Bi-simulation Between P Colonies and P Systems with Multi-stable Catalysts

Erzsébet Csuhaj-Varjú[1] and Sergey Verlan[2]

[1] Department of Algorithms and Their Applications, Faculty of Informatics, ELTE Eötvös Loránd University, Budapest, Hungary, Pázmány Péter sétány 1/c, 1117
csuhaj@inf.elte.hu
[2] Université Paris Est, LACL (EA 4219), UPEC, F-94010, Créteil, France
verlan@u-pec.fr

Abstract. The general concept, called the formal framework of P systems provides a representation to study and analyze different models of P systems. In this paper, two well-known models, P colonies and P systems with multi-stable catalysts are considered. We show that the obtained representations are identical, thus both models can be related using a bi-simulation. This fact opens new approaches both for studying P colonies and catalytic P systems.

1 Introduction

Due to different motivations, there have been several variants of P systems introduced. However, all models have some common basic features as summarized in [5, 9]. Among these characteristics we find

- a description of the initial structure or architecture (indicating the graph relation between the compartments and any additional information as labels, charges, etc.),
- a list of the initial multisets of objects present in each compartment at the beginning of the computation,
- a set of rules, acting over objects and / or over the structure.

Usually, the configuration of a P system is represented by the current contents of the compartments and the current structure of the system.

P systems work with transitions between configurations; a finite sequence of such transitions of a P system Π starting with the initial configuration and ending in some final configuration is called a computation. The final configuration is usually given by halting.

To give a more precise description of the semantics, the following notions (functions) were defined:

- $Applicable(\Pi, \mathcal{C}, \delta)$ – the set of multisets of rules of Π applicable to the configuration $\mathcal{C}$, according to some derivation mode δ.

- $Apply(\Pi, \mathcal{C}, R)$ – the configuration obtained by the (usually parallel) application of the multiset of rules R to the configuration $\mathcal{C}$.
- $Halt(\Pi, \mathcal{C}, \delta)$ – a predicate that yields true if $\mathcal{C}$ is a halting configuration of the system Π using the derivation mode δ.
- $Result(\Pi, \mathcal{C})$ – a function giving the result of the computation of the P system Π when the halting configuration $\mathcal{C}$ has been reached. Usually, this is an integer function. However, generalizations as for example, Boolean or vector functions can also be considered.

We note that δ, above, differs from the dissolution symbol used in some P system models.

The transition of a P system Π according to the derivation mode δ (usually, the maximally parallel derivation mode) is defined as follows: the system changes from a configuration $\mathcal{C}$ to $\mathcal{C}'$ (written as $\mathcal{C} \Rightarrow \mathcal{C}'$) iff

$$\mathcal{C}' = Apply(\Pi, \mathcal{C}, R), \text{ for some } R \in Applicable(\Pi, \mathcal{C}, \delta)$$

The result of the computation of a P system is usually interpreted as the union of the results of all possible computations.

The precise interpretation of the four notions (functions) above depends on the chosen model of P systems. The goal of works [4, 5, 9] was to provide a concrete family of P systems based on the structure of *network of cells* together with a series of definitions of the functions above. The obtained model as well as the accompanying tools and methods together are called the *formal framework of P systems*. It has the property that most of the existing models of P systems could be obtained by a strong bi-simulation of a restricted version (eventually, using a simple encoding) of this formal framework with respect to different parameters, see [10] for some examples. We recall that a simulation of one transitional system by another corresponds to an order relation on corresponding equivalent states [6]. Basically, this means that a step in the simulated system corresponds to one or several steps in the simulating one. In the case of a strong simulation, one step of the simulated system is performed using one step in the simulating system. If two systems can simulate each other, then we speak about bi-simulation.

In this paper, based on formal framework we provide a strong bi-simulation between two well-known models, namely P colonies and multi-stable (purely) catalytic P systems. P colonies are a finite collection of agents which interact with a shared environment via their own sets of programs. Each program is a limited number of very simple rules. Under functioning, the agents act in a maximally parallel manner and they change their own state and exchange symbols with the environment. Purely catalytic P systems are given with multiset rewriting rules where each rule has occurrences of distinguished symbols called catalysts. In the original model, catalysts cannot change, in case of multi-stable catalytic P systems catalysts are allowed to change only to some other, distinguished catalysts.

