Improving the Universality Results of Enzymatic Numerical P Systems

Cristian Ioan Vasile[1], Ana Brânduşa Pavel[1], and Ioan Dumitrache[1]

Department of Automatic Control and Systems Engineering
Politehnica University of Bucharest
Splaiul Independenţei 313, 060042 Bucharest, Romania
{cvasile, apavel, idumitrache}@ics.pub.ro

Summary. This paper provides the proof that Enzymatic Numerical P Sytems with deterministic, but parallel, execution model are universal, even when the production functions used are polynomials of degree 1. This extends previous known results and provides the optimal case in terms of polynomial degree.

1 Enzymatic Numerical P Systems

Numerical P Systems (NP Systems) are a type of P systems [8], inspired by the cell structure, in which numerical variables evolve inside the compartments by means of programs; a program (or rule) is composed of a production function and a repartition protocol. The variables have a given initial value and the production function is a multivariate polynomial. The value of the production function for the current values of the variables is distributed among variables in certain compartments according to a repartition protocol. Formal definition of NPS can be found in [7] where the authors introduce this type of P systems with possible applications in economics.

NP systems were designed both as deterministic and non-deterministic systems [7]. Non-deterministic NPS allow the existence of more rules per each membrane and the best rule is selected by an "oracle", while the deterministic NPS can have only one or no rule per each membrane. NP Systems were used as a naturally parallel and distributed modeling tool for the design of robot controllers [1], [5], [6]. Designing robot controllers requires deterministic mechanisms. Therefore, an extension of NPS, Enzymatic Numerical P systems (EN P Systems), in which enzyme-like variables allow the existence of more than one program (rule) in each membrane, while keeping the deterministic nature of the system, were introduced in [3]. Due to their properties, EN P Systems represent a more powerful modeling tool for robot behaviors than classical N P Systems [5], [6].

2 The power of Enzymatic Numerical P Systems

In [9] the authors prove and analyze the universality of EN P Systems. The main results in [7] and [9] regarding the power of NP Systems and ENP Systems are the following:

Theorem 1. $NRE = N^+P_8(poly^5(5), seq) = N^+P_7(poly^5(6), seq) = NP_7(poly^5(5), enz, seq) = NtP_*(poly^1(11), enz, oneP) = NP_{254}(poly^2(253), enz, allP, det)$.

The family of sets of numbers $N^+(\Pi)$ computed by NP Systems with at most m membranes, production functions which are polynomials of degree at most n, with integer coefficients, with at most r variables in each polynomial, is denoted by $N^+P_m(poly^n(r), seq)$, $m \geq 1, n \geq 0, r \geq 0$, where the fact that they work in the sequential mode (in each step, only one program is applied), is indicated by *seq*. If one of the parameters m, n, r is not bounded, then it is replaced by $*$. (Both in $N^+(\Pi)$ and in $N^+P_m(poly^n(r), seq)$, the superscript + indicates the fact that as the result of a computation we only consider positive natural numbers, zero excluded. If any value of x_{j_0,i_0} is accepted, then the superscript + is removed.) When tissue-like systems are used, we write $NtP_m(poly^n(r), \alpha, \beta)$.

In the next section, the authors present an improvement of the universality results of EN P Systems by reducing the number of membranes, the polynomial degree of the production functions and the number of variables of the production functions.

3 Improving the universality results

The main result proposed here is the following theorem about the power of EN P Systems. The theorem extends previous results about the benefits of adding the enzymatic mechanism in terms of the computational power of the model. It also provides the optimal result regarding the polynomial degree of the production functions, namely 1. The execution model considered in the theorem is a deterministic, but parallel one, in which all active rules are executed in parallel. Rules, which share variables, will use the current value of the variable and execute independently of each other.

Theorem 2. $NRE = NP_4(poly^1(6), enz, allP, det)$.

Proof. The proof is done by constructing a membrane system which enumerates the positive values of some polynomial with integer coefficients corresponding to tuples of natural numbers. It is proven in [2], that polynomials of degree at most 5 with 5 variables are sufficient to imply the universality of the models. This technique is used to show that standard NP Systems are universal [7]. The following system is a modified version of the one used in [9] (it is also shown in graphical form in figure 1):

