Membrane Computing and Economics: Numerical P Systems

Gheorghe PĂUN
Institute of Mathematics of the Romanian Academy
PO Box 1-764, 014700 Bucureşti, Romania
george.paun@imar.ro
and
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

Radu PĂUN
University of Maryland, Department of Economics
3105 Tydings Hall
College Park, Maryland 20742-7211, USA
paun@econ.umd.edu

Abstract. With inspiration from the economic reality, where *numbers* are basic entities to work with, we propose a genuinely new kind of P systems, where numerical *variables* evolve, starting from initial values, by means of *production functions* and *repartition protocols*. We prove that non-deterministic systems of this type, using polynomial production functions, characterize the Turing computable sets of natural numbers, while deterministic systems, with polynomial production functions having non-negative coefficients, compute strictly more than semilinear sets of natural numbers. A series of research topics to be addressed in this framework are mentioned.

1 Introduction

Membrane computing is a bio-inspired branch of natural computing, abstracting computing models from the structure and functioning of living cells and from the organization of cells in tissues or other higher order structures. In short, in a cell-like membrane system (currently called P system), multisets of abstract objects are processed by means of

rewriting-like rules (similar to the bio-chemical reactions) or by other types of rules of a biological inspiration; both objects and evolution rules are localized, associated with the compartments defined by membranes. The generality of this approach (as well as its advantages as a modelling tool: easy extensibility, understandability, and programmability, the intrinsic modularity and non-linearity, etc.) makes membrane computing a promising framework for applications in various areas, such as biology, linguistics, computer science – as well as economics.

Details about membrane computing can be found in [7], with a comprehensive bibliography and recent developments available at website [11]. A series of applications can be found in [2], where, however, no application in economics is included. Such applications can be found in [1], [4], [5], and further titles in Polish can be found at [11]; while a more systematic investigation of the possibility of using membrane computing for modelling economic processes was started in [8].

In all previously mentioned papers, one works in a symbolic framework, and deals with multisets of abstract objects. The present paper proposes a completely different approach, at first sight closer to the economic style, using numerical values in the compartments of a cell-like P system. These variables have initial values given (real or natural numbers). Associated with each compartment there are "programs" consisting of two parts, a "production function" and a "repartition protocol". The production function takes the local variables and computes a number, and the "production" corresponds to variables' current values. This number is then distributed among the local variables, the variables from the immediately upper compartment, and those from the immediately lower compartments (much similar to the commands *here, out, in* from "standard" P systems). In each compartment of the system one uses one program at the time, and this happens in parallel in all compartments. This way, new values for variables are obtained. The process continues iteratively, thus making the system evolve over time.

The idea is rather simple, but a lot of possible variations can be considered, in what concerns the forms of production functions and repartition protocols, the way to use them, the kind of observables we are interested in, etc. This paper has a preliminary character, as it only illustrates the definitions with a few examples, mentions a series of research topics of interest, and only proves few results about the computing power of our devices.

The main result is a computational universality one, for the class of non-deterministic numerical P systems with production functions represented by polynomials with integer coefficients. We consider the output of a system to be the set of values a specific variable can take during system's evolution. If we only preserve the positive numbers from this set, then a characterization of recursively enumerable sets of positive natural numbers is obtained, by means of numerical P systems using polynomial production functions of a rather low degree and using a small number of variables, working in a non-deterministic way (several programs are given in some membranes and in each time unit one of them is applied, non-deterministically chosen). The result, significant for the power and properties of numerical P systems, is a direct consequence of Davis–Robinson–Matijasevitch characterization of recursively enumerable sets of natural numbers as (positive) values of polynomials (see [6]).

In the restrictive case of using deterministic systems and polynomials with non-negative coefficients, we show that the respective family strictly includes the family of semilinear sets of positive natural numbers.

2 Numerical P Systems

We do not recall here any detail from membrane computing (the reader can consult the sources mentioned above), except for the notion of membrane structure. This is considered here as illustrated in Figure 1, as a cell-like three-dimensional hierarchical arrangement of vesicles, precisely delimiting compartments between them and inside the elementary membranes (we identify the membranes and their associated compartments with labels – in figure, they are natural numbers).

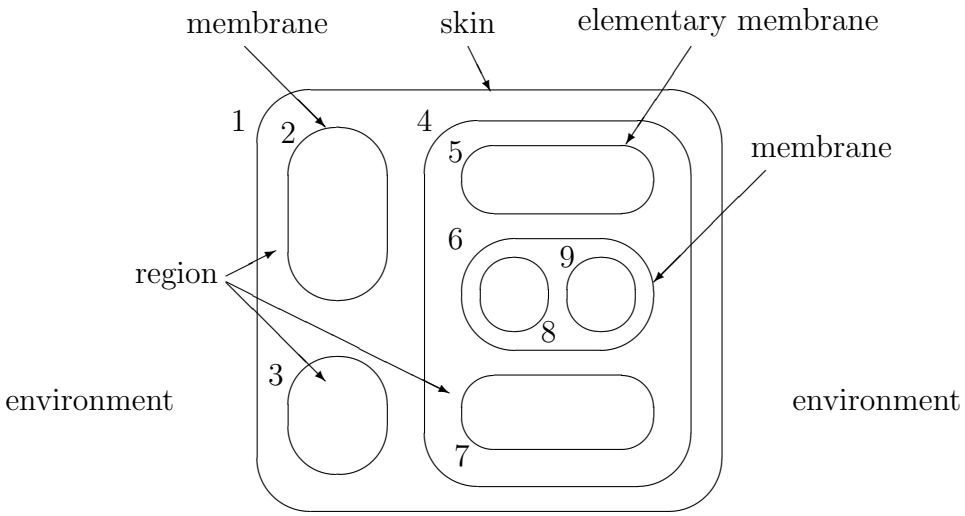

Figure 1: A membrane structure

Clearly, such a structure corresponds to a tree, and it can be represented in a natural/compact way by an expression of correctly matching labelled square brackets; for example, the expression corresponding to the membrane structure from Figure 1 is the following:

$$[_1[_2\ ]_2[_3\ ]_3[_4[_5\ ]_5[_6[_8\ ]_8[_9\ ]_9]_6[_7\ ]_7]_4]_1.$$

In "standard" membrane computing, in the compartments of a membrane structure one places multisets of objects and evolution rules for processing them. Here we consider instead *numerical variables* and *programs* for evolving the variables.