Our bi-simulation demonstrates that although the two models are formally different, one can be used to solve problems concerning the other one. For ex-

ample, both models are computationally complete, thus a proof for one of the models can be "translated" to a proof for the other one.

After providing the bi-simulation and some examples, we discuss the results and propose topics for future research.

2 Definitions and Notations

We assume that the reader is familiar with basic notions of formal language theory and membrane computing; for further details consult [7] and [8].

For a finite multiset of symbols M over an alphabet V, $supp(M)$ denotes the set of symbols in M (the support of M) and $|M|$ denotes its size, i.e., the total number of its symbols. By $|M|_x$, the number of occurrences of symbol x in M is denoted. By V° we denote the set of all finite multisets over V.

Throughout the paper, every finite multiset M is given as a string w, where M and w have the same number of occurrences of symbol a, for each $a \in V$.

2.1 Network of Cells

In this section we provide a summarized version of the definition of a *network of cells*, the class containing all networks of cells forming the structure of the formal framework. The definitions are based on those given in [5]. This version considers only static P systems where the membrane structure does not change under the computation (this also includes systems with the dissolution of membranes). We note that in [4], an extension of the formal framework to P systems with dynamically evolving structure is proposed. However, in order to have a more simple presentation, in this paper we will only consider the first variant. We remark that in the case of static structures both variants coincide, although the notation is slightly different.

Definition 1 ([5]). *A* network of cells of degree $n \geq 1$ *is a construct*

$$\Pi = (n, V, w, Inf, R)$$

where

1. *n is the number of cells;*
2. *V is an* alphabet*;*
3. *$w = (w_1, \ldots, w_n)$ where $w_i \in V^\circ$, for all $1 \leq i \leq n$, is the* finite multiset initially associated to cell *i;*
4. *$Inf = (Inf_1, \ldots, Inf_n)$ where $Inf_i \subseteq V$, for all $1 \leq i \leq n$, is the* set of symbols occurring infinitely often in cell *i (in most of the cases, only one cell, called the* environment*, will contain symbols occurring with infinite multiplicity);*
5. *R is a finite set of rules of the form*

$$(X \to Y; P, Q)$$

where $X = (x_1, \ldots, x_n)$, $Y = (y_1, \ldots, y_n)$, $x_i, y_i \in V^\circ$, $1 \leq i \leq n$, are vectors of multisets over V and $P = (p_1, \ldots, p_n)$, $Q = (q_1, \ldots, q_n)$, p_i, q_i, $1 \leq i \leq n$ are finite sets of multisets over V. We will also use the notation

$$(1, x_1) \ldots (n, x_n) \rightarrow (1, y_1) \ldots (n, y_n) \, ; [(1, p_1) \ldots (1, p_n)]; [(1, q_1) \ldots (n, q_n)]$$

for a rule $(X \rightarrow Y; P, Q)$; moreover, if some p_i or q_i is an empty set or some x_i or y_i is equal to the empty multiset, $1 \leq i \leq n$, then we may omit it from the specification of the rule.

The semantics of the above rule is as follows: objects x_i from cells i are rewritten into objects y_j in cells j, $1 \leq i, j \leq n$, if every cell k, $1 \leq k \leq n$, contains all multisets from p_k and does not contain any multiset from q_k. In other words, the first part of the rule specifies the rewriting of symbols, the second part of the rule specifies permitting conditions and the third part of the rule specifies the forbidding conditions.

For a rule r of the form above, the set

$$\{i \mid x_i \neq \lambda \text{ or } y_i \neq \lambda \text{ or } p_i \neq \emptyset \text{ or } q_i \neq \emptyset\}$$

induces a (hypergraph) relation between the interacting cells. However, this relation does not need to give rise to a *structure* relation like a tree as in P systems or a graph as in tissue P systems.

A *configuration* C of Π is an n-tuple of multisets over V $(u_1, \ldots, u_n)$ satisfying $u_i \cap Inf_i = \emptyset$, $1 \leq i \leq n$.

In the sequel, networks of cells as intermediate models will assist to establish a bi-simulation between two variants of P systems, namely P colonies and P systems with multi-stable catalysts.

2.2 P Colonies

Next we provide the concept of a P colony, based on the formalism given in [7].

