A Kernel P System

Marian Gheorghe[1,2], Florentin Ipate[2], and Ciprian Dragomir[1]

[1] Department of Computer Science
University of Sheffield
Portobello Street, Regent Court, Sheffield, S1 4DP, UK
{m.gheorghe, c.dragomir}@sheffield.ac.uk

[2] Department of Computer Science
University of Pitesti
Str Targu din Vale, 1, Pitesti, Romania
florentin.ipate@ifsoft.ro

Summary. A basic P system, called kernel P system (kP system for short), covering features of different P systems introduced and studied so far is defined and discussed. It is a relatively low level specification system aiming to cover features exhibited by most of the problems modelled so far using P system formalisms. A small set of rules and specific strategies to run the system step by step are presented. Some preliminary results regarding the relationships between kP systems and other classes of P systems, like neural-like P systems and P systems with active membranes, are presented. Examples illustrating the behaviour of kP systems or showing how a sorting algorithm is modelled with various classes of P systems are provided. Further developments of this class of P systems are finally briefly discussed.

1 Introduction

Different variants of P systems have been used for specifying simple algorithms [4, 2], classes of NP-complete problems [7] and other various applications [5]. More specific classes of P systems have been recently considered for modelling various distributed algorithms and problems [9]. In many cases the specification of the system investigated requires features, constraints or type of behaviour which are not always provided by the model in its initial definition. It helps in many cases to have some flexibility with modelling approaches, especially in the early stages of modelling, as it might simplify the model, shorten associated processes and clarify more complex or unknown aspects of the system. The downside of this is the lack of a coherent and well-defined framework that allows us to analyse, verify and test this behaviour and simulate the system. In this respect we engage now on defining a *kernel P system* (*kP system*, for short) that, at least for this stage, will be a low level specification language including the most used concepts from P systems. In a later stage its key features will be formally defined in an operational style and finally implemented within a model checker (SPIN [3], Maude [6]) and integrated into the P-lingua platform.

We will be working with P systems having a graph-like structure (so called, *tissue P systems*) using a set of symbols, labels of membranes, rules of various types and various strategies to run them against the multiset of objects available in each region. The rules in each compartment will be of two types, (i) *object processing rules* which transform and transport objects between compartments or exchange objects between compartments and environment and (ii) *system structure rules* responsible for changing the system's topology. Each rule has a guard, defined using activators and inhibitors in a more general way than in traditional P system classes. An execution strategy can now be specified individually, for each compartment, allowing for more complex rule selection and iteration procedures in addition to the classical maximal parallelism and sequential methods. We consider rewriting and communication rules based on promoters and inhibitors as they seem to be amongst the most flexible and general processing rules, and a special set of symport/antiport rules; additional features like membrane division, dissolution, bond creation and destruction are also considered. Two types of P systems, neural-like P systems and P systems with active membranes, are simulated by the newly introduced P systems. We analyse a specific case study based on a sorting algorithm which is described using the currently introduced model, kP systems, and some other formalisms, using electrical charges, states and labels.

2 kP Systems

A kP system is a formal framework that uses some well-known features of existing P systems and includes some new elements and, more importantly, it offers a coherent view on integrating them into the same formalism. The key elements of a kP system will be formally defined in this section, namely objects, types of rules, internal structure of the system and strategies for running such systems. Some preliminary formal concepts describing the syntax of kP systems and an informal description of the way these systems are executed will be introduced.

We consider that standard concepts like strings, multisets, rewriting rules, and computation are well-known concepts in P systems and indicate [11] as a comprehensive source of information in this respect. First we introduce the key concept of a compartment.

Definition 1. *Given a finite set, A, called* alphabet*, of elements, called* objects*, and a finite set, L, of elements, called* labels*, a* compartment *is a tuple $C = (l, w_0, R^\sigma)$, where $l \in L$ is the label of the compartment, w_0 is the initial multiset over A and R^σ denotes the DNA code of C, which comprises the set of rules, denoted R, applied in this compartment and a regular expression, σ, over $Lab(R)$, the labels of the rules of R.*

The precise format and the types of rules used in this context will be discussed in Section 2.1.

Definition 2. *A* kernel P system *of degree n is a tuple*

$$k\Pi = (A, L, IO, \mu, C_1, \dots, C_n, i_0),$$

where A and L are, as in Definition 1, the alphabet *and the set of* labels*, respectively; IO is a multiset of objects from A, called* environment*; μ defines the membrane structure, which is a graph, (V, E), where V are vertices, $V \subseteq L$ (the nodes are labels of these compartments), and E edges; $C_1, \dots, C_n$ are the n compartments of the system - the inner part of each compartment is called* region*, which is delimited by a* membrane*; the labels of the compartments are from L and initial multisets are over A; i_o is the* output compartment *where the result is obtained.*

As usual in P systems, the environment contains an arbitrary number of copies of each object. Each compartment is specified according to Definition 1.

2.1 kP System Rules

The discussion below assumes that the rules introduced belong to the same compartment, C_i, labelled l_i.

Each rule r may have a **guard** g, in which case r is applicable when g is evaluated to true. Its generic form is $r\ \{g\}$. The guards are constructed according to certain criteria described below. Before presenting these criteria we introduce some notations.