In general, variables will be denoted by letters from the end of the Latin alphabet, especially by x, with subscripts. For instance, $x_{1,i}, x_{2,i}, x_{3,i}$ identify three variables from compartment i of the system. Then, for $t \in \mathbf{N}$ (with $\mathbf{N} = \{0, 1, 2, \ldots\}$ being the set of

natural numbers; by $\mathbf{N}^+$ we denote the set of positive natural numbers, that is, $\mathbf{N}^+ = \mathbf{N} - \{0\}$), we denote by $x_{j,i}(t)$ the *value* of variable $x_{j,i}$ at time t.

Variables' values are considered real numbers, but interesting problems can appear when imposing restrictions, for instance, considering integers as allowed values.

The *programs* that process the variables (that change their values) are made of two components, the *production function* and the *repartition protocol*.

The former component is a general real function having as variables those from a given compartment. For instance, if $x_{j,i}, 1 \leq j \leq k_i$, are the variables from a compartment i, we write $F_{l,i}(x_{1,i}, \ldots, x_{k_i,i})$ to identify the lth production function from compartment i (as we will immediately see, we can have several programs, hence production functions, in each compartment). Of course, we can impose several restrictions on the production functions; for example, we can consider only polynomials (of a restricted degree and/or number of variables), only non-negative coefficients, etc.

It is not equally easy to fix the latter component of a program, the repartition protocol. The idea is to take the current values of variables from a compartment, compute the value of the production function, and then distribute this value among the local and the neighboring variables. Without providing here a complete formalization, let us denote by $par(i)$ the (label of the) parent membrane and by $ch(i)$ the (labels of the) children membranes of a given membrane i (the parent of the skin membrane is the environment, identified, when needed, by *env*, while the set $ch(i)$ of an elementary membrane is empty). In what follows, we propose only one type of repartition protocol: take all variables from compartment i (hence $x_{j,i}$), from $par(i)$ (hence $x_{j,par(i)}$), and from all compartments $k \in ch(i)$ (hence $x_{j,k}$; in each case, j has different ranges); let these variables be $v_1, v_2, \ldots, v_{n_i}$. Then, a repartition procedure is an expression of the form

$$c_1|v_1 + c_2|v_2 + \ldots + c_{n_i}|v_{n_i},$$

where $c_1, \ldots, c_{n_i}$ are natural numbers (they may be also 0, case in which we omit to write "$+0|x_r$"). These coefficients $c_1, \ldots, c_{n_i}$ specify the proportion of the current production distributed to each variable $v_1, \ldots, v_{n_i}$.

More precisely, we proceed as follows. Consider a program

$$pr_{l,i} = (F_{l,i}(x_{1,i}, \ldots, x_{k_i,i}), c_{l,1}|v_1 + c_{l,2}|v_2 + \ldots + c_{l,n_i}|v_{n_i})$$

(also the subscripts of coefficients specify the program). Let

$$C_{l,i} = \sum_{s=1}^{n_i} c_{l,s}.$$

At any time $t \geq 0$, compute $F_{l,i}(x_{1,i}(t), \ldots, x_{k_i,i}(t))$. The value $q = F_{l,i}(x_{1,i}(t), \ldots, x_{k_i,i}(t))/C_{l,i}$ represents the "unitary portion" to be distributed to variables $v_1, \ldots, v_{n_i}$, according to coefficients $c_{l,1}, \ldots, c_{l,n_i}$ in order to obtain the values of these variables at time $t+1$. Specifically, $v_{l,s}$ will receive $q \cdot c_{l,s}, 1 \leq s \leq n_i$. If a variable receives such "contributions" from several neighboring compartments, then they are added in order to produce the next value of the variable.

Thus, formally, we consider systems – called *numerical P systems* – of the form

$$\Pi = (m, H, \mu, (Var_1, Pr_1, Var_1(0)), \ldots, (Var_m, Pr_m, Var_m(0))),$$

where $m \geq 1$ is the number of membranes used in the system (also called the *degree* of Π), H is an alphabet with m symbols (used as labels of membranes), μ is a membrane structure with m membranes, labelled in a one-to-one manner with the elements of H, Var_i is the set of variables from compartment i, Pr_i is the set of programs from compartment i of μ, $1 \leq i \leq m$ (all sets $Var_i, Pr_i, 1 \leq i \leq m$, are finite), and $Var_i(0)$ are numerical values for the variables in compartment $i, 1 \leq i \leq m$; these values are considered as *initial values*, at time 0 of the system evolution. (We have identified here the ith label from H with i; in general, when this will be easy and natural, we will use $H = \{1, 2, \ldots, m\}$.)

Each program is of the form specified above: $pr_{l,i} = (F_{l,i}(x_{1,i}, \ldots, x_{k_i,i}), c_{l,1}|v_1 + c_{l,2}|v_2 + \ldots + c_{l,n_i}|v_{n_i})$ denotes the lth program from compartment i, where $Var_i = \{x_{1,i}, \ldots, x_{k_i,i}\}$.

A system as above evolves as informally described in the Introduction by simultaneously using one program in each compartment in each time unit (the time is supposed to be marked by a universal clock, the same for all membranes). Initially, the variables have values specified by $Var_i(0), 1 \leq i \leq m$. From a time t to time $t + 1$ we pass by (i) choosing non-deterministically one program from each compartment, (ii) computing the value of the respective production function for the values of local variables at time t, and then (iii) computing the values of variables at time $t + 1$ as repartition protocols indicate (this means that previous values of variables which appear in the production functions are "consumed", hence they are reset to zero, and the portions attributed to variables by the repartition protocols are added to form the new value; if a variable does not appear in a production function, then its previous value is not consumed, hence new portions are added to it). This way, we obtain a transition of the system from a configuration to the next (a configuration represents the values of all system's variables at a given moment). A sequence of transitions forms a *trajectory*. Of course, because we can have several programs in each compartment, we can have branchings, hence the trajectories can be arranged in an *evolution tree* (then, each path starting in the root is a trajectory). Obviously, this tree is infinite, with the levels corresponding to configurations which can be reached at the same time starting from the initial configuration.