A P colony $\Pi = (O, e, f, C_1, \ldots, C_n)$, consists of n cells (agents) C_i, $1 \leq i \leq n$, each of them consisting of a multiset of exactly k symbols and an environment consisting of initially a distinguished symbol e in an unbounded number of copies. Every cell C_i has a set of programs $\{p_{i,1}, \ldots, p_{i,k_i}\}$, where each program $p_{i,j}$ consists of exactly k rules of the forms $a \rightarrow b$ (*evolution rule* or *internal point mutation*), $c \leftrightarrow d$ (*one object exchange* with the environment), or r_1/r_2 (*priority rule*, where r_1 and r_2 are arbitrary combinations of point mutation and/or exchange rules).

The computation starts in the initial configuration, i.e., the n-tuple of the initial contents of the cells. It can be performed in the maximally parallel (*par*) or in the sequential (*seq*) mode, the computation mode is assigned to the system at the beginning. If no more program is applicable, then the P colony halts and the result is collected as the number of distinguished symbols f in the environment. The result of the computation of Π is denoted by $N(\Pi)$.

We note that the result can be defined in such a way, too, that we consider the number of all symbols in the environment which are different from e.

The number of cells, the maximal number of programs in a cell, and the maximal number of rules in each program in a given P colony Π are called the degree, the height, and the capacity of Π, respectively.

The family of sets of numbers computed in the derivation mode x for $x \in \{par, seq\}$ by P colonies of capacity k, degree at most $n \geq 1$ and height at most $h \geq 1$, without (resp. with) using priority rules in their programs, is denoted by $NPCol_x(k, n, h)$ (resp. $NPCol_xK(k, n, h)$).

Notice that a strong bi-simulation of the P colony model and the formal framework can be given as follows.

- each rule $a \to b$ in p_{ij} becomes $r_{ij} : (i, a) \to (i, b)$;
- each rule $a \leftrightarrow b$ in p_{ij} becomes $r_{ij} : (i, a)(0, b) \to (i, b)(0, a)$;
- each rule r_1/r_2 in p_{ij} becomes:
  - $p^1_{ij} : r_1$, $p^2_{ij} : r_2; [\emptyset]; [\{(i, a)\}]$ if r_1 is an evolution rule $(a \to b)$
  - $p^1_{ij} : r_1$, $p^2_{ij} : r_2; [\emptyset]; [\{(i, a)(0, b)\}]$ if r_1 is an exchange rule $(a \leftrightarrow b)$.

For the derivation mode, each program becomes a rule partition and then the derivation mode requires to be maximal, but using exactly k rules from each partition (or using all rules from a partition). In the sequential case, the derivation mode prescribes to use only one partition (but all rules from that partition).

Example 1 ([10]). Consider the following P colony Π having 3 cells. For simplicity, we provide only the initial multisets and the programs of the cells.

- C_1 contains the initial multiset aa and the following programs: $p_{11} : a \to b, a \leftrightarrow e$, $p_{12} : a \to c, a \leftrightarrow e$, $p_{13} : b \to a, e \to a$.
- C_2 contains the initial multiset be and the following program: $p_{21} : b \leftrightarrow e, e \to b$.
- C_3 contains the initial multiset ee and the following programs: $p_{31} : e \leftrightarrow a, e \leftrightarrow b$, $p_{32} : b \to f, a \to b$, $p_{33} : f \leftrightarrow a, b \to b$.

Figure 1 shows a graphical representation of this system.

We transform this system to a network of cells Π' having 4 cells (numbered from 0 to 3). Cell 0 corresponds to the environment. Cells 1, 2, 3 correspond to the cells of Π and have the same initial contents as the corresponding agent. We define $Inf_0 = \{e\}$. System Π' contains the following rules:

Rules simulating programs from the first cell:

$r_{111} : (1, a) \to (1, b)$ $\quad$ $r_{112} : (1, a)(0, e) \to (1, e)(0, a)$

$r_{121} : (1, a) \to (1, c)$ $\quad$ $r_{122} : (1, a)(0, e) \to (1, e)(0, a)$

$r_{131} : (1, b) \to (1, a)$ $\quad$ $r_{132} : (1, e) \to (1, a)$

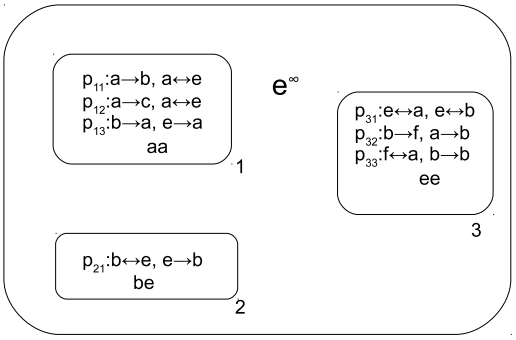

Fig. 1. The P colony from Example 1.

