Journal of Membrane Computing (2022) 4:11–20
https://doi.org/10.1007/s41965-021-00090-2

REGULAR PAPER

Array P systems and pure 2D context-free grammars with independent mode of rewriting

Somnath Bera[1] · Rodica Ceterchi[2] · Sastha Sriram[3] · K. G. Subramanian[4]

Received: 15 September 2021 / Accepted: 12 November 2021 / Published online: 26 November 2021

Abstract
Rewriting array P systems for generation of rectangular picture arrays have been considered with the rules in membranes and the application of the rules as in a pure 2D context-free grammar *(P2DCFG)* and its variants. Here, we introduce in *P2DCFG*, a different mode of rewriting of an array, which we call as *independent* mode. We then consider rewriting array P systems involving *P2DCFG* type of rules but with the independent mode of rewriting. We show that the array generative power is increased in the framework of P systems. This framework also allows for the treatment of the so-called "extended" array grammars.

Keywords Two dimensional languages · Pure context-free grammars · P systems

1 Introduction

Based on pure context-free grammars [15] which have been extensively investigated for their language generating power, a simple but effective non-isometric 2D grammar model, called pure 2D context-free grammar (*P2DCFG*) was introduced in [29, 30] in the area of two-dimensional picture languages [12, 20, 21, 33], to generate rectangular picture array languages. In a *P2DCFG*, all symbols in any column or any row of the rectangular array are rewritten at a time by equal length strings, thus maintaining the array to be rectangular. Several properties [1, 2] and variants [14, 31] of *P2DCFG* have been studied.

Somnath Bera, Rodica Ceterchi, Sastha Sriram and K. G. Subramanian contributed equally to this work.

✉ K. G. Subramanian
kgsmani1948@gmail.com

1 School of Advanced Sciences-Mathematics, Vellore Institute of Technology, Chennai 600 127, Tamil Nadu, India

2 Faculty of Mathematics and Computer Science, University of Bucharest, 14 Academiei St, Bucharest 010014, Romania

3 Department of Mathematics, School of Arts, Sciences, Humanities and Education, SASTRA Deemed University, Tanjore 613 401, Tamil Nadu, India

4 School of Mathematics, Computer Science and Engineering, Liverpool Hope University, Hope Park, Liverpool L16 9JD, UK

On the other hand, in the area of membrane computing [18, 19], a computing model based on the membrane structure and the functioning of living cells, was introduced by Păun in [17] and is now referred to as P system. This area of membrane computing has seen a vast growth both in terms of theoretical results [19] and application studies [34].

Formal language theory [22, 23], which is a classical area of theoretical computer science, has close connections with membrane computing. The two areas of membrane computing and two-dimensional picture array grammars were linked in [4] by developing an array P system for dealing with the problem of generation of two dimensional (2*D*) objects or picture arrays based on Chomsky type array grammars of the isometric variety. Several models of array P systems (see, for example, [5, 28, 32]) have been subsequently introduced and studied in the area of two-dimensional picture languages.

Freund [10] has given a very clear and detailed description of different derivation modes in P systems as well as a discussion of the role of halting conditions in the context of computational power of P systems. Here we consider pure 2D context-free grammars (*P2DCFG*), which belong to the non-isometric variety of array grammars, with a variation in the mode of rewriting of a picture array, which we call as *independent mode*. In a *P2DCFG*, at a step of a derivation, all the symbols in a column or in a row of a rectangular array are to be rewritten but in the independent mode, a single symbol but any symbol in each of the columns or in each of the rows is rewritten at a step of a derivation. The resulting class of picture languages is shown to be incomparable with the family of picture languages generated by *P2DCFG*. We then consider an array P system with objects in the regions of this P system as rectangular picture arrays and rules to generate picture arrays as *P2DCFG* kind of rules but with an independent mode of rewriting. We show that the use of two membranes gives more picture array generative power. We also provide an application of this array P system in generating certain floor designs, referred to as "kolam" patterns.

2 Preliminaries

Let T be a finite alphabet. A word or a string $w = w_1 w_2 \cdots w_n$, $(n \geq 1)$ over T is a finite sequence of symbols from T. We denote by $len(w)$, the length of the word w. The set of all words over T, including the empty word λ with no symbols, is denoted by T^*. We call words of T^* also as *row words*. For any word $w = a_1 a_2 \cdots a_n$, we denote by w^t the word w written vertically as follows and call w^t as a *column word*:

$$\begin{matrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{matrix}.$$

Since the transpose of a row is a column and vice versa, we have $(w^t)^t = w$.

A rectangular $m \times n$ array M over T, called picture array, is of the form

$$M = \begin{matrix} p_{11} & \cdots & p_{1n} \\ \vdots & \ddots & \vdots \\ p_{m1} & \cdots & p_{mn} \end{matrix}$$

where each $p_{ij} \in T, 1 \leq i \leq m,\ 1 \leq j \leq n$. The set of all picture arrays over T is denoted by T^{**}, which includes the empty array λ. We write $T^{++} = T^{**} - \{\lambda\}$.

We refer to [22, 23] for concepts related to formal languages and to [20, 21] for array grammars. For notions related to P Systems we refer to [17, 18]. We now recall a pure 2D context-free grammar introduced in [29, 30].

Definition 1 A pure 2D context-free grammar (*P2DCFG*) is a 4-tuple $G = (T, P_1, P_2, I)$ where

i) T is a finite set of symbols;
ii) P_1 is a finite set of column tables c, where c is a finite set of context-free rules of the form $a \to \alpha, a \in T, \alpha \in T^*$ satisfying the property that for any two rules $a \to \alpha$, $b \to \beta$ in c, we have $len(\alpha) = len(\beta)$; i.e. the words α and β have equal length;
iii) P_2 is a finite set of row tables r, where r is a finite set of rules of the form $d \to \gamma^t, d \in T, \gamma \in T^*$ such that for any two rules $d \to \gamma^t, e \to \delta^t$ in r, we have $len(\gamma) = len(\delta)$;
iv) $I \subseteq T^{**} - \{\lambda\}$ is a finite set of initial (axiom) arrays.