$$
\begin{aligned}
\Pi = (4, H, \mu, &(Var_{Generate}, Pr_{Generate}, Var_{Generate}(0))\\
&(Var_{Compute}, Pr_{Compute}, Var_{Compute}(0), enum),\\
&(Var_{Pow5}, Pr_{Pow5}, Var_{Pow5}(0)),\\
&(Var_{Mult}, Pr_{Mult}, Var_{Mult}(0))),\\
H &= \{Generate, Compute, Pow5, Mult\},\\
\mu &= [_{Generate}[_{Compute}[_{Pow5}\ [_{Mult}]_{Mult}\ ]_{Pow5}]_{Compute}]_{Generate},\\
Var_{Generate} &= \{x_i, e_j, ez_k, er_i, n, e_t, g, gc :\ 1 \le i \le 5, 1 \le i \le 7, 1 \le k \le 5\},\\
Pr_{Generate} &= \{n \to 1|n,\ e_t \to 1|gc,\\
&\quad 1 + x_1|_{e_1} \to 1|er_1,\ -1 + g|_{e_1} \to 1|x_1,\ 1 + n + x_1|_{e_1} \to 1|x_1\}\\
&\cup \{j \cdot e_j \to 1|e_{j+1} + (j-1)|e_t :\ 1 \le j \le 5\}\\
&\cup \{1 + x_i|_{e_1} \to 1|ez_i,\ 1 - i + e_t|_{er_{i-1}} \to 1|x_i :\ 2 \le i \le 5\}\\
&\cup \{g + (ez_i + er_{i-1})|_{e_i} \to 1|er_i,\ 2 - i + n + e_t|_{er_i} \to 1|x_i\\
&\quad :\ 2 \le i \le 5\}\\
&\cup \{2 + e_t|_{er_5} \to \sum_{i=1}^{5} 1|x_i + 1|n,\ er_5|_{e_6} \to 1|gc,\ e_6 \to 1|e_7,\\
&\quad e_7 \to 1|e_1^c\}\\
&\cup \{g + 2 \cdot x_i|_{e_7} \to 1|x_i + 1|x_i^c\ :\ 1 \le i \le 5\},\\
Var_{Generate}(0) &= (5,5,5,5,5,1,0,\ldots,0,5,0,0,0),\\
Var_{Compute} &= \{x_i^c, e_j^c, t, g^*, ep_t, f_Q, enum, aux, e_f :\ 1 \le i \le 5, 1 \le j \le 506\},\\
Pr_{Compute} &= \{g^* + \left(\sum_{i=1}^{5} a_{i,k} \cdot x_i^c + a_{6,k}\right)|_{e^c_{2k-1}} \to 1|s_1,\\
&\quad 3 \cdot e^c_{2k-1} \to 1|e^c_{2k} + 2|ep_1,\ e^c_{2k}|_{ep_t} \to 1|e^c_{2k+1},\\
&\quad g^* - 2 \cdot \beta_k t|_{e^c_{2k+1}} \to 1|aux + 1|e_f\ :\ 1 \le k \le 252\}\\
&\cup \{2 \cdot e^c_{505} \to 1|e_f + 1|e^c_{506},\ aux|_{e_f} \to 1|f_Q,\ -f_Q|_{g*} \to 1|enum,\\
&\quad -(f_Q + e^c_{506}) \to 1|e_f,\ enum + f_Q + ep_t \to 1|gc,\ e^c_{506} \to 1|e_1\}\\
Var_{Compute}(0) &= (0,0,\ldots,0),\\
Var_{Pow5} &= \{s_1, s_2, ep_i, e_M, gc^*, z\ :\ 1 \le i \le 7\},\\
Pr_{Pow5} &= \{z + 3 \cdot s_1|_{ep_1} \to 1|a + 1|b + 1|s_1,\ z + 2 \cdot s_2|_{ep_3} \to 1|a + 1|b,\\
&\quad z + s_1|_{ep_5} \to 1|a,\ z + s_2|_{ep_5} \to 1|b,\ z + s_2|_{ep_7} \to 1|t,\\
&\quad ep_7 \to 1|ep_t,\ e_M \to 1|gc^*\}\\
&\cup \{3 \cdot ep_{2k-1} \to 1|ep_{2k} + 2|e_s,\ ep_{2k}|_{e_M} \to 1|ep_{2k+1},\ 1 \le k \le 3\}\\
Var_{Pow5}(0) &= (0,0,\ldots,0),\\
Var_{Mult} &= \{a, b, z^*, d, u, e_s\},\\
Pr_{Mult} &= \{z^* + 1.5 \cdot a|_b \to 2|a + 1|s_2,\ z^* - (1+d)|_b \to 1|d,\ d \to 1|b
\end{aligned}
$$

$$e_s + b|_u \to 1|e_M,\ a + b|_u \to 1|gc^*\}$$
$$Var_{Mult}(0) = (0, 0, 0, 0, 1, 0).$$