Rules simulating programs from the second cell:

$$r_{211} : (2, b)(0, e) \to (2, e)(0, b) \qquad r_{212} : (2, e) \to (2, b)$$

Rules simulating programs from the third cell:

$$r_{311} : (3, e)(0, a) \to (3, a)(0, e) \qquad r_{312} : (3, e)(0, b) \to (3, b)(0, e)$$
$$r_{321} : (3, b) \to (3, f) \qquad r_{322} : (3, a) \to (3, b)$$
$$r_{331} : (3, f)(0, a) \to (3, a)(0, f) \qquad r_{332} : (3, b) \to (3, b)$$

We remark that the derivation mode of P colonies groups rules corresponding to programs, uses maximal parallelism or sequential mode, and it requires that all rules from a group should be used. Since working with one symbol, the group r_{111} and r_{112} from the above example is equivalent to the application of a single rule $r_{11} : (1, aa)(0, e) \to (1, be)(0, a)$. Hence, we obtain that a program corresponds to a more complicated rule, and k is the size of the left-hand side (LSH) of this rule (and equal to the right-hand side, i.e., RHS). By considering such rules, the evolution of a P colony becomes just maximally parallel or sequential.

This consideration yields to the following network of cells Π'' (working in sequential or maximally-parallel manner):

$$r_{11} : (1, aa)(0, e) \to (1, be)(0, a) \qquad r_{12} : (1, aa)(0, e) \to (1, ce)(0, a)$$
$$r_{13} : (1, be) \to (1, aa)$$
$$r_{21} : (2, be)(0, e) \to (2, be)(0, b)$$
$$r_{31} : (3, ee)(0, ab) \to (3, ab)(0, ee) \qquad r_{32} : (3, ab) \to (3, fb)$$
$$r_{33} : (3, bf)(0, a) \to (3, ab)(0, f)$$

Since the number of combinations of objects in an agent is finite, it can be represented by a single symbol, a state. Also, symbol e from cell 0 can be ignored as it carries no information. This permits to deduce that a P colony corresponds to a cooperative rewriting mechanism with the size of LHS or RHS at most $k+1$

and forbidding conditions (if checking rules are present). In the next section we refine this observation by showing that the rewriting is performed in a catalytic-like manner.

2.3 P systems with Multi-stable Catalysts

In this section we extend the notion of a P system with catalysts to that variant where the catalysts can have multiple states. For catalytic P systems, consult [7].

Let V and C be two disjoint alphabets, let $k > 0$, and let C have a partition $C = C_1 \cup \cdots \cup C_n$ such that $1 \leq |C_i| \leq k$. We say that each partition is a multi-stable catalyst and we define $Period(C_i) = |C_i|$ the *period* of the catalyst C_i, $1 \leq i \leq n$.

In the sequel, the elements of a multi-stable catalyst C_i having period k will be denoted by $c_i^{(j)}$, $1 \leq j \leq k$.

A k-states multi-stable (purely) catalytic P system with n catalysts is a construct $\Gamma = (V, C, R, w)$, where V is the set of non-catalytic objects of Γ, $C = C_1 \cup \cdots \cup C_n$ with catalysts C_i, having period at most k, $1 \leq i \leq n$.

R is a finite set of rules where each rule is of the following form

$$c_i^{(j)} u \to c_i^{(t)} v, \text{ where } 1 \leq j, t \leq Period(C_i), 1 \leq i \leq n \text{ and } u, v \in V^\circ.$$

The initial configuration of Γ, w is a multiset over $V \cup C$, with at most one element of each multi-stable catalyst C_i, i.e., $w \subseteq (V \cup C)^\circ$, with the condition that $\sum_{j=1}^{Period(C_i)} |w|_{c_i^{(j)}} \leq 1$, $1 \leq i \leq n$.