We do not give here a formal definition of the notions considered above, but we will illustrate them by examples in the next section.

Also, we do not go into details concerning the possible economic interpretations of the elements of a numerical P system or of its functioning. The economic inspiration is visible, with the note that the simplest interpretation of variable values would be in terms of monetary units.

In the previous discussion, a numerical P system was conceived as a dynamical system, and the interest was in its evolution over time. We can also consider such a system as a computing device. In what follows we examine only the case where we take as the *set of numbers* computed by a system the set of values a specified variable assumes during the system evolution – therefore, each system can compute several sets of numbers, one associated with each variable. More details will be given in the next sections.

3 Examples

We proceed with a complex example, in order to clarify the way a transition in a numerical P system is defined. The example also illustrates the intricate behavior such a system can have, although the system we consider is a *deterministic* one, that is, having only one program in each compartment (hence, no branching is possible during the evolution).

The system is

$$\Pi_1 = (4, H, \mu, (Var_1, Pr_1, Var_1(0)), (Var_2, Pr_2, Var_2(0)), (Var_3, Pr_3, Var_3(0)), (Var_4, Pr_4, Var_4(0))),$$

with the following components (because there is only one program in each compartment, the first subscript of programs' elements is omitted):

$$\begin{aligned}
H &= \{1,2,3,4\},\\
\mu &= [_1[_2[_3\ ]_3[_4\ ]_4]_2]_1,\\
Var_1 &= \{x_{1,1}\},\\
Pr_1 &= (2x_{1,1}^2, 1|x_{1,1}+1|x_{1,2}),\\
Var_1(0) &= (1),\\
Var_2 &= \{x_{1,2}, x_{2,2}, x_{3,2}\},\\
Pr_2 &= (x_{1,2}^3 - x_{1,2} - 3x_{2,2} - 9, 1|x_{2,2}+1|x_{3,2}+1|x_{2,3}),\\
Var_2(0) &= (3,1,0),\\
Var_3 &= \{x_{1,3}, x_{2,3}\},\\
Pr_3 &= (2x_{1,3} - 4x_{2,3} + 4, 2|x_{1,3}+1|x_{2,3}+1|x_{1,2}),\\
Var_3(0) &= (2,1),\\
Var_4 &= \{x_{1,4}, x_{2,4}, x_{3,4}\},\\
Pr_4 &= (x_{1,4}x_{2,4}x_{3,4}, 1|x_{1,4}+1|x_{2,4}+1|x_{3,4}+1|x_{3,2}),\\
Var_4(0) &= (2,2,2).
\end{aligned}$$

For an easier understanding, the system is also given in Figure 2 in a graphical representation, with the initial values specified in square brackets for each variable, and with the programs given in the form *production function* $\rightarrow$ *repartition procedure*.

Before examining the way this system evolves, let us point that we have polynomial production functions, that use both positive and negative coefficients, that variable $x_{3,2}$ does not appear in F_2, hence it is not a "productive" one (we leave to the reader the task to interpret this in economic terms), but both compartments 2 and 4 contribute to the value of this variable; then, besides contributing to values of local variables, both compartments 1 and 3 contribute to the value of variable $x_{1,2}$, while compartment 2 also contributes to the value of $x_{2,3}$.

Let us discuss in detail the transition from $t = 0$ to $t = 1$ (the production functions were chosen in such a way that their values are always divisible by the corresponding C_i – the sums of the distribution coefficients; these sums are $C_1 = 2, C_2 = 3, C_3 = 4, C_4 = 4$).

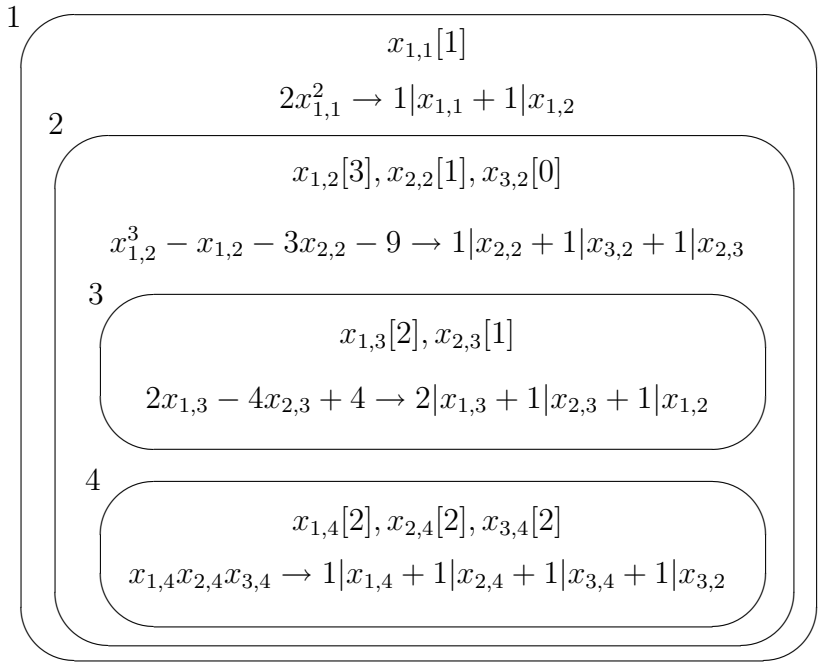

Figure 2: A representation of the numerical P system Π_1

The productions in each compartments are as follows:

$$F_1(1) = 2, \quad F_2(3, 1, 0) = 12, \quad F_3(2, 1) = 4, \quad F_4(2, 2, 2) = 8.$$

