



# Computationally universal P systems without priorities: two catalysts are sufficient

Rudolf Freund<sup>a,\*</sup>, Lila Kari<sup>b</sup>, Marion Oswald<sup>a</sup>, Petr Sosík<sup>b,c</sup>

<sup>a</sup>*Faculty of Informatics, Vienna University of Technology, Favoritenstr. 9–11, A–1040 Vienna, Austria*

<sup>b</sup>*Department of Computer Science, The University of Western Ontario, London, Ont., Canada, N6A 5B7*

<sup>c</sup>*Institute of Computer Science, Silesian University, Opava, Czech Republic*

Received 18 November 2003; received in revised form 16 April 2004; accepted 4 June 2004

## Abstract

The original model of P systems with symbol objects introduced by Păun was shown to be computationally universal, provided that catalysts and priorities of rules are used. By reduction via register machines Sosík and Freund proved that the priorities may be omitted from the model without loss of computational power. Freund, Oswald, and Sosík considered several variants of P systems with catalysts (but without priorities) and investigated the number of catalysts needed for these specific variants to be computationally universal. It was shown that for the classic model of P systems with the minimal number of two membranes the number of catalysts can be reduced from six to five; using the idea of final states the number of catalysts could even be reduced to four. In this paper we are able to reduce the number of catalysts again: two catalysts are already sufficient. For extended P systems we even need only one membrane and two catalysts. For the (purely) catalytic systems considered by Ibarra only three catalysts are already enough.

© 2004 Elsevier B.V. All rights reserved.

*Keywords:* Membrane computing; P systems; Catalysts; Complexity; Universality

## 1. Introduction

In the original paper introducing membrane systems (P systems) in [14] as a symbol manipulating model catalysts as well as priority relations on the rules were used to prove

\* Corresponding author.

*E-mail addresses:* [rudi@emcc.at](mailto:rudi@emcc.at) (R. Freund), [lila@csd.uwo.ca](mailto:lila@csd.uwo.ca) (L. Kari), [marion@emcc.at](mailto:marion@emcc.at) (M. Oswald), [sosik@csd.uwo.ca](mailto:sosik@csd.uwo.ca), [Petr.Sosik@fpf.slu.cz](mailto:Petr.Sosik@fpf.slu.cz) (P. Sosík).

them to be computationally universal; in [18,19] it was shown that a priority relation on the rules is not necessary to obtain this universality result. In [8] the number of catalysts was reduced by one for the variants of P systems with two membranes considered there; moreover, the number of catalysts could even be reduced by one more when considering computations reaching some finitely specified final states as in the model of P automata introduced in [2] instead of halting computations. We will now show that even two catalysts are already sufficient for all these variants.

In extended P systems we specify a terminal alphabet and only consider the terminal symbols contained in the skin membrane at the end of a successful computation (in effect this means that we ignore the catalysts, which of course can never be eliminated); again the skin membrane and two catalysts are sufficient. In [10] (purely) catalytic P systems were introduced and from results obtained in [8,18] it was observed that seven catalysts are sufficient if we only allow rules with catalysts; here we show that even three catalysts are already enough.

In the following section, after some prerequisites from formal language theory, we give a precise definition of the model of register machines used in the subsequent proofs. Then we define the specific variants of P systems considered in this paper. In the further parts of this paper we show how we can reduce the number of catalysts in P systems with specific stopping conditions by using new proof techniques for simulating register machines. A short summary finally concludes the paper.

## 2. Definitions

For well-known notions and basic results from the theory of formal languages, the reader is referred to [3,17]. We only give some basic notations first. For an alphabet  $V$ , by  $V^*$  we denote the free monoid generated by  $V$  under the operation of concatenation; the *empty string* is denoted by  $\lambda$  and  $V^* \setminus \{\lambda\}$  is denoted by  $V^+$ . Any subset of  $V^*$  ( $V^+$ ) is called a ( $\lambda$ -free) *language*. Two languages  $L, L' \subseteq V^*$  are considered to be equal if  $L \setminus \{\lambda\} = L' \setminus \{\lambda\}$ . Moreover, by  $\mathbf{N}_0$  we denote the set of non-negative integers and by  $N_0^\beta RE$  we denote the family of recursively enumerable sets of  $\beta$ -vectors  $(y_1, \dots, y_\beta)$  of non-negative integers. Two sets of  $\beta$ -vectors are considered to be equal if they only differ at most by the zero-vector  $(0, \dots, 0)$ .

Let  $m \geq 2$  and let  $k, l$  be two positive integers not greater than  $m$ ; then we define

$$l \ominus_m k := \begin{cases} l - k, & \text{for } l > k, \\ l - k + m, & \text{for } l \leq k. \end{cases}$$

### 2.1. Register machines

In this subsection we briefly recall the concept of Minsky's register machine (e.g. see [12]). Such an abstract machine uses a finite numbers of registers for storing arbitrarily large non-negative integers and runs a program consisting of numbered instructions of various simple types. Several variants of the machine with different numbers of registers and different

instruction sets were shown to be computationally universal (e.g. see [12] for some original definitions and proofs as well as [5–7] for the definitions and results we use in this paper).

An  $n$ -register machine is a construct  $M = (n, P, i, h)$ , where

- $n$  is the number of registers,
- $P$  is a set of labelled instructions of the form  $j : (op(r), k, l)$ , where  $op(r)$  is an operation on register  $r$  of  $M$ ,  $j, k, l$  are labels from the set  $Lab(M)$  (which numbers the instructions of the program of  $M$  represented by  $P$ ),
- $i$  is the initial label, and
- $h$  is the final label.

The machine is capable of the following instructions:

$(A(r), k, l)$ : Add one to the contents of register  $r$  and proceed to instruction  $k$  or to instruction  $l$ ; in the deterministic variants usually considered in the literature we demand  $k = l$ .

$(S(r), k, l)$ : If register  $r$  is not empty then subtract one from its contents and go to instruction  $k$ , otherwise proceed to instruction  $l$ .

**Halt**: Stop the machine. This additional instruction can only be assigned to the final label  $h$ .

In their *deterministic variant*, such  $n$ -register machines can be used to compute any partial recursive function  $f : \mathbb{N}_0^\alpha \rightarrow \mathbb{N}_0^\beta$ ; starting with  $(n_1, \dots, n_\alpha) \in \mathbb{N}_0^\alpha$  in registers 1 to  $\alpha$ ,  $M$  has computed  $f(n_1, \dots, n_\alpha) = (r_1, \dots, r_\beta)$  if it halts in the final label  $h$  with registers 1 to  $\beta$  containing  $r_1$  to  $r_\beta$ . If the final label cannot be reached,  $f(n_1, \dots, n_\alpha)$  remains undefined.

A deterministic  $n$ -register machine can also accept an input  $(n_1, \dots, n_\alpha) \in \mathbb{N}_0^\alpha$  in registers 1 to  $\alpha$ , which is recognized if the register machine finally stops by the halt instruction with all its registers being empty. If the machine does not halt, the analysis is not successful.

In their *non-deterministic variant*  $n$ -register machines can compute any recursively enumerable set of non-negative integers (or of vectors of non-negative integers). Starting with all registers being empty, we consider a computation of the  $n$ -register machine to be successful, if it halts with the result being contained in the first ( $\beta$ ) register(s) and with all other registers being empty.

