

# Asynchronous P Systems

**Rudolf FREUND**

Faculty of Informatics  
Vienna University of Technology  
Favoritenstr. 9, A-1040 Wien, Austria  
E-mail: rudi@emcc.at

## Abstract

In the area of P systems, applying the rules in a maximally parallel way is one of the most common features of many models introduced so far. Whereas the idea of membranes as well as many operations and rules used in membrane systems have a concrete biological background, the universal clock assumed to control the parallel application of rules is unrealistic, but on the other hand relevant for many interesting theoretical results, especially when proving computational completeness and solving hard problems. Based on a quite general definition of tissue P systems, we investigate several models of P systems and compare their computational power in the classic case (i.e., applying the rules in a maximally parallel way) and in the case of applying the rules in an asynchronous (i.e., sequential) way. Moreover, we also recall some results for (tissue) P systems working in an asynchronous way already in the original definition. Finally, we also raise some questions for future research in this subarea of asynchronous P systems.

## 1 Introduction

When in 1998 Gheorghe Păun in [17] introduced membrane systems (which soon afterwards were called P systems), the way of applying the evolution rules in a maximally parallel way was one of the intrinsic features of this new model. Although biological processes in living organisms happen in parallel, they are not synchronized by a universal clock as assumed in the original model of membrane system, instead many processes involve several objects in parallel, but the processes themselves are carried out in an asynchronous way, which feature formally can be captured by letting these processes happen in a sequential/unsynchronized way.

Many variants of P systems have been investigated so far (see [18] for a comprehensive overview as well as [20] for the actual state of research). We here consider several of these models of P systems, and we assume the reader to be familiar with the original definitions and explanations given for these models, as going into more details would go far beyond the scope of an overview article as this one is intended to be.

Already in his first papers on P systems, the author investigated generalized models of membrane systems with sequential application of (quite complex) rules

(*generalized P systems*, e.g., see [7], [8], [9]). We will not recall the complex definitions of these systems, as they also used a kind of bounded parallel application of rules, i.e., when applying a complex rule to several objects, the simple rules working together in this complex rule themselves may be consumed, whereas other simple rules become applicable afterwards. Yet the complex rules were applied in a sequential way and in that way were already examples for asynchronous P systems.

In order to show some common features of various more recent models, in the third section we will define a general model of *tissue P systems* using states for the communication channels between the cells (compare [14]) and priorities on the rules used in these channels as well as with energy assigned to the cell membranes. On the other hand, this general model is only meant for static membrane structures (although we could deal with deletion of elementary membranes), not for dynamic ones (especially, we do not consider membrane division or generation). As long as no dynamic membrane features as membrane division and membrane generation are involved, the graph structure representing the channels between the cells in a tissue P system is just a generalization of the tree structure of the membranes in a classic P system, hence many results known from literature only for classic tree structures of membranes can be directly carried over or at least quite easily be adapted for this general case of tissue P systems.

Based on the general model of tissue P systems defined in the third section, we consider several variants of P systems and tissue P systems known from literature, redefine them in the sense of the general model, and investigate their generative power when applying the rules in a sequential or asynchronous way; for some models we also compare the variant of asynchronous application of rules with the variant of applying the rules in a maximally parallel way.

In the fourth section, we consider *P systems with symbol objects* and report on some first results on asynchronous P systems already mentioned in [18]. Then we consider the model of *tissue(-like) P systems with channel states* as defined in [14] which formed the background for definition of the general model of tissue P systems defined in the third section; we show that one-cell tissue P systems with channel states and antiport rules (i.e., asynchronous one-cell tissue P systems with antiport rules) characterize the (Parikh sets of) languages generated by matrix grammars without appearance checking. When using antiport rules transporting symbol objects between the cells and between cells and the environment, respectively, as well as maximal parallelism, we already reach computational completeness with only one membrane (see [13]); on the other hand, *asynchronous P systems with antiport rules* characterize the sets of numbers recognized by partially blind counter automata (see [15]). (*Tissue*) *P systems with unit rules and energy assigned to (cell) membranes* are another example for asynchronous (tissue) P systems (see [11]); for obtaining computational completeness, we need a priority relation on the rules.

In the fifth section, we deal with string objects. We first consider *gemmating P systems* (see [3], [2], [1]); the rules used there specify on which end of the current string a rule has to be applied, which additional feature allows for the simulation of Post systems in normal form (see [12]); in the same way, computational completeness

can even be obtained with the corresponding model of asynchronous gemmating (tissue) P systems. (Tissue) P systems with splicing rules or cutting/recombination rules by definition are working in an asynchronous way and reach computational completeness with only one cell (again taking advantage of the fact that one can make the rules working at the ends of the strings).

A short summary of the models of (tissue) P systems and of the results, especially for asynchronous (tissue) P systems, exhibited in sections four and five as well as an outlook to future research conclude the paper.

## 2 Preliminary Definitions

The set of non-negative integers is denoted by  $\mathbf{N}$ . An *alphabet*  $V$  is a finite non-empty set of abstract *symbols*. Given  $V$ , the free monoid generated by  $V$  under the operation of concatenation is denoted by  $V^*$ ; the *empty string* is denoted by  $\lambda$ , and  $V^* \setminus \{\lambda\}$  is denoted by  $V^+$ ;  $|w|$  denotes the length of the string  $w \in V^*$ . For more details on formal language theory we refer to [5] and [21].

A *register machine* is a construct  $M = (n, R, l_0, l_h)$ , where  $n$  is the number of registers,  $R$  is a finite set of instructions injectively labelled with elements from a given set  $lab(M)$ ,  $l_0$  is the initial/start label, and  $l_h$  is the final label.

The instructions are of the following forms:

- $l_1 : (add(r), l_2, l_3)$ ,  
Add 1 to the contents of register  $r$  and proceed to the instruction (labelled with)  $l_2$  or  $l_3$ . (We say that we have an ADD instruction.)
- $l_1 : (sub(r), l_2, l_3)$ ,  
If register  $r$  is not empty, then subtract 1 from its contents and go to instruction  $l_2$ , otherwise proceed to instruction  $l_3$ . (We say that we have a SUB instruction.)
- $l_h : halt$ ,  
Stop the machine. The final label  $l_h$  is only assigned to this instruction.