Therefore, the "unitary portions" in each compartment are: 1, 4, 1, 2, respectively. The two units from compartment 1 are distributed to $x_{1,1}$ and $x_{1,2}$. This latter variable also receives one unit from compartment 3. Compartment 2 contributes with 4 units to local variables $x_{2,2}$ (a productive one) and $x_{3,2}$ (non-productive), as well as to variable $x_{2,3}$ from membrane 3. Compartment 3 also gives two units to its variable $x_{1,3}$ and one to $x_{2,3}$. Compartment 4 gives 2 units to each of its variables, as well as to $x_{3,2}$. After considering all repartitions, we get the following values for variables:

$$\begin{aligned} &x_{1,1}(1) = 1, \\ &x_{1,2}(1) = 2, \quad x_{2,2}(1) = 4, \quad x_{3,2}(1) = 6, \\ &x_{1,3}(1) = 2, \quad x_{2,3}(1) = 5, \\ &x_{1,4}(1) = 2, \quad x_{2,4}(1) = 2, \quad x_{3,4}(1) = 2. \end{aligned}$$

Now, the productions are

$$F_1(1) = 2, \quad F_2(2, 4, 6) = -15, \quad F_3(2, 5) = -12, \quad F_4(2, 2, 2) = 8,$$

with the portions $1, -5, -3, 2$, respectively, hence we get

$$x_{1,1}(2) = 1,$$

$$x_{1,2}(2) = -2, \quad x_{2,2}(2) = -5, \quad x_{3,2}(2) = 3,$$
$$x_{1,3}(2) = -6, \quad x_{2,3}(2) = -8,$$
$$x_{1,4}(2) = 2, \quad x_{2,4}(2) = 2, \quad x_{3,4}(2) = 2.$$

Continuing, we get the productions

$$F_1(1) = 2, \quad F_2(-2, -5, 3) = 0, \quad F_3(-6, -8) = 24, \quad F_4(2, 2, 2) = 8,$$

therefore

$$x_{1,1}(2) = 1,$$
$$x_{1,2}(2) = 7, \quad x_{2,2}(2) = 0, \quad x_{3,2}(2) = 5,$$
$$x_{1,3}(2) = 12, \quad x_{2,3}(2) = 6,$$
$$x_{1,4}(2) = 2, \quad x_{2,4}(2) = 2, \quad x_{3,4}(2) = 2.$$

The jump from positive to negative and again to positive values for several variables looks hard to predict and this feeling is stressed by the next transition, where we have

$$F_1(1) = 2, \quad F_2(7, 0, 5) = 327, \quad F_3(12, 6) = 4, \quad F_4(2, 2, 2) = 8,$$

and

$$x_{1,1}(4) = 1,$$
$$x_{1,2}(4) = 2, \quad x_{2,2}(4) = 109, \quad x_{3,2}(4) = 116,$$
$$x_{1,3}(4) = 2, \quad x_{2,3}(4) = 110,$$
$$x_{1,4}(4) = 2, \quad x_{2,4}(4) = 2, \quad x_{3,4}(4) = 2.$$

The reader can examine further the evolution of Π_1; we choose to stop here, with the remark that compartments 1 and 4 will always keep the same values for their variables, whereas the "unpredictable behavior" of the system is induced by the programs from compartments 2 and 3.

Let us now consider a much simpler system:

$$\Pi_2 = (1, \{1\}, [_1\ ]_1, (\{x_{1,1}\}, \{(2x_{1,1}, 1|x_{1,1})\}, (1))).$$

It is easy to see that, starting from the initial value 1, the value of $x_{1,1}$ is doubled step by step, hence the trajectory of the system (again deterministic) is 1, 2, 4, 8,..., that is, the powers of 2.

Let us also examine the system Π_3 given in Figure 3, looking for the set of values taken by variable $x_{1,1}$ – we denote this set by $Set(\Pi_3, x_{1,1})$. The reader can easily see that variable $x_{1,3}$ increases by 1 at each step, also transmitting its value to $x_{1,2}$. In turn, compartment 2 transmits the value $2x_{1,2} + 1$ to $x_{1,1}$, which is never consumed, hence its value increases continuously. We start with all variables equal to 0. Thus, $x_{1,1}$ starts from

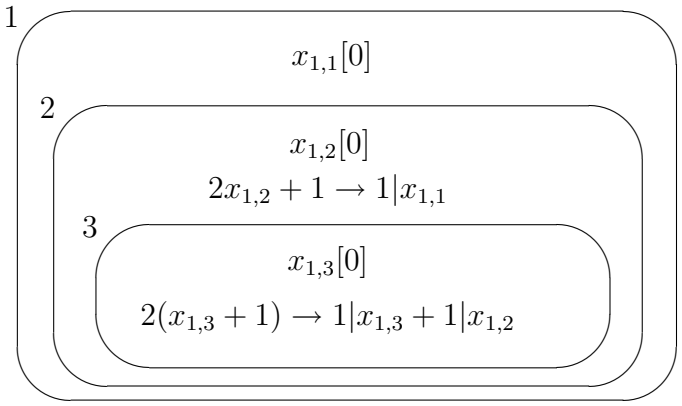

Figure 3: The system Π_3

0 and continuously receives $2i+1$, for $i = 0, 1, 2, 3, \ldots$, which implies that in n steps we get for $x_{1,1}$ the value $\sum_{i=0}^{n-1}(2i+1) = n^2$, that is, $Set(\Pi_3, x_{1,1}) = \{n^2 \mid n \geq 0\}$.

We conclude this section with an example of a non-deterministic system:

$$\Pi_4 = (1, \{1\}, [\,_1\,]_1, (\{x_{1,1}\}, \{(2x_{1,1}, 1|x_{1,1}),\ (3x_{1,1}, 1|x_{1,1})\}, (1))).$$

Again, the production is assigned to the unique variable, but in each step we can choose either the first program or the second one; in the former case the value of the variable is multiplied by 2, in the latter case it is multiplied by 3. Thus, the values of $x_{1,1}$ will be of the form 2^i3^j, with $i \geq 0, j \geq 0$. Actually, *all numbers of this form are values of* $x_{1,1}$, with all 2^i3^j being reachable in step $i+j$. Therefore, the trajectories of Π_4 are paths in the graph whose first five levels are indicated in Figure 4, starting in node 1; on the arrows we have mentioned the program used in that transition.