We consider multisets over $A \cup \bar{A}$, where A and $\bar{A}$ are interpreted as **promoters** and **inhibitors**, respectively; $\bar{A} = \{\bar{a} | a \in A\}$. For a multiset w over $A \cup \bar{A}$ and an element a from the same set we denote by $\#_a(w)$ the number of a's occurring in w. We also consider the set of well-known relational operators $Rel = \{<, \leq, =, \neq, \geq, >\}$. For a multiset $w = a_1^{n_1} \dots a_k^{n_k}$, $a_j \in A \cup \bar{A}$, $1 \leq j \leq k$, and $\alpha_j \in Rel$, $1 \leq j \leq k$, we introduce the following notation $w' = \alpha_1 a_1^{n_1} \dots \alpha_k a_k^{n_k}$; a_j is not necessarily unique in w or w' (as it will transpire from the explanations below, this case may occur when the multiplicity of a symbol belongs to an interval); w' is called *multiset* over $A \cup \bar{A}$ with *relational operators* over *Rel*.

If g is a guard defined according to the criteria below and pr, a predicate over this set of guards, then:

- $g = \epsilon$ means $pr(\epsilon)$ is always *true*, i.e., no condition is associated with the rule r; this guard is almost always ignored from the syntax of the rule;
- g is a *multiset* over $A \cup \bar{A}$ with *relational operators* over *Rel*, i.e., $g = \alpha_1 a_1^{n_1} \dots \alpha_k a_k^{n_k}$, then $pr(w)$ is *true* iff for z, the current multiset of C_i, we have, for every $1 \leq j \leq k$, either (i) if $a_j \in A$ then $\#_{a_j}(z)\ \alpha_j\ n_j$ holds, or (ii) if $a_j \in \bar{A}$, i.e., $a_j = \bar{a}$, $a \in A$, then $\#_a(z)\ \alpha_j\ n_j$ does not hold;
- $g = w_1 | \dots | w_p$, i.e., g is a *finite disjunction* of *multisets* over $A \cup \bar{A}$ with *relational operators* over *Rel*, then $pr(w_1 | \dots | w_p)$ is *true* iff there exists $1 \leq j \leq p$, such that $pr(w_j)$ is *true*.

We denote by $FE(A \cup \bar{A})$, from Finite regular Expressions over $A \cup \bar{A}$ with relational operators, the set of expressions defined above. When a compound guard, cg, referring to compartments l_i and l_j is used, its generic format is $cg = l_i.g_1 \; op \; l_j.g_2$, where g_1, g_2 are finite expressions referring to compartments l_i and l_j, respectively; then, obviously, $pr(cg) = pr(g_1) \; op \; pr(g_2)$, $op \in \{\&, |\}$, where & stands for *and* and | for *or*, meaning that either both guards are true or at least one is true. Simpler forms, where one of the operands is missing, are also allowed as well as $cg = \epsilon$. A compound guard defines a Boolean condition defined across the two compartments.

Example 1. If the rule is $r : ab \to c \; \{\leq a^3 \geq b^7 = \bar{c}\}$, then this can be applied iff the current multiset consists of at most 3 a's and at least 7 b's and does not contain a single c (either none or more than 2 c's are allowed).

A rule can have one the following types:

- (a) **rewriting and communication** rule: $x \to y \; \{g\}$,
  where $x \in A^+$, $y \in A^*$, $g \in FE(A \cup \bar{A})$; the right hand side, y, has the form $y = (a_1, t_1) \dots (a_h, t_h)$, where $a_j \in A$ and $t_j \in L$, $1 \leq j \leq h$, is an object and a target, i.e., the label of a compartment, respectively; the target, t_j, must be either the label of the current compartment, l_i, (more often ignored) or of an existing neighbour of it ($(l_i, t_j) \in E$) or an unspecified one, $*$; otherwise the rule is not applicable; if a target, t_j, refers to a label that appears more than once then one of the involved compartments will be non-deterministically chosen; if t_j is $*$ then the object a_j is sent to a neighbouring compartment arbitrarily chosen;
- (b) **input-output** rule, is a form of symport/antiport rule: $(x/y) \; \{g\}$,
  where $x, y \in A^*$, $g \in FE(A \cup \bar{A})$; x from the current region, l_i, is sent to the environment and y from the environment is brought into the current region;
- (c) **system structure rules**; the following types are considered:
  - (c1) **membrane division** rule: $[]_{l_i} \to []_{l_{i_1}} \dots []_{l_{i_h}} \; \{g\}$,
    where $g \in FE(A \cup \bar{A})$; the compartment l_i will be replaced by h compartments obtained from l_i, i.e., the content of them will coincide with that of l_i; their labels are $l_{i_1}, \dots, l_{i_h}$, respectively; all the links of l_i are inherited by each of the newly created compartments;
  - (c2) **membrane dissolution** rule: $[]_{l_i} \to \lambda \; \{g\}$;
    the compartment l_i will be destroyed together with its links;
  - (c3) **link creation** rule: $[]_{l_i}; []_{l_j} \to []_{l_i} - []_{l_j} \; \{cg\}$;
    the current compartment, l_i, is linked to l_j and if more than one l_j exists then one of them will be non-deterministically picked up; cg, called compound guard, describes an expression $l_i.g_1 \; op \; l_j.g_2$ as defined above;
  - (c4) **link destruction** rule: $[]_{l_i} - []_{l_j} \to []_{l_i}; []_{l_j} \; \{cg\}$;
    is the opposite of link creation and means that compartments l_i, l_j are disconnected; as usual, when more than a link, $(l_i, l_j) \in E$, exists then only one is considered by this rule; cg is a compound guard.

2.2 Regular Expressions and their Interpretation for kP Systems

In kP Systems the way in which rules are executed is described using regular expressions (over the sets of labels of rules). This approach allows the usual behaviour of P systems - requiring the rewriting and communication, and input-output rules to be applied in a maximal parallel way and structural rules (e.g. membrane division and dissolution, creation and destruction of links) to be executed one per membrane - as well as other alternative or additional features to be expressed in a consistent and elegant manner.