Notice that the rules of a multi-stable catalytic P system with multiple states can easily be represented in the formal framework by [5],[10] as follows:

$$(0, c_i^{(j)} u) \to (0, c_i^{(t)} v), \text{ for all } c_i^{(j)} u \to c_i^{(t)} v \in R.$$

As standard P systems, the k-states multi-stable (purely) catalytic P systems Γ with n catalysts work by transitions of their configurations where the rules are applied in the maximally parallel manner. A successful computation performed by Γ is a finite sequence of transitions starting in its initial configuration and ending by halting; the result of the computation is the number of non-catalytic objects in the halting configuration. The result of the computation is denoted by $N(\Gamma)$.

3 Bi-simulation of the Two Models

In this section we demonstrate the equivalence of P colonies and multi-stable (purely) catalytic P systems by using their representation in the above formal framework.

We first show that any (recursively enumerable) set of numbers that can be computed by a P colony (in the sense defined above) can be computed by a multi-stable catalytic P system as well.

Theorem 1. *For any P colony $\Pi = (O, e, w_0, P_1, \ldots, P_n)$ of size (k, n, h) there exists a h'-states multi-stable purely catalytic P system $\Gamma = (O, C, w, R)$ with n catalysts with $h' \leq h + 1$ such that $N(\Pi) = N(\Gamma)$.*

Proof. To simplify the presentation, we consider P colonies that do not contain checking rules.

According to the discussion above (see also [10]), every P colony can be represented by the formal framework. To this goal, any program p located in cell i is replaced by a rule of the corresponding network of cells. Let $p = p_1 \cup p_2$, where p_c contains all the communication rules and p_r contains all the rewriting rules of p. Let $lhs_c(p)$ (resp. $rhs_c(p)$) denote the multiset of letters of all left-hand (resp. right-hand) sides of the communication rules; we consider the same notation for the rewriting rules, using the index r. For simplicity, we will speak of sum of the left-hand sides (resp. right-hand sides) of the rules in the sequel and we will use notation $+$.

Since the definition of the P colony requires that if a program is used, then all of its rules should be applied, therefore we obtain that the execution of a program p is equivalent to the following rule given in terms of the formal framework:

$$\begin{aligned}(i, x)(0, y) \to (i, x')(0, y'), \text{ where} \\ x = lhs_c(p) + lhs_r(p), y = rhs_c(p), \\ x' = rhs_c(p) + rhs_r(p), y' = lhs_c(p).\end{aligned} \tag{1}$$

Since in every step of the computation every cell in a P colony contains a constant number of objects equal to its capacity k, each cell contents can be interpreted as a number z in base $k + 1$ having exactly $|O|$ bits. Alphabet O is equal to $\{o_1, \ldots, o_s\}$ and any o_i (and e) can appear in any contents in at most k copies. Thus $|O|$ bits represent the number of occurrences of object o_i in a cell contents c. Under this interpretation, the rules of a program specify some other number z' equal to the value of the contents of the cell after the application of the program. Since the number of rules in a program is exactly k, for each number z and program p there is exactly one number z' associated.

We remark that for a cell i having h programs there are at most $h+1$ different possible configurations of the cell contents. We number these configurations from 1 to i_h, where $i_h \leq h + 1$. Let f be a bijection between all possible values of cell configurations and $1, \ldots, i_h$. Thus, we can rewrite 1 as follows:

$$\begin{aligned}(i, c^{(f(z))})(0, y) \to (i, c^{(f(z'))})(0, y'), \text{ where} \\ y = rhs_c(p), y' = lhs_c(p), 1 \leq z, z' \leq i_h.\end{aligned} \tag{2}$$

We can further transform this rule as follows:

$$(0, c_i^{(f(z))} y) \to (0, c_i^{(f(z'))} y') \tag{3}$$

It can clearly be seen that this rule corresponds to a rule of a multi-stable (purely) catalytic P system.

Hence, starting from the P colony Π, components of Γ can be constructed. We first remark that object e in P colonies act as an empty symbol, so we replace all its occurrences by λ in the obtained catalytic rules. First, the initial multiset of Γ is determined from the initial configuration of Π. Since every rule $c_i^{(f(z))}y \to c_i^{(f(z'))}y'$ of Γ correspond to the application of a program p in Π described above, it can easily be seen that any transition from configuration c_1 to configuration c_2 of Π corresponds to the application of an m-tuple of rules in Γ, where $m \leq n$. Notice that depending on the applicability of their programs, some components may remain inactive. Since the initial multiset of Γ contains at most n catalysts, Γ has only catalytic rules, at any computation step as many catalytic rules are applied in parallel as possible, i.e. at most n. Since these rules correspond to programs of pairwise different components of Π, every computation in Γ corresponds to a computation in Π as well. Thus, it is easy to see that the number of non-catalytic objects at halting of Γ is equal to the number of objects in the environment of Π which are different from e at halting. □

Example 2. Let us consider P colony Π from Example 1. We recall the corresponding rules.

