cells, as well as the organization of cells in tissues, organs, and other higher order structures [Păun 2000]. Membrane system, also called P system, provides a non-deterministic distributed parallel model for dealing with mathematical or computational problems [Gutiérrez-Naranjo et al. 2006, Pan et al. 2011a]. As we know, most P systems are proved to be universal [Păun 2002], and computationally efficient [Alhazov et al. 2003, Pan and Alhazov 2006, Pan and Martín-Vide 2006, Pan and Pérez-Jiménez 2010, Pan and Martín-Vide 2005]. Nowadays, applications of P systems [Ciobanu et al. 2006], as well as exploring new models [Pan et al. 2011b, Pan et al. 2012a, Pan et al. 2012b, Zhang et al. 2012], are getting more and more attention. A most recent overview of the field of membrane computing can be found in [Păun et al. 2010].
Shuffle on trajectories is useful in modeling parallel composition of words and languages [Păun et al. 1995]. Informally, trajectory is a segment of a line in the plane, starting in the origin of axes and continuing in parallel with the axes Ox and Oy. The line can change its direction only in points of nonnegative integer coordinates. A trajectory defines how to skip from a word to another word during the shuffle operation. Shuffle on trajectories [Mateescu et al. 1998, Kadrie et al. 2001] provides a method of great flexibility to handle the operation of parallel composition of processes from the catenation to the usual shuffle of processes.

In [Annadura et al. 2008], trajectories are drawn into P system for the first time, and a so-called trajectory P system is proposed. In a trajectory P system, shuffle operation deals with words from the same language set in each region, and this operation is applied in a special kind of sequential way. Specifically, if \(i\) is the elementary membrane and \(L_i\) is the set of words in this membrane, then \(L_i \overset{T}{\rightarrow} = \bigcup_{\alpha, \beta \in L_i} \alpha \uplus \beta\) is the language computed in the \(i\)-th membrane. Sequences are considered as objects in membranes. However, the number of each kind of objects is not considered in that kind of P systems. Here, we introduce a new kind of P system with shuffle operation, which uses languages as objects and contains catalytic-like rules. This kind of system has a structure of basic P systems. All the evolution rules are based on shuffle operation and applied in a maximally parallel way as usual in P systems. Results of the shuffle operation are the only new products after applying evolution rules. It deals with words from different languages in each region. For example, if \(i\) is the elementary membrane and \(L_{i1}, L_{i2}\) are two languages in this membrane, then \(L_{i1} \uplus L_{i1} \overset{T}{\rightarrow} L_{i2} \uplus L_{i2} = \bigcup_{\alpha \in L_{i1}, \beta \in L_{i2}} \alpha \uplus \beta\) is the new product in some step of the computation. If \(L_{i1} = L_{i2}\) in our system, \(L_{i1} \uplus L_{i1} \overset{T}{\rightarrow} L_{i2}\) is equal to \(L_{i1} \uplus L_{i1}\) in trajectory P systems. In some sense, trajectory P system [Annadura et al. 2008] can be considered as a special case of the system proposed here. Some comparison results are given, which show that the string-language generative power of shuffle operation is further extended in a membrane structure. Moreover, we extend string objects to picture objects and propose another variant, a P system with shuffle operation on picture languages. It generates languages which can be generated by Siromoney matrix grammars [Siromoney et al. 1972] or array-rewriting P systems [Ceterchi et al. 2003]. The picture-language generative power of P system with shuffle operation on picture languages is illustrated with several examples. The trajectory P system considered in [Venkatesan et al. 2010] which is an extension to arrays of trajectory P system [Annadura et al. 2008] is also in some sense a special case of the P system with shuffle operation on picture languages considered here.

2 Preliminaries

An alphabet \(V\) is a non empty set, whose elements are called symbols. An ordered sequence of symbols is a string. The number of symbols in a string \(u\) is the length of the string, and it is denoted by \(|u|\). As usual, the empty string (with length 0) is
denoted by $\lambda$. The set of strings of length $n$ built with symbols from the alphabet $V$ is denoted by $V^n$ and $V^* = \cup_{n \geq 0} V^n$. A language over $V$ is a subset of $V^*$. The families of languages, which are regular, context free, context sensitive and recursively enumerable, are denoted by $\text{REG}$, $\text{CF}$, $\text{CS}$ and $\text{RE}$, respectively. A picture $A$ over $V$ is a rectangular $m \times n$ array of elements of the form

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} = [a_{ij}]_{m \times n}.$$  

The set of all pictures or arrays over $V$ is denoted by $V^{**}$. A picture or an array language over $V$ is a subset of $V^{**}$. The number of columns in an array $A$ is denoted by $|A|_c$. The number of rows in an array $A$ is denoted by $|A|_r$. The empty array is denoted by $\Lambda$, $|\Lambda|_c = |\Lambda|_r = 0$.

Let $A = [a_{ij}]_{m \times p}, B = [b_{ij}]_{n \times q}$. The column concatenation $A \Phi B$ of $A$ and $B$ is defined only when $m = n$, which is given by

$$A \Phi B = \begin{bmatrix} a_{11} & \cdots & a_{1p} & b_{11} & \cdots & b_{1q} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mp} & b_{n1} & \cdots & b_{nq} \end{bmatrix}.$$  

Similarly, the row concatenation $A \Theta B$ of $A$ and $B$ is defined only when $p = q$, which is given by

$$A \Theta B = \begin{bmatrix} a_{11} & \cdots & a_{1p} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mp} \\ b_{11} & \cdots & b_{1q} \\ \vdots & \ddots & \vdots \\ b_{n1} & \cdots & b_{nq} \end{bmatrix}.$$  

$A \Phi P = P \Phi A = P$ and $A \Theta P = P \Theta A = P$ for any $P \in V^{**}$.