The results proved in [5] (based on the results established in [12]) as well as in [6,7] immediately lead us to the following results which differ from the original results mainly by the fact that the result of a computation is stored in registers that must not be decremented.

**Proposition 1.** *For any partial recursive function  $f : \mathbb{N}_0^\alpha \rightarrow \mathbb{N}_0^\beta$ , there exists a deterministic  $(\alpha + 2 + \beta)$ -register machine  $M$  computing  $f$  in such a way that when starting with  $(n_1, \dots, n_\alpha) \in \mathbb{N}_0^\alpha$  in registers 1 to  $\alpha$ ,  $M$  has computed  $f(n_1, \dots, n_\alpha) = (r_1, \dots, r_\beta)$  if it halts in the final label  $h$  with registers  $\alpha + 3$  to  $\alpha + 2 + \beta$  containing  $r_1$  to  $r_\beta$ , and all other registers being empty; if the final label cannot be reached,  $f(n_1, \dots, n_\alpha)$  remains undefined. The registers  $\alpha + 3$  to  $\alpha + 2 + \beta$  are never decremented.*

The following two corollaries are immediate consequences of the preceding proposition (by taking  $\alpha = 0$  and  $\beta = 0$ , respectively).

**Corollary 2.** For any recursively enumerable set  $L \subseteq \mathbf{N}_0^\beta$  of vectors of non-negative integers there exists a non-deterministic  $(\beta + 2)$ -register machine  $M$  generating  $L$  in such a way that, when starting with all registers 1 to  $\beta + 2$  being empty,  $M$  non-deterministically computes and halts with  $n_i$  in registers  $i + 2$ ,  $1 \leq i \leq \beta$ , and registers 1 and 2 being empty if and only if  $(n_1, \dots, n_\beta) \in L$ . The registers 3 to  $\beta + 2$  are never decremented.

**Corollary 3.** For any recursively enumerable set  $L \subseteq \mathbf{N}_0^\alpha$  of vectors of non-negative integers there exists a deterministic  $(\alpha + 2)$ -register machine  $M$  accepting  $L$  in such a way that  $M$  halts with all registers being empty if and only if  $M$  starts with some  $(n_1, \dots, n_\alpha) \in L$  in registers 1 to  $\alpha$  and the registers  $\alpha + 1$  to  $\alpha + 2$  being empty.

## 2.2. The standard model of P systems and variants

The standard type of membrane systems (P systems) has been studied in many papers and several monographs; we refer to [1,4, 13–15] for details, motivation, and examples. In the definition of the P system below we omit some ingredients (like priority relations on the rules) not needed in the following:

A P system (of degree  $d$ ,  $d \geq 1$ ) is a construct

$$\Pi = (V, C, \mu, w_1, \dots, w_d, R_1, \dots, R_d, i_o),$$

where

- (i)  $V$  is an alphabet; its elements are called *objects*;
- (ii)  $C \subseteq V$  is a set of *catalysts*;
- (iii)  $\mu$  is a *membrane structure* consisting of  $d$  membranes (usually labelled with  $i$  and represented by corresponding brackets  $[_i$  and  $]_i$ ,  $1 \leq i \leq d$ );
- (iv)  $w_i$ ,  $1 \leq i \leq d$ , are strings over  $V$  associated with the regions  $1, 2, \dots, d$  of  $\mu$ ; they represent multisets of objects present in the regions of  $\mu$  (the multiplicity of a symbol in a region is given by the number of occurrences of this symbol in the string corresponding to that region);
- (v)  $R_i$ ,  $1 \leq i \leq d$ , are finite sets of *evolution rules* over  $V$  associated with the regions  $1, 2, \dots, d$  of  $\mu$ ; these evolution rules are of the forms  $a \rightarrow v$  or  $ca \rightarrow cv$ , where  $c$  is a catalyst,  $a$  is an object from  $V \setminus C$ , and  $v$  is a string from  $((V \setminus C) \times \{here, out, in\})^*$ ;
- (vi)  $i_o$  is a number between 1 and  $d$  and it specifies the *output* membrane of  $\Pi$ .

The membrane structure and the multisets represented by  $w_i$ ,  $1 \leq i \leq d$ , in  $\Pi$  constitute the *initial configuration* of the system. A transition between configurations is governed by an application of the evolution rules which is done in parallel: all objects, from all membranes, which *can be* the subject of local evolution rules *have to* evolve simultaneously.

The application of a rule  $u \rightarrow v$  in a region containing a multiset  $M$  results in subtracting from  $M$  the multiset identified by  $u$ , and then in adding the multiset identified by  $v$ . The objects can eventually be transported through membranes due to the targets *in* and *out* (we usually omit the target *here*). We refer to [1] and [15] for further details and examples. According to [10], the P system  $\Pi$  is called *catalytic*, if every evolution rule involves a catalyst.

The system continues parallel steps until there remain no applicable rules in any region of  $\Pi$ ; then the system halts. We consider the number of objects from  $V$  contained in the

output membrane  $i_o$  at the moment when the system halts as the *result* of the underlying computation of  $\Pi$ . The set of results of all computations possible in  $\Pi$  is denoted by  $N(\Pi)$ . The class of all sets of  $\beta$ -vectors  $(y_1, \dots, y_\beta)$  of non-negative integers computable by P systems (as the numbers of  $\beta$  different symbols to be found in the output membrane  $i_o$  at the end of halting computations) of the above type with  $d$  membranes and the set of catalysts containing at most  $m$  elements is denoted by  $N_0^\beta OP_{gen}(d, cat_m, halt)$ .

We may relax the condition that the output membrane has to be an elementary membrane where all the elements found there at the end of a halting computation count for the result of this computation; instead, we may specify a set of terminal objects  $\Sigma$  and only count the number of the  $\beta$  different symbols of  $\Sigma \subseteq V$  to be found in any specified output membrane at the end of halting computations; in this way, we obtain *extended P systems* of the form

$$\Pi = (V, \Sigma, C, \mu, w_1, \dots, w_d, R_1, \dots, R_d, i_o)$$

and the class  $N_0^\beta EP_{gen}(d, cat_m, halt)$ .

In addition to these generating membrane systems we may also consider accepting P systems where the multiset to be analyzed is put into region  $i_o$  together with  $w_{i_o}$  and accepted by a halting computation. The classes of all sets of  $\alpha$ -vectors  $(y_1, \dots, y_\alpha)$  of non-negative integers accepted in that way by halting computations in P systems of these types with  $d$  membranes and the set of catalysts containing at most  $m$  elements are denoted by  $N_0^\alpha WP_{acc}(d, cat_m, halt)$ ,  $W \in \{O, E\}$ .

In [2] accepting P systems were introduced as P automata using final states as accepting conditions, i.e., instead of the halting condition an input is accepted if the P system reaches a configuration where the contents of (specified) membranes coincides with the multisets given by a final state. In more detail, for a P system as defined above a final state over  $V$  is of the form  $(f_1, \dots, f_d)$  where each  $f_i$ ,  $1 \leq i \leq d$  is either a final multiset over  $V$  or (a special symbol denoted by)  $A$ ; then the P system accepts its input (given in  $i_o$ ) by this final state if during the computation a configuration is reached such that the contents of every membrane  $i$  with  $f_i \neq A$  coincides with  $f_i$ . The special symbol  $A$  indicates that we do not care about the contents of membrane  $i$  if  $f_i = A$ . Hence, a P system with final states is a construct of the form

$$\Pi = (V, C, \mu, w_1, \dots, w_d, R_1, \dots, R_d, i_o, F),$$

where  $V, C, \mu, w_1, \dots, w_d, R_1, \dots, R_d, i_o$  are defined as above and  $F$  is a finite set of final states over  $V$ . The class of all sets of  $\alpha$ -vectors  $(y_1, \dots, y_\alpha)$  of non-negative integers accepted in P systems with  $d$  membranes and the set of catalysts containing at most  $m$  elements by computations reaching a final state is denoted by  $N_0^\alpha OP_{acc}(d, cat_m, final\ state)$ .