We first consider the set of labels of the rules, from the set R, in a given compartment, denoted by $Lab(R)$. We can define regular expressions over this set, $REG(Lab(R))$. A regular expression $\sigma \in REG(Lab(R))$ is interpreted as follows

- $\sigma = \epsilon$ means no rule from the current compartment will be executed;
- $\sigma = r$, $r \in Lab(R)$, means the rule r is executed;
- $\sigma = \alpha\beta$ means first are executed rules designed by α and then those in β;
- $\sigma = \alpha|\beta$ means either the rules designed by α or those by β are executed; often we use the notation defining sets where $|$ is replaced by ,;
- $\sigma = \gamma^*$ means rules designed by γ are executed in a maximal parallel way.

Regular expressions allows the definition of various execution strategies, including well-known maximal parallelism (and also sequential) behaviour, but also to encode more subtle concepts like order relationships between rules, which introduce a form of sequential execution. Given the above introduced types of rules we can also specify in a more coherent way the fact that maximal parallelism imposes some constraints on the way the rules dealing with the system structure are handled; it is always the case that such rules are applied one per compartment and at the end of each step. Indeed, this assumption can be made in this case as the left hand side of any of the rules c1–c4 , does not contain any object, so they are applied only when there guards are satisfied. These cases are briefly analysed below.

- Naturally, * is used to capture the maximal parallelism of a set of rules; for rules R, with $Lab(R) = \{r_1, \ldots, r_k\}$, we mostly write either $Lab(R)^*$ or $\{r_1, \ldots, r_k\}^*$, instead of $(r_1| \ldots |r_k)^*$.
- In order to express the fact that *maximal parallelism* means that *object processing rules are applied in a maximal parallel way and at the end only one of the system structure rules is applied*, we first split R, the set of rules, into R_1, containing all the object processing rules, and R_2, with all the structure defining rules, associated with the current compartment; then given the convention introduced for the set of regular expressions over $Lab(R)$, the above behaviour is expressed by $Lab(R_1)^*Lab(R_2)$.
- Now suppose that a certain order relationship exists, e.g. $r_1, r_2 > r_3, r_4$, which means that when weak priority is applied, the first two rules are executed first, if possible, then the next two. If both are executed with maximal parallelism, this is described by $\{r_1, r_2\}^*$ $\{r_3, r_4\}^*$.

These regular expressions define the strategy of executing the rules of the current compartment and together with them they form the true *DNA code* of each compartment. Other ways of executing the rules of a compartment, like equal to, less than or greater than a given number of steps, can be also considered.

The result of a computation will be the number of objects collected in the output compartment. For a kP systems $k\Pi$, the set of all these numbers will be denoted by $M(k\Pi)$.

2.3 kP System Examples

In this section we illustrate the newly introduced P system model with some examples.

Example 2. Let us consider the following kP system with $n = 4$ compartments, $k\Pi_1 = (A, L, IO, \mu, C_1, \dots, C_4, 1)$, where
$A = \{a, b, c, p\}$,
$L = \{1, 2, 3\}$,
IO contains an arbitrary number of objects over $\{b, c\}$,
$C_1 = (1, w_{1,0}, R_1^\sigma), C_2 = (2, w_{2,0}, R_2^\sigma), C_3 = (2, w_{3,0}, R_2^\sigma), C_4 = (3, w_{4,0}, R_3^\sigma)$,
μ is given by the following graph with edges $(1, 2), (1, 3)$; $(1, 2)$ appears twice as $n = 4$ and there are two compartments, C_2, C_3, with label 2;
$w_{1,0} = a^3p$, $w_{2,0} = p$, $w_{3,0} = p$, $w_{4,0} = \lambda$, and
R_1^σ is $R_1 = \{r_1 : a \to a(b, 2)(c, 3)\ \{\geq p\}; r_2 : p \to p; r_3 : p \to \lambda\}$, and $\sigma_1 = Lab(R_1)^*$,
R_2^σ is $R_2 = \{r_1 : (b/c)\ \{\geq p\}; r_2 : p \to p; r_3 : p \to \lambda\}$, and $\sigma_2 = Lab(R_2)^*$,
R_3^σ is $R_3 = \emptyset$ and $\sigma_3 = Lab(R_3)^*$.

Please note that we do not use targets for objects meant to stay in the current compartment (i.e. we have $r_1 : a \to a(b, 2)(c, 3)\ \{\geq p\}$ instead of $r_1 : a \to (a, 1)(b, 2)(c, 3)\ \{\geq p\}$).

In this example there are only rewriting and communication rules (all the rules, but r_1 from R_2) and an input-output one (r_1 from R_2); some rules have a guard, $\geq p$ (p is a promoter), others do not have any and in each compartment the rules are applied in maximal parallel way in every step, as indicated by σ_j, $1 \leq j \leq 3$. As two instances of the compartment labelled 2, C_2, C_3, appear in the system, when the rule r_1 from the first compartment is applied, the object b goes non-deterministically to one of the two compartments labelled 2 as long as p remains in compartment 1; object c goes always to compartment labelled 3, C_4.