A derivation in a *P2DCFG* G is defined as follows: For $p, q \in T^{**}$, q is derived in G from a picture p, written $p \Rightarrow q$, either (i) by rewriting in parallel all the symbols in a column of p, rewriting each symbol by a rule in some column table or (ii) by rewriting in parallel all the symbols in a row of p, rewriting each symbol by a rule in some row table. All the rules used to rewrite a column (or a row) of symbols should belong to the same table. The reflexive, transitive closure of $\Rightarrow$ is denoted by $\Rightarrow^*$.

The picture language generated by G is the set of picture arrays $L(G) = \{M \in T^{**} \mid M_0 \Rightarrow^* M \text{ for some } M_0 \in I\}$. The family of picture languages generated by *P2DCFGs* is denoted by *P2DCFL*.

Example 1 Consider the *P2DCFG* $G_1 = (T, P_1, P_2, \{M_0\})$ where $T = \{a, b, d, e\}$, $P_1 = \{c_1, c_2\}, P_2 = \{r\}$ where $c_1 = \{a \to ab, d \to da\}$, $c_2 = \{a \to a, d \to e\}$,

$$r = \left\{ d \to \begin{matrix} a \\ d \end{matrix}, a \to \begin{matrix} b \\ a \end{matrix} \right\}, \text{ and } M_0 = \begin{matrix} a & b & b \\ a & b & b \\ d & a & a \end{matrix}.$$

G_1 generates a picture language L_1 consisting of picture arrays p of size (m, n), $m, n \geq 3$ with $p(i, 1) = p(m, j) = a$, for $1 \leq i \leq m - 1$, $2 \leq j \leq n$; $p(m, 1) = d$ or $p(m, 1) = e$; $p(i, j) = b$, otherwise. We note that a derivation in G_1, starting from the axiom array M_0, generates picture arrays of the forms

$$\begin{matrix} a & b & \cdots & b \\ \vdots & \vdots & \ddots & \vdots \\ a & b & \cdots & b \\ d & a & \cdots & a \end{matrix} \quad \text{and} \quad \begin{matrix} a & b & \cdots & b \\ \vdots & \vdots & \ddots & \vdots \\ a & b & \cdots & b \\ e & a & \cdots & a \end{matrix}$$

since the column table c_1 is applicable to only the leftmost column $(a \cdots ad)^t$, rewriting in parallel all the symbols a and d in that column, thereby adding the symbol b to the immediate right of each a while adding the symbol a to the immediate right of d. Likewise, the row table r is applicable to only the bottommost row and adds a row of the form $ab \cdots b$ just above it. Likewise the column table c_2 is applicable to only the leftmost column $(a \cdots ad)^t$, rewriting in parallel all the symbols a as a itself but changing d in that column as e, and after this no table of rules is applicable.

3 Pure 2D context-free grammar in independent mode

We now introduce a different mode of rewriting in a pure 2D context-free grammar, which we call as independent mode, based on a corresponding notion in the study of two-dimensional insertion systems considered in [11].

Definition 2 A pure 2D context-free grammar in independent mode (abbreviated as *IP2DCFG*), $G_I = (T, P_1, P_2, I)$ has its components T, P_1, P_2, I as in the *P2DCFG* in Definition 1, with a difference in the mode of rewriting of a picture array, which we call as independent mode, and which is done as described below:

A direct derivation of a picture array M_2 from an $m \times n$ picture array M_1, written as $M_1 \Rightarrow_i M_2$, is done in the following manner: In applying the rules of a row (respy. column) table r (respy. c) to the $m \times n$ picture array M_1, only one symbol in each of the m rows (respy. n columns) is rewritten at a time and it can be any symbol in that row (respy. column). All the symbols (chosen for rewriting) should have rules in the row table r (respy. column table c). Otherwise, the table of rules is not applicable. If a picture array Y is obtained from a picture array X using a *IP2DCFG*, through a sequence of direct derivation steps, we write $X \Rightarrow_i^* Y$. Note that the lengths of the right sides of all the rules in a column table (respy. a row table), are the same and so the derived array is also a rectangular array.

The picture language generated by a *IP2DCFG* G_I is the set of picture arrays $L(G) = \{M \in T^{**} \mid M_0 \Rightarrow_i^* M$ for some $M_0 \in I\}$. The family of picture languages generated by *IP2DCFGs* is denoted by *IP2DCFL*.

We illustrate with an example.

Example 2 Consider the *IP2DCFG* $G_2 = (T, P_1, P_2, \{M_0\})$ where $T = \{a, b, d, e, x\}$, $P_1 = \{c_1, c_2\}, P_2 = \{r\}$ where $c_1 = \{a \to ab, x \to xx, d \to bd\}$, $c_2 = \{a \to a, x \to x, d \to e\}$, $r = \left\{ a \to \begin{matrix} a \\ b \end{matrix}, x \to \begin{matrix} x \\ x \end{matrix}, d \to \begin{matrix} b \\ d \end{matrix} \right\}$, and

$$M_0 = \begin{matrix} a & b & b \\ b & x & b \\ b & b & d \end{matrix}.$$

G_2 generates a picture language L_2 consisting of picture arrays p of size $m \times n$, $m, n \geq 3$ with $p(1,1) = a$, $p(1,j) = p(i,1) = b$, for $2 \leq j \leq n$ and $2 \leq i \leq m$, $p(m,j) = p(i,n) = b$, for $2 \leq j \leq n-1$ and $2 \leq i \leq m-1$, $p(m,n) = d$ or e, $p(i,j) = x$, otherwise. We note that a derivation in G_2, starting from the axiom array M_0, generates picture arrays of the forms

$$\begin{matrix} a & b & \cdots & b & b \\ b & x & \cdots & x & b \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ b & x & \cdots & x & b \\ b & b & \cdots & b & d \end{matrix} \quad \text{and} \quad \begin{matrix} a & b & \cdots & b & b \\ b & x & \cdots & x & b \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ b & x & \cdots & x & b \\ b & b & \cdots & b & e \end{matrix}.$$

A sample derivation $M_0 \Rightarrow_i^* M$ using the tables c_1, r, r, c_2 in this order in independent mode is shown in Fig. 1.

We have indicated the symbols rewritten by enclosing these in rectangular boxes. Note that when a column table of rules is used, a symbol in each row (not necessarily in the same column) is rewritten while a symbol in each column (not necessarily in the same row) is rewritten when a row table of rules is used.