- C_1 contains the initial multiset aa and the following programs:
  $p_{11} : a \to b, a \leftrightarrow e$, $p_{12} : a \to c, a \leftrightarrow e$, $p_{13} : b \to a, e \to a$.
- C_2 contains the initial multiset be and the following program:
  $p_{21} : b \leftrightarrow e, e \to b$.
- C_3 contains the initial multiset ee and the following programs:
  $p_{31} : e \leftrightarrow a, e \leftrightarrow b$, $p_{32} : b \to f, a \to b$, $p_{33} : f \leftrightarrow a, b \to b$.

Let $O = \{a, b, c, e, f\}$, and let $o_1 = a$,...,$o_5 = f$, in this order. The different cell contents are $A = (aa, be, ce, ee, ab, bf)$ which correspond to numbers 00002, 01010, 01100, 02000, 00011, 010010 in base $k + 1 = 6$. For simplicity, let us denote these numbers by s_1, s_2, s_3, s_4, s_5, and s_6, respectively.

Then by constructing the multi-stable catalytic P system we obtain the following rules:

- $C_1^{s_1} \to C_1^{s_1}a$, , $C_1^{s_1} \to C_1^{s_3}a$, $C_1^{s_2} \to C_1^{s_1}$,
- $C_2^{s_2} \to C_2^{s_2}b$,
- $C_3^{s_4}ab \to C_3^{s_5}$, $C_3^{s_5} \to C_3^{s_6}$, $C_3^{s_6}a \to C_3^{s_5}f$.

Next we show that the sets of numbers computed by multi-stable catalytic P systems can be computed by P colonies as well.

Theorem 2. *For any h-states multi-stable catalytic P system $\Gamma = (O, C, w, R)$ with n catalysts there exists a P colony $\Pi = (O, e, w_0, P_1, \ldots, P_n)$ of size (k, n, h) such that $N(\Pi) = N(\Gamma)$ holds.*

Proof. We construct Π as follows. P colony Π has n cells and each cell i has $Period(C_i)$ programs. Now we will show how these programs are constructed.

Consider a rule $c_i^j u \to c_i^t v \in R$. We suppose that $|u| = |v|$. If this is not the case, then we complement the smaller multiset by adding the needed amount of symbols e. That is, if $|u| < |v|$ then let $u' = u + e^{|v|-|u|}$. Suppose that $u = u_1 \dots u_s$ and $v = v_1 \dots v_s$. We will construct the program p_j corresponding to this rule. It will be composed from two parts. The first part will contain communication rules that simulate the rewriting of u to v in the above rule. The second part contains rewriting rules that allow to complement the encoding of the catalyst state by the contents of the cell.

In order to determine the corresponding rewriting rules we should first find an encoding for each state of the catalyst. This encoding can be obtained as a solution of the following integer optimization problem.

$$\begin{aligned}
&k \to min,\\
&\sum_{a \in O} x_a^{i,j} = k, \quad 1 \le j \le Period(C_i),\\
&x_a^{i,j} \ge |v|_a, \qquad x_a^{i,t} \ge |u|_a, \qquad \text{for any } c_i^j u \to c_i^t v \in R,\\
&x_a^{i,j} \in \mathbb{N}, \quad a \in O, 1 \le i \le n, 1 \le j \le Period(C_i).
\end{aligned} \tag{4}$$

The inequalities state that the symbols that are sent out (resp. received in) by the exchange rules of the P colony belong to the coding of state j (resp. t) of the catalyst C_i.

We remark that since inequalities 4 do not impose an upper bound value for $x_a^{i,j}$, there is always a solution for this system. In case of several possible solutions, we prefer solutions having the maximal number of symbols e.

The capacity of the P colony is the value k.