Without loss of generality, one can assume that in each ADD instruction  $l_1 : (add(r), l_2, l_3)$  and in each SUB instruction  $l_1 : (sub(r), l_2, l_3)$  the labels  $l_1, l_2, l_3$  are mutually distinct.

A *Post system*  $G$  (e.g., see [16]) is a construct

$$(V, T, P, w_0)$$

where

- $V$  is a set of (non-terminal and terminal) symbols,
- $T \subseteq V$  is a set of terminal symbols,

- $P = \{(x_i, y_i) \mid 1 \leq i \leq n\}$  is a set of productions with  $(x_i, y_i) \in V^+ \times V^*$  for  $1 \leq i \leq n$ , and
- $w_0$  is the axiom.

The derivation relation  $\Longrightarrow_G$  is defined as follows: For each  $(x, y) \in P$ , we define the derivation relation  $\Longrightarrow_{(x,y)}$  by  $wx_i \Longrightarrow_{(x,y)} y_iw$  for all  $w \in V^*$ , i.e.,  $x$  is cut away on the right-hand side of the underlying sentential form and  $y_i$  is inserted on its left-hand side;  $\Longrightarrow_G$  then is the union of all  $\Longrightarrow_{(x,y)}$ , i.e.,  $\Longrightarrow_G := \bigcup_{(x,y) \in P} \Longrightarrow_{(x,y)}$ . Whenever clear from the context, we will only write  $\Longrightarrow$  instead of  $\Longrightarrow_G$ . The reflexive and transitive closure of  $\Longrightarrow_G$  is denoted by  $\Longrightarrow_G^*$ . The *language generated by the Post system  $G$*  is the set of all terminal strings derivable from the axiom (in other words, it contains the results of successful computations in  $G$ ), i.e.,

$$L(G) = \{w \in T^* \mid w_0 \Longrightarrow_G^* w\}.$$

A Post system  $(V, T, P, w_0)$  is said to be in *normal form* if

$$\begin{aligned} P &= \{(x_i, y_i) \mid 1 \leq i \leq n\} \\ \text{and, for some } m \text{ and } k \text{ with } 0 \leq m \leq k \leq n, \\ (|x_i|, |y_i|) &\in \{(2, 1)\} \text{ for } 1 \leq i \leq m, \\ (|x_i|, |y_i|) &\in \{(1, 2)\} \text{ for } m+1 \leq i \leq k, \\ (|x_i|, |y_i|) &\in \{(1, 0), (1, 1)\} \text{ for } k+1 \leq i \leq n. \end{aligned}$$

For *matrix grammars without appearance checking*, we have a special normal form:  $G = (N, T, S, M)$  (where  $N$  and  $T$  are the sets of terminal and non-terminal symbols, respectively,  $S$  is the start symbol, and  $M$  is the set of matrices) is said to be in the *f-binary normal form*, if  $N = N_1 \cup N_2 \cup \{S, f\}$ , with these three sets being mutually disjoint, and the matrices in  $M$  are in one of the following forms:

1.  $(S \rightarrow XA)$ , with  $X \in N_1, A \in N_2$ ,
2.  $(X \rightarrow Y, A \rightarrow x)$ , with  $X, Y \in N_1, A \in N_2, x \in (N_2 \cup T)^*, |x| \leq 2$ ,
3.  $(X \rightarrow f, A \rightarrow x)$ , with  $X \in N_1, A \in N_2$ , and  $x \in T^*, |x| \leq 2$ ,
4.  $(f \rightarrow \lambda)$ .

Moreover, there is only one matrix of type 1 and only one matrix of type 4, which is only used in the last step of a derivation yielding a terminal result.

### 3 A General Model of Tissue P Systems

The reader is assumed to be familiar with the main ingredients and variants of the basic models of P systems and tissue systems, we especially refer to [18] and the original papers cited there. In this section we define the general model of (tissue)

P systems we are going to use for representing the different models of (tissue) P systems considered in this paper.

A *tissue P system* (of degree  $m \geq 1$ ) with channel states, priorities on the channel rules and energy assigned to cell membranes (*tcpeP system* for short) is a construct

$$\Pi = (m, O, T, K, O_\infty, W, E, ch, (s_{(i,j)}, R_{(i,j)}, \rho_{(i,j)})_{(i,j) \in ch}, i_o),$$

where

- $m$  is the number of *cells* assumed to be labelled with  $1, 2, \dots, m$ ;
- $O$  is the alphabet of *objects*;
- $T \subseteq O$  is the alphabet of *terminal* objects;
- $K$  is the alphabet of *states* (not necessarily disjoint of  $O$ );
- $O_\infty$  consists of  $m + 1$  sets of objects indicating those objects from  $O$  that are present in arbitrarily many copies in the environment and the  $m$  cells;
- $W$  consists of  $m + 1$  strings over  $O - O_\infty$  representing the *initial finite multiset* of objects present in the environment and the  $m$  cells of the system;
- $E \subseteq N^m$  are  $m$  numbers indicating the *initial energy values* assigned to the cell membranes of the  $m$  cells;
- $ch \subseteq \{(i, j) \mid i, j \in \{0, 1, 2, \dots, m\}, (i, j) \neq (0, 0)\}$  is the set of *links* (called *synapses* in [14]; in the following, we will use the term *channels*) between two cells or a cell and the environment (indicated by 0);
- $s_{(i,j)}$  is the *initial state* of the channel  $(i, j) \in ch$ ;
- $R_{(i,j)}$  is a finite set of *rules*, associated with the channel  $(i, j) \in ch$ , of the form  $(s, r, s')$ , for some  $s, s' \in K$  and  $r$  being a rule which involves objects in the cells  $i$  and  $j$  and yields new objects in these cells and also involves and possibly changes the energies assigned to these cells  $i$  and  $j$ ; the rules also have to obey to a priority relation  $\rho_{i,j}$ , i.e., a rule from  $R_{(i,j)}$  can only be applied if no other rule of higher priority could be applied, too;
- $i_o \in \{0, 1, 2, \dots, m\}$  is the *output cell* (0 means that the output is to be found in the environment).