Theorem 1 *The families of P2DCFL and IP2DCFL are incomparable but not disjoint.*

Proof That the families are not disjoint can be seen from the picture language consisting of $m \times n \, (m, n \geq 2)$ picture arrays p over $\{a\}$ where $p(i,j) = a$, for all $1 \leq i \leq m, 1 \leq j \leq n$, which is generated by a *P2DCFG* as well as a *IP2DCFG* with a column table of rules $c = \{a \to aa\}$, a row table of rules $\left\{ a \to \begin{matrix} a \\ a \end{matrix} \right\}$ and axiom array $\begin{matrix} a & a \\ a & a \end{matrix}$.

The incomparability can be seen as follows: Consider the picture language L_{abd} consisting of $3 \times n, (n \geq 3)$ picture arrays p_1, p_2 such that (i) $p_1(1,1) = p_1(2,1) = a$, $p_1(3,1) = d$, $p_1(3,j) = a, 2 \leq j \leq n$ and $p_1(i,j) = b$, otherwise and (ii) $p_2(1,1) = p_2(2,1) = b$, $p_2(3,1) = d$,

Fig. 1 Derivation $M_0 \Rightarrow_i^* M$

$$M_0 = \begin{matrix} \boxed{a} & b & b \\ b & \boxed{x} & b \\ b & b & \boxed{d} \end{matrix} \Rightarrow_i \begin{matrix} \boxed{a} & b & b & b \\ b & \boxed{x} & \boxed{x} & b \\ b & b & b & \boxed{d} \end{matrix} \Rightarrow_i \begin{matrix} \boxed{a} & b & b & b \\ b & x & \boxed{x} & b \\ b & \boxed{x} & x & b \\ b & b & b & \boxed{d} \end{matrix} \Rightarrow_i \begin{matrix} \boxed{a} & b & b & b \\ b & x & \boxed{x} & b \\ b & \boxed{x} & x & b \\ b & x & \boxed{x} & b \\ b & b & b & \boxed{d} \end{matrix}$$

$$\Rightarrow_i \begin{matrix} a & b & b & b \\ b & x & x & b \\ b & x & x & b \\ b & x & x & b \\ b & b & b & e \end{matrix} = M$$

$p_2(3,j) = b, 2 \leq j \leq n$ and $p_2(i,j) = a$, otherwise. Two arrays M_1 and M_2 of L_{abd} are shown below.

$$M_1 = \begin{matrix} a & b & \cdots & b \\ a & b & \cdots & b \\ d & a & \cdots & a \end{matrix} \quad \text{and} \quad M_2 = \begin{matrix} b & a & \cdots & a \\ b & a & \cdots & a \\ d & b & \cdots & b \end{matrix}.$$

The picture language L_{abd} is in $P2DCFL$ generated by a $P2DCFG$ with two column tables of rules $c_1 = \{a \rightarrow ab, d \rightarrow da\}, c_2 = \{b \rightarrow ba, d \rightarrow db\}$ and axiom arrays

$$\begin{matrix} a & b & b \\ a & b & b \\ d & a & a \end{matrix} \quad \text{and} \quad \begin{matrix} b & a & a \\ b & a & a \\ d & b & b \end{matrix}.$$

But it can be seen that this picture language L_{abd} is not in $IP2DCFL$. In fact, if we assume that L_{abd} can be generated by a $IP2DCFG$, then arrays of the form p_1 can be generated in the independent mode only by a column table of rules $c = \{b \rightarrow bb, d \rightarrow da\}$, since rules for rewriting a can not be included in such a table as it will result in picture arrays not in the language. But then the column table c could be applied to a picture array of the form p_2 in the independent mode, again resulting in picture arrays not in the language. Similar reasoning can be done for picture arrays p_2. This shows that L_{abd} cannot be generated by any $IP2DCFG$.

On the other hand, consider the picture language L' consisting of $2 \times n, (n \geq 4)$ picture arrays q such that $q(1,2) = q(2,3) = a$, $q(i,j) = b$, otherwise. An array in L' is shown below:

$$\begin{matrix} b & a & b & b & \cdots & b \\ b & b & a & b & \cdots & b \end{matrix}.$$

The picture language L' is in $IP2DCFL$ generated by a $IP2DCFG$ with an axiom array $\begin{matrix} b & a & b & b \\ b & b & a & b \end{matrix}$ and a column table $c = \{a \rightarrow ab\}$. It cannot be generated by any $P2DCFG$. If there is such a $P2DCFG$, then due to the requirement that in a $P2DCFG$, all symbols in a single column should be rewritten at a time, this grammar should have column tables of rules to rewrite one or more of the columns $\begin{matrix} b \\ b \end{matrix}, \begin{matrix} a \\ b \end{matrix}$ and $\begin{matrix} b \\ a \end{matrix}$. This means that the grammar should have a column table having a pure context-free rule to rewrite only b and / or a column table of pure context-free rules to rewrite a and b. But it is clear that having such column tables of rules, the grammar will generate picture arrays not in the language L'. For instance a column table with rule for b will allow rewriting of the first column of $b's$ as well, yielding picture arrays not in the language. □

4 Array P system based on *IP2DCFG*

We now introduce an array P system model with the membranes of the P system containing picture array objects and column or row tables of rules as in $IP2DCFG$ in the sense that the rewriting is in independent mode.

Definition 3 An array P system (of degree $m \geq 1$) with $IP2DCFG$ kind of rules is a construct

$$\Pi = (T, \mu, A_1, \cdots, A_m, R_1, \cdots, R_m, i_o),$$

where T is the alphabet consisting of terminal symbols, μ is a membrane structure with m membranes labelled in a one-to-one manner with $1, 2, \cdots, m$; $A_1, \cdots, A_m$ are finite sets (can be empty) of rectangular picture arrays over T with A_i in the membrane or region labelled i for $1 \leq i \leq m$; $R_1, \cdots, R_m$ are finite sets of column tables or row tables of context-free rules over T (as in a $IP2DCFG$) with R_i in the membrane or region labelled i for $1 \leq i \leq m$. A region can contain both column tables of rules and row tables of rules. The application of a column or row table is as done in a $IP2DCFG$. The tables have attached targets *here, out, in,* in_j (in general, *here* is omitted) and i_o is the label of the output membrane which is an elementary membrane of μ.

