On the Power of Enzymatic Numerical P Systems

Cristian Ioan Vasile[1], Ana Brânduşa Pavel[1], Ioan Dumitrache[1], and Gheorghe Păun[2,3]

[1] Department of Automatic Control and Systems Engineering
Politehnica University of Bucharest
Splaiul Independenţei 313, 060042 Bucharest, Romania
{cvasile, apavel, idumitrache}@ics.pub.ro
[2] Institute of Mathematics of the Romanian Academy
PO Box 1-764, 014700 Bucharest, Romania
george.paun@imar.ro
[3] Research Group on Natural Computing
Department of Computer Science and Artificial Intelligence
University of Sevilla
Avda. Reina Mercedes s/n, 41012 Sevilla, Spain
gpaun@us.es

Abstract. The main result of the paper is 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.

Keywords: Membrane computing, Numerical P Systems, Enzymatic Numerical P Systems, Turing universality

1 The main result

The main result of the paper 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 productions 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 1. $NRE = NP_4(poly^1(6), enz, allP, det)$.

Proof. The proof is as always done by constructing a membrane system which enumerates the positive values of some polynomial with integer coeficients corresponding to tuples of natural numbers. It is proven in [1], that polynomials of degree at most 5 with 5 variables are suficient to imply the universality of

the models. This technique is employed to show that standard NP Systems are universal [2]. The following system is a modified version of the one used in [3] (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, \\
& 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_{2k-1}^c} \to 1|s_1, \\
& \quad 3 \cdot e_{2k-1}^c \to 1|e_{2k}^c + 2|ep_1,\ e_{2k}^c|_{ep_t} \to 1|e_{2k+1}^c, \\
& \quad g^* - 2 \cdot \beta_k t|_{e_{2k+1}^c} \to 1|aux + 1|e_f\ :\ 1 \le k \le 252\} \\
& \cup \{2 \cdot e_{505}^c \to 1|e_f + 1|e_{506}^c,\ aux|_{e_f} \to 1|f_Q,\ -f_Q|_{g*} \to 1|enum, \\
& \quad -(f_Q + e_{506}^c) \to 1|e_f,\ enum + f_Q + ep_t \to 1|gc,\ e_{506}^c \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),
\end{aligned}
$$

$$
\begin{aligned}
Var_{Mult} &= \{a, b, z^*, d, u, e_s\},\\
Pr_{Mult} &= \{z^* + 1.5 \cdot a|_b \to 2|a + 1|s_2,\ z^* - (1 + d)|_b \to 1|d,\ d \to 1|b\\
&\qquad e_s + b|_u \to 1|e_M,\ a + b|_u \to 1|gc^*\}\\
Var_{Mult}(0) &= (0, 0, 0, 0, 1, 0).
\end{aligned}
$$

The system is composed of 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 [3]. Only the last rule of the membrane, $e_7 \to 1|e_1^c$, was changed in order to sychronize it with the *Computation* membrane. The 5-tuple generation process is described in detail in [3]. 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 cuurent 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 from 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 indefinetly, thus generating all 5-tuples of natural numbers in a deterministic way.

For the next part of the argument it is important to recall that every polynomial f of degree 5 with 5 variables can be put in the following form (lemma from [3]):

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

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 second part of system, the computational part, 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 taking the 5-th power of a number, more specifically a natural number [3]. Taking 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 productions all production functions is reduced to 1, the optimal value.