Yet the idea of final states can also be carried over to generating P systems, i.e., a P system with final states as above can be used as a generative device, too; instead of considering the contents of the output membrane  $i_o$  in halting computations, we consider the contents of the output membrane  $i_o$  in computations having reached a final state  $(f_1, \dots, f_d)$  (obviously, in general we must have  $f_{i_o} = A$ ). Then the class of all sets of  $\beta$ -vectors  $(y_1, \dots, y_\beta)$  of non-negative integers generated in P systems with  $d$  membranes and the set of catalysts containing at most  $m$  elements by computations having reached a final state is denoted by  $N_0^\beta OP_{gen}(d, cat_m, final\ state)$ .

Considering *extended P systems with final states* of the form

$$\Pi = (V, \Sigma, C, \mu, w_1, \dots, w_d, R_1, \dots, R_d, i_0, F),$$

where again we only take into account the terminal symbols in the specified membrane  $i_0$ , we obtain the corresponding classes  $N_0^\gamma EP_X(d, cat_m, final\ state)$ ,  $X \in \{gen, acc\}$ .

If in the variants of P systems defined above only catalytic rules are used, we add the superscript *cat* thus obtaining the classes  $N_0^\gamma WP_X^{cat}(d, cat_m, Y)$ ,  $W \in \{O, E\}$ ,  $X \in \{gen, acc\}$ ,  $Y \in \{halt, final\ state\}$ .

### 3. Universality results

In order to prove the main results of this paper we elaborate a more general result using Proposition 1 that any partial recursive function  $f : \mathbf{N}_0^\alpha \rightarrow \mathbf{N}_0^\beta$  can be computed by a P system (halting or with final states) with only two membranes and with only  $\alpha + 2$  catalysts.

#### 3.1. P systems for partial recursive functions

Consider a register machine  $M$  with  $m$  registers,  $m \geq 1$ , and let  $P$  be the program for  $M$  with  $n$  instructions  $i_1, i_2, \dots, i_n$  computing  $f$  such that the last  $\beta$  registers are never decremented. We now construct a P system with  $m = m' - \beta$  catalysts simulating  $M$ . Informally, each register  $a$  is represented by objects  $o_a$  playing the rôles of counter elements. The value of register  $a$  at each moment corresponds to the number of symbols  $o_a$  in the system. There are also special objects  $p_j$ ,  $1 \leq j \leq n$ , which play the rôle of program labels; their marked variants guide the simulation of the instruction labelled by  $p_j$  within the P system.

The presence of the marked variants  $p_j^{(h,1)}$ ,  $1 \leq h \leq m$  of the object  $p_j$ —for each catalyst there has to be such a marked variant to keep it busy—starts the sequence of operations corresponding to the instruction  $j$ . For each of the  $m$  registers not representing an output value (where according to the result stated in Proposition 1 conditional decrementing may be necessary), in contrast to the proofs given in [18] and then in [8] we now need only one catalyst because we use the concept of “paired catalysts”: together with the catalyst  $c_a$  associated with register  $a$  we also associate (“pair”) another catalyst (we shall take  $c_{a \ominus_m 1}$ ) which together with  $c_a$  will do the correct simulation of an instruction  $j : (S(a), k, l) \in P$  in four steps; the remaining catalysts  $c_{a \ominus_m h}$  with  $2 \leq h < m$  are occupied by the marked variants of  $p_j$ ,  $p_j^{(h,i)}$ ,  $1 \leq i \leq 4$ , during these four steps, and the  $p_j^{(h,4)}$  are eliminated in the fourth step, before in the next step the new multiset  $p_k^{(1,1)} \dots p_k^{(m,1)}$  or  $p_l^{(1,1)} \dots p_l^{(m,1)}$  of (marked) program labels appears. The simulation of an instruction  $j : (A(a), k, k) \in P$  needs only one step. Finally, if the multiset  $p_n^{(1,1)} \dots p_n^{(m,1)}$  representing the final label  $n$  appears, these objects are also eliminated in one step, whereafter the computation halts if and only if it has been successful, i.e., no trap symbol # is present (after having been generated during the simulation of some subtract-instruction).

**Theorem 4.** For each partial recursive function  $f : \mathbb{N}_0^\alpha \rightarrow \mathbb{N}_0^\beta$ , there is a P system  $\Pi = (V, C, [1[2]2]_1, w, \lambda, R, \emptyset, 2)$  with  $\alpha + 2$  catalysts and with the objects  $o_a \in V$  satisfying the following conditions: For any arbitrary  $(x_1, \dots, x_\alpha) \in \mathbb{N}_0^\alpha$ , denote

$$\Pi_{(x_1, \dots, x_\alpha)} = (V, C, [1[2]2]_1, wo_1^{x_1} \cdots o_\alpha^{x_\alpha}, \lambda, R, \emptyset, 2).$$

The system  $\Pi_{(x_1, \dots, x_\alpha)}$  can halt if and only if  $f(x_1, \dots, x_\alpha)$  is defined and if it halts, then in the skin membrane only the catalysts remain and in the output membrane 2 only terminal symbols  $o_{\alpha+3}$  to  $o_{\alpha+2+\beta}$  appear in such a way that  $N(\Pi_{(x_1, \dots, x_\alpha)}) = \{f(x_1, \dots, x_\alpha)\}$ .

**Proof.** Consider a (deterministic) register machine  $M$  as defined above with  $m'$  registers, the last  $\beta$  registers being special output registers which are never decremented. (From the result stated in Proposition 1 we know that  $m' = \alpha + 2 + \beta$  is sufficient.) Now let  $m = m' - \beta$  and let  $P$  be a program which computes the function  $f$  such that the initial instruction has the label 1 and the halting instruction has the label  $n$ . The input values  $x_1, \dots, x_\alpha$  are expected to be in the first  $\alpha$  registers and the output values from  $f(x_1, \dots, x_\alpha)$  are expected to be in registers  $m + 1$  to  $m'$ . Moreover, at the beginning of a computation all the registers except possibly the registers 1 to  $\alpha$  contain zero.