**Definition 1** A trajectory is an element $t$, $t \in V^*$, where $V = \{r, u\}$.

Let $\Sigma$ be an alphabet and let $t$ be a trajectory, $t = t_1t_2 \cdots t_n$, where $t_i \in V$, $1 \leq i \leq n$. We define $|t|_r$ the number of $r$ in $t$, and $|t|_u$ is defined as the number of $u$ in $t$. Let $\alpha, \beta$ be two words over $\Sigma$, $\alpha = a_1a_2 \cdots a_p$, $\beta = b_1b_2 \cdots b_q$, where $a_i, b_j \in \Sigma$, $1 \leq i \leq p$ and $1 \leq j \leq q$.

**Definition 2** The shuffle of $\alpha$ with $\beta$ on the trajectory $t$, denoted by $\alpha \shuffle_t \beta$, is defined as follows: if $|\alpha| \neq |t|_r$ or $|\beta| \neq |t|_u$, then $\alpha \shuffle_t \beta = \emptyset$; else $\alpha \shuffle_t \beta = c_1c_2 \cdots c_{p+q}$, where

$$c_i = \begin{cases} a_{k_1+1} & \text{if } t_i = r, \\ b_{k_2+1} & \text{if } t_i = u, \end{cases}$$

$$k_1 + 1 = \min \{ k \geq 1 | \sum_{j=1}^k a_j = |t|_r \}, \quad k_2 + 1 = \min \{ k \geq 1 | \sum_{j=1}^k b_j = |t|_u \}.$$
where \( k_1 \) is defined as the number of \( r \) in \( t_1, \ldots, t_{i-1} \), and \( k_2 \) is defined as the number of \( u \) in \( t_1, \ldots, t_{i-1} \).

If \( T \) is a set of trajectories, the shuffle of \( \alpha \) with \( \beta \) on the set \( T \) of trajectories, denoted \( \alpha \sqcup \beta \), is \( \alpha \sqcup \beta = \bigcup_{t \in T} \alpha \sqcup t \beta \). If \( L_1, L_2 \subseteq \Sigma^* \), then \( L_1 \sqcup \beta L_2 = \bigcup_{\alpha \in L_1, \beta \in L_2} \alpha \sqcup \beta \) [Mateescu et al. 1998].

**Theorem 1.** [Mateescu et al. 1998] Let \( L_1, L_2 \) and \( T, T \subseteq \{r, u\}^* \) be three languages.

(i) If all three languages are regular, then \( L_1 \sqcup T \sqcup L_2 \) is regular.

(ii) If two languages are regular languages and the third one is a context-free language, then \( L_1 \sqcup T \sqcup L_2 \) is a context-free language.

**Definition 3** Let \( V \) be a finite alphabet, \( t \) a string over \( \{r, u\}, v \in \{r, u\} \), \( P, Q \in V^* \) having the same number of rows and \( |P|_r = |t|_r, |Q|_c = |t|_c \). The column shuffle of \( P \) with \( Q \) on the trajectory \( vt \), denoted by \( P \sqcup \| vt Q \), is recursively defined as follows.

- If \( v = r \), then \( P \sqcup \| vt Q = (A\Phi X) \sqcup \| vt Q = A\Phi (X \sqcup vt Q) \), where \( A \in V^* \) is the first column of \( P \).
- If \( v = u \), then \( P \sqcup \| vt Q = P \sqcup \| vt (B\Phi Y) = B\Phi (P \sqcup vt Y) \), where \( B \in V^* \) is the first column of \( Q \).
- If \( v = r \), then \( A \sqcup \| vt P = \emptyset, P \sqcup \| vt A = (A\Phi X) \sqcup \| vt A = A\Phi (X \sqcup vt A) \), where \( A \in V^* \) is the first column of \( P \).
- If \( v = u \), then \( A \sqcup \| vt P = A \sqcup \| vt (A\Phi X) = A\Phi (X \sqcup vt A), P \sqcup \| vt A = \emptyset \), where \( A \in V^* \) is the first column of \( P \).
- If \( t \neq \lambda \), then \( A \sqcup \| t A = \emptyset \); otherwise, \( A \sqcup \| t A = A \).

**Definition 4** Let \( V \) be a finite alphabet, \( t \) a string over \( \{l, d\}, v \in \{l, d\} \), \( P, Q \in V^* \) having the same number of columns and \( |P|_r = |t|_l, |Q|_d = |t|_d \). The row shuffle of \( P \) with \( Q \) on the trajectory \( vt \), denoted by \( P \sqcup \| vt Q \), is recursively defined as follows.

- If \( v = l \), then \( P \sqcup \| vt Q = (A\Theta X) \sqcup \| vt Q = A\Theta (X \sqcup vt Q) \), where \( A \in V^* \) is the first row of \( P \).
- If \( v = d \), then \( P \sqcup \| vt Q = P \sqcup \| vt (B\Theta Y) = B\Theta (P \sqcup vt Y) \), where \( B \in V^* \) is the first row of \( Q \).
- If \( v = l \), then \( A \sqcup \| vt P = \emptyset, P \sqcup \| vt A = (A\Theta X) \sqcup \| vt A = A\Theta (X \sqcup vt A) \), where \( A \in V^* \) is the first row of \( P \).
- If \( v = d \), then \( A \sqcup \| vt P = A \sqcup \| vt (A\Theta X) = A\Theta (X \sqcup vt A), P \sqcup \| vt A = \emptyset \), where \( A \in V^* \) is the first row of \( P \).
If $t \neq \lambda$, then $A \perp_{\perp}^{r} A = \emptyset$; otherwise, $A \perp_{\perp}^{r} A = A$.

**Definition 5** [Siromoney et al. 1972] A Siromoney matrix grammar is a 2-tuple $(G_1, G_2)$, where $G_1 = (H_1, I_1, P_1, S)$ is a regular, a context-free or a context-sensitive grammar

- $H_1$ is a finite set of horizontal non-terminals,
- $I_1 = \{S_1, S_2, \ldots, S_k\}$, a finite set of intermediates, $H_1 \cap I_1 = \emptyset$,
- $P_1$ is a finite set of production rules called horizontal production rules,
- $S \in H_1$ is the start symbol.