A rule of the form  $(s, r, s') \in R_{(i,j)}$  changes the state of the channel between the cells  $i$  and  $j$  from  $s$  to  $s'$ , it can only be applied if the current state is  $s$ . In contrast to the definition of tissue P systems with channel states in [14] where a rule had to be used on each channel for which a rule could be used, we here will consider an asynchronous or even sequential model, i.e., usually in one derivation step only one rule in one channel is applied. Moreover, the channels in this general model are directed, i.e., we distinguish between  $R_{(i,j)}$  and  $R_{(j,i)}$ .

The computation starts with the initial configuration specified by  $W$ ,  $E$ , and  $(s_{(i,j)})_{(i,j) \in ch}$ ; in each time unit, one rule is used in one channel. The results of a computation are described either by the multiplicity of symbol objects from  $T$  or by the string objects over  $T$  present in cell  $i_0$  in a halting configuration or by the terminal strings appearing in cell  $i_0$  during an arbitrary computation (we will usually restrict ourselves to these variants).

## 4 (Tissue) P Systems Working on Symbol Objects

In this section we consider several models of (tissue) P systems working on symbol objects. We first recall some results for the original model of P systems when membrane systems were introduced by Gheorghe Păun in [17]. Then we exhibit a result from [14] saying that one-cell tissue P systems with channel states and antiport rules (i.e., asynchronous one-cell tissue P systems with antiport rules) characterize the (Parikh sets of) languages generated by matrix grammars without appearance checking. Moreover, we recall the optimal result (with respect to the number of membranes and the weight of the rules) for P systems with antiport rules for the case of applying rules in the maximally parallel way (e.g., see [13]) as well as mention the result proved in [15] showing that when applying rules in an asynchronous way, only the generative power of partially blind counter automata can be obtained. Finally, we consider (tissue) P systems with unit rules and energy assigned to (cell) membranes, which work in an asynchronous way, but for obtaining computational completeness need a priority relation on the rules.

### 4.1 The classic model

For the classic model of P systems, we refer to the original article by Gheorghe Păun and the detailed explanations given in [18]. As shown in [10], *P systems with catalysts* are able to generate any arbitrary recursively enumerable set of numbers with only two catalysts in only one membrane provided we impose the condition of maximal parallelism. On the other hand, one of the first results dealing with asynchronous P systems (see [18], subsection 3.4.5) says that *asynchronous P systems with catalysts* can only generate regular sets of numbers.

### 4.2 Tissue P systems with channel states and antiport rules

The background for the definition of the general model of tissue P systems defined in the third section was the model of tissue(-like) P systems with channel states as defined in [14]. One main difference between the two models is that there for each set  $\{i, j\}$  of cell labels  $i, j$  only for one channel  $(i, j)$  or  $(j, i)$  a set of rules was defined. We now consider our general model of *tissue P systems with channel states and antiport rules* (but without energy assigned to cell membranes and without priorities on the rules, therefore we omit specifying these ingredients  $E$  and  $\rho_{(i,j)}$ ): A rule of the form  $(s, x/y, s') \in R_{(i,j)}$  is interpreted as an antiport rule for the ordered pair  $(i, j)$  of cells, acting only if the channel  $(i, j)$  has the state  $s$ ; the application of the rule means moving the objects specified by  $x$  from cell  $i$  (from the environment, if  $i = 0$ ) to

cell  $j$ , at the same time moving the objects specified by  $y$  in the opposite direction, as well as changing the state of the channel from  $s$  to  $s'$ . (The rules with one of  $x, y$  empty are, in fact, symport rules, but we do not explicitly consider here this distinction, as it is not relevant for what follows.) The weight of an antiport rule  $x/y$  is the maximum of the lengths of  $x$  and  $y$ . If, at the end of a halting computation, the contents of the final cell labelled by  $i_0$  consists only of terminal symbols, then the (vector of) numbers represented by the copies of terminal symbols constitutes the result of this successful computation in the tissue P system with channel states and antiport rules.

Based on the results proved in [14] for tissue P systems with channel states and antiport rules we now exhibit a characterization of (Parikh sets of) matrix languages:

**Theorem 1.** *Any Parikh set of a language that can be generated by a matrix grammar without appearance checking can be generated by a tissue P system with only one cell, only one channel state and antiport rules of weight two.*

*Proof.* Consider a matrix grammar  $G = (N_1 \cup N_2 \cup \{S, f\}, T, S, M)$  in the f-binary normal form and construct the tissue P system with channel states and antiport rules

$$\begin{aligned}
\Pi &= (1, O, T, \{s\}, (O, \emptyset), (\lambda, X_0 A_0 Z), \{(1, 0)\}, (s, R_{(1,0)}), 1), \\
O &= N_1 \cup \{f\} \cup N_2 \cup T \cup \{\langle X, \alpha\beta \mid X \in N_1 \cup \{f\}, \alpha, \beta \in N_2 \cup T\}, \\
R_{(1,0)} &= \{(s, XA/Yx, s) \mid (X \rightarrow Y, A \rightarrow x) \in M \\
&\quad X \in N_1, Y \in N_1 \cup \{f\}, A \in N_2, x \in N_2 \cup T \cup \{\lambda\}\} \\
&\cup \{(s, XA/Y\langle Y, \alpha_1\alpha_2 \rangle, s), (s, \langle Y, \alpha_1\alpha_2 \rangle/\alpha_1\alpha_2, s) \mid \\
&\quad (X \rightarrow Y, A \rightarrow \alpha_1\alpha_2) \in M, \\
&\quad X \in N_1, Y \in N_1 \cup \{f\}, A \in N_2, \alpha_1, \alpha_2 \in N_2 \cup T\} \\
&\cup \{(s, \alpha/\alpha, s) \mid \alpha \in N_1 \cup N_2\} \cup \{(f, \lambda/Z, f)\},
\end{aligned}$$

where  $(S \rightarrow X_0 A_0)$  is the initial matrix of  $M$ .