We construct the P system

$$\Pi = (V, \{c_i \mid 1 \leq i \leq m\}, [1[2]2]_1, c_1 \dots c_m p_1^{(1,1)} \cdots p_1^{(m,1)}, \lambda, R, \emptyset, 2)$$

with

$$\begin{aligned} V = \{ \# \} \cup \{ c_i, c'_i, c''_i \mid 1 \leq i \leq m \} \cup \{ o_k \mid 1 \leq k \leq m' \} \cup \\ \{ p_n^{(h,1)} \mid 1 \leq h \leq m \} \cup \{ p_j^{(h,1)} \mid 1 \leq h \leq m, j : (A(a), k, k) \in P \} \cup \\ \{ p_j^{(h,1)} \mid 1 \leq h \leq m, j : (S(a), k, l) \in P \} \cup \\ \{ p_j^{(h,l)} \mid 2 \leq h < m, 1 \leq l \leq 4, j : (S(a), k, l) \in P \} \cup \\ \{ p'_j, p''_j, \bar{p}_j, \bar{p}'_j, \bar{p}''_j, \hat{p}_j, \hat{p}'_j, \hat{p}''_j \mid j : (S(a), k, l) \in P \} \end{aligned}$$

and

$$\begin{aligned} R = \{ x \rightarrow \# \mid x \in V \setminus (C \cup \{ o_k \mid 1 \leq k \leq m' \} \\ \cup \{ \bar{p}'_j, \hat{p}'_j \mid j : (S(a), k, l) \in P \}) \} \\ \cup \{ c_{m \ominus_m h} p_n^{(h,1)} \rightarrow c_{m \ominus_m h} \mid 1 \leq h \leq m \} \\ \cup \{ c_{m \ominus_m h} p_j^{(h,1)} \rightarrow c_{m \ominus_m h} \mid 1 \leq h < m, 1 \leq a \leq m' \} \end{aligned}$$

$$\begin{aligned}
& j : (A(a), k, k) \in P \} \\
& \cup \left\{ c_m p_j^{(m,1)} \rightarrow c_m p_k^{(1,1)} \cdots p_k^{(m,1)} o_a \mid 1 \leq a \leq m, j : (A(a), k, k) \in P \right\} \\
& \cup \left\{ c_m p_j^{(m,1)} \rightarrow c_m p_k^{(1,1)} \cdots p_k^{(m,1)} (o_a, in) \mid m < a \leq m', \right. \\
& j : (A(a), k, k) \in P \} \\
& \cup \left\{ c_{a \ominus_m h} p_j^{(h,l)} \rightarrow c_{a \ominus_m h} p_j^{(h,l+1)} \mid 2 \leq h < m, 1 \leq a \leq m, \right. \\
& 1 \leq l \leq 3, j : (S(a), k, l) \in P \} \\
& \cup \left\{ c_{a \ominus_m h} p_j^{(h,4)} \rightarrow c_{a \ominus_m h} \mid 2 \leq h < m, 1 \leq a \leq m, j : (S(a), k, l) \in P \right\} \\
& \cup \left\{ c_a p_j^{(m,1)} \rightarrow c_a \hat{p}_j \hat{p}'_j, c_a p_j^{(m,1)} \rightarrow c_a \bar{p}_j \bar{p}'_j \bar{p}''_j, \right. \\
& c_a o_a \rightarrow c_a c'_a, c_a c'_a \rightarrow c_a c''_a, \\
& c_a \bar{p}_j \rightarrow c_a, c_a \ominus_m 1 c''_a \rightarrow c_a \ominus_m 1, c_a \hat{p}_j \rightarrow c_a \#, c_a \ominus_m 1 \hat{p}'_j \rightarrow c_a \ominus_m 1 \hat{p}''_j, \\
& c_a \ominus_m 1 \bar{p}''_j \rightarrow c_a \ominus_m 1 p''_j, c_a \ominus_m 1 p''_j \rightarrow c_a \ominus_m 1 p'_j, \\
& c_a \hat{p}''_j \rightarrow c_a p_k^{(1,1)} \cdots p_k^{(m,1)} \\
& c_a p'_j \rightarrow c_a p_l^{(1,1)} \cdots p_l^{(m,1)} \mid 1 \leq a \leq m, j : (S(a), k, l) \in P \} \\
& \cup \left\{ c_a \ominus_m 1 y \rightarrow c_a \ominus_m 1 \mid y \in \left\{ p_j^{(1,1)}, \hat{p}_j, \bar{p}'_j \right\}, \right. \\
& 1 \leq a \leq m, j : (S(a), k, l) \in P \}. \quad \square
\end{aligned}$$

Then for an arbitrary  $(x_1, \dots, x_\alpha) \in \mathbb{N}_0^\alpha$  the axiom of the corresponding system  $\Pi_{(x_1, \dots, x_\alpha)}$  is  $c_1 \dots c_m p_1^{(1,1)} \cdots p_1^{(m,1)} o_1^{x_1} \dots o_\alpha^{x_\alpha}$ . The contents of register  $a$ ,  $1 \leq a \leq m$  is represented by the sum of the number of symbols  $o_a$  and conditional decrementing actions on this register are guarded by the pair of catalysts  $c_a$  and  $c_{a \ominus_m 1}$ . The set of rules  $R$  depends on the instructions of  $P$ ; the halting instruction as well as each add-instruction is simulated in one step, whereas each subtract-instruction is simulated in four steps; in more detail, the simulation works as follows:

- (1) Every simulation of a rule starts with the program labels  $p_1^{(1,1)}, \dots, p_1^{(m,1)}$ . The halting instruction eliminates the final labels  $p_n^{(1,1)}, \dots, p_n^{(m,1)}$  by using the rules  $c_m \ominus_m h p_n^{(h,1)} \rightarrow c_m \ominus_m h$ ,  $1 \leq h \leq m$ ; if the computation has been successful, then only the catalysts remain in the skin membrane, whereas the result of the computation, i.e., the number of symbols  $o_{m+1}$  to  $o_{m+\beta}$ , can be found in the output membrane 2.
- (2) Each add-instruction  $j : (A(a), k, k) \in P$ ,  $1 \leq a \leq m$  ( $m < a \leq m'$ , respectively) is simulated in one step by using the catalytic rules  $c_m \ominus_m h p_j^{(h,1)} \rightarrow c_m \ominus_m h$ ,  $1 \leq h < m$  as well as  $c_m p_j^{(m,1)} \rightarrow c_m p_k^{(1,1)} \cdots p_k^{(m,1)} o_a$  (and  $c_m p_j^{(m,1)} \rightarrow c_m p_k^{(1,1)} \cdots p_k^{(m,1)} (o_a, in)$ ),



respectively, i.e., if  $a$  is the index of a register representing a component of the result vector of the computation, then the symbol  $o_a$  is immediately moved to the output membrane 2). Observe that by definition  $a \ominus_m m = a$  for all  $a$  with  $1 \leq a \leq m$ .