A computation in Π is done as in an array-rewriting P system [4] with the successful computations being the halting ones; each rectangular picture array in each region of the system, which can be rewritten by a column table of rules or a row table of rules, associated with that region, should be rewritten. This means that a region can contain column tables of rules and/or row tables of rules but one table (column or row) of rules is applied at a time to a picture array and the rewriting is done as in a $IP2DCFG$. The picture array obtained by rewriting is retained in the same region if the target associated with the table used is *here* or sent to an immediate outer region (respy. directly inner region, nondeterministically chosen), if the target is *out* (respy. *in*). If the target is in_j then the array is immediately sent to a directly inner membrane with label j. If no internal membrane exists in a region, then a table with the target indication *in* cannot be used.

A computation is successful only if it stops, that is, a configuration, called a halting configuration, is reached where no table of rules can be applied to the existing arrays in the regions. The result of a successful computation consists of rectangular picture arrays over T collected in the output membrane with label i_o in the halting configuration. Note that all the picture arrays that stay at the output membrane in the halting configuration will belong to the picture language since there are only one kind of symbols, namely terminal symbols.

The set of all picture arrays generated by such a system Π is denoted by $IAL(\Pi)$. The family of all array languages $IAL(\Pi)$ generated by such systems Π as above, with at most m membranes, is denoted by $IAP_m(IP2DCFG)$.

Example 3 Consider the picture language L_{sq} consisting of square sized $n \times n, n \geq 3$, picture arrays p with $p(1,1) = a, p(1,j) = p(i,1) = b$, for $2 \leq j \leq n$ and $2 \leq i \leq n$, $p(n,j) = p(i,n) = b$, for $2 \leq j \leq n-1$ and $2 \leq i \leq n-1$, $p(n,n) = e$, $p(i,j) = x$, otherwise. A picture array in L_{sq} is shown below.

$$\begin{array}{llllll} a & b & \cdots & b & b \\ b & x & \cdots & x & b \\ b & x & \cdots & x & b \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ b & x & \cdots & x & b \\ b & b & \cdots & b & e \end{array}$$

We construct an array P system Π_{sq} with only one membrane and with the only region containing tables having $IP2DCFG$ kind of rules applied in independent mode. Π_{sq} is given by

$\Pi_{sq} = (T, \mu, A_1, R_1, 1)$,

where

i) $T = \{a, b, d_1, d_2, e, x\}$
ii) $\mu = [_1\ ]_1$
iii) $A_1 = \left\{ \begin{array}{lll} a & b & b \\ b & x & b \\ b & b & d_1 \end{array} \right\}$,
iv) R_1 consists of a row table r and two column tables c_1 and c_2, each with target *here*. The tables of rules are given as follows:

$$c_1 = \{a \to ab, d_1 \to bd_2, x \to xx\}, c_2 = \{a \to a, d_1 \to e, x \to x\},$$

$$r = \{a \to \begin{array}{c} a \\ b \end{array}, x \to \begin{array}{c} x \\ x \end{array}, d_2 \to \begin{array}{c} b \\ d_1 \end{array}\}.$$

In the array P system Π_{sq}, the only membrane or region labelled 1 which is also the output membrane, initially, contains the array

$$\begin{array}{lll} a & b & b \\ b & x & b \\ b & b & d_1 \end{array}.$$

If the applicable column table c_2 is applied, then the array generated is

$$\begin{array}{lll} a & b & b \\ b & x & b \\ b & b & e \end{array}$$

which is collected in the language as membrane 1 is the output membrane and no table of rules is applicable in the region at this moment with the computation coming to a halt. If the column table c_1 (instead of c_2) is applied to the axiom array in region 1, then the array generated is

$$\begin{array}{llll} a & b & b & b \\ b & x & x & b \\ b & b & b & d_2 \end{array}.$$

The row table r can be applied now which generates the array

$$\begin{array}{llll} a & b & b & b \\ b & x & x & b \\ b & x & x & b \\ b & b & b & d_1 \end{array}$$

and the process can be repeated. Application of the rules of c_1 followed by the rules of r with both the tables of rules applied an equal number of times yields the $n \times n$ array (for some $n \geq 3$)

$$\begin{array}{lllll} a & b & \cdots & b & b \\ b & x & \cdots & x & b \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ b & x & \cdots & x & b \\ b & b & \cdots & b & d_1 \end{array}$$

If the column table c_2 is applied during the process instead of c_1 in region 1, the array

$$\begin{array}{lllll} a & b & \cdots & b & b \\ b & x & \cdots & x & b \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ b & x & \cdots & x & b \\ b & b & \cdots & b & e \end{array}$$

is generated changing the symbol d_1 to e in the array. The computation comes to a halt and the array is collected in the language generated by Π_{sq}. Note that the generated array at the halting configuration will have an equal number of rows and columns and will be an element of L_{sq}. Thus the system Π_{sq} generates the language L_{sq}.

We now examine the generative power of the array-rewriting P system with $IP2DCFG$ kind of rules in independent mode of derivation.

Theorem 2 $IP2DCFL \subset IAP_1(IP2DCFG) \subset IAP_2(IP2DCFG)$.

Proof The inclusion $IP2DCFL \subseteq IAP_1(IP2DCFG)$ can be seen as follows: Let $L \in IP2DCFL$ and $G_I = (T, P_1, P_2, I)$ be an $IP2DCFG$ generating L. We construct an array P system of degree 1 with $IP2DCFG$ kind of rules, $\Pi = (T \cup \overline{T}, [_1\ ]_1, A, R, 1)$ where $\overline{T} = \{\overline{a} \mid a \in T\}$. In other words the alphabet of Π contains all the symbols of the alphabet T of G_I and in addition contains the "barred" version of every symbol of T; A contains all the picture arrays of I but every symbol in each picture array of I is replaced by its barred version. Likewise R contains all the column tables of P_1 and all the row tables of P_2 but each symbol in the right and left sides of every rule in the tables, is replaced by the corresponding barred symbol. In addition R contains a new column table $c = \{\overline{a} \to a \mid a \in T\}$. We denote by $\overline{M}$ the array obtained from the array M by replacing each symbol of M by the corresponding barred symbol. It can be seen that for every direct derivation $M_1 \Rightarrow_i M_2$ in G_I, there is a computation in Π with the array $\overline{M_1}$ generating $\overline{M_2}$ which then generates M_2 by the application of the rules of the column table c as many times as needed and the computation halts. Hence every picture array in L generated in G_I from an axiom array in I is also computed by Π. Thus $L \in IAP_1(IP2DCFG)$. □

The proper inclusion follows from the picture array language L_{sq} in Example 3 which shows that $L_{sq} \in IAP_1(IP2DCFG)$. On the other hand this picture language cannot be generated by any $IP2DCFG$ since we need to have some control on the application of column table of rules and row table of rules to maintain the square shape of the picture arrays generated by a $IP2DCFG$. We can use some "intermediate" symbols to alternate the application of the column and row tables of rules but then this will result in picture arrays not in the language.