The initial configuration of $k\Pi_1$ is $M_0 = (a^3p, p, p, \lambda)$. The only applicable rules are r_1, r_2 and r_3 from C_1 and r_2, r_3 from C_2, C_3. If r_1, r_2 are chosen in C_1 and r_2 in C_2, C_3, then a^3p is rewritten by r_1, r_2 in C_1 and p in C_2, C_3 by r_2; then three a's stay in C_1, three b's go non-deterministically to C_2, C_3, three c's go to compartment labelled 3, C_4, and each p in C_2, C_3 stays in its compartment. Let us assume that two of them go to C_2 and one to C_3. Hence, the next configuration is $M_1 = (a^3p, b^2p, bp, c^3)$. If in the next step the same rules are applied identically in the first compartment, C_1, and rules r_1, r_2 are used in C_2 and r_1, r_3 in C_3, then the next configuration is $M_2 = (a^3p, b^2c^2p, bc, c^6)$. If now r_1, r_3 are used in C_1, with r_1 used in the same way and r_1, r_3 in C_2 (no rule is available in C_3) then $M_3 = (a^3, b^2c^4, b^2c, c^9)$; this is a final configuration as there is no promoter to trigger a further step.

Observation. If $r_1 : a \to a(b,2)(c,3)\ \{\geq p\}$ from C_1 is changed to $r_1 : a \to a(b,2)(c,*)\ \{\geq p\}$, then the three resulting c's (obtained after applying r_1 to a^3) go non-deterministically to any of the three neighbours of C_1.

Example 3. Let us reconsider the example above enriched with rules dealing with the system structure. First we will show how the system handles the multiplication of compartments with label 2, C_2 and C_3, when a certain condition holds; we will consider the guard $\geq b^2 \geq p$. In this case the new kP system, denoted $k\Pi_2$, will have the same structure and content as $k\Pi_1$ except R_2^{σ} which is now defined as follows
R_2^{σ} is $R_2 = R_2^{(1)} \cup R_2^{(2)}$, where $R_2^{(1)} = \{r_1 : (b/c)\ \{\geq p\}; r_2 : p \to p; r_3 : p \to \lambda\}$, $R_2^{(2)} = \{r_4 : []_2 \to []_2[]_2\ \{\geq b^2 \geq p\}\}$ and $\sigma_2 = Lab(R_2^{(1)})^* Lab(R_2^{(2)})$.
We can notice that the regular expression σ_2 tells us that first the rewriting rules are applied in a maximal parallel manner and then one of system structure rules is chosen to be executed.

If the system follows the same pathway as $k\Pi_1$ then M_2 shows a different configuration given that in C_2 after applying $R_2^{(1)}$ in a maximal parallel manner, $R_2^{(2)}$ is applied as indicated by σ_2, when the guard of r_4 is true. The compartment C_2 is divided into two compartments, $C_{2,1}, C_{2,2}$, with the same label 2 and appearing on positions 2 and 3 in the new configuration, $M_2' = (a^3p, b^2c^2p, b^2c^2p, bc, c^6)$; the new compartments labelled 2 are linked to compartment C_1. In the next step both are divided as they contain the guard triggering the membrane division rule r_4. The process will stop when either p will be rewritten to λ or b^2 stops coming.

If we aim to either dissolve or disconnect a compartment labelled by 2 from compartment C_1, once a certain condition is true, for instance b^2c^2p appears in it, then two more rules will be added to $R_2^{(2)}$, namely
$r_5 : []_2 \to \lambda\ \{\geq b^2 \geq c^2 \geq p\}$, $r_6 : []_2 - []_1 \to []_2; []_1\ \{\geq b^2 \geq c^2 \geq p\}$. The same regular expression, $\sigma_2 = Lab(R_2^{(1)})^* Lab(R_2^{(2)})$, is used, but in this case $R_2^{(2)}$ contains three elements and at most one is applied at each step, in every compartment with label 2.

3 Neural-like P Systems and P Systems with Active Membranes versus kP Systems

In order to prove how powerful and expressive kP systems are, we will show how two of the most used variants of P systems are simulated by kP system. More precisely, we will show how neural-like P systems and P systems with active membranes are simulated by some reduced versions of kP systems.

Definition 3. *A* neural-like P system *(tissue P system with states) of degree n is a construct $\Pi = (O, \sigma_1, ...\sigma_n, syn, i_0)$ ([10], p. 249), where:*

- *O is a finite, non-empty set of objects, the* alphabet*;*
- *$\sigma_i = (Q_i, s_{i,0}, w_{i,0}, R_i)$, $1 \leq i \leq n$, represents a* cell *and*
  - *Q_i is the finite set of* states *of cell σ_i;*
  - *$s_{i,0} \in Q_i$ is the* initial state*;*
  - *$w_{i,0} \in O*$ is the* initial multiset *of objects contained in cell σ_i;*
  - *R_i is a finite set of* rewriting and communication rules*, of the form $sw \to s'xy_{go}z_{out}$; when such a rule is applied, x will replace w in cell σ_i, the objects from y will be sent to neighbouring cells, according to the transmission mode (see th next Observation) and the objects from z will be sent out into the environment; cell σ_i will move from state s to s';*
- *$syn \subseteq \{1, ..., n\} \times \{1, ..., n\}$, the connections between cells,* synapses*;*
- *i_0 is the output cell.*

Observations.

1. For such systems, three processing modes are considered, called "max", "min", "par", and three transmission modes, namely "one", "repl", "spread". For formal definitions and other details we refer to [10].
2. We denote by *simple neural-like P systems* the class of P systems given by Definition 3, with rewriting and communication rules $sw \to s'x(a_1, t_1) \cdots (a_p, t_p)$, where t_h, $1 \leq h \leq p$, denotes the target cell (σ_h), and processing mode "max", transmission mode defined by the target indications mentioned in each rule.

Notation. For a given P system, Π, the set of numbers computed by Π will be denoted by $M(\Pi)$.