- (3) Each subtract-instruction  $j : (S(a), k, l) \in P$  is simulated in four steps. We have to distinguish between two cases depending on the contents of register  $a$ ; in both cases the catalysts  $c_{a \ominus_m h}$ ,  $2 \leq h < m$ , are busy with the objects  $p_j^{(h,i)}$ ,  $1 \leq i \leq 4$ ; the objects  $p_j^{(h,4)}$  finally are eliminated in the fourth step. The main part of the simulation is accomplished by the catalyst  $c_a$  and its “paired companion”  $c_{a \ominus_m 1}$ :
- (a) We non-deterministically assume that the contents of register  $a$  is not empty; we start with the rules  $c_a p_j^{(m,1)} \rightarrow c_a \hat{p}_j \hat{p}'_j$  and  $c_{a \ominus_m 1} p_j^{(1,1)} \rightarrow c_{a \ominus_m 1}$ . In the second step, the number of symbols  $o_a$  is decremented by using the rule  $c_a o_a \rightarrow c_a c'_a$ ; if in contrast to our choice, no such symbol  $o_a$  is present (i.e., the contents of the register represented by the number of symbols  $o_a$  is empty), then by the enforced application of the rule  $c_a \hat{p}'_j \rightarrow c_a \#$  the trap symbol  $\#$  is introduced, which causes a non-halting computation due to the rule  $\# \rightarrow \#$ . If  $\hat{p}'_j$  could wait until being used in the third step by the rule  $c_{a \ominus_m 1} \hat{p}'_j \rightarrow c_{a \ominus_m 1} \hat{p}''_j$ , then the simulation will be successful: In the second step,  $c_{a \ominus_m 1}$  is used in the rule  $c_{a \ominus_m 1} \hat{p}_j \rightarrow c_{a \ominus_m 1}$ , and in the third step  $c_a$  is used in the rule  $c_a c'_a \rightarrow c_a c''_a$ . We finish with the application of the rules  $c_a \hat{p}''_j \rightarrow c_a p_k^{(1,1)} \cdots p_k^{(m,1)}$  and  $c_{a \ominus_m 1} c''_a \rightarrow c_{a \ominus_m 1}$ .
- (b) For the other case, we non-deterministically assume that the contents of register  $a$  is empty; we start with the two rules  $c_a p_j^{(m,1)} \rightarrow c_a \bar{p}_j \bar{p}'_j \bar{p}''_j$  and  $c_{a \ominus_m 1} p_j^{(1,1)} \rightarrow c_{a \ominus_m 1}$ . In the second step, we are forced to use the two rules  $c_a \bar{p}_j \rightarrow c_a$  and  $c_{a \ominus_m 1} \bar{p}'_j \rightarrow c_{a \ominus_m 1} p'_j$  in order not to introduce the trap symbol  $\#$ . In the third step, we only use  $c_{a \ominus_m 1} p''_j \rightarrow c_{a \ominus_m 1} p'_j$  and finish with applying the two rules  $c_a p'_j \rightarrow c_a p_l^{(1,1)} \cdots p_l^{(m,1)}$  and  $c_{a \ominus_m 1} \bar{p}''_j \rightarrow c_{a \ominus_m 1}$  in the fourth step. In the third step the catalyst  $c_a$  is not used if our non-deterministic choice has been correct, i.e., if there is no symbol  $o_a$  present in the skin membrane; otherwise, the rule  $c_a o_a \rightarrow c_a c'_a$  has to be applied in the third step, but in this case both  $c'_a$  and  $p'_j$  would need the catalyst  $c_a$  in the fourth step of the simulation in order not to be sent to the trap symbol  $\#$ .

Any other behavior of the system as the one described above for the correct simulation of the instructions of  $P$  by the rules in  $R$  leads to the appearance of the trap object  $\#$  within the system, hence, the system never halts.

It follows from the description given above that after each simulation of an instruction the number of objects  $o_a$  equals the contents of register  $a$ ,  $1 \leq a \leq m'$ . Hence, after having simulated the instruction *Halt* and halting the system, the number of symbols  $o_{m+1}$  to  $o_{m+\beta}$  in the output membrane 2 equals the output of the program  $P$ . The only other objects remaining within the system are the  $m$  catalysts in the skin membrane; according to the result about register machines stated in Proposition 1,  $m = \alpha + 2$  and therefore  $\alpha + 2$  catalysts are enough.  $\square$

For  $P$  systems with final states, we can immediately take over the construction given in the preceding proof:

**Corollary 5.** For each partial recursive function  $f : \mathbf{N}_0^\alpha \rightarrow \mathbf{N}_0^\beta$  there is a P system with final states  $\Pi^F = (V, C, [1[2]2]_1, w, \lambda, R, \emptyset, 2, F)$  with  $\alpha+2$  catalysts and with the objects  $o_a \in V$  satisfying the following conditions: For any arbitrary  $(x_1, \dots, x_\alpha) \in \mathbf{N}_0^\alpha$ , denote

$$\Pi_{(x_1, \dots, x_\alpha)}^F = (V, C, [1[2]2]_1, wo_1^{x_1} \cdots o_\alpha^{x_\alpha}, \lambda, R, \emptyset, 2, F).$$

The system  $\Pi_{(x_1, \dots, x_\alpha)}^F$  reaches a final state if and only if  $f(x_1, \dots, x_\alpha)$  is defined, and in the final state the output membrane 2 contains only terminal symbols  $o_{m+1}$  to  $o_{m+\beta}$  in such a way that  $N(\Pi_{(x_1, \dots, x_\alpha)}^F) = \{f(x_1, \dots, x_\alpha)\}$ .

**Proof.** The only difference to the P system constructed in Theorem 4 is that we have to define the final state for successful computations, which simply is the contents of the skin membrane at the end of a halting computation, i.e.,  $c_1 \dots c_m$ . Hence, taking  $F = \{(c_1 \dots c_m, A)\}$  we obtain the P system with final states  $\Pi'$  is  $(\Pi, \{(c_1 \dots c_m, A)\})$ , where  $\Pi$  is the P system constructed in the proof of Theorem 4.  $\square$

In catalytic systems we only need one more catalyst for the rules handling the trap symbol #:

**Corollary 6.** For each partial recursive function  $f : \mathbf{N}_0^\alpha \rightarrow \mathbf{N}_0^\beta$  there is  
(1) a halting catalytic P system

$$\Pi^{\text{cH}} = (V \cup \{c_0\}, C \cup \{c_0\}, [1[2]2]_1, w, \lambda, R_C, \emptyset, 2),$$

(2) a catalytic P system with final states

$$\Pi^{\text{cF}} = (V \cup \{c_0\}, C \cup \{c_0\}, [1[2]2]_1, w, \lambda, R_C, \emptyset, 2, F),$$

respectively, with  $\alpha + 3$  catalysts and with the objects  $o_a \in V$  satisfying the following conditions: For any arbitrary  $(x_1, \dots, x_\alpha) \in \mathbf{N}_0^\alpha$ , denote

$$(1) \Pi_{(x_1, \dots, x_\alpha)}^{\text{cH}} = (V \cup \{c_0\}, C \cup \{c_0\}, [1[2]2]_1, wo_1^{x_1} \cdots o_\alpha^{x_\alpha}, \lambda, R_C, \emptyset, 2)$$

and

$$(2) \Pi_{(x_1, \dots, x_\alpha)}^{\text{cF}} = (V \cup \{c_0\}, C \cup \{c_0\}, [1[2]2]_1, wo_1^{x_1} \cdots o_\alpha^{x_\alpha}, \lambda, R_C, \emptyset, 2, F),$$

respectively. The system

$$(1) \Pi_{(x_1, \dots, x_\alpha)}^{\text{cH}} \text{ halts,}$$

$$(2) \Pi_{(x_1, \dots, x_\alpha)}^{\text{cF}} \text{ reaches a final state,}$$

respectively, if and only if  $f(x_1, \dots, x_\alpha)$  is defined and

(1) in the halting computation or

(2) in the final state,

respectively, in the skin membrane only the catalysts remain and the output membrane 2 contains only terminal symbols  $o_{m+1}$  to  $o_{m+\beta}$  in such a way that

$$N(\Pi_{(x_1, \dots, x_\alpha)}^{\text{cH}}) = N(\Pi_{(x_1, \dots, x_\alpha)}^{\text{cF}}) = \{f(x_1, \dots, x_\alpha)\}.$$

**Proof.** The rules in  $R_C$  are obtained from the rules in  $R$  constructed in the proof of Theorem 4 by just replacing the rules in

$$\left\{x \rightarrow \# \mid x \in V \setminus \left( C \cup \left\{ \bar{p}'_j, \hat{p}'_j \mid j : (S(a), k, l) \in P \right\} \cup \{o_k \mid 1 \leq k \leq m'\} \right) \right\}$$

with the rules in

$$\left\{ c_0 x \rightarrow c_0 \# \mid x \in V \setminus \left( C \cup \left\{ \bar{p}'_j, \hat{p}'_j \mid j : (S(a), k, l) \in P \right\} \cup \{o_k \mid 1 \leq k \leq m'\} \right) \right\}$$

using the additional catalyst  $c_0$ .  $\square$

In extended P systems we do not need the additional output membrane, i.e.,  $\alpha + 2$  catalysts ( $\alpha + 3$  catalysts in catalytic systems) in the skin membrane are sufficient.