The inclusion $IAP_1(IP2DCFG) \subseteq IAP_2(IP2DCFG)$ in the statement of the theorem follows from the definition of the family $IAP_m(IP2DCFG)$. For the proper inclusion we consider the language L_{ab} consisting of picture arrays p of size $m \times (4n+2)$, $m \geq 2$, $n \geq 1$, where p is such that every row of p is of the form $xa^nb^na^nb^ny$ $(n \geq 1)$ over the terminal symbols $\{x, y, a, b\}$. The language L_{ab} belongs to $IAP_2(IP2DCFG)$ generated by the P system with two membranes having the membrane structure $[_1\ [_2\ ]_2\ ]_1$ and the only axiom array

$$\begin{matrix} x & d & e & y \\ x & d & e & y \end{matrix}$$

in membrane 1 initially. Membrane 1 has a row table r with target *here* and two column tables c_1, c_2 with target *in*. Membrane 2, which is the output membrane, has a column table c_3 with target *out* and another column table c_4 with target *here*.

The tables of rules are given below:

$$r = \{x \to \begin{matrix} x \\ x \end{matrix}, a \to \begin{matrix} a \\ a \end{matrix}, b \to \begin{matrix} b \\ b \end{matrix}, y \to \begin{matrix} y \\ y \end{matrix}, d \to \begin{matrix} d \\ d \end{matrix}, e \to \begin{matrix} e \\ e \end{matrix}\},$$

$c_1 = \{d \to adb\}$, $c_2 = \{e \to ab\}$, $c_3 = \{e \to aeb\}$, $c_4 = \{d \to ab\}$.

A computation can start with an application of the rules of the column table c_1 or c_2 in membrane 1 to the axiom array. If the rules of the column table c_1 in membrane 1 are applied to the axiom array, then the generated array

$$\begin{matrix} x & a & d & b & e & y \\ x & a & d & b & e & y \end{matrix}$$

will be sent to membrane 2 and if the rules of the column table c_3 are applied, then the generated array is

$$\begin{matrix} x & a & d & b & a & e & b & y \\ x & a & d & b & a & e & b & y \end{matrix}$$

which is sent back to membrane 1. This process can repeat. The rules of the row table r in membrane 1 can be applied any number of times when the array is in membrane 1 at any step of a computation. The application of the rules of the row table r will increase the number of rows of the array by one and the array will remain in membrane 1 itself. After the application of the rules of c_1 in membrane 1 and the rules of c_3 in membrane 2, with both the tables of rules applied an equal number of times, say, $n-1$ times $(n \geq 1)$ with the rules of the row table r applied $m+2$, $(m \geq 0)$ times during the process, the array arriving in membrane 1 will have $m+2$ rows and will be of the form given below:

$$\begin{matrix} x & a^{(n-1)} & d & b^{(n-1)} & a^{(n-1)} & e & b^{(n-1)} & y \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x & a^{(n-1)} & d & b^{(n-1)} & a^{(n-1)} & e & b^{(n-1)} & y \end{matrix}$$

If the rules of the column table c_2 are now applied in membrane 1 to the array mentioned above (instead of c_1), the generated array

$$\begin{matrix} x & a^{(n-1)} & d & b^{(n-1)} & a^n & b^n & y \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x & a^{(n-1)} & d & b^{(n-1)} & a^n & b^n & y \end{matrix}$$

is sent to membrane 2. Here only the column table c_4 is applicable generating the array

$$\begin{matrix} x & a^n & b^n & a^n & b^n & y \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x & a^n & b^n & a^n & b^n & y \end{matrix}$$

which is collected in the language as the computation halts.

On the other hand, if in membrane 1 at any step of a computation, the rules of the column table c_1 are applicable and applied, then the generated array is sent to the inner membrane 2 and if the rules of the column table c_4 are applied which replaces the symbol d by ab in the rows, then the array remains there but the computation does not halt as the rules of the column table c_3 are applicable generating and sending the resulting array to membrane 1. Here the rules of the column table c_2 are only applicable with the resulting array sent to the region 2 and the computation halts. Here again it is collected in the language. Note that no other sequence of application of the rules of the tables will be a correct sequence. Thus the picture language L_{ab} is generated and hence belongs to $IAP_2(IP2DCFG)$. This picture language cannot belong to $IAP_1(IP2DCFG)$ as there is only one membrane and so the technique of alternately generating the first "block" of $a^n db^n$ and the second "block" of $a^n eb^n$ using "intermediate" symbols, cannot be managed for ever as the number of rows can keep increasing, thus forcing the cardinality of the alphabet of "intermediate" symbols to grow infinitely.

Remark Analogous to the statement in Theorem 2, in the case of $P2DCFL$ [29, 30], we have $P2DCFL \subset AP_1(P2DCFG) \subset AP_2(P2DCFG)$, which corrects the erroneous statements in [32, Theorem 1, Page 1905]. The proof of this statement is similar to the proof of Theorem 2.

Based on splicing system, which is a theoretical model of DNA recombination proposed by Head [13], Berstel et al. [3] introduced flat splicing system, which is a variant of splicing system and which involves the operation known as flat splicing on words. A picture array generating model, called array flat splicing system (AFS), was introduced and investigated in [16] by defining the operation of flat splicing to picture arrays. Informally expressed, the idea is that between two adjacent columns (or to the left / right of a column) in a rectangular array M, a rectangular array N having the same number of rows as M and with a specific "prefix array" and / or a specific "suffix array" gets inserted by the "flat splicing rules". The family of picture array languages generated by AFS is denoted by $L(AFS)$. Now we exhibit a picture array language that cannot be generated by any AFS but can be generated by an array P system as in Definition 3, with $IP2DCFG$ kind of rules and only one membrane. For a formal definition of AFS and for an illustration of how this system works, we refer to [16].