Theorem 1. *If Π is neural-like P system of degree n, then there is a kP system, Π', of degree n and using only rules of type (a), rewriting and communication rules, simulating Π and such that $M(\Pi') \subseteq M(\Pi) \cup \{2\}$.*

Proof. Let Π be a simple neural-like P systems of degree n, as defined above. We construct Π' as follows:
$\Pi' = (A, L, IO, \mu, C_1, ..., C_n)$ where:

- $A = O \cup (\bigcup_{1 \leq i \leq n} Q_i) \cup \{\gamma\}$; γ is a new symbol neither in O nor in $\bigcup_{1 \leq i \leq n} Q_i$;
- $L = \{1, ..., n\}$; $IO = \emptyset$;
- $\mu = syn$;
- $C_i = (i, w'_{i,0}, R'^{\sigma}_i), 1 \leq i \leq n$; and
  - $w'_{i,0} = \gamma, 1 \leq i \leq n$;
  - R'_i contains the following rules:
    1. $\gamma \to s_{i,0}tw_{i,0}$, where $s_{i,0}, w_{i,0}$ are the initial state and initial multiset, respectively, associated with cell σ_i, and $t \in Q^{(i)}_{s_{i,0}}$. For $s \in Q_i$, denote by $Q^{(i)}_s = \{t | t \in Q_i, sx \to ty \in R_i\}$; i.e., $Q^{(i)}_s$ gives, when the cell σ_i is in state s, all the states where σ_i can move to. In the first step, in

compartment C_i, a rule $\gamma \rightarrow s_{i,0}tw_{i,0}$ is applied and the current multiset becomes $w'_i = s_{i,0}w_{i,0}$.

2. For each pair $(s,t), t \in Q_s^{(i)}$, there are rules
$sx_i \rightarrow ty_i \in R_i, 1 \leq i \leq p \quad (*)$.
If there are no rules in R_i from s to t then another pair is considered. For the above rules, the following rules are considered in R'_i:
$x_i \rightarrow y_i$ $\{= s = t\}$, $1 \leq i \leq p$, and $st \rightarrow tq$ $\{\geq x_1|...| \geq x_p\}$, $q \in Q_t^{(i)}$ $(**)$
In the above guards the notation $\geq x_i$, if $x_i = a_{i,1} \dots a_{i,l_i}$, denotes $\geq a_{i,1} \dots \geq a_{i,l_i}$. The rules $(**)$ make use of guards; the first p rules are applied iff the current multiset contains one s and one t, whereas the last one is applicable iff at least one or more of the occurrences of one of the multisets x_i, $1 \leq i \leq p$, is included in the current multiset. Clearly, in state s only the rules $(*)$ of Π are applicable for this P system, depending on the availability of the multisets occurring on the left hand side of them; the next state Π is moving to is t. Similarly, in Π' only the rules denoted by $(**)$ are applicable; the rule $st \rightarrow tq$ $\{\geq x_1|...| \geq x_p\}$ is applied once whereas the first p rules are applied as many times as their corresponding $(*)$ rules are applied.

If the set $Q_t^{(i)}$ used in $st \rightarrow tq$ $\{\geq x_1|...| \geq x_p\}$ of $(**)$ is empty, i.e., there are no rules from state t, then the rule is replaced by $st \rightarrow \lambda$. When $Q_{s_{i,0}}^{(i)} = \emptyset$ then the rule $\gamma \rightarrow w_{i,0}$ is introduced in R'_i.

At any moment the component C_i of the kP system Π' contains a multiset which is the multiset of σ_i augmented by the current state of σ_i, s, and one of the next states, t, if it exists.

The process will stop in component C_i of Π' when no pair of rules of type $(**)$ is applicable, which means no $sx_i \rightarrow ty_i$ rule is applicable in state s.

The multiset $M(\Pi')$ contains $M(\Pi)$ and maybe two states s, t occurring in the last step of the computation. Hence $M(\Pi') \subseteq M(\Pi) \cup \{2\}$. $\square$

Comments.

1. The above simulation can be assessed with respect to number of compartments, objects and rules as well as the computation steps.
2. When rules $sw \rightarrow txy_{go}$ are used in the "spread" mode, this means that any $a \in O$ occurring in y may go to any of the neighbours. In this case if $y = y_1ay_2$ then for each such $a \in O$, the rules of R'_i corresponding to $sw \rightarrow txy_{go}$, denoted $(**)$ above, will show $w \rightarrow xy$ $\{= s = t\}$ replaced by $w \rightarrow xy_1(a,j)y_2$ $\{= s = t\}$, where j the label of one of the neighbours of the current compartment. For "one" mode all a's in y will point to the same target, j, for all neighbours of the compartment i.
3. The transmission replicative mode - when a symbol is sent to all the neighbours, can also be simulated. Indeed if $j_1, \cdots, j_h$ are the neighbours of i, then $w \rightarrow xy_1(a,j)y_2$ is transformed into $w \rightarrow xy_1(a,j_1) \cdots (a,j_h)y_2$ for each a.

4. If a rewriting rule contains z_{out} on its right side, i.e., $sw \to txy_{go}x_{out}$ then in the set of rules transcribing it, $w \to \alpha$, we will have $\alpha = xy_1(a,j)y_2z'$, where if $z = a_1 \cdots a_k$, then $z' = a'_1 \cdots a'_k$; also rules (a'/λ) will be added to R'_i, for any $a \in O$. In this way in the next step all the prime elements are sent out into the environment. If there is a need to synchronize the behaviour of the system with the environment then this should be done a bit differently. For this transmission mode the kP system is a bit more complex as it must contain input-output rules and the environment definition needs to be considered.
5. If we want to simulate the "min" processing mode then this can be obtained by specifying the sequential behaviour of the component i - by changing the regular expression of the component.