Let $x_a^{i,j}$, $a \in O$, $1 \le j \le Period(C_i)$ be a solution of the above problem. Let $Code(c_i^j) = \sum_{a \in O} a^{x_a^{i,j}}$. Let $c_i^j u \to c_i^t v \in R$ and let $|u| = |v| = s$, $d^j = Code(c_i^j) - v$ and $d^t = Code(c_i^t) - u$. Suppose that $d^l = d_1^l \dots d_m^l, l \in \{j, t\}$. Then

$$p_j = (v_1 \leftrightarrow u_1; \dots v_s \leftrightarrow u_s; d_1^j \to d_1^t; \dots d_m^j \to d_m^t).$$

Now we are able to construct the colony: for every C_i, $1 \le i \le n$, P colony Π will have component P_i. The programs belonging to P_i are obtained from the rules $C_i^j u \to C_i^t v$, in the above described manner. Notice that any program of Π, determined above encodes the application of the corresponding catalytic rule, thus, the programs to be applied and the catalytic rules correspond to each other. Since at the beginning of the computation the initial state of Γ contains at most one element of each multi-stable catalyst and both systems apply the maximally parallel computation, we obtain that the two systems compute the same set of numbers. □

Example 3. To demonstrate the previous construction, we add an example.

Consider the following multi-stable catalytic P system $\Pi = (O, C, w_1, R_1)$.

$1.1 : C_1^1 a \to C_1^2 bc$ $\quad$ $2.1 : C_2^1 \to C_2^2$ $\quad$ $3.1 : C_3^1 \to C_3^2 a$

$1.2 : C_1^1 a \to C_1^3 c$ $\quad$ $2.2 : C_2^2 b \to C_2^2$ $\quad$ $3.2 : C_3^2 c \to C_3^1 b$

$1.3 : C_1^2 ac \to C_1^1 aa$ $\quad$ $3.3 : C_3^1 \to C_3^3$

We transform rules 1.1, 2.2 and 3.1 by adding symbol e in order to balance the number of symbols at both sides:

$1.1 : C_1^1 ae \to C_1^2 bc$ $\quad$ $2.2 : C_2^2 b \to C_2^2 e$ $\quad$ $3.1 : C_3^1 \to C_3^2 a$

Then the corresponding minimization problem is the following:

$$
\begin{aligned}
& k \to min \\
& x_a^{i,j} + x_b^{i,j} + x_c^{i,j} + x_e^{i,j} = k, \quad 1 \le i, j \le 3 \\
& x_b^{1,1} \ge 1,\ x_c^{1,1} \ge 1,\ x_a^{1,2} \ge 1,\ x_e^{1,2} \ge 1, \\
& x_c^{1,1} \ge 1,\ x_a^{1,3} \ge 1, \\
& x_a^{1,1} \ge 1,\ x_c^{1,1} \ge 1,\ x_a^{1,2} \ge 2 \\
& x_e^{2,2} \ge 1,\ x_b^{2,2} \ge 1 \\
& x_a^{3,1} \ge 1,\ x_e^{3,2} \ge 1,\ x_c^{3,1} \ge 1,\ x_b^{3,2} \ge 1 \\
& x_a^{i,j} \in \mathbb{N}, \quad a \in O, 1 \le i, j \le 3.
\end{aligned}
$$

We can regroup inequalities by the corresponding state:

$$
\begin{aligned}
& k \to min \\
& x_a^{i,j} + x_b^{i,j} + x_c^{i,j} + x_e^{i,j} = k, \quad 1 \le i, j \le 3 \\
& x_a^{1,1} \ge 1,\ x_b^{1,1} \ge 1,\ x_c^{1,1} \ge 1 \\
& x_a^{1,2} \ge 2,\ x_e^{1,2} \ge 1 \\
& x_a^{1,3} \ge 1 \\
& x_b^{2,2} \ge 1,\ x_e^{2,2} \ge 1 \\
& x_a^{3,1} \ge 1,\ x_c^{3,1} \ge 1 \\
& x_b^{3,2} \ge 1,\ x_e^{3,2} \ge 1 \\
& x_a^{i,j} \in \mathbb{N}, \quad a \in O, 1 \le i, j \le 3.
\end{aligned}
$$

The minimal value of k is equal to 3 and one of possible solutions yields to the following codes for c_i^j:

x	c_1^1	c_1^2	c_1^3	c_2^1	c_2^2	c_3^1	c_3^2	c_3^3
$Code(x)$	abc	aae	aaa	eee	bee	ace	bee	eee

We remark that catalysts C_2 and C_3 can be represented only using two symbols.