The state plays no rôle, the matrices of  $M$  are simulated by the antiport rules. As long as at least one non-terminal symbol from  $N_1 \cup N_2$  is present, the computation must continue. Obviously,  $\Pi$ , by halting computations, generates exactly the same numbers (in cell 1) as  $G$ .  $\square$

As shown in [14], for tissue P systems with only one cell also the converse of the preceding theorem holds true, i.e., any Parikh set of a language that can be generated by a one-cell tissue P system with channel states and antiport rules can be generated by a matrix grammar without appearance checking (we here omit the proof which could be an adequate adaptation of the proof given in [14]).

### 4.3 P systems with antiport rules

The preceding results have shown that with asynchronous (tissue) P systems with antiport rules and only one cell we can only get (Parikh sets of) matrix languages.

With imposing the condition of maximal parallelism, we obtain full computational power. The proof of the following theorem follows the proof given in [13]; for the representation of the P system we take a similar model as before in the proof of Theorem 1, but we omit the state and when applying the rules have in mind the condition of maximal parallelism:

**Theorem 2.** *Any Parikh set of a recursively enumerable language can be generated by a P system with only one cell (membrane) and antiport rules of weight two.*

*Proof.* For a given register machine  $M = (n, R, l_0, l_n)$  we consider  $lab(M)$  to be the set of instruction labels and the alphabet  $U = \{a_1, \dots, a_m\}$  (the symbol  $a_i$  is associated with register  $i$  and the contents of this register will be represented by the multiplicity of object  $a_i$  in the P system we are going to construct); we now construct the P system

$$\begin{aligned} \Pi &= (1, O, T, (O, \emptyset), (\lambda, l_0), \{(1, 0)\}, R_{(1,0)}, 1), \\ O &= U \cup \{l, l', l'', l''', l^{iv} \mid l \in lab(M)\}, \\ R_{(1,0)} &= \{(l_1, out; l_2 a_r, in), (l_1, out; l_3 a_r, in) \mid l_1 : (add(r), l_2, l_3) \in R\} \\ &\cup \{(l_1, out; l'_1 l''_1, in), \\ &\quad (l'_1 a_r, out; l''_1, in), \\ &\quad (l''_1, out; l^{iv}_1, in), \\ &\quad (l^{iv}_1 l'''_1, out; l_2, in), \\ &\quad (l^{iv}_1 l'_1, out; l_3, in) \mid l_1 : (sub(r), l_2, l_3) \in R\}. \end{aligned}$$

We start with  $l_0$  present in the system, and then we simulate  $M$  :

Each add-instruction  $l_1 : (add(r), l_2, l_3)$  of  $M$  is simulated by means of a rule  $(l_1, out; l_2 a_r, in)$  or a rule  $(l_1, out; l_3 a_r, in)$  in  $\Pi$ , while a subtract-instruction  $l_1 : (sub(r), l_2, l_3) \in R$  is simulated as follows. The available object  $l_1$  is sent out, in exchange of  $l'_1$  and  $l''_1$ . The first object checks whether the register is non-empty, and in the affirmative case it exits the system (together with a copy of  $a_r$ ) and is replaced by  $l'''_1$ ; if no copy of  $a_r$  is available, then  $l'_1$  remains in the system and waits. The object  $l''_1$  checks what the other object has done, allowing it a step for acting (this is the step when the rule  $(l''_1, out; l^{iv}_1, in)$  is used). The object  $l^{iv}_1$  will find inside either one of the objects  $l'_1$  (if the register  $r$  was empty) and  $l'''_1$  (if the register  $r$  was not empty), and in each case the next object to be introduced in the system,  $l_3$  or  $l_2$ , respectively, is the correct label to be used in the program of  $M$ . Hence, halting computations in the P system  $\Pi$  generate the same numbers as  $M$ .  $\square$

In the case of *asynchronous P systems with antiport rules*, we obtain a characterization of the family languages generated by partially blind counter automata (see [15]).

#### 4.4 (Tissue) P systems with unit rules and energy assigned to (cell) membranes

In [11] P systems with unit rules and energy assigned to (cell) membranes were introduced as an asynchronous model of P systems, too; adapting the definitions

given there, we get the following definition (we omit the states and  $O_\infty$ , because all symbols are assumed to occur in a finite number only):

A *tissue P system with unit rules and energy assigned to cell membranes (of degree  $m$ )* is a construct

$$\Pi = (m, O, T, W, E, ch, (R_{(i,j)}, \rho_{(i,j)})_{(i,j) \in ch}, )$$

where the rules in the sets  $R_{(i,j)}$  are of the form  $(a, b, \Delta e_i, \Delta e_j)$  with  $a, b \in O$ , and  $|\Delta e_i|$  and  $|\Delta e_j|$ , respectively, is the amount of energy that - for  $\Delta e_i \geq 0$  and  $\Delta e_j \geq 0$  - is added to or - for  $\Delta e_i < 0$  and  $\Delta e_j < 0$  - is subtracted from  $e_i$  and  $e_j$ , respectively (the energy assigned to cell  $i$  and  $j$ , respectively) by the application of the rule which in addition moves an object  $a$  from cell  $i$  to cell  $j$  thereby changing it to  $b$ . Observe that negative values for the energy assigned to a cell are not allowed. We restrict ourselves to a general priority relation  $\rho$  on the rules saying that the rules with maximal value for  $|\Delta e_i + \Delta e_j|$  have to be applied.

In contrast to other models, the output values have nothing to do with the objects in some specific cell, but are constituted by the energy values assigned to the cell membranes at the end of a halting computation.

The following theorem is an immediate consequence of the corresponding proof given in [11]:

**Theorem 3.** Let  $L \subseteq \mathbf{N}^\beta$  be a recursively enumerable set of (vectors of) non-negative integers. Then  $L$  can be generated by a tissue P system with unit rules and energy assigned to cell membranes with (at most)  $\beta + 2$  cells.

*Proof.* Consider a register machine  $M = (m, P, 1, n)$  with  $m$  registers, where  $m = \beta + 2$ , and  $lab(M)$  to be the set of instruction labels. The output values from  $M$  are expected to be in registers 1 to  $\beta$  at the end of a successful computation. Moreover, without loss of generality, we may assume that at the beginning of a computation all the registers contain zero.