We study now how P systems with active membranes are simulated by kP systems. In this case we are dealing with a cell-like system, so the underlying struture is a tree and a set of labels (types) for the compartments of the system. The system will start with a number of compartment and its structure will evolve. In the study below it will be assumed that the number of compartments simultaneously present in the system is bounded.

Definition 4. *A P system with active membranes of initial degree n is a tuple (see [11], Chapter 11) $\Pi = (O, H, \mu, w_{1,0}, \dots, w_{n,0}, R, i_0)$ where:*

- *O, $w_{1,0}, \dots, w_{n,0}$ and i_0 are as in Definition 3;*
- *H is the set of labels for compartments;*
- *μ defines the tree structure associated with the system;*
- *R consists of rules of the following types*
  - *(a)* rewriting rules*: $[u \to v]_h^e$, for $h \in H$, $e \in \{+,-,0\}$ (set of electrical charges), $u \in O^+$, $v \in O^*$;*
  - *(b)* in communication rules*: $u[]_h^{e_1} \to [v]_h^{e_2}$, for $h \in H$, $e_1, e_2 \in \{+,-,0\}$, $u \in O^+$, $v \in O^*$;*
  - *(c)* out communication rules*: $[u]_h^{e_1} \to []_h^{e_2}v$, for $h \in H$, $e_1, e_2 \in \{+,-,0\}$, $u \in O^+$, $v \in O^*$;*
  - *(d)* dissolution rules*: $[u]_h^e \to v$, for $h \in H \backslash \{s\}$, s denotes the skin membrane (the outmost one), $e \in \{+,-,0\}$, $u \in O^+$, $v \in O^*$;*
  - *(e)* division rules for elementary membranes*: $[u]_h^{e_1} \to [v]_h^{e_2}[w]_h^{e_3}$, for $h \in H$, $e_1, e_2, e_3 \in \{+,-,0\}$, $u \in O^+$, $v, w \in O^*$;*

The following result shows how a P system with active membranes starting with n_1 compartments and having no more than n_2 simultaneously present ones can be simulated by a kP systems using only rules of type (a).

Theorem 2. *If Π is a P system with active membrane having n_1 initial compartments and utilising no more than n_2 compartments at any time, then there is a kP system, Π', of degree 1 and using only rules of type (a), rewriting and communication rules, such that Π' simulates Π.*

Proof. Let us denote $J_0 = \{(i,h)|1 \leq i \leq n_2, h \in H\}$; for a multiset $w = a_1 \dots a_m$, (w,i,h), $(i,h) \in J_0$, denotes $(a_1,i,h)\dots(a_m,i,h)$. Let us consider the P system with active membranes, $\Pi = (O, H, \mu, w_{1,0}, \dots, w_{n_1,0}, R, i_0)$. The polarizations of the n_1 compartments are all 0, i.e., $e_1 = \dots = e_{n_1} = 0$.

We construct Π' as follows:
$\Pi' = (A, L, IO, \mu', C_1)$ where:

- $A = \bigcup_{(i,h)\in J_0}\{(a,i,h)|a \in O \cup \{+,-,0\} \cup \{\delta\}\}$, where δ is a new symbol; let us denote by $= \overline{\delta_{all}}$ the guard $= \overline{(\delta,1,1)} \dots = \overline{(\delta,n_2,|H|)}$, $|H|$ is the number of elements in H ($= \overline{\delta_{all}}$ stands for none of the (δ,i,h), $(i,h) \in J_0$);
- $L = \{1\}$; $IO = \emptyset$; $\mu' = [\,]_1$;
- $C_1 = (1, w'_{1,0}, R_1'^{\sigma})$, and
  - $w'_{1,0} = (w_{1,0},1,h_1)\dots(w_{n_1,0},n_1,h_{n_1})(e_1,1,h_1)\dots(e_{n_1},n_1,h_{n_1})$, $e_1 = \dots = e_{n_1} = 0$; let $J_c = J_0 \setminus \{(i,h_i)|1 \leq i \leq n_1\}$ (J_c denotes indexes available for new compartments and $J_0 \setminus J_c$ the set of indexes of the current compartments);
  - R'_1 contains the following rules
    - (a') for each $h \in H$ and each rule $[u \to v]_h^e \in R$, $e \in \{+,-,0\}$, we add the rules $(u,i,h) \to (v,i,h)$ $\{= (e,i,h) = \overline{\delta_{all}}\}$, $1 \leq i \leq n_2$; these rules are applied to every multiset containing elements with $h \in H$, only when the polarization (e,i,h) appears and none of the (δ,j,h') appears;
    - (b') for each $h \in H$ and each rule $u[\,]_h^{e_1} \to [v]_h^{e_2} \in R$, $e_1, e_2 \in \{+,-,0\}$, we add the rules $(u,j,l)(e_1,i,h) \to (v,i,h)(e_2,i,h)$ $\{= \overline{\delta_{all}}\}$, $1 \leq i \leq n_2$, j is the parent of i of label l; these rules will transform (u,j,l) corresponding to u from the parent compartment j to (v,i,h) corresponding to v from compartment i of label h, the polarization is changed; for each polarization, (e_1,i,h) only one single rule can be applied at any moment of the computation;
    - (c') for each $h \in H$ and each rule $[u]_h^{e_1} \to [\,]_h^{e_2}v \in R$, $e_1, e_2 \in \{+,-,0\}$, we add the rules $(u,i,h)(e_1,i,h) \to (v,j,l)(e_2,i,h)$ $\{= \overline{\delta_{all}}\}$, $1 \leq i \leq n_2$, j is the parent of i of label l;
    - (d') for each $h \in H$ and each rule $[u]_h^e \to v \in R$, $e \in \{+,-,0\}$, we add the rules $(u,i,h)(e,i,h) \to (v,j,l)(\delta,i,h)$ $\{= \overline{\delta_{all}}\}$, $1 \leq i \leq n_2$, j is the parent of i of label l; all the elements corresponding to those in compartment i must be moved to j - this will happen in the presence of (δ,i,h) when no other transformation will take place; this is obtained by using rules $(a,i,h) \to (a,j,l)$ $\{= (\delta,i,h)\}$, $a \in O$ and $(\delta,i,h) \to \lambda$ $\{= (\delta,i,h)\}$; the set of available indexes will change now to $J_c = J_c \cup \{(i,h)\}$;
    - (e') for each $h \in H$ and each rule $[u]_h^{e_1} \to [v]_h^{e_2}[w]_h^{e_3} \in R$, $e_1, e_2, e_3 \in \{+,-,0\}$; if j_1, j_2 are the indexes of the new compartments, we add $(u,i,h)(e_1,i,h) \to (v,j_1,k_1)(e_2,j_1,k_1)(w,j_2,k_2)(e_3,j_2,k_2)(\delta,i,h)$ $\{= \overline{\delta_{all}}\}$, $1 \leq i \leq n_2$; the content corresponding to compartment i should be moved to j_1 and j_2, hence rules $(a,i,h) \to (a,j_1,k_1)(a,j_2,k_2$ $\{= (\delta,i,h)\}$, $a \in O$ and finally $(\delta,i,h) \to \lambda$ $\{= (\delta,i,h)\}$; J_c is updated, $J_c = J_c \cup \{(i,h)\} \setminus \{(j_1,k_1),(j_2,k_2)\}$.