4 On the Computing Power of Numerical P Systems

Let us start by fixing some notations. As already used in the previous section, let us denote by $Set(\Pi, x_{j,i})$ the set of values assumed by variable $x_{j,i}$ during the evolution of system Π; if values less than or equal to 0 are omitted, then we write $Set^+(\Pi, x_{j,i})$.

From now on, only polynomials with integer coefficients are considered as production functions. Two important parameters describing the complexity of a polynomial are the *degree* of the polynomial and the *number of variables* (we also call this as the *dimension* of the polynomial).

Then, we can distinguish between polynomials with arbitrary integer coefficients and those with non-negative coefficients.

A delicate point appears when considering the repartition protocols, when dividing the current production to the sum of the repartition coefficients. Let us first distinguish the restricted case where all values of all polynomials in a system are divisible by the respective sums $C_{l,i}$ of repartition coefficients (we associate with this class of programs

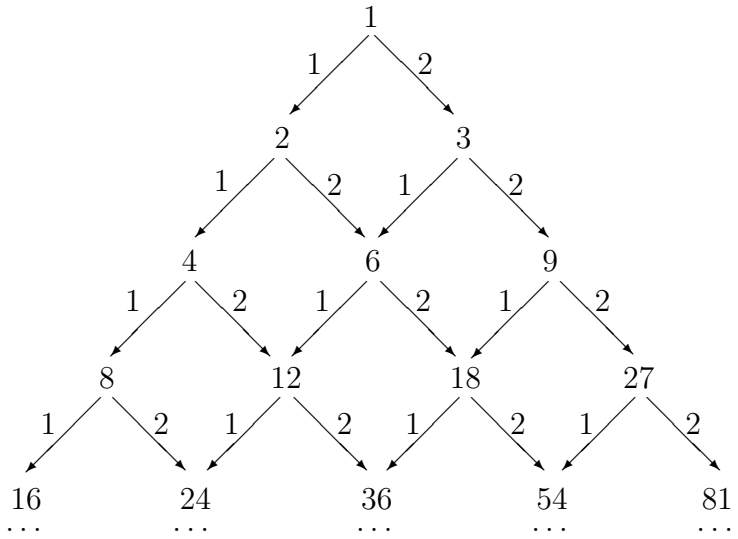

Figure 4: The evolution graph of system Π_4

the notation *div*). When a current production is not divisible by the associated coefficients total, then we can take the following decisions: (i) the remainder is lost (the production which is not immediately distributed is lost), (ii) the remainder is added to the production obtained in the next step (the non-distributed production is carried over to the next step), (iii) the system simply stops and aborts, no result is associated with that computation. We denote these three cases with *lost, carry, stop*, respectively.

Thus, we can distinguish many types of systems, depending on the programs and their use. Accordingly, we define the following families of sets of numbers. By

$$SET^+P_m(poly^n(r), nneg, \alpha),\ m \geq 1, n \geq 0, r \geq 0, \alpha \in \{div, lost, carry, stop\},$$

we denote the family of all sets $Q \subseteq \mathbf{N}^+$ such that there are a system Π and a variable $x_{j,i}$ of Π such that $Q = Set^+(\Pi, x_{j,i})$, and Π has at most m membranes, the polynomials from its programs have degrees at most n and dimensions at most r, the coefficients positive or zero, and the distribution is of type indicated by α. If we work with deterministic systems, then we also add the letter D in front of SET. If arbitrary coefficients are allowed, then the indication "nneg" is removed. If one of the parameters m, n, r is not bounded, then it is replaced by $*$. The case $n = 0$ corresponds to constant polynomials, and $r = 0$ to the absence of variables.

Because we have written the polynomials in the form $F_{l,i}(x_{1,i}, \ldots, x_{k_i,i})$, that is, mentioning all variables from compartment i (even if some of them do not appear in the polynomial expression), parameter r above can be interpreted as representing the maximum number of variables in the compartments of the system.

Note that for a given triple (m, n, r) we have 16 families, obtained by combining the other three classification criteria: deterministic or not, non-negative or arbitrary, and *div, lost, carry, stop*.

When using non-negative coefficients, the values of polynomials are non-negative, but the writing SET^+ still makes sense, because it reminds the fact that number 0 is omitted. This is much similar to the usual convention in automata and language theory (membrane computing, too), where the empty string (number 0) is ignored when comparing the computing power of two devices.

Let us also denote by N^+RE the family of recursively enumerable (Turing computable) sets of positive natural numbers.

Directly from the definitions above, we have a series of relations between our families. (When the two terms of a relation contain parameters written in square brackets, then the relation is true both with the two parameters present and with the two parameters missing.)

Lemma 4.1 *1.* $[D]SET^+P_m(poly^n(r)[,nneg],\alpha) \subseteq [D]SET^+P_{m'}(poly^{n'}(r')[,nneg],\alpha)$, *for all* $1 \leq m \leq m', 0 \leq n \leq n', 0 \leq r \leq r', \alpha \in \{div, lost, carry, stop\}$,

2. $DSET^+P_m(poly^n(r)[,nneg],\alpha) \subseteq SET^+P_m(poly^n(r)[,nneg],\alpha)$, *for all* $m \geq 1, n \geq 0, r \geq 0$, $\alpha \in \{div, lost, carry, stop\}$,

3. $[D]SET^+P_m(poly^n(r),nneg,\alpha) \subseteq [D]SET^+P_m(poly^n(r),\alpha)$, *for all* $m \geq 1, n \geq 0, r \geq 0$, $\alpha \in \{div, lost, carry, stop\}$,

4. $[D]SET^+P_m(poly^n(r)[,nneg],div) \subseteq [D]SET^+P_m(poly^n(r)[,nneg],\alpha)$, *for all* $m \geq 1$, $n \geq 0$, $r \geq 0$, $\alpha \in \{lost, carry, stop\}$

These relations can be extended to cases where one, two, or all three of m, n, r are replaced with $*$, but we leave the details to the reader. Also, it is true that all these families are included in N^+RE; these inclusions can be proved in a straightforward (but tedious) way (constructing a Turing machine for simulating a given P system) or we can invoke for them the Turing-Church thesis.