$G_2 = (G_{21}, G_{22}, \ldots, G_{2k})$, where $G_{2i} = (V_{2i}, T, P_{2i}, S_i), 1 \leq i \leq k$ are regular grammars,

- $V_{2i}$ is a finite set of vertical non-terminals, $V_{2i} \cap V_{2j} = \emptyset, i \neq j$,
- $T$ is a finite set of terminals,
- $P_{2i}$ is a finite set of right linear production rules of the form $X \rightarrow aY$ or $X \rightarrow a$, where $X, Y \in V_{2i}, a \in T$,
- $S_i \in V_{2i}$ is the start symbol of $G_{2i}$.

The type of $G_1$ gives the type of $G$. $G$ is called regular, context-free or context-sensitive Siromoney matrix grammars if $G_1$ is regular, context-free or context-sensitive, respectively.

Derivations are defined as follows. A string $S_{i_1}S_{i_2}\ldots S_{i_n} \in I_1^*$ is generated horizontally using the horizontal production rules of $P_1$ in $G_1$, i.e., $S \Rightarrow S_{i_1}S_{i_2}\ldots S_{i_n} \in I_1^*$. Vertical derivations proceed as follows. We write

$A_{i_1} \ldots A_{i_n}$
$\downarrow$
$a_{i_1} \ldots a_{i_n}$
$B_{i_1} \ldots B_{i_n}$

if $A_{ij} \rightarrow a_{ij}B_{ij}$ are rules in $P_{2j}, 1 \leq j \leq n$, where $A_{ij}, B_{ij}$ are the $j$-th non-terminals of some strings. The derivation terminates if $A_{mj} \rightarrow a_{mj}$ are all terminal rules in $G_2$.

The set $L(G)$ of picture arrays generated by $G$ consists of all $m \times n$ arrays $[a_{ij}]$ such that $1 \leq i \leq m, 1 \leq j \leq n$ and $S \Rightarrow^{*}_{G_1} S_{i_1}S_{i_2}\ldots S_{i_m} \Rightarrow^{*}_{G_2} [a_{ij}]$. We denote the picture language classes of regular, context-free Siromoney matrix grammars by $RML$, $CFML$, respectively.
3 P System with Shuffle Operation and Catalytic-Like Rules

3.1 P System with Shuffle Operation on String Languages

**Definition 6** A P system with shuffle operation on string languages is defined as:

\[ \Pi = (V, T, \mu, L_1, \cdots, L_m, T, T_1, \cdots, T_m, R_1, \cdots, R_m, i_o) \]

where:

- \( V \) is an alphabet;
- \( T = \{ r, u \} \) is the control alphabet;
- \( \mu \) is a membrane structure consisting of \( m \) membranes;
- \( L_1, \cdots, L_m \) are initial sets of languages associated with regions 1, \( \cdots \), \( m \) of \( \mu \), respectively;
- \( T_i \subseteq T^*, 1 \leq i \leq m \) are sets of trajectories associated with regions 1, \( \cdots \), \( m \) of \( \mu \), respectively;
- \( R_i \) is a finite set of evolution rules associated with regions \( i \) of \( \mu \); and evolution rules are of the forms: \( A_{h_1}B_{h_2} \rightarrow (C, \text{tar}) \), or catalytic-like rules: \( A_{h_1}B_{h_2} \rightarrow (A, \text{here})(C, \text{tar}) \) or \( A_{h_1}B_{h_2} \rightarrow (C, \text{tar})(B, \text{here}) \), where \( A, B, C \) are languages defined over alphabet \( V \), \( h_1, h_2 \in T \), \( h_1 \neq h_2 \), and \( \text{tar} \subseteq \{ \text{here, out, in}_j \mid 1 \leq j \leq m \} \). Each language is assigned with either \( r \) or \( u \), which means that it can appear in right or up direction in the shuffle operation. If there exist \( A', B' \in L_i, A' \subseteq A, B' \subseteq B, A' \sqcup \sqcup T_i B' \neq \emptyset \), then \( A', B' \) can be assigned to one rule of the three forms non-deterministically, and \( C = A' \sqcup \sqcup T_i B' \) is produced. \( C \) is the shuffle of \( A' \) with \( B' \) on the set \( T_i \) of trajectories.
- \( i_o \) is the output cell.

The symbols \( \text{here, out, in}_j, 1 \leq j \leq m \), are called target commands or target indications. The objects can be transported through membranes according to the targets. Specifically, suppose \( (C, \text{tar}) \) is present in the right-hand side of a rule in membrane \( i \). If \( \text{here} \in \text{tar} \), one copy of object \( C \) is placed in the same region \( i \) where the rule is applied. If \( \text{out} \in \text{tar} \), one copy of object \( C \) is moved to the region immediately outside membrane \( i \). If \( \text{in}_j \in \text{tar} \), one copy of object \( C \) should be moved to the membrane with label \( j \), provided that this membrane is immediately inside membrane \( i \).

Rules from sets of \( R_i, 1 \leq i \leq m \), are applied to sets of languages in corresponding region of \( \mu \) synchronously, in a non-deterministic maximally parallel manner. The \( m \)-tuple of sets of languages present at any moment in the \( m \) regions of \( \Pi \) constitutes the configuration of the system at that moment. The \( m \)-tuple \( (L_1, \cdots, L_m) \) is the initial configuration of \( \Pi \). A transition between configurations is governed by application
of evolution rules, all of which are based on shuffle operation as defined above. The results of shuffle operations are the only new products of each rule. A sequence of transitions between configurations of a given system \( \Pi \) is called a computation with respect to \( \Pi \). A computation is successful if and only if it halts: there is no rule applicable to the objects present in the last configuration. The computational result defined here is the union set of objects (languages) which are present in cell \( i_o \) in the halting configuration.