The obtained P colony is shown in Fig. 3.

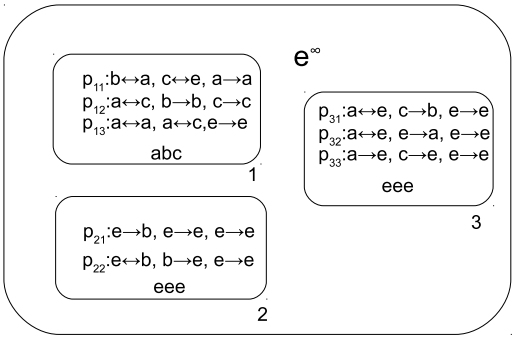

Fig. 2. The P colony constructed in Example 3.

4 Conclusions

In this paper we have shown a strong bi-simulation between the model of P colonies and pure multi-stable catalytic P systems. This result was obtained by using the formal framework for P systems as intermediate step.

As immediate consequence of the results of this paper, it is possible to rewrite existing results from the area of P colonies in terms of multi-stable catalytic P systems and conversely. These investigations are topics of future research. Another consequence is the possibility to conduct proofs in terms of purely catalytic P systems (that tend to be simpler) and automatically transform them to P colonies.

Furthermore, this article allows to establish the correspondence between different extensions of P colonies (see [3]) and particular variants of catalytic P systems. For example, the evolving environment extension [2] corresponds to the same multi-stable catalytic P systems, so it can be simulated by a P colony with a greater capacity. Homogeneous P colonies [1] correspond to catalytic P systems having same rules for all catalysts.

Other possible extensions can also be discussed. For example, non-pure catalytic systems would correspond to P colonies having special rules allowing to evolve objects by themselves in the environment. Another possibility that follows from our constructions is to consider P colonies where the capacity is different in each cell.

5 Acknowledgement

The work of E. CS-V. was supported by NKFIH (National Research, Development, and Innovation Office), Hungary, grant no. K 120558.

References

1. L. Cienciala, L. Ciencialová, and A. Kelemenová. Homogeneous P colonies. *Computing and Informatics*, 27(3):481–496, 2008.
2. L. Ciencialová, L. Cienciala, and P. Sosík. P colonies with evolving environment. In A. Leporati and C. Zandron, editors, *Proceedings of the 17th International Conference on Membrane Computing (CMC17)*, pages 105–118, 2016.
3. L. Ciencialová, E. Csuhaj-Varjú, L. Cienciala, and P. Sosík. P colonies. *Bulletin of the International Membrane Computing Society*, 1(2):119–156, 2016.
4. R. Freund, I. Pérez-Hurtado, A. Riscos-Núñez, and S. Verlan. A formalization of membrane systems with dynamically evolving structures. *International Journal of Computer Mathematics*, 90(4):801–815, 2013.
5. R. Freund and S. Verlan. A formal framework for static (tissue) P systems. In G. Eleftherakis, P. Kefalas, G. Păun, G. Rozenberg, and A. Salomaa, editors, *Membrane Computing, 8th International Workshop, WMC 2007, Revised Selected and Invited Papers*, volume 4860 of *LNCS*, pages 271–284. Springer, 2007.
6. R. Milner. An algebraic definition of simulation between programs. In *Proceedings of the 2Nd International Joint Conference on Artificial Intelligence*, IJCAI'71, pages 481–489, San Francisco, CA, USA, 1971. Morgan Kaufmann Publishers Inc.
7. G. Păun, G. Rozenberg, and A. Salomaa, editors. *The Oxford Handbook of Membrane Computing*. Oxford University Press, 2009.
8. G. Rozenberg and A. Salomaa, editors. *Handbook of Formal Languages*, volume 1–3. Springer, 1997.
9. S. Verlan. Study of language-theoretic computational paradigms inspired by biology. Habilitation thesis, Université Paris Est, 2010.
10. S. Verlan. Using the formal framework for P systems. In A. Alhazov, S. Cojocaru, M. Gheorghe, Y. Rogozhin, G. Rozenberg, and A. Salomaa, editors, *Membrane Computing - 14th International Conference, CMC 2013, Chişinău, Republic of Moldova, August 20-23, 2013, Revised Selected Papers*, volume 8340 of *Lecture Notes in Computer Science*, pages 56–79. Springer, 2013.