Anyway, the most restricted families are $DSET^+P_m(poly^n(r),nneg,div)$, especially for small m, n, r, hence these families deserve a special interest. As we will immediately show, these families – with arbitrary m – strictly include the family of semilinear sets of (positive) natural numbers.

A subset of $\mathbf{N}^k, k \geq 1$, is semilinear if it is a finite union of linear sets; a set $M \subseteq \mathbf{N}^k$ is linear if there are $v_0, v_1, \ldots, v_s \in \mathbf{N}^k$ such that $M = \{v_0 + a_1v_1 + \ldots + a_sv_s \mid a_1, \ldots, a_s \in \mathbf{N}\}$. We denote by $SLIN_k^+, k \geq 1$, the family of semilinear sets of vectors from $(\mathbf{N}^+)^k$ (hence $SLIN_1^+$ is the family of semilinear sets of positive natural numbers).

Theorem 4.1 *(i)* $DSET^+P_1(poly^1(1),nneg,div) - SLIN_1^+ \neq \emptyset$.
(ii) $SLIN_1^+ \subset DSET^+P_*(poly^1(1),nneg,div)$.

Proof. (i) This is a consequence of the fact that, as shown in Section 3, the non-semilinear sets $\{2^n \mid n \geq 0\}$ and $\{n^2 \mid n \geq 1\}$ belong to $DSET^+P_1(poly^1(1),nneg,div)$.

(ii) The strictness of the inclusion follows from (i), hence we only have to prove the inclusion. To this aim, we use the following characterization of semilinear sets of numbers

(see, e.g., [3]): a set $Q \subseteq \mathbf{N}$ is semilinear (that is, it is the length set of a regular language) if either Q is finite, or it is the union of a finite set with an infinite arithmetical progression. Clearly, the same assertion is true if number 0 is ignored. Let us assume that we have the more complex case, of an infinite set, and suppose that

$$Q = \{n_1, n_2, \ldots, n_k\} \cup \{m_0 + i \cdot m_1 \mid i \geq 0\},$$

for some $n_1, \ldots, n_k, m_0, m_1 \in \mathbf{N}^+$. We construct the system

$$\begin{aligned}
\Pi &= (k+2, H, \mu, (Var_0, Pr_0, Var_0(0)), \\
&\quad (Var_1, Pr_1, Var_1(0)), \ldots, (Var_{k+1}, Pr_{k+1}, Var_{k+1}(0))), \\
H &= \{0, 1, 2, \ldots, k+1\}, \\
\mu &= [_0[_1[_2 \cdots [_{k+1}\,]_{k+1} \cdots]_2]_1]_0, \\
Var_i &= \{x_{1,i}\}, \text{ for all } 0 \leq i \leq k+1, \\
Pr_0 &= \{(2(x_{1,0} + m_1), 1|x_{1,0} + 1|x_{1,1})\}, \\
Pr_i &= \{(x_{1,i}, 1|x_{1,i+1})\}, \text{ for all } 1 \leq i \leq k, \\
Pr_{k+1} &= \emptyset, \\
Var_0(0) &= (m_0), \\
Var_i(0) &= (n_i), \text{ for all } 1 \leq i \leq k, \\
Var_{k+1}(0) &= 0.
\end{aligned}$$

For the reader convenience, this system is also represented in Figure 5. One can easily see that in each time unit all variables from compartments $1, 2, \ldots, k$ "push down" their values towards the central membrane, hence all numbers $n_1, n_2, \ldots, n_k$ become once values of $x_{1,k}$. Similarly, $x_{1,0}$ increases in every step with m_1, hence it follows the values of the arithmetical progression $\{m_0 + i \cdot m_1 \mid i \geq 0\}$, simultaneously also moving its value to $x_{1,1}$, so that the respective value is pushed towards $x_{1,k}$. In each step, variable $x_{1,k}$ is reset to 0, by transferring its value to $x_{1,k+1}$. Thus, $Set^+(\Pi, x_{1,k}) = Q$, and $Q \in DSET^+P_{k+2}(poly^1(1), nneg, div)$. □

We pass now to a larger class of systems, removing the restrictions of working in a deterministic way and of using only non-negative coefficients. This is enough to ensure the Turing completeness of our systems.

Theorem 4.2 $N^+RE = SET^+P_*(poly^*(*), div)$.

Proof. In view of the remarks above, we only prove here the inclusion $N^+RE \subseteq SET^+P_*(poly^*(*), div)$.

To this aim, we use the characterization of N^+RE by means of positive values of polynomials with integer coefficients ([6]): for each set $Q \in N^+RE$ there is a polynomial $f_Q(x_1, \ldots, x_n)$ with integer coefficients such that $r \in Q$ iff $r = f_Q(a_1, \ldots, a_n)$ for some $a_1, \ldots, a_n \in \mathbf{N}$.