The size of the multiset obtained in i_0 by using Π computation is the same as the size of the multiset in Π, when only (a, i_0, h) are considered, minus 1 (the polarization is also included). $\square$

4 Case Study - Static Sorting

In this section we analyze the newly introduced kP systems by comparing them with established P system classes by using them to specify a static sorting algorithm. This algorithm was first written with symport/antiport rules [4] and then reconsidered in some other cases [2]. The specification below mimics this algorithm.

4.1 Static Sorting with kP Systems

Let us consider a kP system having the following $n = 6$ compartments:
$C_i = (i, w_{i,0}, R_i^\sigma)$, $1 \leq i \leq n$, where
$w_{1,0} = a^3; w_{2,0} = a^6 p; w_{3,0} = a^9; w_{4,0} = a^5 p; w_{5,0} = a^7; w_{6,0} = a^8 p.$

The rules in compartment R_i, $1 \leq i \leq n$, are :
$r_1 : a \rightarrow (b, i-1)\ \{\geq p\}$, only for $i > 1$
$r_2 : p \rightarrow p'$
$r_3 : p' \rightarrow (p, i-1)$, for i even and $r_3' : p' \rightarrow (p, i+1)$, for i odd
$r_4 : ab \rightarrow a(a, i+1)$, $i < n$
$r_5 : b \rightarrow a$, $i < n$.

We assume that any two compartments, C_i, C_{i+1}, $1 \leq i < n$, are connected. The aim of this problem is to order the content of these compartments such that the highest element (a^9) will be in the left most compartment, C_1, and the smallest one (a^3) in the right most compartment, C_n, $(n = 6)$.

Remarks:

- the set of objects is $A = \{a, b, p, p'\}$;
- compartment C_i has the label i, $1 \leq i \leq n$; so any two compartments have distinct labels;
- the rule r_1 is absent from the compartment C_1;
- the last two rules, r_4, r_5, are only present in compartments C_1 to C_{n-1};
- for $n = 2k+1$ we need an auxiliary compartment, C_{n+1}, which will start with an initial multiset p and will contain a set of rules with $r_2 : p \rightarrow p'$ and $r_3 : p' \rightarrow (p, n)$; whereas C_n should have an additional rule $r_3' : p' \rightarrow (p, n+1)$;
- the regular expression corresponding to the execution of the rules in a compartment C_i is $\sigma_i = \{r_1, r_2, r_3, r_4\}^* \{r_5\}^*$, if i is even; for odd values of i, r_3 is replaced by r_3'; the regular expression tells us that firstly the rules from the first set are applied in a maximal parallel manner and then r_5, also in a maximal way.

Observation. The regular expression, σ_i, describes an order relationship, $r_1, r_2, r_3, r_4 > r_5$. So we can replace this kP system by a P system with promoters and having an order relationships on the set of rules associated with each membrane.

The table below presents the first steps of the computation. In the first step the only applicable rules are r_1, r_2; given the presence of the promoter p, rule r_1 moves all a's from each even compartment to the left compartment as b's and rule r_2 transforms the promoter into p'. Next, rules r_3, r_4, r_5 are applicable; first r_3 and r_4 are applied, this means p' is moved as p to the left compartments and for each ab an a is kept in the current compartment and a b is moved as an a to the right compartment; finally, the remaining b's, if any, are transformed into a's. These two steps implement a sort of comparators between two adjacent compartments moving to the left bigger elements. In the previous steps the comparators have been considered between odd and even compartments. In the next step the promoters appear in even compartments and the comparators are now acting between an even and an odd compartment. The algorithm does not have a stopping condition. It must stop when no changes appear in two consecutive steps. Given that the algorithm must stop in maximum $2(n-1)$ steps, then we can introduce such a counter, c, in each compartment and rules $c \rightarrow c_1$, $c_i \rightarrow c_{i+1}$, $1 \leq i \leq 2(n-1)-2$ and $c_{2(n-1)-1}p \rightarrow \lambda$. These rules should be executed before the rest, so the regular expression associated with them should be a prefix of the regular set associated with each compartment.