We construct the tissue P system

$$\begin{aligned} \Pi &= (m, O, \{p_n\}, W, E, ch, (R_{(i,j)}, \rho)_{(i,j) \in ch}, ) \text{ with} \\ O &= \{p_j, \tilde{p}_j \mid 1 \leq j \leq n, j \in lab(M)\}, \\ W &= (p_1, \lambda, \dots, \lambda), \\ E &= (0, \dots, 0), \\ ch &= \{(0, i), (i, 0) \mid 1 \leq i \leq m\}, \\ R_{(0,i)} &= \{(p_j, \tilde{p}_j, 0, 0) \mid j : (add(i), k, l) \in P\} \\ &\cup \{(p_j, \tilde{p}_j, 0, 0) \mid j : (sub(i), k, l) \in P\} \\ &\quad \text{for } 1 \leq i \leq m, \\ R_{(i,0)} &= \{(\tilde{p}_j, p_k, 1, 0), (\tilde{p}_j, p_l, 1, 0) \mid j : (add(i), k, l) \in P\} \\ &\cup \{(\tilde{p}_j, p_k, -1, 0), (\tilde{p}_j, p_l, 0, 0) \mid j : (sub(i), k, l) \in P\} \\ &\quad \text{for } 1 \leq i \leq m. \end{aligned}$$

The contents of register  $i$ ,  $1 \leq i \leq m$ , is represented by the energy value  $e_i$  of membrane  $i$ .

The sets of rules  $R_i$  depend on the instructions of  $P$ ; in more detail, the simulation works as follows:

1. Each add-instruction  $j : (add(i), k, l) \in P$ ,  $1 \leq i \leq m$  is simulated in two steps by using the rules  $(p_j, \tilde{p}_j, 0, 0)$  and  $(\tilde{p}_j, p_k, 1, 0), (\tilde{p}_j, p_l, 1, 0)$ .
2. Each conditional subtract-instruction  $j : (sub(i), k, l) \in P$  is simulated in two steps by the rules  $(p_j, \tilde{p}_j, 0, 0)$  as well as  $(\tilde{p}_j, p_k, -1, 0)$  or  $(\tilde{p}_j, p_l, 0, 0)$ .

The condition of priority guarantees that  $(\tilde{p}_j, p_k, -1, 0)$  is applied as long as  $e_i$  has a positive value. Only if in the current configuration  $e_i = 0$ , i.e., register  $i$  is empty, the rule  $(\tilde{p}_j, p_l, 0, 0)$  can be used.

It follows from the description given above that after each simulation of an instruction each energy value  $e_i$  equals the contents of register  $i$ ,  $1 \leq i \leq m$ . Hence, after having simulated the halt instruction labelled by  $n$  and halting the system by just doing nothing with the halting symbol  $p_n$  anymore, the energy values  $e_1, \dots, e_m$  equal the output of the program  $P$ . The only object remaining within the system is the final label  $p_n$  in region 0.  $\square$

Without priorities, we only get a characterization of (Parikh sets of) languages generated by matrix languages without appearance checking (see the corresponding proofs in [11]).

## 5 (Tissue) P Systems Working on String Objects

In this section we consider (tissue) P systems working on string objects. As a specific model, we consider gemmating P systems which, in contrast to the original definitions given in [3] (also see [2] and [1]) turns out to reach computational completeness in the asynchronous case, too; in the setting of the general model of this paper, the number of cells can even be reduced to two. Moreover, we also mention some universality results when using molecular operations as splicing or cutting/recombination in asynchronous (tissue) P systems.

### 5.1 Gemmating P Systems

The model of gemmating P systems first was examined in [3] and is abstracted from the way bigger substances like proteins are moved across cell membranes by means of vesicles, i.e., some string objects can be transported in a mobile membrane to a target membrane and then being fused with it. For more detailed explanations we refer to [3], [2], and [1]. We only define a restricted variant of the general model considered there:

An (*extended*) *gemmating P system* is a construct

$$\Pi = (V, T, \mu, M_1, \dots, M_n, D_1, \dots, D_n),$$

where

- $V$  is an alphabet,

- $T \subseteq V$  is the terminal alphabet,
- $\mu = [0[1 \ ]_1 \dots [n \ ]_n]_0$  is a membrane structure of depth 2 and degree  $n + 1$ ,
- $M_i, 1 \leq i \leq n$ , are finite multisets of strings over  $V$ ,
- $D_i, 1 \leq i \leq n$ , are sets of *pre-dynamic evolution rules* associated with membrane  $i$ , i.e., sets of mutation rules of the form  $a \rightarrow v$  with  $a \in V$  and  $v \in V^* \{ @_j \} \cup \{ @_j \} V^*$  where  $@_j \notin V, 0 \leq j \leq n$  and  $i \neq j$ . (Note that the special symbol  $@_j$  can only appear on either end of the string).

Starting from an initial configuration consisting of  $\mu$  and  $M_i, 1 \leq i \leq n$ , in membrane  $i$ , the system proceeds from one configuration to the next one by non-deterministically applying the rules in the sets  $D_i, 1 \leq i \leq n$ , in a maximally parallel way: A string can only be rewritten by one rule per step and the resulting strings then are transported by mobile membranes to the membranes specified by the target indications given by  $@_j$ , i.e., in sum, applying the rule  $a \rightarrow u@_j$  ( $a \rightarrow @_j u$ ) in membrane  $i, i \neq j$ , to a string  $wa$  ( $aw$ ) means removing this string from membrane  $i$  and instead adding the string  $wu$  ( $uw$ ) in membrane  $j$  in case  $j \geq 1$ , whereas for  $j = 0$  this means that the string is sent out of the system. The *language generated by the (extended) gemmating P system* is considered to be the set of terminal strings that have been sent out of the system during a halting computation.

In the general model defined above, an extended gemmating P system can be represented as a *gemmating tissue P system*

$$\Pi = (m, O, T, W, ch, (R_{(i,j)})_{(i,j) \in ch}, 0),$$

where we have omitted  $O_\infty$ , because  $(O_\infty)_i = \emptyset, 1 \leq i \leq m$ , as well as  $E$ , because we do not use energies assigned to the cell membranes, and we also omitted  $K, s_{(i,j)}$  and  $\rho_{(i,j)}$ , because we do not need states (in fact, we use only one steady state) and we also do not need priorities on the rules.