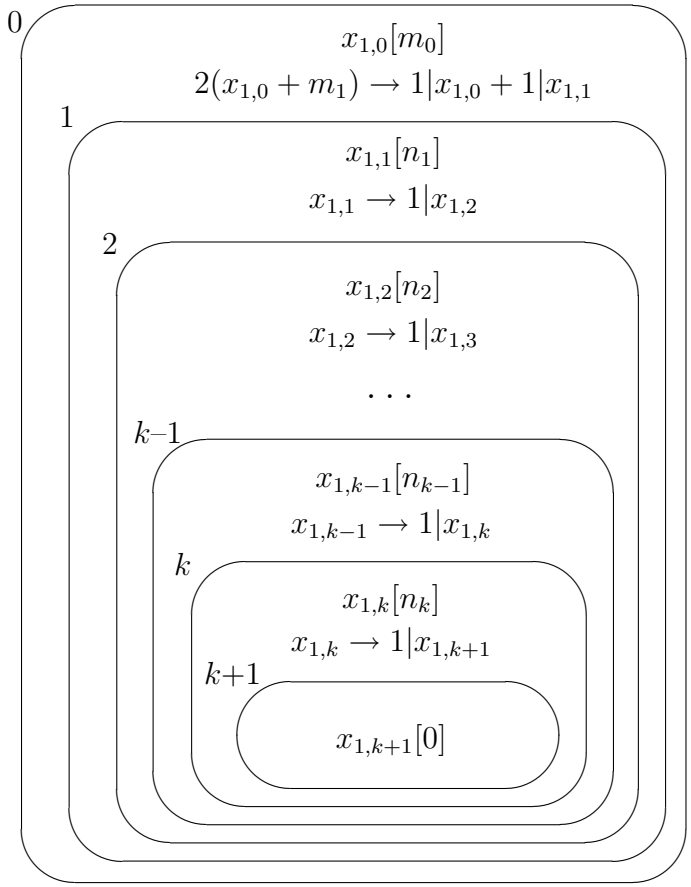

Figure 5: The system from the proof of Theorem 4.1

Take an arbitrary set $Q \in N^+RE$ and let $f_Q(x_1, \ldots, x_n)$ be a polynomial as above. We construct the following numerical P system:

$$
\begin{aligned}
\Pi &= (n+3, H, \mu, (Var_s, Pr_s, Var_s(0)), (Var_0, Pr_0, Var_0(0)), \\
&\quad (Var_g, Pr_g, Var_g(0)), (Var_1, Pr_1, Var_1(0)), (Var_2, Pr_2, Var_2(0)), \\
&\quad \ldots, (Var_n, Pr_n, Var_n(0))), \\
H &= \{s, 0, g, 1, 2, \ldots, n\}, \\
\mu &= [_1[_0[_g\,]_g]_0[_1\,]_1[_2\,]_2 \cdots [_n\,]_n]_s, \\
Var_s &= \{x_{1,s}, x_{2,s}, \ldots, x_{n,s}\}, \\
Pr_s &= \{(f_Q(x_{1,s}, x_{2,s}, \ldots, x_{n,s}), 1|x_{1,0})\}, \\
Var_s(0) &= (0, 0, \ldots, 0), \\
Var_0 &= \{x_{1,0}\}, \\
Pr_0 &= \{(x_{1,0}, 1|x_{1,g})\}, \\
Var_0(0) &= (0), \\
Var_g &= \{x_{1,g}\},
\end{aligned}
$$

$$
\begin{aligned}
Pr_g &= \emptyset, \\
Var_g(0) &= (0), \\
Var_i &= \{x_{1,i}\}, \\
Pr_i &= \{(x_{1,i}, 1|x_{1,i}), \quad (x_{1,i}+1, 1|x_{1,i}), \quad (x_{1,i}, 1|x_{i,s})\}, \\
Var_i(0) &= (0), \text{ for all } i = 1, 2, \ldots, n.
\end{aligned}
$$

(The labels from H have the following "meanings": s indicates the skin membrane, 0 is the compartment where the result of the computation is collected, g is a "garbage collector", and membranes 1, 2,..., n are associated with the n variables of the polynomial f_Q.)

This system works as follows. All variables starts from 0. In each membrane $1, 2, \ldots, n$ there are three programs; the first one leaves the local variable unchanged, the second one increases it by one, the third program moves the value of the local variable to the corresponding variable from compartment s. Specifically, compartment i provides a value to variable $x_{i,s}, 1 \leq i \leq n$. By non-deterministically choosing the programs to apply, in compartments $1, 2, \ldots, n$ we can produce all vectors $(a_1, a_2, \ldots, a_n)$ of natural numbers.

In each step, in compartment s we compute a value of polynomial f_Q, therefore we can compute such values for all vectors of natural numbers. The respective values are immediately transferred to variable $x_{1,0}$ from compartment 0. In each step, this variable is also reset to 0, by transferring its value to the "garbage collector" variable $x_{1,g}$. Therefore, variable $x_{1,0}$ takes all values of f_Q, which means that $Set^+(\Pi, x_{1,0}) = Q$, hence $Q \in SET^+P_*(poly^*(*), div)$. This concludes the proof. □

We do not know whether a similar result can be obtained for deterministic numerical P systems. Because the non-determinism is used only in membranes $1, 2, \ldots, n$, hence for obtaining all vectors $(a_1, \ldots, a_n)$ of natural numbers, the problem reduces to the question whether or not the set of all such vectors can be generated in a deterministic way. Another interesting research topic is whether or not the positive values of a variable can be selected inside the system (changing in a minimal way the definition), not by means of an external condition.

In what concerns the number of membranes, the degree and the dimension of the polynomials used, these parameters can be bounded by rather small values, making use of similar results about the degree and the number of variables used in polynomials f_Q which characterize recursively enumerable sets of numbers. For instance, polynomials of degree 5 using 5 variables suffice, hence we obtain:

Corollary 4.1 $N^+RE = SET^+P_8(pol^5(5), div) = SET^+P_7(poly^5(6), div)$.

Proof. The equality $N^+RE = SET^+P_8(pol^5(5), div)$ is a direct consequence of the previous proof. Then, membrane g can be omitted, at the price of using one further variable in compartment s: instead of the program $Pr_0 = \{(x_{1,0}, 1|x_{1,g})\}$ we use $Pr_0 = \{(x_{1,0}, 1|x_{n+1,s})\}$, with variable $x_{n+1,s}$ added to Var_s, with initial value 0. □

In view of point 4 in Lemma 4.1, similar results hold true also for families obtained by replacing *div* with any of *lost, carry, stop*.