Theorem 3 $IAP_1(IP2DCFG) \setminus L(AFS) \neq \emptyset$.

Proof We consider the picture language L_{xy} consisting of $n \times n, (n \geq 3)$ picture arrays p such that each of the first $(n-1)$ rows of p has the word $xa^n y, (n \geq 1)$ and the last row has the word $pa^n q$, $(n \geq 3)$ over the terminal symbols $\{x, y, p, q, a\}$. This picture language L_{xy} is not in $L(AFS)$ as it cannot be generated by any AFS. In fact if we assume that there is an AFS generating L_{xy}, then the array $\begin{matrix} x & a & y \\ x & a & y \\ p & a & q \end{matrix}$ should be an axiom array. An $n \times n$ array in L_{xy} has $(n-2), n \geq 3$ middle columns of $a's$ and to generate such arrays, the AFS needs to insert an array over a but such arrays are not members of the picture language L_{xy}. On the other hand, inserting the axiom array $\begin{matrix} x & a & y \\ x & a & y \\ p & a & q \end{matrix}$ into itself, or an array of the language into this axiom array, will yield an array with more than one column of the form $(x \cdots xp)^t$ as well as column of the form $(y \cdots yq)^t$ and hence such an array cannot be an element of L_{xy}. □

5 Array P systems and extended 2D grammars

An earliest two-dimensional picture array generating model introduced by the Siromoney group [25], originally called context-free matrix grammar [26] and subsequently referred to as $2DCFG$ in [22], involves two phases of rewriting. The first phase generates strings over "intermediate" symbols using context-free string grammar rules and in the second phase, these strings are rewritten by groups of normal form regular grammar rules such as a group of nonterminal rules of the form $A \rightarrow aB$ or a group of terminal rules of the form $A \rightarrow a$ in the vertical direction to produce the columns of a rectangular picture array over a set of terminal symbols. The "intermediate" symbols of the first phase will be the start nonterminal symbols in the second phase. This model has been extensively used in 2D grammar studies.

An extension of this 2D grammar was introduced in [27]. We do not give here the formal definition of this extended 2D grammars but informally mention the details in the extended model. The first phase is similar to the Siromoney matrix grammar [26] but the second phase has tables of rules which are sets of normal form regular nonterminal rules or sets of normal form regular terminal rules. An applicable table of rules is used (applied) for the rewriting in the second phase. We will call the extended 2D grammar model as $T2DCFG$ or $T2DCSG$ depending on whether the first phase involves a context-free grammar or a context-sensitive grammar. The corresponding families of picture languages are denoted by $T2DCFL$ and $T2DCSL$. We now compare the array P system considered here with these extended 2D grammars.

Theorem 4 (i) $IAP_2(IP2DCFG) \cap T2DCSL \neq \emptyset$.

(ii) $IAP_2(IP2DCFG) \setminus T2DCFL \neq \emptyset$.

Proof We consider the picture language L consisting of $m \times (4n+2), (m \geq 4, n \geq 1)$ picture arrays p such that the first row of p is of the form $xa^nb^na^nb^ny$ $(n \geq 1)$ and the next few rows are of the form $xz^{4n}y$ $(n \geq 1)$ followed by a row of the form $qz^{4n}r(n \geq 1)$ and the remaining rows are of the form $sz^{4n}t$ $(n \geq 1)$ over the terminal symbols $\{x, y, q, r, s, t, z, a, b\}$. This picture language is in *T2DCSL* but is not in *T2DCFL* since the first row is a strictly context-sensitive language. In fact the corresponding *T2DCSG* will generate in the first phase the *CSL* $\{XA^nB^nA^nB^nY \mid n \geq 1\}$ where X, Y, A, B are "intermediate" symbols which will serve as the start symbols for the second phase. The tables of rules in the second phase are $t_1 = \{X \to xX, A \to aZ, B \to bZ, Y \to yY\}$, $t_2 = \{X \to xX_1, Z \to zZ, Y \to yY_1\}$, $t_3 = \{X_1 \to xX_1, Z \to zZ, Y_1 \to yY_1\}$, $t_4 = \{X_1 \to qU, Z \to zZ, Y_1 \to rV\}$, $t_5 = \{U \to sU, Z \to zZ, V \to tV\}$, and $t_6 = \{U \to s, Z \to z, V \to t\}$. Application of the rules of the table t_1 will generate the first row $xa^nb^na^nb^ny$ of the array p. This can be followed by the application of the rules of the table t_2 once and t_3 certain number of times, yielding the rows of the form $xz^{4n}y$ till the rules of the table t_4 are applied. This will yield the row $qz^{4n}r$. Likewise using the table t_5 the rows of the form $sz^{4n}t$ are generated and the derivation ends when the table t_6 is used generating the required array. □

The language L belongs to $IAP_2(IP2DCFG)$ generated by the P system with two membranes having the membrane structure $[_1\ [_2\ ]_2\ ]_1$ and the only axiom array

$$\begin{matrix} x & d & e & y \\ x & z & z & y \\ q & z & z & r \\ s & z & z & t \end{matrix}$$

in membrane 1 initially. Membrane 1 has two row tables r_1, r_2 with target *here* and two column tables c_1, c_3 with target *in*. Membrane 2 has a column table c_2 with target *out* and another column table c_4 with target *here*. The output membrane is 2.

The tables of rules are given below: $r_1 = \{x \to \begin{matrix} x \\ x \end{matrix}, z \to \begin{matrix} z \\ z \end{matrix}, y \to \begin{matrix} y \\ y \end{matrix}\} r_2 = \{s \to \begin{matrix} s \\ s \end{matrix}, z \to \begin{matrix} z \\ z \end{matrix}, t \to \begin{matrix} t \\ t \end{matrix}\}$, $c_1 = \{d \to adb, z \to zzz\}$, $c_2 = \{e \to aeb, z \to zzz\}$, $c_3 = \{e \to ab, z \to zz\}$, $c_4 = \{d \to ab, z \to zz\}$. At any step of a computation, application of the rules of the row tables r_1, r_2 will add the rows $xz^{4n}y$ and $sz^{4n}t$ independently any number of times above and below the row $qz^{4n}r$ with the array remaining in membrane 1. A computation starts with an application of the rules of the column table c_1 in membrane 1 to the axiom array, sending the generated array

$$\begin{matrix} x & a & d & b & e & y \\ x & z & z & z & z & y \\ q & z & z & z & z & r \\ s & z & z & z & z & t \end{matrix}$$

to membrane 2. If the rules of the column table c_2 in membrane 2, are applied, then the generated array

$$\begin{matrix} x & a & d & b & a & e & b & y \\ x & z & z & z & z & z & z & y \\ q & z & z & z & z & z & z & r \\ s & z & z & z & z & z & z & t \end{matrix}$$

will be sent back to membrane 1. This process can repeat. If in membrane 1, the rules of the column table c_3 (instead of c_1) are applied, then the generated array is sent to the inner membrane 2 where the rules of the column table c_4 are applied which replaces the symbol d by ab in the first row and replaces one symbol z in each of the remaining rows by zz. The resulting array is in the form of the picture array p, which remains in the output region 2 and the computation halts. Here it is collected in the language. Thus the picture language L is generated. Note that no other correct sequence of applications of the tables of rules is possible.