**Corollary 7.** For each partial recursive function  $f : \mathbf{N}_0^\alpha \rightarrow \mathbf{N}_0^\beta$  there is

(1) an extended (halting) P system with  $\alpha + 2$  catalysts

$$\Pi^E = (V, \Sigma, C, [1]_1, w, R_E),$$

(2) an extended P system with final states with  $\alpha + 2$  catalysts

$$\Pi^{EF} = (V, \Sigma, C, [1]_1, w, R_E, \{c_1 \dots c_m\}),$$

(3) a catalytic extended (halting) P system with  $\alpha + 3$  catalysts

$$\Pi^{cE} = (V \cup \{c_0\}, \Sigma, C \cup \{c_0\}, [1]_1, w, R_E),$$

(4) a catalytic extended P system with final states with  $\alpha + 3$  catalysts

$$\Pi^{cEF} = (V \cup \{c_0\}, \Sigma, C \cup \{c_0\}, [1]_1, w, R_E, \{c_0 c_1 \dots c_m\}),$$

respectively, with  $\Sigma = \{o_k \mid m + 1 \leq k \leq m + \beta\}$  and the objects  $o_a \in V$  satisfying the following conditions: for any arbitrary  $(x_1, \dots, x_\alpha) \in \mathbf{N}_0^\alpha$ , denote the corresponding P system by

$$(1) \Pi_{(x_1, \dots, x_\alpha)}^E = (V, \Sigma, C, [1]_1, w o_1^{x_1} \dots o_\alpha^{x_\alpha}, R_E),$$

$$(2) \Pi_{(x_1, \dots, x_\alpha)}^{EF} = (V, \Sigma, C, [1]_1, w o_1^{x_1} \dots o_\alpha^{x_\alpha}, R_E, \{c_1 \dots c_m\}),$$

$$(3) \Pi_{(x_1, \dots, x_\alpha)}^{cE} = (V \cup \{c_0\}, \Sigma, C \cup \{c_0\}, [1]_1, w o_1^{x_1} \dots o_\alpha^{x_\alpha}, R_E),$$

$$(4) \Pi_{(x_1, \dots, x_\alpha)}^{cEF} = (V \cup \{c_0\}, \Sigma, C \cup \{c_0\}, [1]_1, w o_1^{x_1} \dots o_\alpha^{x_\alpha}, R_E, \{c_0 c_1 \dots c_m\}),$$

respectively.

$$(1) \Pi_{(x_1, \dots, x_\alpha)}^E \text{ halts,}$$

$$(2) \Pi_{(x_1, \dots, x_\alpha)}^{cE} \text{ halts,}$$

- (3)  $\Pi_{(x_1, \dots, x_\alpha)}^{\text{EF}}$  reaches the final state  $c_1 \dots c_m$ ,
- (4)  $\Pi_{(x_1, \dots, x_\alpha)}^{\text{cEF}}$  reaches the final state  $c_0 c_1 \dots c_m$ ,  
 respectively, if and only if  $f(x_1, \dots, x_\alpha)$  is defined, and after halting the computation or after having reached the final state, respectively, in the skin membrane only the catalysts and the terminal symbols  $o_{m+1}$  to  $o_{m+\beta}$  remain in such a way that

$$N\left(\Pi_{(x_1, \dots, x_\alpha)}^G\right) = \{f(x_1, \dots, x_\alpha)\} \text{ for } G \in \{E, \text{EF}, \text{cE}, \text{cEF}\}.$$

**Proof.** The rules in  $R_E$  are obtained from the rules in  $R$  constructed in the proof of Theorem 4 (and Corollary 5) as well as from the rules in  $R_C$  constructed in the proof of Corollary 6 by just replacing each occurrence of  $(o_a, \text{in})$  by  $o_a$  (which in fact means  $(o_a, \text{here})$ ).  $\square$

### 3.2. Generating P systems

The following corollaries are immediate consequences of Theorem 4 as well as Corollaries 5–7 by taking  $\alpha = 0$  and simulating non-deterministic register machines:

**Corollary 8.**  $N_0^\beta OP_{gen}(d, \text{cat}_2, \text{halt}) = N_0^\beta RE$  for every  $d \geq 2$ .

**Proof.** We only prove the inclusion  $N_0^\beta RE \subseteq N_0^\beta OP_{gen}(2, \text{cat}_2, \text{halt})$ . In the same way as in the proof of Theorem 4 the P system  $\Pi$  was constructed in order to simulate the (deterministic) register machine from Proposition 1, we now construct a P system  $\Pi'$  which simulates the non-deterministic register machine from Corollary 2 and in that way non-deterministically generates a representation of any vector from the given language  $L$  in  $N_0^\beta RE$  by the corresponding numbers of symbols  $o_3$  to  $o_{2+\beta}$ . Hence, we define

$$\Pi' = (V, C, [1[2]2]_1, w, \lambda, R', \emptyset, 2),$$

where  $R'$  is constructed in a similar way as  $R$  in the proof of Theorem 4, except that now in the non-deterministic case we have add-instructions of the form  $j : (A(a), k, l)$  for some  $a, k, l$  with  $a \in \{1, 2\}$  and  $1 \leq k, l \leq n$ ; for their simulation we now not only need the rule  $c_m p_j^{(m,1)} \rightarrow c_m p_k^{(1,1)} \dots p_k^{(m,1)} o_a$ , but also  $c_m p_j^{(m,1)} \rightarrow c_m p_l^{(1,1)} \dots p_l^{(m,1)} o_a$  in  $R'$ . Obviously,  $N(\Pi') = L$ . By the given construction, we only need 2 catalysts.  $\square$