Characterizations of N^+RE as positive images of polynomials can be found in [6] for other combinations degree–dimension, and they imply results similar to that in Corollary 4.1. A systematic examination of these results, with the possibility of improving them in various parameters, remains as a research topic. It also remains to investigate the size and properties of families $SET^+P_m(poly^n(r), div)$ with values of m, n, r smaller than those in Corollary 4.1 or in similar universality results.

5 A Wealth of Research Topics

As already mentioned, the goal of the present paper is only to introduce the numerical P systems, to suggest some of their basic features, and to leave the mathematical study of these systems and the possible applications in economics for further research. While the issue of applications in economics is easy to formulate (find relevant case studies, write adequate numerical P systems, investigate them analytically or, presumably easier and still relevant, write simulation programs and carry computer experiments, look for problems of a practical or theoretical interest for economics which can be addressed in this framework), the mathematical problems are much more intricate, as they are rather numerous and diverse.

First, we have seen that there are many different classes of systems, and still more can be considered. For instance, in Theorems 4.1 and 4.2, all productions are simply transferred to only one variable, which is a rather particular case of the class *div*. Then, it is crucial whether we work with real numbers or only with integers – in the general case of using real coefficients, no difficulty appears with respect to the divisibility of the production.

All this discussion assumes that the repartition protocols are of the form used in previous sections, but also this issue can be questioned: what other forms of repartition protocols can be used? For instance, what about some procedures which evolve in time (depending on the moment of time, on the previous value of the variables, etc.), not fixed in advance as considered here?

After fixing the components' form, we have to settle a basic question: which are the "observables", which parameters of the system are to be examined. We have considered here the set of values assumed by a specified variable, but there exist other possibilities as well: taking the vector of values of all variables, of a specified set of variables, the vector of productions of all/certain compartments or only the production from one compartment. In all cases, we can consider either the set or the sequence of the specified values.

Now, it follows the distinction between deterministic and non-deterministic systems, a distinction relevant for all classes of systems and for all types of observed behavior. Actually, even for non-deterministic systems (i.e., which have several programs in compartments) we can define the evolution in a deterministic way, for instance, using the programs from any compartment cyclically, or according to other "external controls".

Combining all these possibilities, we get a large panoply of classes of "outputs" of numerical P systems. An interesting endeavor would be to compare these classes, among

themselves and with classes of vector sequences or number sequences known in other areas, or of vector sets or number sets from other areas. Does the degree of polynomials induce an infinite hierarchy of families for a specific class of systems? The same with respect to the dimension of polynomials. Does the use of negative coefficients add power? What about the classic question whether deterministic systems are less powerful than the non-deterministic ones? (In particular, can a deterministic system produce the same set of numbers as the non-deterministic system Π_4 from Section 3?)

In what concerns sequences of numbers from other areas of mathematics, the possibilities are multiple. Just consider sequences from [10], or the sequences investigated in the theory of Lindenmayer systems – see, e.g., [9].

Besides studying properties of sequences or of sets (of vectors or of numbers), we can also consider decidability and complexity problems, and such problems can also have relevance for economics. Does a variable take a negative value? Does a variable take a value greater than a specified threshold? Is a specified vector reachable by some specified variables? Is the set of numbers reached by a specified variable finite? Is in each step the difference between the largest and the lowest value of a variable bounded during the system evolution? (This can be considered as a fairness property.) Is each of these questions solvable algorithmically? If so, which is the complexity of solving it?

Of course, these questions should be addressed for particular classes of numerical P systems. For instance, in view of Theorem 4.2, all non-trivial (Rice theorem) decidability questions have a negative answer for non-deterministic systems using polynomials with arbitrary integer coefficients as production functions.

There also are a series of standard problems in membrane computing: find infinite hierarchies on the number of membranes, define and find normal forms, define and investigate descriptional complexity measures, investigate closure properties, etc.

The list of such research topics can be continued, but we conclude with a question of a theoretical computer science nature: how can a numerical P system be converted into a computing device of the same kind as usual P systems? More precisely, what kind of a halting condition or another way of signaling the end of a computation can be considered, such that a grammar-like generative device is obtained? How can a numerical P system work in the accepting mode or in a functional mode (taking an input, we stop or we output yes/no, or a numerical value, respectively).

6 Final Remarks

The present paper is an invitation for the reader to consider numerical P systems, with inspiration from economics: variables evolve in the compartments of a cell-like membrane structure by means of production functions and repartition protocols. The work of such systems looks rather intricate and their mathematical investigation seems to request a serious effort; in particular, a large number of research topics are open in this area (some of them mentioned in the paper). Thus, we believe that numerical P systems deserve a systematic study, both from mathematical and economics points of view.

References

[1] J. Bartosik: Paun's systems in modeling of human resource management. *Second Conference on Tools and Methods of Data Transformation*, WSU Kielce, 2004.

[2] G. Ciobanu, Gh. Păun, M.J. Pérez-Jiménez, eds.: *Applications of Membrane Computing*. Springer-Verlag, Berlin, 2005.

[3] S. Ginsburg: *The Mathematical Theory of Context-Free Languages*. McGraw-Hill Book Comp., New York, 1966.

[4] W. Korczynski: On a model of economic systems. *Second Conference on Tools and Methods of Data Transformation*, WSU Kielce, 2004.

[5] W. Korczynski: Păun's systems and accounting. In *Pre-Proceedings of Sixth Workshop on Membrane Computing, WMC6*, Vienna, July 2005, 461–464.

[6] Y. Matijasevitch: *Hilbert's Tenth Problem*. MIT Press, Cambridge, London, 1993.

[7] Gh. Păun: *Membrane Computing – An Introduction*. Springer-Verlag, Berlin, 2002.

[8] Gh. Păun, R. Păun: Membrane computing as a framework for modelling economic processes. Submitted, 2005.

[9] G. Rozenberg, A. Salomaa: *The Mathematical Theory of L Systems*. Academic Press, New York, 1980.

[10] N.J.A. Sloane, S. Plouffe: *The Encyclopedia of Integer Sequences*. Academic Press, New York, 1995.

[11] The P Systems Web Page: `http://psystems.disco.unimib.it`.