6 Generation of floor designs

As an application of the array P System model considered in Sect. 4, we use a well-known technique [24] developed to generate certain interesting classes of floor designs, called "kolam patterns" [24, 26]. The idea is that with each symbol a of a rectangular picture array which is considered to occupy a unit square in the rectangular grid, a primitive picture pattern $i(a)$ of the "kolam pattern" is associated. The picture array is then interpreted as a "kolam pattern" replacing the symbols in the picture array generated by the array P system, by the corresponding primitive picture patterns, to yield the required "kolam pattern".

We illustrate this by considering the picture language L_{kolam} consisting of $n \times n, n \geq 3$ picture arrays p such that $p(1,1) = a, p(1,n) = y$, $p(n,1) = x, p(n,n) = e$, $p(1,j) = q, p(n,j) = r$, for $2 \leq j \leq n-1$, $p(i,1) = s, p(i,n) = t$, for $2 \leq i \leq n-1$, $p(i,j) = d$, otherwise. A picture array of L_{kolam} is shown below.

$$\begin{matrix} a & q & q & q & y \\ s & d & d & d & t \\ s & d & d & d & t \\ s & d & d & d & t \\ x & r & r & r & e \end{matrix}$$

The corresponding "kolam pattern" with the "kolam" primitives used, is shown in Fig. 2. L_{kolam} is generated by an array P system as in Definition 2 with membrane structure

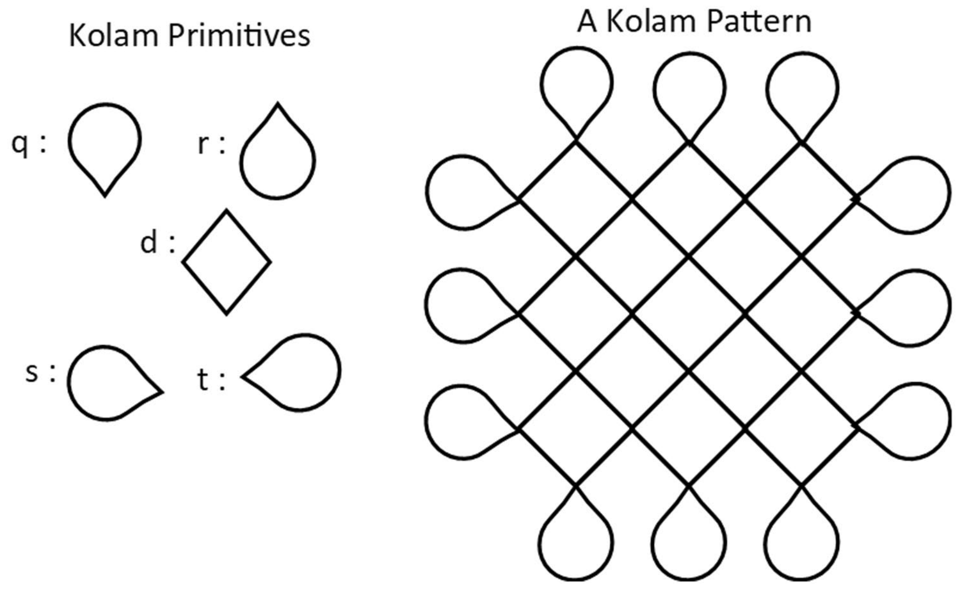

Fig. 2 A kolam pattern with its primitives

$[_1\,[_2\,]_2\,]_1$, two column tables c_1, c_2 in region 1 and a row table r in region 2 given by $c_1 = \{a \to aq, d \to dd, b \to rb\}$, $c_2 = \{a \to a, d \to d, b \to e\}, r = \left\{ a \to \begin{matrix} a \\ s \end{matrix}, d \to \begin{matrix} d \\ d \end{matrix}, b \to \begin{matrix} t \\ b \end{matrix} \right\}$ with c_1, c_2 having target *in* and r having target *out*. The only initial axiom array in membrane 1, is

$$M_0 = \begin{matrix} a & q & y \\ s & d & t \\ x & r & b \end{matrix}.$$

The primitive picture patterns associated with q, r, s, t, d are shown in Fig. 2 and $i(a) = i(e) = i(x) = i(y) = blank$.

7 Conclusions and future work

We have introduced here a new rewriting mode in pure 2D context-free grammars, called independent mode, for the generation of certain picture array languages, inspired from two-dimensional insertion systems with independent mode considered in [11]. We have shown the incomparability of the two classes of array languages, namely *P2DCFL* [29, 30] and *IP2DCFL*, introduced here. Using P systems as a control mechanism for array rewriting, we have generated square pictures of a certain type (Example 3) using only one membrane and target agreement for rules. We also sketch an application of this formalism of *P2DCFG* with *independent* mode to the generation of certain simple floor designs, known as "kolam" patterns. It will be of interest to compare the array models considered here with other kinds of array grammars [4, 6–9] which is for future work.

Acknowledgements An earlier version of this paper was presented in the International conference on Membrane Computing (ICMC 2021). The comments of the reviewers on the conference version and the journal version were very helpful in preparing this improved revised version.

Declarations

Funding No funding was received for conducting this study.

Conflict of interest The authors declare no conflict of interest.