As the result is interesting of its own, we completely specify  $\Pi'$ ; as only two catalysts are needed, we can use a less complex notation, because these two catalysts form the only pair used in the simulation of any subtract-instruction, i.e., we do not need the objects  $p_j^{(h,i)}$ ,  $2 \leq h < m$ ,  $1 \leq i \leq 4$ , for the subtract-instructions  $j : (S(a), k, l) \in P$ . Using  $p_j$  and  $\tilde{p}_j$  instead of  $p_j^{(m,1)}$  and  $p_j^{(1,1)}$ , we obtain the following P system  $\Pi'$ :

$$\Pi' = (V', \{c_1, c_2\}, [1[2]2]_1, c_1 c_2 p_1 \tilde{p}_1, \lambda, R', \emptyset, 2)$$

with

$$\begin{aligned} V' = & \{\#\} \cup \{c_1, c'_1, c''_1, c_2, c'_2, c''_2\} \cup \{o_k \mid 1 \leq k \leq m'\} \\ & \cup \left\{ p_j, \tilde{p}_j, p'_j, p''_j, \bar{p}_j, \bar{p}'_j, \bar{p}''_j, \hat{p}_j, \hat{p}'_j, \hat{p}''_j \mid j : (S(a), k, l) \in P \right\} \\ & \cup \left\{ p_j, \tilde{p}_j \mid j : (A(a), k, l) \in P \right\} \end{aligned}$$

and

$$\begin{aligned} R' = & \left\{ x \rightarrow \# \mid x \in \left\{ p_j, \tilde{p}_j, p'_j, p''_j, \bar{p}_j, \bar{p}'_j, \hat{p}_j, \hat{p}'_j \mid j : (S(a), k, l) \in P \right\} \right\} \\ & \cup \left\{ x \rightarrow \# \mid x \in \{c'_1, c''_1, c'_2, c''_2\} \right\} \cup \{\# \rightarrow \#\} \cup \{c_1 p_n \rightarrow c_1, c_2 \tilde{p}_n \rightarrow c_2\} \\ & \cup \left\{ c_1 \tilde{p}_j \rightarrow c_1 \mid j : (A(a), k, l) \in P \right\} \\ & \cup \left\{ c_2 p_j \rightarrow c_2 p_k \tilde{p}_k o_a, c_2 p_j \rightarrow c_2 p_l \tilde{p}_l o_a \mid a \in \{1, 2\}, \right. \\ & \quad \left. j : (A(a), k, l) \in P \right\} \\ & \cup \left\{ c_2 p_j \rightarrow c_2 p_k \tilde{p}_k (o_a, in) \mid 2 < a \leq m', j : (A(a), k, k) \in P \right\} \\ & \cup \left\{ c_a p_j \rightarrow c_a \hat{p}_j \hat{p}'_j, c_a p_j \rightarrow c_a \bar{p}_j \bar{p}'_j \bar{p}''_j, c_a o_a \rightarrow c_a c'_a, c_a c'_a \rightarrow c_a c''_a, \right. \\ & \quad c_{3-a} c''_a \rightarrow c_{3-a}, c_a \hat{p}'_j \rightarrow c_a \#, c_{3-a} \hat{p}'_j \rightarrow c_{3-a} \hat{p}''_j, c_a \hat{p}'_j \rightarrow c_a p_k \tilde{p}_k, \\ & \quad c_a \bar{p}_j \rightarrow c_a, c_{3-a} \bar{p}''_j \rightarrow c_{3-a} p''_j, c_{3-a} p''_j \rightarrow c_{3-a} p'_j, \\ & \quad \left. c_a p'_j \rightarrow c_a p_l \tilde{p}_l \mid a \in \{1, 2\}, j : (S(a), k, l) \in P \right\} \\ & \cup \left\{ c_{3-a} y \rightarrow c_{3-a} \mid y \in \left\{ \tilde{p}_j, \hat{p}_j, \bar{p}'_j \right\}, a \in \{1, 2\}, j : (S(a), k, l) \in P \right\}. \end{aligned}$$

The following table shows how a subtract-instruction  $j : (S(a), k, l) \in P$  is simulated depending on the contents of register  $a$ :

Simulation of the subtract-instruction $j : (S(a), k, l)$ if the contents of register $a$ is not empty		the contents of register $a$ is empty	
$c_a p_j \rightarrow c_a \hat{p}_j \hat{p}'_j$	$c_a p_j \rightarrow c_a \bar{p}_j \bar{p}'_j \bar{p}''_j$	$c_a p_j \rightarrow c_a \bar{p}_j \bar{p}'_j \bar{p}''_j$	$c_a p_j \rightarrow c_a \bar{p}_j \bar{p}'_j \bar{p}''_j$
$c_{3-a} \tilde{p}_j \rightarrow c_{3-a}$	$c_{3-a} \tilde{p}_j \rightarrow c_{3-a}$	$c_{3-a} \tilde{p}_j \rightarrow c_{3-a}$	$c_{3-a} \tilde{p}_j \rightarrow c_{3-a}$
$c_a o_a \rightarrow c_a c'_a$	$c_a \bar{p}_j \rightarrow c_a$	$c_a \bar{p}_j \rightarrow c_a$	$c_a \bar{p}_j \rightarrow c_a$
$c_{3-a} \hat{p}_j \rightarrow c_{3-a}$	$c_{3-a} \bar{p}''_j \rightarrow c_{3-a} p''_j$	$c_{3-a} \bar{p}''_j \rightarrow c_{3-a} p''_j$	$c_{3-a} \bar{p}''_j \rightarrow c_{3-a} p''_j$
$c_a c'_a \rightarrow c_a c''_a$	$c_{3-a} p''_j \rightarrow c_{3-a} p'_j$	$c_{3-a} p''_j \rightarrow c_{3-a} p'_j$	$c_{3-a} p''_j \rightarrow c_{3-a} p'_j$
$c_a \hat{p}''_j \rightarrow c_a p_k \tilde{p}_k$	$c_a p'_j \rightarrow c_a p_l \tilde{p}_l$	$c_a p'_j \rightarrow c_a p_l \tilde{p}_l$	$c_a p'_j \rightarrow c_a p_l \tilde{p}_l$
$c_{3-a} c''_a \rightarrow c_{3-a}$	$c_{3-a} \bar{p}'_j \rightarrow c_{3-a}$	$c_{3-a} \bar{p}'_j \rightarrow c_{3-a}$	$c_{3-a} \bar{p}'_j \rightarrow c_{3-a}$

We should like to mention that at any time  $c_a$  can be used in the rule  $c_a o_a \rightarrow c_a c'_a$ , but carried out at the wrong time, the application of this rule will immediately cause the introduction of the trap symbol # and therefore lead to a non-halting computation. Moreover, making the wrong choice when simulating a subtract-instruction also leads to the (enforced) introduction of the trap symbol and therefore to a non-halting computation.  $\square$

**Corollary 9.**  $N_0^\beta OP_{gen}(d, cat_2, final\ state) = N_0^\beta RE$  for every  $d \geq 2$ .

**Proof.** In the same way as in the proof of Corollary 5 the P system  $\Pi^F$  was constructed from the P system  $\Pi$  constructed in the proof of Theorem 4 we now can construct the P system with final states generating a set  $L \in N_0^\beta RE$  from the P system constructed in the proof of Corollary 8.  $\square$

Obviously, the results obtained so far are optimal with respect to the number of membranes in the P systems constructed in the proofs of Theorem 4 and Corollaries 5–9.

For catalytic P systems, in [10] it was proved that with one catalyst we cannot reach universal computational power; hence, only the case of two catalysts in catalytic P systems remains for a suitable characterization, because from Corollaries 8 and 9 we immediately infer the following results (in the same way as Corollary 6 was an obvious consequence of Theorem 4 and Corollary 5):

**Corollary 10.** For every  $d \geq 2$ , we have

$$N_0^\beta RE = N_0^\beta OP_{gen}^{cat}(d, cat_3, halt) = N_0^\beta OP_{gen}^{cat}(d, cat_3, final\ state).$$

The proofs of the following results immediately follow from preceding proofs, too (see Corollary 7):

**Corollary 11.** For every  $d \geq 1$ , we have

$$N_0^\beta RE = N_0^\beta EP_{gen}(d, cat_2, Y) = N_0^\beta EP_{gen}^{cat}(d, cat_3, Y)$$

for every  $Y \in \{halt, final\ state\}$ .