If there are some objects that exist in the output membrane in the initial configuration and remain unchanged during the whole computation, they are not included in the computational result.

The language generated by a P system with shuffle operation on string languages \( \Pi \) is denoted by \( L(\Pi) \). The family of all languages \( L(\Pi) \) generated by systems \( \Pi \) as above, with at most \( m \) membranes, is denoted by \( LLP_m(FL_1, FL_2, \text{type}) \), where the second \( L \) indicates that the objects are languages, \( FL_1 \) is the family from which languages \( L_i \) are taken, \( FL_2 \) is the family from which the sets of trajectories are taken, and \( \text{type} \) indicates the type of rules, \( \text{type} \in \{\lambda, \text{cat}\} \); while the system is with catalytic-like rules, it is denoted by \( LLP_m(FL_1, FL_2, \text{cat}) \).

### 3.1.1 The Language Generative Power of P System with Shuffle Operation on String Languages

In this subsection, the language generative power of P system with shuffle operation on string languages is investigated. When comparing with Theorem 1 in Section 2, the following theorems show that the shuffle operation has more power in the framework of P systems than in the standard set-up.

**Theorem 2.** \( LLP_3(\text{REG}, \text{REG}, \text{cat}) - \text{REG} \neq \emptyset \).

**Proof.** Let us consider the system with catalytic-like rules

\[
\Pi_1 = (V, T, \mu, L_1, L_2, L_3, T_1, T_2, T_3, R_1, R_2, R_3, i_o),
\]

where:

- \( V = \{a, b\}; T = \{r, u\} \);
- \( \mu = [[2][3]]_1 \);
- \( L_1 = \{\{a\}, \{b\}\}, L_2 = \{\{b\}\}, L_3 = \emptyset \);
- \( T_1 = \{ru^n \mid n \geq 1\}, T_2 = \{r^nu \mid n \geq 1\}, T_3 = \emptyset \);
- \( R_1: \{a\}_r \{a^{n-1}b^n \mid n \geq 1\}_u \rightarrow (C', \text{in}_j), j = 2, 3, R_2: \{a^n b^n \mid n \geq 1\}_r \{b\}_u \rightarrow (C'' \text{out})\{b\}, R_3 = \emptyset \);
- \( i_o = 3 \).
The initial configuration of this system is shown in Fig. 1. Initially, there are objects \{a\} and \{b\} in membrane 1. According to \(T_1\), one rule of \(R_1\) is chosen non-deterministically, and object \{ab\} is sent to membrane 2 or membrane 3 in a non-deterministic way. If it reaches membrane 3, there is no rule applicable in the next step, the computation halts, and object \{ab\} is obtained in the output membrane.

If \{ab\} reaches membrane 2, by using the rule \(\{a^n b^n \mid n \geq 1\}_r\{b\}_u \rightarrow (C', \text{out})\{b\}\), object \(ab^2\) is introduced and sent to membrane 1. Object \(ab^2\) is sent to membrane 2 or membrane 3 in a non-deterministic way by using rule \(\{a\}_r\{b^2\}_u \rightarrow (C', \text{in}_j), j = 2, 3\). If it reaches membrane 3, the computation halts and there is object \(ab^2\) in the output membrane. If it reaches membrane 2, the computation continues in the same way as described above. According to the definition of the computational result, system \(\Pi_1\) generates language \(L = \{ab\} \cup \{a^2b^2\} \cup \cdots \cup \{a^n b^n \mid n \geq 1\} \cup \cdots = \{a^n b^n \mid n \geq 1\}\). Language \(\{a^n b^n \mid n \geq 1\}\) is context-free, but not regular. Therefore, \(\text{LLP}_3(\text{REG}, \text{REG}, \text{cat}) - \text{REG} \neq \emptyset\), and this concludes the proof.

**Theorem 3.** \(\text{LLP}_2(\text{REG}, \text{CF}) - \text{CF} \neq \emptyset\).

**Proof.** Let us consider a system without catalytic-like rules

\[\Pi_2 = (V, T, \mu, L_1, L_2, T_1, T_2, R_1, R_2, i_o),\]

where:

- \(V = \{a, b, c\}; \) \(T = \{r, u\};\)
- \(\mu = \{[\_], 2\};\)
- \(L_1 = \{\{a^n \mid n \geq 1\}, \{b^n \mid n \geq 1\}\}, \) \(L_2 = \{\{c^n \mid n \geq 1\}\};\)
- \(T_1 = \{r^u a^n \mid n \geq 1\}, \) \(T_2 = \{r^u a^n \mid n \geq 1\};\)
- \(R_1: \{a^n \mid n \geq 1\}_r\{b^n \mid n \geq 1\}_u \rightarrow (C', \text{out}), \)
- \(R_2: \{a^n b^n \mid n \geq 1\}_r\{c^n \mid n \geq 1\}_u \rightarrow (C'', \text{in});\)
The initial configuration of this system is shown in Fig. 2.

![Figure 2: Initial structure of system $\Pi_2$](image)

The computation starts from membrane 1. By using rule $\{a^n | n \geq 1\}_u \rightarrow \text{(C', out)}$ object $\{a^n b^n | n \geq 1\}$ is sent out. In the next step, by using rule $\{a^n b^n | n \geq 1\}_u \rightarrow \text{(C'', in)}$, language $\{a^n b^n c^n | n \geq 1\}$ is obtained as the computational result of $\Pi_2$. That is, $L(\Pi_2) = \{a^n b^n c^n | n \geq 1\}$.

As we know, $\{a^n b^n c^n | n \geq 1\}$ is context-sensitive, but not context-free. Therefore, $\text{LLP}_2(\text{REG, CF}) - \text{CF} \neq \emptyset$, and this concludes the proof.