A rule  $a \rightarrow u@_j$  ( $a \rightarrow @_j u$ ) in  $D_i$  in the extended gemmating P system now means putting the rule  $a \rightarrow u@$  ( $a \rightarrow @u$ ) into  $R_{(i,j)}$  in the gemmating tissue P system. As results of computations in gemmating tissue P systems we take the terminal strings appearing in the environment at any derivation step.

The proof of the following result follows the proof given in [12]:

**Theorem 4.** *For every recursively enumerable string language  $L$  there exists a gemmating tissue P system  $\Pi$  with (at most) three cells such that  $L(\Pi) = L$ .*

*Proof.* We take a Post system in normal form  $G = (V, T, P, w_0)$  which generates  $L$ . Then we construct the gemmating tissue P system

$$\Pi = (3, V_\Pi, T, W, ch, (R_{(1,2)}, R_{(1,3)}, R_{(2,1)}, R_{(3,0)}, R_{(3,1)}, 0)$$

having the following components:

$$\begin{aligned}
V_{\Pi} &= V \cup \{(\varepsilon, i), (\lambda, i) \mid 0 \leq i \leq n+m\} \cup \{\lambda_i \mid 1 \leq i \leq k\} \\
&\cup \{(\lambda_i, j) \mid 1 \leq i \leq m, 0 \leq j \leq i\}, \\
W &= (\lambda, w_0, \lambda, \lambda), \\
ch &= \{(1, 2), (1, 3), (2, 1), (3, 0), (3, 1)\}, \\
R_{(1,2)} &= \{x_i \rightarrow (\varepsilon, i) @ \mid m+1 \leq i \leq n\} \\
&\cup \{x_{i,2} \rightarrow (\varepsilon, i) @, x_{i,1} \rightarrow (\varepsilon, n+i) @ \mid 1 \leq i \leq m\} \\
&\cup \{(\varepsilon, j) \rightarrow (\varepsilon, j-1) @ \mid n+m \geq j \geq 1\}, \\
R_{(1,3)} &= \{(\varepsilon, 0) \rightarrow \lambda @\}, \\
R_{(2,1)} &= \{E \rightarrow @(\lambda, i) E y_i \mid m+1 \leq i \leq n, E \in V\} \\
&\cup \{E \rightarrow @(\lambda_i, i) E \mid 1 \leq i \leq m, E \in V\} \\
&\cup \{\lambda_i \rightarrow @(\lambda, n+i) y_i \mid 1 \leq i \leq m\} \\
&\cup \{(\lambda_i, j) \rightarrow @(\lambda_i, j-1) \mid m \geq j \geq 1\} \\
&\cup \{(\lambda, j) \rightarrow @(\lambda, j-1) \mid n+m \geq j \geq 1\}, \\
R_{(3,0)} &= \{(\lambda, 0) \rightarrow @\lambda\}, \\
R_{(3,1)} &= \{(\lambda, 0) \rightarrow @\lambda\} \cup \{(\lambda_i, 0) \rightarrow @\lambda_i \mid 1 \leq i \leq m\}.
\end{aligned}$$

The main idea for simulating a rule from  $G$  in  $\Pi$  has already quite often been used in the area of P systems: For the rules  $(x_i, y_i)$  with  $m+1 \leq i \leq n$ , we only have to erase one symbol on the right-hand side of the current string  $w x_i$  by the rule  $x_i \rightarrow (\varepsilon, i) @$  in  $R_{(1,2)}$ , which string now on the left-hand side gets the corresponding substring  $y_i$  by the application of the rules  $E \rightarrow @(\lambda, i) E y_i$  from  $R_{(2,1)}$ ; the resulting string  $(\lambda, i) y_i w (\varepsilon, i)$  is sent back to cell 1. By applying the rules  $(\varepsilon, j) \rightarrow (\varepsilon, j-1) @$  from  $R_{(1,2)}$  and  $(\lambda, j) \rightarrow @(\lambda, j-1)$  from  $R_{(2,1)}$  the string oscillates between cells 1 and 2 until the indices of  $\varepsilon$  and  $\lambda$  reach 0. By applying the rule  $(\varepsilon, 0) \rightarrow \lambda @$  (from  $R_{(1,3)}$ ) and the rule  $(\lambda, 0) \rightarrow @\lambda$  (from  $R_{(3,1)}$ ) (or  $(\lambda, 0) \rightarrow @\lambda$  from  $R_{(3,0)}$ , respectively) we finally obtain the string  $y_i w$ , i.e., we have successfully simulated the application of the rule  $(x_i, y_i)$ .

If in the second step of the simulation, i.e., when applying a rule  $E \rightarrow @\lambda_j E y_j$  from  $R_{(2,1)}$ , we do not choose the same index as when applying the rule  $x_i \rightarrow (\varepsilon, i) @$  from  $R_{(1,2)}$  in the first step, then the simulation will stop either in cell 2 (in case  $i > j$ ) with  $(\lambda, 0) y_i w (\varepsilon, i-j)$  or in cell 3 (in case  $i < j$ ) with  $(\lambda, j-i) y_i w (\varepsilon, 0)$  without having sent out a string.

Rules  $(x_i, y_i)$  with  $1 \leq i \leq m$  are simulated in two cycles, the first cycle starting with the rules  $x_{i,2} \rightarrow (\varepsilon, i) @$  (from  $R_{(1,2)}$ ) and  $E \rightarrow @(\lambda_i, i) E$  (from  $R_{(2,1)}$ ). This time we have to remember in  $\lambda_i$  of  $(\lambda_i, i)$  which rule of  $G$  we are going to simulate, hence, after decrementing the indices of  $(\varepsilon, i)$  and  $(\lambda_i, i)$  (by the rules  $(\varepsilon, j) \rightarrow (\varepsilon, j-1) @_2$  and  $(\lambda_i, j) \rightarrow @(\lambda_i, j-1)$ , from  $R_{(1,2)}$  and  $R_{(2,1)}$ , respectively) we reach cell 3 with  $(\lambda_i, 0)$  and can apply the rule  $(\lambda_i, 0) \rightarrow @\lambda_i$  from  $R_{(3,1)}$ . Now the rule  $x_{i,1} \rightarrow (\varepsilon, n+i) @$  from  $R_{(1,2)}$  has to be applied so that after the application of the rule  $\lambda_i \rightarrow @(\lambda, n+i) y_i$  (from  $R_{(2,1)}$ ) again the checking phase with oscillating between cells 1 and 2 can start. Observe that now we have to use  $n+i$  as starting level for checking the correct erasing of  $x_{i,1}$  and not  $i$  in order not to be confused with the process for the first symbol  $x_{i,2}$ .