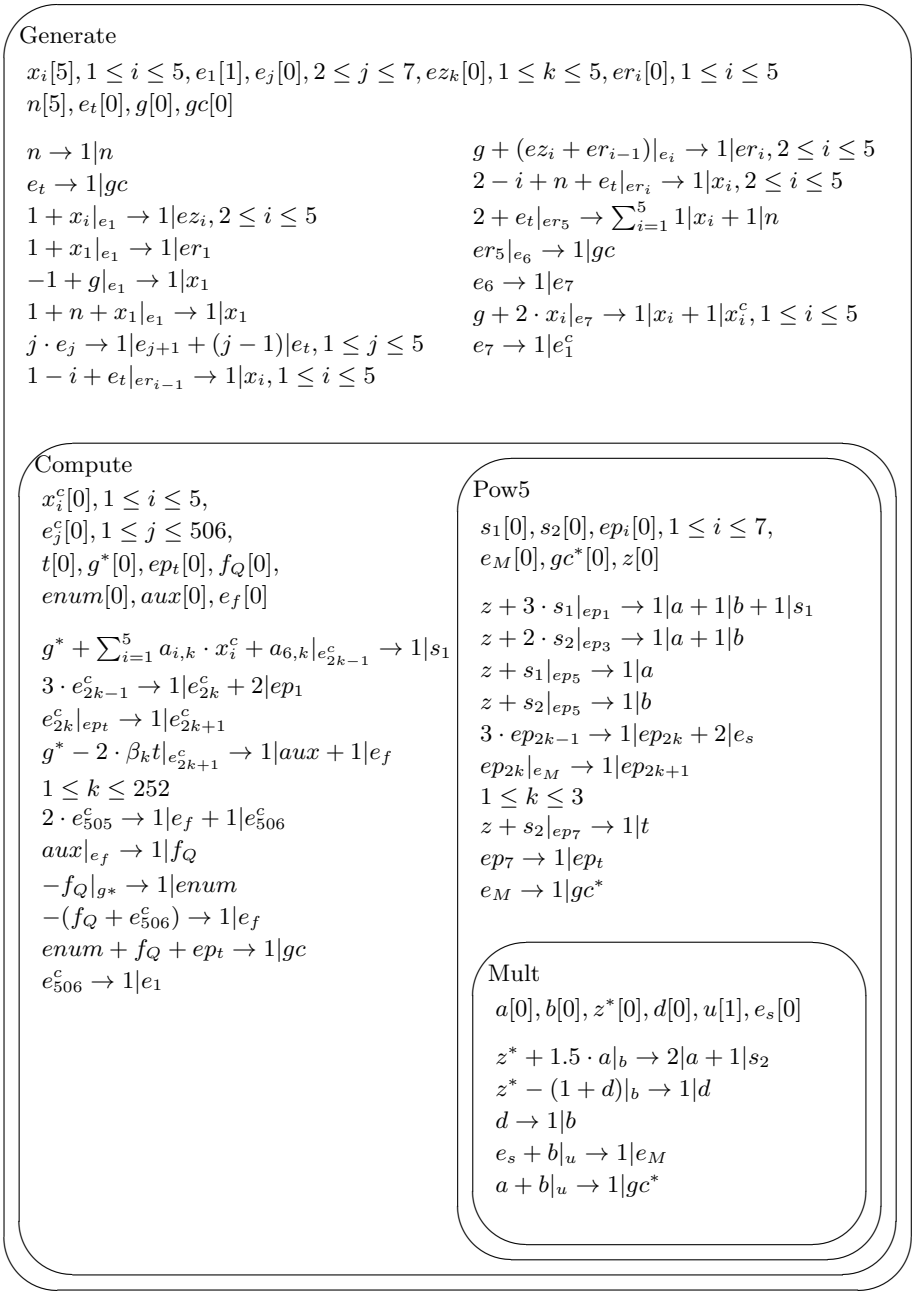

Fig. 1. The EN P system from the proof of Theorem 1

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. □

2 Remarks

In the proof of theorem 1 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 convinient to have more membranes that compute in parallel, because it allows the underlying runtime environment to performe 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 1 than there are in the one from theorem 4 in [3]. 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 porgram flow control are of degree 1 and also, most rules with higher degree polynomial productions fucntions have few terms. These observations can be used in order to optimize the data structures and algorithms employed in simulating EN P Systems.

Acknowledgments.

This paper is suported by the Sectorial Operational Programme 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. Minsky, M. (ed.): Computation: Finite and Infinite Machines. Prentice-Hall (1967)
2. Paun, G., Paun, A.: Membrane Computing and Economics: Numerical P Systems. Fundamenta Informaticae pp. 213–227 (2004)
3. Vasile, C.I., Pavel, A.B., Dumitrache, I., P˘ aun, G.: On the Power of Enzymatic Numerical P Systems. Acta Informatica (submitted)