**Theorem 4.** There are $P$ systems $\Pi$ with shuffle operation of degree two, with regular initial languages, one regular and one context-free sets of trajectories and catalytic-like rules such that $L(\Pi)$ is not context-free.

**Proof.** Let us consider a system with catalytic-like rules

$\Pi_3 = (V, T, \mu, L_1, L_2, T_1, T_2, R_1, R_2, i_o),$

where:

- $V = \{a, b, c\}; T = \{r, u\};$
- $\mu = [[1] 2];$
- $L_1 = \{\{a\}, \{b\}\}, L_2 = \{\{b\}, \{c^n | n \geq 1\}\};$
- $T_1 = \{ru^n | n \geq 1\}, T_2 = \{r^n u, r^{2n} u^n | n \geq 1\};$
- $R_1: \{a\}_r \{a^n b^n | n \geq 1\}_u \rightarrow \{a\}(C', \text{in}),$
  $R_2: \{a^n b^n | n \geq 1\}_r \{b\}_u \rightarrow (C'', \text{out})\{b\},$
  $\{a^n b^n | n \geq 1\}_r \{c^n | m \geq 1\}_u \rightarrow C''\{c^n | m \geq 1\};$
- $i_o = 2.$
The initial configuration of this system is shown in Fig. 3. Obviously, system $\Pi_3$ generates language $L = \{abc\} \cup \{a^2b^2c^2\} \cup \cdots \cup \{a^n b^n c^n\} \cup \cdots = \{a^n b^n c^n \mid n \geq 1\}$. We do not describe the process in details. That is, $L(\Pi_3) = \{a^n b^n c^n \mid n \geq 1\}$. Clearly, this language is not context-free. This concludes the proof.

Figure 3: Initial structure of system $\Pi_3$

### 3.2 P System with Shuffle Operation on Picture Languages

**Definition 7** A P system with shuffle operation on picture languages is defined as:

$$\Pi = (V, T, \mu, L_1, \cdots, L_m, T_1, \cdots, T_m, R_1, \cdots, R_m, i_0),$$

where:

- $V$ is an alphabet;
- $T = T_c \cup T_r$ where $T_c = \{r, u\}$, $T_r = \{l, d\}$, is the control alphabet;
- $\mu$ is a membrane structure consisting of $m$ membranes;
- $L_1, \cdots, L_m$ are initial sets of languages associated with regions $1, \cdots, m$ of $\mu$, respectively;
- $T_i = T_{ic} \cup T_{ir}, T_{ic} \subseteq T^*_c, T_{ir} \subseteq T^*_r, 1 \leq i \leq m$ are sets of trajectories associated with regions $1, \cdots, m$ of $\mu$, respectively;
- $R_i$ is a finite set of evolution rules associated with regions $i$ of $\mu$; and evolution rules are of the forms: $A_{h_1} B_{h_2} \xrightarrow{T_{ic}} (C, \text{tar})$, $A_{h_1} B_{h_2} \xrightarrow{T_{ir}} (C, \text{tar})$, or catalytic-like rules: $A_{h_1} B_{h_2} \xrightarrow{T_{ic}} (A, \text{here})(C, \text{tar})$, $A_{h_1} B_{h_2} \xrightarrow{T_{ir}} (A, \text{here})(C, \text{tar})$ or $A_{h_1} B_{h_2} \xrightarrow{T_{ic}} (C, \text{tar})(B, \text{here})$, $A_{h_1} B_{h_2} \xrightarrow{T_{ir}} (C, \text{tar})(B, \text{here})$, where $A, B, C$ are picture languages defined over alphabet $V$, $h_1, h_2 \in T_c$ or $h_1, h_2 \in T_r$, $h_1 \neq h_2$, and $\text{tar} \subseteq \{\text{here}, \text{out}, \text{in}_j \mid 1 \leq j \leq m\}$.
A, B' ⊆ B, A' ⊔⊔ T_i, B' ≠ ∅, then A', B' can be assigned to one rule of the six forms non-deterministically, and C = A' ⊔⊔ T_i, B' is produced. C is the shuffle of A' with B' on the set T_i of trajectories.

- io is the output cell.

The symbols here, out, in_j, 1 ≤ j ≤ m, are target commands which are the same as that defined in Definition 6. Rules from sets of R_i, 1 ≤ i ≤ m, are applied to sets of languages in corresponding region of μ synchronously, in a non-deterministic maximally parallel manner. The m-tuple of sets of languages present at any moment in the m regions of Π constitutes the configuration of the system at that moment. The m-tuple (L_1, ⋯, L_m) is the initial configuration of Π. A transition between configurations is governed by application of evolution rules, all of which are based on shuffle operation as defined above. A sequence of transitions between configurations of a given system Π is called a computation with respect to Π. A computation is successful if and only if it halts; there is no rule applicable to the objects present in the last configuration. The computational result is defined as the union set of objects (languages) which are present in cell io in the halting configuration. If there are some objects that exist in the output membrane in the initial configuration and remain unchanged during the whole computation, they are not included in the computational result. The language generated by a P system with shuffle operation on picture languages Π is denoted by L(Π). The family of all languages L(Π) generated by systems Π as above, with at most m membranes, is denoted by LL^pP_m(FL_1, FL_2, type); while the system is with catalytic-like rules, it is denoted by LL^pP_m(FL_1, FL_2, cat).

3.2.1 The Language Generative Power of P System with Shuffle Operation on Picture Languages

In this subsection, the language generative power of P system with shuffle operation on picture languages is investigated. Some examples are given to prove that this kind of P system can generate picture languages that can be generated by Siromoney matrix grammars [Siromoney et al. 1972] or array-rewriting P systems [Ceterchi et al. 2003].