### 3.3. Accepting P systems/P automata

The following corollaries are immediate consequences of Theorem 4 as well as Corollaries 5–7 by taking  $\beta = 0$ . Although for P automata we now have the minimal number of only one membrane, the number of catalysts depends on the number  $\alpha$  of components of the vector of non-negative integers to be analyzed.

**Corollary 12.** For every  $d \geq 1$ , we have

$$N_0^\alpha RE = N_0^\alpha XP_{acc}(d, cat_{\alpha+2}, Y) = N_0^\alpha XP_{acc}^{cat}(d, cat_{\alpha+3}, Y)$$

for every  $X \in \{O, E\}$  and  $Y \in \{halt, final\ state\}$ .

**Proof.** We first prove the inclusion  $N_0^\alpha RE \subseteq N_0^\alpha OP_{acc}(1, cat_{\alpha+2}, halt)$ . In the same way as in the proof of Theorem 4 the P system there was constructed in order to simulate the (deterministic) register machine from Proposition 1, we now construct a P system which simulates the (deterministic) register machine from Corollary 3. As we have no output, we simply omit the output membrane; moreover, we have no rules sending an object into another membrane. The rest of the construction is exactly the same as in Theorem 4. For the remaining variants of accepting P systems and P automata we only refer to the proof ideas elaborated in the preceding proofs.  $\square$

For the simplest case of  $\alpha = 1$ , therefore the maximal number of catalysts needed for accepting languages from  $N_0^1 RE$  by P systems is 3 and by catalytic P systems is 4.

#### 4. Conclusion

The number of catalysts can be seen as a complexity measure for P systems with catalysts. Only the characterization of functions computed or sets generated/accepted by the variants of P systems considered in this paper having one catalyst less remains as an interesting open question for future research; yet we conjecture that for computationally universal P systems the results obtained in this paper are already optimal not only with respect to the number of membranes, but also with respect to the number of catalysts.

Even some more results can be found in [9]; in particular, there we also consider several variants of P systems with catalysts generating/accepting strings and show that only two catalysts (three catalysts for the catalytic variants) in only one membrane are already sufficient for obtaining universality. Again we conjecture that these results are already optimal even with respect to the number of catalysts.

In [11], the bounds for the number of catalysts and/or membranes were improved (with respect to the optimal results known before this paper) by introducing more powerful types of catalysts like so-called bi-stable catalysts and mobile catalysts. The authors showed that a P system can generate all recursively enumerable number sets using (a) five membranes, two catalysts and one bi-stable catalyst, (b) three membranes and one mobile catalyst, (c) two membranes and two mobile catalysts. From these results, case (a) has become obsolete by the results obtained in this paper, whereas case (b) may give a chance for improving this result for mobile catalysts. Using only one catalyst in several membranes is another interesting case to be investigated.

#### Acknowledgements

We gratefully acknowledge the most interesting discussions with Gheorghe Păun and other participants of the brainstorming week on P systems in Tarragona at the beginning of February 2003 as well as of the Fifth International Workshop Descriptive Complexity of Formal Systems in Budapest, the stays of the Austrian authors being supported by MolCoNet project IST-2001-32008. The research of Petr Sosík was partly supported by the Grant

Agency of Czech Republic, Grant No. 201/02/P079. The research of Lila Kari was supported by the Canada Research Chair Program and NSERC Discovery Grant.

## References

- [1] C.S. Calude, Gh. Păun, *Computing with Cells and Atoms*, Taylor & Francis, London, 2001.
- [2] E. Csuhaj-Varjú, Gy. Vaszil, P automata or purely communicating accepting P systems, in: Gh. Păun, G. Rozenberg, A. Salomaa, C. Zandron (Eds.), *Membrane Computing, Internat. Workshop, WMC-CdeA 2002, Curtea de Argeş, Romania, August 2002, Lecture Notes in Computer Science*, vol. 2597, Springer, Berlin, 2003, pp. 219–233.
- [3] J. Dassow, Gh. Păun, *Regulated Rewriting in Formal Language Theory*, Springer, Berlin, 1989.
- [4] J. Dassow, Gh. Păun, On the power of membrane computing, *J. Univ. Comput. Sci.* 5 (2) (1999) 33–49.
- [5] R. Freund, Gh. Păun, On the number of non-terminals in graph-controlled, programmed, and matrix grammars, in: M. Margenstern, Y. Rogozhin (Eds.), *Machines, Computations and Universality, 3rd Internat. Conf., MCU 2001, Lecture Notes in Computer Science*, vol. 2055, Springer, Berlin, 2001, pp. 214–225.
- [6] R. Freund, Gh. Păun, From regulated rewriting to computing with membranes: collapsing hierarchies, *Theoret. Comput. Sci.* 312 (2004) 143–188.
- [7] R. Freund, M. Oswald, GP systems with forbidding context, *Fund. Inform.* 49 (1–3) (2002) 81–102.
- [8] R. Freund, M. Oswald, P. Sosík, Reducing the number of catalysts needed in computationally universal systems without priorities, in: E. Csuhaj-Varjú, C. Kintala, D. Wotschke, Gy. Vaszil (Eds.), *Fifth Internat. Workshop Descriptive Complexity of Formal Systems, Budapest, Hungary, July 12–14, 2003, MTA SZTAKI, Budapest, 2003*, pp. 102–113.
- [9] R. Freund, L. Kari, M. Oswald, P. Sosík, Computationally universal P systems without priorities: two catalysts are sufficient, *Techn. Report*, Vienna University of Technology, 2003.
- [10] O.H. Ibarra, Z. Dang, O. Egecioglu, G. Saxena, Characterizations of catalytic membrane computing systems, in: B. Rován, P. Vojtás (Eds.), *28th Internat. Sympos. on Mathematical Foundations of Computer Science 2003, MFCS 2003, Bratislava, Slovakia, August 25–29, 2003, Lecture Notes in Computer Science*, vol. 2747, Springer, Berlin, 2003, pp. 480–489.
- [11] S.N., Krishna, A. Păun, Three universality results on P systems, in: M. Cavaliere, C. Martín-Vide, Gh. Păun (Eds.), *Brainstorming Week on Membrane Computing; Tech. Report No. 26, Tarragona, February 5–11, 2003, Rovira i Virgili University*, 2003, pp. 198–206.
- [12] M.L. Minsky, *Finite and Infinite Machines*, Prentice-Hall, Englewood Cliffs, NJ, 1967.
- [13] Gh. Păun, Computing with Membranes: an Introduction, *Bull. EATCS* 67 (1999) 139–152.
- [14] Gh. Păun, Computing with Membranes, *J. Comput. System Sci.* 61 (1) (2000) 108–143.
- [15] Gh. Păun, *Membrane Computing: An Introduction*, Springer, Berlin, 2002.
- [17] G. Rozenberg, A. Salomaa (Eds.), *Handbook of Formal Languages*, Springer, Berlin, 1997.
- [18] P. Sosík, The power of catalysts and priorities in membrane systems, *Grammars* 6 (1) (2003) 13–24.
- [19] P. Sosík, R. Freund, P Systems without priorities are computationally universal, in: Gh. Păun, G. Rozenberg, A. Salomaa, C. Zandron (Eds.), *Membrane Computing. Internat. Workshop, WMC-CdeA 2002, Curtea de Argeş, Romania, August 2002, Lecture Notes in Computer Science*, vol. 2597, Springer, Berlin, 2003, pp. 400–409.