If the rule applied in  $\Pi$  in the last step of the simulation of a rule from the Post system  $G$  is  $(\lambda, 0) \rightarrow @\lambda$  from  $R_{(3,1)}$  then (possibly) we may continue with simulating another rule from  $G$  (if no such rule exists, we halt without having sent out any

string); otherwise, we may assume that the computation in  $G$  has finished with the rule just simulated and therefore apply the rule from  $R_{(3,0)}$ , i.e.,  $(\lambda, 0) \rightarrow @\lambda$ . By applying this rule, the computation in  $\Pi$  halts, because the current string is sent out; it contributes to  $L(G)$  if and only if it consists of terminal symbols only. In sum, a halting computation in  $\Pi$  yields a terminal string if and only if this string is the result of a successful computation in  $G$ , hence,  $L(\Pi) = L(G)$ . This observation concludes the proof.  $\square$

The preceding proof shows that, in contrast to the original definition of gemmating P systems, we do not need maximal parallelism; instead, throughout any computation, there is exactly one string in the system, and when this string is sent out (contributing to the generated language if and only if it consists of terminal symbols), the computation halts in any case. If we start with an arbitrary number of axioms  $w_0$ , then each copy may evolve according to the rules given in the proof, and we may take each terminal string sent out during any computation (halting or non-halting) as the result of a successful computation in the gemmating tissue P system. The derivation of different strings happens in a completely unsynchronized way, and we even may assume that in one derivation step of the gemmating tissue P systems several strings are affected by (possibly different) rules in parallel.

Taking advantage of the more general definition, we can improve the result proved for gemmating tissue P systems with respect to the number of cells needed to obtain computational completeness:

**Corollary 5.** *For every recursively enumerable string language  $L$  there exists a gemmating tissue P system  $\Pi'$  with (at most) two cells such that  $L(\Pi') = L$ .*

*Proof.* Take the gemmating tissue P system

$$\Pi = (3, V_{\Pi}, T, W, ch, (R_{(1,2)}, R_{(1,3)}, R_{(2,1)}, R_{(3,0)}, R_{(3,1)}, 0)$$

and define the corresponding reduced gemmating tissue P system

$$\Pi' = (2, V'_{\Pi}, T, W', ch', (R'_{(0,1)}, R'_{(1,0)}, R'_{(1,2)}, R'_{(2,1)}), 0)$$

with

$$\begin{aligned} V'_{\Pi} &= V_{\Pi} \cup \{\lambda_{out}\}, \\ W' &= (\lambda, w_0, \lambda), \\ ch' &= \{(0, 1), (1, 0), (1, 2), (2, 1)\}, \\ R'_{(0,1)} &= R_{(3,1)} \cup \{(\lambda, 0) \rightarrow @\lambda_{out}\}, \\ R'_{(1,0)} &= R_{(1,3)} \cup \{\lambda_{out} \rightarrow @\lambda\}, \\ R'_{(1,2)} &= R_{(1,2)}, \\ R'_{(2,1)} &= R_{(2,1)}. \end{aligned}$$

In some sense, we take the environment as the third cell. Instead of applying  $(\lambda, 0) \rightarrow @\lambda$  from  $R_{(3,0)}$  in order to get a terminal string, we now have to apply  $(\lambda, 0) \rightarrow @\lambda_{out}$  from  $R'_{(0,1)}$  and  $\lambda_{out} \rightarrow @\lambda$  from  $R'_{(1,0)}$ . The other details of the preceding proof remain unchanged.  $\square$

## 5.2 (Tissue) P systems with splicing or cutting/ recombination rules

P systems with splicing or cutting/recombination rules were considered in [6], see there for detailed definitions and explanations. Expressed in the general model used in this paper, we deal with (tissue) P systems using splicing or cutting/ recombination rules as the rules used in the channels. For obtaining computational completeness, we do not need states, energy assigned to membranes or priorities on the rules, and moreover, only one cell is needed. The proofs given in [6] can directly be expressed in the notions of this article, hence, we do not repeat the extensive definitions and proofs given there and state the following result without proof:

**Proposition 6.** *For every recursively enumerable string language  $L$  there exists a one-cell tissue P system with splicing or cutting/recombination rules generating  $L$ .*

We should like to stress the fact that the derivation of an arbitrary number of initial strings (the axioms are available in an unbounded number) happens in a completely unsynchronized way; in fact, we could even assume that several strings are affected even in a parallel manner (but without enforcing maximal parallelism).

## 6 Summary and Future Research

From the big variety of (tissue) P systems, we could only investigate a small number for the case of applying rules not in the maximally parallel way, but instead in a sequential or asynchronous way. A strict interpretation of the sequential way of applying rules says that in each step exactly one rule is applied and the next rule can only be applied after finishing the previous step. In a more natural sense, an asynchronous way of applying rules allows for the application of an arbitrary number of rules in parallel, but does not enforce maximality. In most cases, there is no real difference between strict sequential and asynchronous application of rules, hence, the notion *asynchronous P system* or *asynchronous tissue P system* in fact will cover most of the models of P systems and tissue P systems, respectively, where the rules are not applied in a maximally parallel way.