Example 1 Consider the system

Π_4 = \{V, T, μ, L_1, L_2, L_3, L_4, T_1, T_2, T_3, T_4, R_1, R_2, R_3, R_4, io\},

where:

- V = \{a, b\}; T = \{r, u, l, d\};
- μ = \[];\[];\[];\[
- L_1 = \{b\}, \{b\}, L_2 = \{(a, b), \{b\}\}, L_3 = ∅, L_4 = \{(a)\}.
\[ T_{1r} = \{ld^n | n \geq 1\}, T_{2r} = \{r^n | n \geq 1\}, T_{4c} = \{r^n | n \geq 1\}, T_{3r} = \{ld^n | n \geq 1\}, T_{1c} = T_{2r} = T_{3c} = T_{3r} = \emptyset; \]

\[ R_1: \{b\}_1 \begin{pmatrix} b \\ \vdots \\ b \end{pmatrix}_{n \times 1} \xrightarrow{T_{4c}} \{b\}(C', \text{here}, \text{out}). \]

\[ R_2: \{(a, b, \ldots, b)_{1 \times n} | n \geq 2\}_r \{b\}_u \xrightarrow{T_{4c}} (C'', \text{here}, \text{out}) \{b\}, \]

\[ R_3 = \emptyset, \]

\[ R_4: \begin{pmatrix} a \\ a \\ a \end{pmatrix}_r \begin{pmatrix} b \\ a \\ a \end{pmatrix}_{n \times 1} \xrightarrow{T_{4c}} (C'', \text{here}) \begin{pmatrix} b \\ a \end{pmatrix}, \]

\[ \begin{pmatrix} a \ b \ldots b \\ \vdots \\ a \ a \ a \end{pmatrix}_{m \times n} \xrightarrow{T_{4c}} (C^{(4)}, \text{here}) \begin{pmatrix} b \\ a \end{pmatrix}_{m \times 1}, \]

\[ \{\begin{pmatrix} b \\ \vdots \\ b \end{pmatrix}_{n \times 1} \xrightarrow{T_{4c}} (C^{(4)}, \text{in}_3) \} \begin{pmatrix} b \\ a \end{pmatrix}_{m \times 1} \]

\[ \{\begin{pmatrix} b \\ \vdots \\ b \end{pmatrix}_{m \times 1} \xrightarrow{T_{4c}} \} \begin{pmatrix} b \\ a \end{pmatrix}_{m \times 1} \]

\[ \{(a, b, \ldots, b)_{1 \times n} | n \geq 2\}_l \begin{pmatrix} a \ b \ldots b \\ \vdots \\ a \ a \ a \end{pmatrix}_{n \times n} \xrightarrow{T_{4c}} \}

\[ \{(a, b, \ldots, b)_{1 \times n} | n \geq 2\}(C^{(5)}, \text{here}), \]

\[ \{(a, b, \ldots, b)_{1 \times n} | n \geq 2\}_l \begin{pmatrix} a \ b \ldots b \\ \vdots \\ a \ a \ a \end{pmatrix}_{m \times n} \xrightarrow{T_{4c}} \}

\[ \{(a, b, \ldots, b)_{1 \times n} | n \geq 2\}_l \begin{pmatrix} a \ b \ldots b \\ \vdots \\ a \ a \ a \end{pmatrix}_{m \times n} \xrightarrow{T_{4c}} \} \]
The initial configuration of this system is shown in Fig. 4. In membrane 1, by applying rules in $R_1$, pictures of the form
\[
\begin{pmatrix}
   b \\
   \vdots \\
   b \\
   a
\end{pmatrix}_{m \times 1}
\]
are generated and sent out to membrane 4. At the same time, by applying rules in $R_2$ pictures of the form $(a, b, \ldots, b)_{1 \times n}$ are generated in membrane 2 and also sent out to membrane 4. They shuffle in the skin membrane and all kinds of tokens $L$ (Fig. 5) can be obtained. So the system can generate the family of all $L$-shaped angles, which is a $RML$.

\[
\{(a, b, \ldots, b)_{1 \times n} \mid n \geq 2\}(C^{(5)}, i_{o3}); \quad \text{and} \quad i_o = 3.
\]

\textbf{Example 2} Consider the system

\[
\Pi_5 = \{V, T, \mu, L_1, L_2, L_3, L_4, T_1, T_2, T_3, T_4, R_1, R_2, R_3, R_4, i_o\}.
\]
where $V, T, \mu, L_1, L_2, L_3, L_4, T_1, T_2, T_3, T_4, R_1, R_2, R_3, i_o$ are the same as those in Example 1.

Example 3 Consider the system

$$\Pi_6 = \{V, T, \mu, L_1, \ldots, L_6, T_1, \ldots, T_6, R_1, \ldots, R_6, i_o\},$$

where:

- $V = \{a, b\}; T = \{r, u, l, d\};$
- $\mu = [[2][3][5][6][4]];$
- $L_1 = \{(a), (b, b)\}, L_2 = \{(a), \{a\}, \{a\}, \{b, b\}\}, L_3 = \{(b, b)\}, L_5 = \{(a), \{(a, a)\}\}, L_4 = L_6 = \emptyset;$
- $T_{1c} = \{r^n u r^n \mid n \geq 1\}, T_{2r} = \{l^m d \mid n \geq 2\}, T_{3c} = \{r^n u \mid n \geq 2\},$
- $T_{3r} = \{l^m d \mid n \geq 2\}, T_{4r} = \{l^m d \mid n \geq 1\}, T_{5c} = \{r^n u \mid n \geq 2\},$
- $T_{1r} = T_{2c} = T_{3c} = T_{5c} = T_{6r} = T_{6r} = \emptyset;$
- $R_1: \{(b \ldots b) \mid m, n \geq 1\}, R_2: \{(a) \mid m \geq 1\},$
\[
\begin{pmatrix}
\vdots \\
\vdots \\
\vdots
\end{pmatrix}
_{m \times 1},
\]
\[
\begin{pmatrix}
a \\
\vdots \\
a
\end{pmatrix}
_{m \times 1}
\]