The proposed membrane system is mainly composed of two parts: the 5-tuple generation part and the computation of the polynomial's value. The generating part, implemented in the *Generate* membrane, is the same as in the proof from [9]. Only the last rule of the membrane, $e_7 \to 1|e_1^c$, was changed in order to synchronize it with the *Computation* membrane. The 5-tuple generation process is described in detail in [9]. The five variables forming the tuple are regarded as a single number with 5 digits in a certain base. The algorithm counts down from the highest 5-digit number to zero. Therefore, if the current base is $b+1$, the membrane will generate the numbers from *bbbbb* to 00000. When the null tuple is reached the variables are reset to the highest 5-digit number of the next base. The algorithm start with base 6 from the tuple $(5, 5, 5, 5, 5)$ and generates $(5, 5, 5, 5, 4)$, ..., $(5, 5, 5, 5, 0)$, $(5, 5, 5, 4, 5)$, ..., $(0, 0, 0, 0, 0)$. At this point it will move to the tuple $(6, 6, 6, 6, 6)$ which corresponds to the highest 5-digit number in base 7. The process repeats indefinitely, thus generating all 5-tuples of natural numbers in a deterministic way.

For the next part of the proof it is important to recall that every polynomial f of degree 5 with 5 variables can be put in the following form (lemma from [9]):

$$f(x_1, \ldots, x_5) = \sum_{i=1}^{m} \beta_i \cdot (a_{1,i}x_1 + \ldots + a_{5,i}x_5 + a_{6,i})^5 \tag{1}$$

where m is 252 and represents the maximum number of terms of f in the general form, β_i are polynomial specific coefficients and $a_{j,i}$ are some constants. This form of the polynomial is used in order to compute the values corresponding to the generated 5-tuples in the first part of the procedure in the *Generate* membrane.

The *Compute* membrane was rewritten such that only polynomials of degree one are used as production functions. This is achieved by noting that the only part where polynomials of degree greater than one are needed is when the 5-th power of a number is computed, more specifically a natural number [9]. Computing the power of a number can be done using only multiplication; computing the 5-th power of x can be done by first computing $a = x \cdot x = x^2$, then $b = a \cdot a = x^4$ and finally $c = x \cdot b = x^5$. Since x in the system is a natural number, multiplication can be performed as a repeated addition, $a \cdot b = \underbrace{a + \ldots + a}_{b}$. This procedure is implemented in the *Pow5* membrane which repeatedly uses the *Mult* membrane to compute the products of natural numbers. Thus the degree of the polynomials in all production functions is reduced to 1, the optimal value.

Also, the number of membranes needed in the computation was reduced by reusing some membranes, *Pow5* and *Mult*. Instead of using $m = 252$ *Pow5* membranes in order to compute the m terms of the polynomial (in the form from equation 1), the membrane is used repeatedly to compute each term. Therefore, the number of membranes is reduced to 4. □