The feature of applying rules in a maximally parallel way seems to be essential for obtaining universal computational power in many cases, at least when dealing with symbol objects: *P systems with catalysts* are able to generate any arbitrary recursively enumerable set of numbers with only two catalysts in only one membrane (see [10]), whereas *asynchronous P systems with catalysts* can only generate regular sets of numbers (see [18], subsection 3.4.5). *Tissue P systems with channel states and antiport rules* with only one cell work in an asynchronous way and thus only allow for a characterization of matrix languages (without appearance checking). P systems with antiport rules are computationally complete with only one membrane (e.g., see [13]), and, due to the cooperative nature of these rules, *asynchronous P systems with antiport rules* can generate any set of numbers which can be obtained by partially blind counter automata (see [15]). (Tissue) P systems with unit rules and energy assigned to (cell) membranes are another example for asynchronous

(tissue) P systems (see [11]); for obtaining computational completeness, we need a priority relation on the rules. In sum, we made the observation that for most models of (tissue) P systems working on symbol objects considered in the literature of membrane systems so far, the feature of applying the rules in a maximally parallel way is essential for obtaining computational completeness or else some other powerful feature has to be used (e.g., priorities), which somehow is not too surprising, because in most proofs to be found in literature this feature is needed for capturing the feature of appearance checking when simulating matrix grammars or the feature of checking the contents of a register for zero when simulating register machines.

When dealing with string objects, some small additional ingredients allow to obtain computational completeness even with asynchronous (tissue) P systems: For example, *gemmating P systems* (see [3], [2], [1]) specify on which end of the current string a rule has to be applied, which additional feature allows for the simulation of Post systems in normal form (see [12]) and in that way for obtaining computational completeness even with the corresponding model of asynchronous (tissue) P systems. (Tissue) P systems with splicing rules or cutting/recombination rules are working in an asynchronous way and reach computational completeness with only one cell (again taking advantage of the fact that one can make the rules working at the ends of the strings).

Many topics remain for future research, for example, we have not investigated the accepting variants of P systems (*P automata*) considered in this paper. Moreover, many other models of (tissue) P systems not considered in this article (e.g., one may simply consider some variants of the big variety of remaining models of membrane systems described in the book of Gheorghe Păun, [18]) deserve to be investigated for the case of asynchronous application of rules, too. Finally, other new variants, already from the beginning omitting the feature of a universal clock (e.g., see [22]) and relying on an asynchronous way for the application of rules (e.g., see [4]), promise interesting new results and applications for the future.

## Acknowledgements

The author acknowledges many interesting and inspiring discussions with Gheorghe Păun and many colleagues working in the area of membrane systems as well as Marion Oswald's help in preparing this article; moreover, he acknowledges IST-2001-32008 project "MolCoNet".

## References

- [1] D. Besozzi, E. Csuhaj-Varjú, G. Mauri, C. Zandron: Size and power of extended gemmating P systems. In: [19] (2004) 92–101
- [2] D. Besozzi, G. Mauri, Gh. Păun, C. Zandron: Gemmating P systems: collapsing hierarchies. *Theoretical Computer Science* **296**, 2 (2003) 253–267
- [3] D. Besozzi, C. Zandron, G. Mauri, N. Sabadini: P systems with gemmation of mobile membranes. In: A. Restivo, S. Ronchi Della Rocca, L. Roversi (eds.):

- Proc. of the 7th Italian Conf. of Theoretical Computer Science 2001, *Lecture Notes in Computer Science* **2202**, Springer-Verlag, Berlin (2001) 136–153
- [4] M. Cavaliere: Towards Asynchronous P Systems. *This volume* (2004)
  - [5] J. Dassow, Gh. Păun: *Regulated Rewriting in Formal Language Theory*. Springer-Verlag, Berlin (1989)
  - [6] R. Freund, F. Freund, M. Margenstern, M. Oswald, Yu. Rogozhin, S. Verlan: P Systems with Cutting/Recombination Rules Assigned to Membranes. In C. Martín-Vide, G. Mauri, Gh. Păun, G. Rozenberg, A. Salomaa (eds.): *Membrane Computing, International Workshop, WMC 2003, Tarragona, Spain, July 17-22, 2003, Lecture Notes in Computer Science* **2933**, Springer-Verlag, Berlin (2004), 191–202
  - [7] R. Freund: Generalized P-Systems. In: G. Ciobanu, Gh. Păun Gh.(eds.): *Proceedings Fundamentals of Computation Theory, Lecture Notes in Computer Science* **1684**, Springer-Verlag, Berlin (1999) 281–292
  - [8] R. Freund, F. Freund: Molecular computing with generalized homogeneous P-systems. In: A. Condon, G. Rozenberg (eds.): *DNA Based Computers*, Proceedings DNA 6, Leiden (2000) 113–125
  - [9] R. Freund: Sequential P-systems. *Romanian Journal of Information Science and Technology*, Vol. 4, Numbers 1-2 (2001) 77–88
  - [10] R. Freund, L. Kari, M. Oswald, P. Sosík: Computationally Universal P Systems without Priorities: Two Catalysts are Sufficient. To appear in *Theoretical Computer Science*
  - [11] R. Freund, A. Leporati, M. Oswald, C. Zandron: Sequential P systems with unit rules and energy assigned to membranes. In: [19] (2004) 168–182
  - [12] R. Freund et alii: Extended gemmating P systems are computationally complete with four membranes. *Submitted*
  - [13] R. Freund, Gh. Păun: Deterministic P systems. *Submitted*
  - [14] R. Freund, Gh. Păun, M.J. Pérez Jiménez: Tissue-like P systems with channel states. In: [19] (2004) 206–223
  - [15] P. Frisco: About P systems with symport/antiport. In: [19] (2004) 224–236
  - [16] M.L. Minsky: *Computation: Finite and Infinite Machines*. Prentice Hall, Englewood Cliffs, New Jersey, USA (1967)
  - [17] Gh. Păun: Computing with membranes. *Journal of Computer and System Sciences* **61**, 1 (2000) 108–143, and TUCS Research Report 208 (1998) (<http://www.tucs.fi>)
  - [18] Gh. Păun: *Membrane Computing: an Introduction*. Springer-Verlag, Berlin (2002)

- [19] Gh. Păun, A. Riscos Nuñez, A. Romero Jiménez, F. Sancho Caparrini (eds.): Dept. of Computer Sciences and Artificial Intelligence, *Univ. of Sevilla Tech. Report 01/2004*, Second Week on Membrane Computing, Sevilla, Spain, Feb. 2-7 (2004)
- [20] The P Systems Web Page: <http://psystems.disco.unimib.it/>
- [21] A. Salomaa, G. Rozenberg (eds.): *Handbook of Formal Languages*. Springer-Verlag, Berlin (1997)
- [22] D. Sburlan: Clock-free P systems. *This volume* (2004)