\[\begin{pmatrix}
\vdots \\
\vdots \\
\vdots
\end{pmatrix}
_{m \times 1},
\]
\[
\begin{pmatrix}
\vdots \\
\vdots \\
\vdots
\end{pmatrix}
_{m \times 1}
\]

\[\begin{pmatrix}
\vdots \\
\vdots \\
\vdots
\end{pmatrix}
_{m \times 1}
\]
\[
\begin{pmatrix}
\vdots \\
\vdots \\
\vdots
\end{pmatrix}
_{m \times 1}
\]

\[\begin{pmatrix}
\vdots \\
\vdots \\
\vdots
\end{pmatrix}
_{m \times 1}
\]
\[
\begin{pmatrix}
\vdots \\
\vdots \\
\vdots
\end{pmatrix}
_{m \times 1}
\]

\[\begin{pmatrix}
\vdots \\
\vdots \\
\vdots
\end{pmatrix}
_{m \times 1}
\]
\[
\begin{pmatrix}
\vdots \\
\vdots \\
\vdots
\end{pmatrix}
_{m \times 1}
\]

\[\begin{pmatrix}
\vdots \\
\vdots \\
\vdots
\end{pmatrix}
_{m \times 1}
\]
\[
\begin{pmatrix}
\vdots \\
\vdots \\
\vdots
\end{pmatrix}
_{m \times 1}
\]

\[\begin{pmatrix}
\vdots \\
\vdots \\
\vdots
\end{pmatrix}
_{m \times 1}
\]
\[
\begin{pmatrix}
\vdots \\
\vdots \\
\vdots
\end{pmatrix}
_{m \times 1}
\]

\[\begin{pmatrix}
\vdots \\
\vdots \\
\vdots
\end{pmatrix}
_{m \times 1}
\]
\[
\begin{pmatrix}
\vdots \\
\vdots \\
\vdots
\end{pmatrix}
_{m \times 1}
\]

\[\begin{pmatrix}
\vdots \\
\vdots \\
\vdots
\end{pmatrix}
_{m \times 1}
\]
\[
\begin{pmatrix}
\vdots \\
\vdots \\
\vdots
\end{pmatrix}
_{m \times 1}
\]

\[\begin{pmatrix}
\vdots \\
\vdots \\
\vdots
\end{pmatrix}
_{m \times 1}
\]
\[
\begin{pmatrix}
\vdots \\
\vdots \\
\vdots
\end{pmatrix}
_{m \times 1}
\]

The initial configuration of system \(\Pi_6\) is shown in Fig. 7. In membrane 3, \(m \times n\) pictures of the form
\[
\begin{pmatrix}
b \ldots b \\
\vdots \\
b \ldots b
\end{pmatrix}
_{m \times n}
\]
are generated and sent out to the skin membrane. At the same time, \(m \times 1\) pictures
\[
\begin{pmatrix}
a \\
\vdots \\
a
\end{pmatrix}
_{m \times 1}
\]
are generated in membrane 2 and also sent out to the skin membrane. So they shuffle and produce pictures of the form

$$\begin{pmatrix}
\ldots & b & a & b & \ldots & b \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
& & b & a & b & \ldots & b \\
\end{pmatrix}_{m \times n}$$

according to $T_{1c}$. These pictures are sent into membrane 4. Pictures of the form $(a, \ldots, a)_{1 \times n}$ are generated in membrane 5 and also sent out to membrane 4. According to $T_{4r}$, they are attached to the top of pictures

$$\begin{pmatrix}
\ldots & b & a & b & \ldots & b \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
& & b & a & b & \ldots & b \\
\end{pmatrix}_{m \times n}$$

Hence, the above system can generate the set of all digitalized pictures of the tokens $T$ (Fig. 6), which is a $CFML$.

**Example 4** Consider the system

$$\Pi_7 = \{V, T, \mu, L_1, \ldots, L_6, T_1, \ldots, T_6, R_1, \ldots, R_6, i_o\},$$

where:

- $V = \{a, b\}; T = \{r, u, l, d\};$
- $\mu = [[3][6][4][5]]_1;$
- $L_2 = \{(b, b), (b, b), (b, b), (b, b)\}, \{(a, a), (a, a), (a, a)\}; L_3 = \{\{a\}a\},$
\[
\{ \begin{pmatrix} a & a \\ a & a \end{pmatrix} \}, L_4 = \{ \{b\}, \{(b, b)\}, \{ \begin{pmatrix} b & b \\ b & b \end{pmatrix} \}\}, L_5 = \{ \{(a, a)\},
\{ \begin{pmatrix} a & a \\ a & a \end{pmatrix} \}\}, L_1 = L_6 = \emptyset;
\]

- \( T_{1r} = \{ ld^m | n \geq 1 \}, T_{2c} = \{ ru^n r | n \geq 1 \}, T_{3c} = \{ r^n u | n \geq 2 \}, \)
  \( T_{4c} = \{ r^n u | n \geq 2 \}, T_{4r} = \{ l^n d | n \geq 2 \}, T_{5r} = \{ ld^m | n \geq 1 \}, \)
  \( T_{1c} = T_{2r} = T_{3r} = T_{5c} = T_{6c} = T_{6r} = \emptyset; \)

- \( R_1: \{ \begin{pmatrix} a \ldots a \\ a \ldots a \end{pmatrix} \}_{2 \times n} | n \geq 3 \} | m \geq 1, n \geq 3 \} \xrightarrow{T_{1r}} \)
  \{ \begin{pmatrix} a \ldots a \\ a \ldots a \end{pmatrix} \}_{2 \times n} | n \geq 3 \}_1 \{ \begin{pmatrix} a \ldots a \\ a \ldots a \end{pmatrix} \}_{m \times n} \}