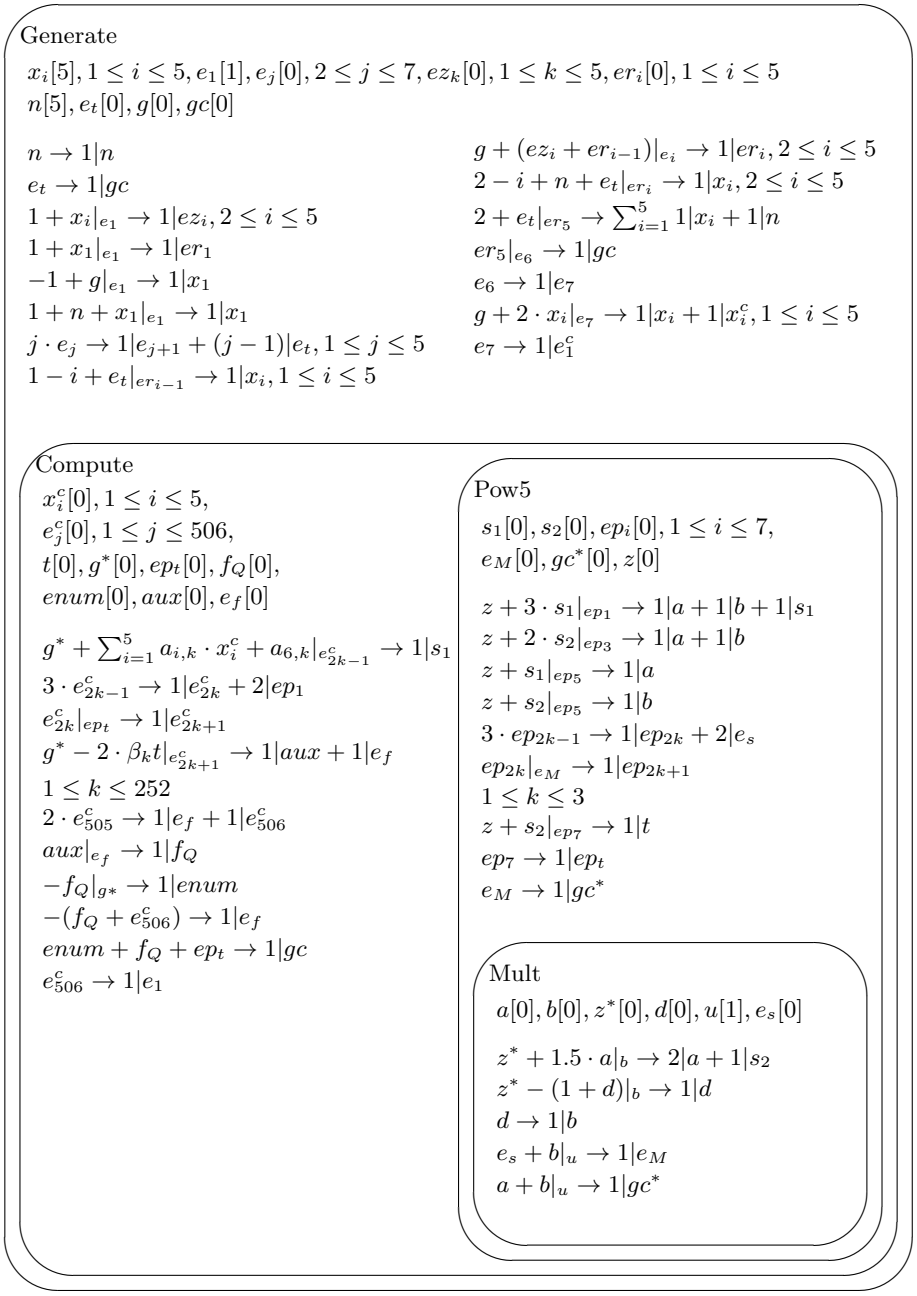

Fig. 1. The EN P system from the proof of Theorem 2

Parts of the proposed membrane system, *Generate* membrane and *Pow5* membrane, were simulated and verified using SimP, an EN P Systems simulator proposed in [4].

4 Remarks

In the proof of theorem 2 a method of reusing membranes was used in order to reduce the number of membranes in the system. It is, however, important to notice that it also constrained the system to perform most important computations in a serial manner. In practice, it may be more convenient to have more membranes that compute in parallel, because it allows the underlying runtime environment to perform optimizations based on available hardware and software platform. It is also important to note that there are more rules dedicated to program control flow in the membrane system from theorem 2 than there are in the one from theorem 4 in [9].

Another important observation is that even though computation can be done with polynomial production functions of degree 1, in some cases it is more convenient to use higher degree polynomials. However, most rules used for program flow control are of degree 1 and also, most rules with higher degree polynomial productions functions have few terms. These observations are relevant for optimizing the data structures and algorithms used for simulating EN P Systems.

Acknowledgments

This paper is supported by the Sectorial Operational Program Human Resources Development, financed from the European Social Fund and by the Romanian Government under the contract number SOP HRD/107/1.5/S/82514.

References

1. Buiu, C., Vasile, C.I., Arsene, O.: Development of membrane controllers for mobile robots. Information Sciences (in press), doi: 10.1016/j.ins.2011.10.007
2. Minsky, M. (ed.): Computation: Finite and Infinite Machines. Prentice-Hall (1967)
3. Pavel, A., Arsene, O., Buiu, C.: Enzymatic numerical P systems - a new class of membrane computing systems. In: The IEEE Fifth International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA 2010) Liverpool. pp. 1331–1336 (September 2010)
4. Pavel, A.B.: Membrane controllers for cognitive robots. Master's thesis, Department of Automatic Control and System Engineering, Politehnica University of Bucharest, Romania (February 2011)
5. Pavel, A.B., Buiu, C.: A software tool for modeling and simulation of numerical P systems. Natural Computing (in press), doi: 10.1007/s11047-011-9286-5
6. Pavel, A.B., Vasile, C.I., Dumitrache, I.: Robot localization implemented with enzymatic numerical P systems (submitted)

7. Păun, G., Paun, A.: Membrane Computing and Economics: Numerical P Systems. Fundamenta Informaticae pp. 213–227 (2004)
8. Păun, G., Rozenberg, G., Salomaa, A. (eds.): The Oxford Handbook of Membrane Computing. Oxford University Press (2010)
9. Vasile, C.I., Pavel, A.B., Dumitrache, I., Păun, G.: On the Power of Enzymatic Numerical P Systems (submitted)