References

1. Bersani, M. M., Frigeri, A., & Cherubini, A. (2011). On some classes of 2D languages and their relations. In J.K. Aggarwal, et al. (Eds.) *Combinatorial Image Analysis* (vol. 4958, pp. 222–234). Lecture Notes on Computer Science.
2. Bersani, M. M., Frigeri, A., & Cherubini, A. (2013). Expressiveness and complexity of regular pure two-dimensional context-free languages. *International Journal of Computer Mathematics, 90,* 1708–1733.
3. Berstel, J., Boasson, L., & Fagnot, I. (2012). Splicing systems and the Chomsky hierarchy. *Theoretical Computer Science, 436,* 2–22.
4. Ceterchi, R., Mutyam, M., Păun, Gh., & Subramanian, K.G. (2003). Array—rewriting P systems. *Natural Computing*, 2, 229–249.
5. Ceterchi, R., Subramanian, K.G., & Venkat, I. P. (2015) . Systems with Parallel Rewriting for Chain Code Picture Languages. In A. Beckmann, V. Mitrana, M. Soskova (Eds.) Evolving Computability. CiE 2015. Lecture Notes in Computer Science (vol. 9136, pp. 145–155). Champaign: Springer.
6. Fernau, H., Freund, R., Schmid, M. L., Subramanian, K. G., & Wiederhold, P. (2015). Contextual array grammars and array P systems. *Annals of Mathematics and Artificial Intelligence, 75,* 5–26.
7. Freund, R. (1994) . Control mechanisms on #-context-free array grammars. In Gh. Păun (Ed.) *Mathematical Aspects of Natural and Formal Languages* (pp. 97–137). Singapore: World Scientific.
8. Freund, R. (2000). Array Grammars. Technical Rep. 15/00, Research Group on Mathematical Linguistics (p. 164), Rovira i Virgili University, Tarragona.

9. Freund, R., Paun, Gh., & Rozenberg, G., et al. (2007). Contextual array grammars. In K. G. Subramanian (Ed.), *Formal Models* (pp. 112–136). Languages and Applications: World Scientific Publishing.
10. Freund, R. (2020). How derivation modes and halting conditions may influence the computational power of P systems. *Journal of Membrane Computing, 2,* 14–25.
11. Fujioka, K. (2014) . A two-dimensional extension of insertion systems. A.-H. Dediu et al. (Eds.) TPNC 2014, Lecture Notes on Computer Science (vol. 8890, pp. 181–192).
12. Giammarresi, D., & Restivo, A. (1997) . Two-dimensional languages, In G. Rozenberg and A. Salomaa (Eds.) Handbook of Formal Languages (vol. 3, pp. 215–267). Berlin: Springer.
13. Head, T. (1987). Formal language theory and DNA: an analysis of the generative capacity of specific recombinant behaviours. *Bulletin of Mathematical Biology, 49,* 735–759.
14. Křivka, Z., Martín-Vide, C., Meduna, A., & Subramanian, K.G. (2014). A variant of pure two-dimensional context-free grammars generating picture languages. In R.P. Barneva, V.E. Brimkov, J. Slapal (Eds.) *Combinatorial Image Analysis* (vol. 8466, pp. 123–133). Lecture Notes Computer Science. Heidelberg: Springer
15. Maurer, H. A., Salomaa, A., & Wood, D. (1980). Pure grammars. *Information Control, 44*, 47–72.
16. Pan, L., Nagar, A. K., Subramanian, K. G., & Song, B. (2016). Picture array generation using flat splicing operation. *Journal of Computational and Theoretical Nanoscience, 13,* 3568–3577.
17. Păun, Gh. (2000). Computing with membranes. *Journal of Computer and System Sciences, 61*, 108–143.
18. Păun, Gh. (2000). *Membrane Computing: An Introduction*. Berlin: Springer.
19. Păun, Gh., Rozenberg, G., & Salomaa, A. (2010). *The Oxford Handbook of Membrane Computing*. New York: Oxford University Press.
20. Rosenfeld, A. (1979). *Picture Languages—Formal Models for Picture Recognition*. New York: Academic Press.
21. Rosenfeld, A., & Siromoney, R. (1993). Picture languages—a survey. *Languages of Design, 1*, 229–245.
22. Rozenberg, G., Salomaa. A. (1997). (Eds.) *Handbook of Formal Languages* (vol. 1–3). Berlin: Springer.
23. Salomaa, A. (1973). *Formal Languages*. London: Academic Press.
24. Siromoney, G., Siromoney, R., & Krithivasan, K. (1974). Array grammars and kolam. *Computer Graphics Image Processing., 3*(1), 63–82.
25. Siromoney, R. (1991). Contributions of Professor Gift Siromoney in the area of pattern recognition. *IETE Journal of Research, 37*(5–6), 409–418.
26. Siromoney, G., Siromoney, R., & Krithivasan, K. (1972). Abstract families of matrices and picture languages. *Computer Graphics Image Processing, 1*, 234–307.
27. Siromoney, R., Subramanian, K. G., & Rangarajan, K. (1977). Parallel/Sequential rectangular arrays with tables. *International Journal of Computer Mathematics, 6*, 143–158.
28. Subramanian, K. G. (2007). P systems and picture languages. *Lecture Notes in Computer Science, 4664*, 99–109.
29. Subramanian, K. G., Ali, R. M., Geethalakshmi, M., & Nagar, A. K. (2009). Pure 2D picture grammars and languages. *Discrete Applied Mathematics, 157*(16), 3401–3411.
30. Subramanian, K. G., Nagar, A. K., & Geethalakshmi, M. (2008). Pure 2D picture grammars (P2DPG) and P2DPG with regular control. In Brimkov, et al. (Eds.), *Combinatorial Image Analysis*. Lecture Notes on Computer Science (vol. 4958, pp. 330–341).
31. Subramanian, K. G., Geethalakshmi, M., David, N. G., & Nagar, A. K. (2015). Picture array generation using pure 2D context-free grammar rules. *Lecture Notes on Computer Science, 9448*, 187–201.
32. Subramanian, K. G., Pan, L., Lee, S. K., & Nagar, A. K. (2010). A P system model with pure context-free rules for picture array generation. *Mathematical and Computer Modelling, 52*, 1901–1909.
33. Wang, P. S. P. (1989). *Array Grammars, Patterns and Recognizers*. Singapore: World Scientific.
34. Zhang, G., Pérez-Jiménez, M.J., & Păun, Gh. (2017). Real-life Applications with Membrane Computing. In Emergence, Complexity and Computation Series