- \( R_2: \{ \begin{pmatrix} a \ldots a \\ a \ldots a \end{pmatrix} \}_{2 \times n} | n \geq 1 \} \xrightarrow{T_{2c}} \)
  \{ \begin{pmatrix} a \ldots a \\ a \ldots a \end{pmatrix} \}_{2 \times n} | n \geq 1 \}_1 \{ \begin{pmatrix} a \ldots a \\ a \ldots a \end{pmatrix} \}_{m \times n} \}

- \( R_3: \{ \begin{pmatrix} a \ldots a \\ a \ldots a \end{pmatrix} \}_{2 \times n} | n \geq 2 \} \xrightarrow{T_{3c}} \)
  \{ \begin{pmatrix} a \ldots a \\ a \ldots a \end{pmatrix} \}_{2 \times n} | n \geq 2 \}_1 \{ \begin{pmatrix} a \ldots a \\ a \ldots a \end{pmatrix} \}_{m \times n} \}

- \( R_4: \{ \begin{pmatrix} a \ldots a \\ a \ldots a \end{pmatrix} \}_{2 \times n} | n \geq 2 \} \xrightarrow{T_{4c}} \)
  \{ \begin{pmatrix} a \ldots a \\ a \ldots a \end{pmatrix} \}_{2 \times n} | n \geq 2 \}_1 \{ \begin{pmatrix} a \ldots a \\ a \ldots a \end{pmatrix} \}_{m \times n} \}

- \( R_5: \{ \begin{pmatrix} a \ldots a \\ a \ldots a \end{pmatrix} \}_{2 \times n} | n \geq 2 \} \xrightarrow{T_{5r}} \)
  \{ \begin{pmatrix} a \ldots a \\ a \ldots a \end{pmatrix} \}_{2 \times n} | n \geq 2 \}_1 \{ \begin{pmatrix} a \ldots a \\ a \ldots a \end{pmatrix} \}_{m \times n} \}

- \( R_6 = \emptyset; \)

- \( i_o = 6. \)
The initial configuration of this system is shown in Fig. 8. $m \times n$ pictures of the form
\[
\begin{pmatrix}
  b & \ldots & b \\
  \vdots & \ddots & \vdots \\
  b & \ldots & b
\end{pmatrix}_{m \times n}
\]
are generated in membrane 4 and sent out to membrane 2. At the same time, $m \times 2$ pictures of the form
\[
\begin{pmatrix}
  a & a \\
  \vdots & \vdots \\
  a & a
\end{pmatrix}_{m \times 2}
\]
are generated in membrane 5 and also sent out to membrane 2. So they shuffle and produce pictures
\[
\begin{pmatrix}
  a & b & \ldots & b & a \\
  \vdots & \vdots & \ddots & \vdots & \vdots \\
  a & b & \ldots & b & a
\end{pmatrix}_{m \times n}
\]
according to $T_{2c}$. These pictures are sent out to the skin membrane. Moreover, $2 \times n$ pictures of the form
\[
\begin{pmatrix}
  a & \ldots & a \\
  a & \ldots & a
\end{pmatrix}_{2 \times n}
\]
are generated in membrane 3 and also sent out to the skin membrane. They shuffle with pictures
\[
\begin{pmatrix}
  a & b & \ldots & b & a \\
  \vdots & \vdots & \ddots & \vdots & \vdots \\
  a & b & \ldots & b & a
\end{pmatrix}_{m \times n}
\]
according to $T_{1_T}$. So system $Π_7$ can generate all solid rectangles marked as shown in Fig. 9, which can also be generated by an array-rewriting P system [Ceterchi et al. 2003].

\[
\begin{array}{cccccccc}
\text{a} & \text{a} & \text{a} & \text{a} & \text{a} & \text{a} & \text{a} & \text{a} \\
\text{a} & \text{b} & \text{b} & \text{b} & \text{b} & \text{b} & \text{b} & \text{a} \\
\text{a} & \text{b} & \text{b} & \text{b} & \text{b} & \text{b} & \text{b} & \text{a} \\
\text{a} & \text{b} & \text{b} & \text{b} & \text{b} & \text{b} & \text{b} & \text{a} \\
\text{a} & \text{b} & \text{b} & \text{b} & \text{b} & \text{b} & \text{b} & \text{a} \\
\text{a} & \text{a} & \text{a} & \text{a} & \text{a} & \text{a} & \text{a} & \text{a} \\
\end{array}
\]

**Figure 9:** Solid rectangle

4 Conclusions and Remarks

A new kind of P system with shuffle operation and catalytic-like rules has been proposed. It has a hierarchical membrane structure with language-objects in each compartment delimited by membranes. Evolution rules, including catalytic-like rules are based on shuffle operations and deal with different languages in each region. They are applied in a maximally parallel way as usual in P systems. The computational result is defined as the union set of languages which are present in the output membrane in halting configurations. In our study, string languages and picture languages are considered as objects of such P systems, respectively.

The language generative power of P systems with shuffle operation on string languages with or without catalytic-like rules is investigated. Comparing with Theorem 1 in section 2, the shuffle operation placed in a membrane structure is more powerful than in the standard set-up. It remains open whether Theorem 2 can be extended to the system without catalytic-like rules. We also illustrate the language generative power of P system with shuffle operation on picture languages with catalytic-like rules by comparing with Siromoney matrix grammars and array-rewriting P systems. This paper is an enhanced version of [Niu et al. 2010]. Another direction which is worth investigating is the way P system with shuffle operation on picture languages generates languages without catalytic-like rules.

Acknowledgment

The authors are grateful to the reviewers for very useful comments which helped in revising the paper. The work of Y. Niu is supported by National Natural Science Foundation of China (61033003, 91130034 and 30870826), Ph.D. Programs Foundation of Ministry of Education of China (20100142110072), Fundamental Research Funds for the Central Universities (2010ZD001 and 2010MS003), and National Science Foundation of Hubei Province (2008CDB113 and 2011CDA027).
References