Compartments - Step	C_1	C_2	C_3	C_4	C_5	C_6
0	a^3	a^6p	a^9	a^5p	a^7	a^8p
1	a^3b^6	p'	a^9b^5	p'	a^7b^8	p'
2	a^6p	a^3	a^9p	a^5	a^8p	a^7
3	a^6p'	a^3b^9	p'	a^5b^8	p'	a^7
4	a^6	a^9p	a^3	a^8p	a^5	a^7p
5	a^6b^9	p'	a^3b^8	p'	a^5b^7	p'

Observation. Bounded number of labels! The above solution is using n labels for n compartments. As the rules are the same in each compartment, with two exceptions involving the components at both ends of the system (compartments C_1 and C_n), it is natural to look for a solutions with a bounded number of labels. If we use the same label everywhere except for the two margins then we face the problem of replacing the rules using targets with different rules where the targets are now the new labels; if these are the same we can no longer distinguish between left and right neighbours, so we should have at least two distinct ones. Additionally, we have to distinguish odd and even positions. Consequently, four labels, and two more for the two ends are enough. Are there further simplifications? The answer to this question and the solution in this case are left as exercises to the reader.

4.2 Static Sorting with States

We consider the same n-compartment tissue-like P system structure as in the previous subsection. Additionally, in this case, the rules in each compartment use states; an order relationship between rules in each compartment is also considered. Initial states are s_1 in odd compartments and s_0 otherwise; the content of the 6 regions is illustrated by the first line, step 0, of the table below.

The addition of states is potentially very useful from a modelling point of view since many widely-used modelling languages are state-based and, therefore, such rules were a strong candidate for inclusion in our kP system model. However, as shown below, states can be effectively simulated by rewriting rules, as shown below .

For the algorithm considered, the rules in each compartment and the order relationships are as follows
Compartment 1:
$r_1 : s_0x \to s_0y$
$r_2 : s_0y \to s_1x$
$r_3 : s_1ab \to s_0a(a, 2)$
$r_4 : s_1b \to s_0a$
The rules satisfy: $r_1, r_2, r_3 > r_4$.

Compartment i, $2 \leq i \leq n - 1$:
$r_1 : s_0a \to s_1(b, i - 1)$
$r_3 : s_1ab \to s_0a(a, i + 1)$
$r_4 : s_1b \to s_0a$
The rules satisfy: $r_1, r_3 > r_4$.

Compartment n:
$r_1 : s^0a \to s^1b_{n-1}$
$r_2 : s^1x \to s^1y$
$r_3 : s^1y \to s^1z$
$r_4 : s^1z \to s^0x$

Membranes - Step	C_1	C_2	C_3	C_4	C_5	C_6
0	$s^1 : a^3x$	$s^0 : a^6$	$s^1 : a^9$	$s^0 : a^5$	$s^1 : a^7$	$s^0 : a^8x$
1	$s^1 : a^3b^6x$	$s^1 : \_$	$s^1 : a^9b^5$	$s^1 : \_$	$s^1 : a^7b^8$	$s^1 : x$
2	$s^0 : a^6x$	$s^1 : a^3$	$s^0 : a^9$	$s^1 : a^5$	$s^0 : a^8$	$s^1 : a^7y$
3	$s^0 : a^6y$	$s^1 : a^3b^9$	$s^1 : \_$	$s^1 : a^5b^8$	$s^1 : \_$	$s^1 : a^7z$
4	$s^1 : a^6x$	$s^0 : a^9$	$s^1 : a^3$	$s^0 : a^8$	$s^1 : a^5$	$s^0 : a^7x$
5	$s^1 : a^6b^9$	$s^1 : \_$	$s^1 : a^3b^8$	$s^1 : \_$	$s^1 : a^5b^7$	$s^1 : x$

In the case where we have an odd number of compartments, the $n-$th region must contain an y instead of x. Thus the starting configuration for $n = 7$ is the

following:
$w_{1,0} = a^3x; w_{2,0} = a^6; w_{3,0} = a^9; w_{4,0} = a^5; w_{5,0} = a^7; w_{6,0} = a^8; w_{7,0} = a^{13}y$.

4.3 Static Sorting with P Systems using Polarizations on Membranes

We now use cell-like P systems with active membranes to specify the same algorithm. P systems with active membranes were introduced with the primary aim of solving NP-complete problems in polynomial (often linear) time [11]. The key features of this variant is the possibility of multiplying the number of compartments during the computation process by using membrane division rules in addition to multiset rewriting and communication rules. Each membrane can have one of the three electrical charges $\{+, -, 0\}$ and a rule can only be executed if the membrane has the required electrical charge; a rule can also change the polarization of the membrane when objects cross it (either in or out).

In our static sorting example compartments with two states were used, so, when the algorithm is implemented using electrical charges, it is expected that two electrical charges would suffice. Indeed, from the list of rules below one may observe that 0 and + are the only polarizations utilised.

There is, however, a problem with this approach, arising from the rule application strategy. In P systems with membrane division and polarizations, only one rule which can change the polarization of a membrane can be applied per step [7]. The sorting algorithm however, employs maximal parallel communication rules to operate the *comparator* procedure between membranes. In order to correctly implement this procedure we will accept maximal parallel communication rules which change the charge of the membrane they traverse to/from *if and only if* they target the same final polarization.

In the case of P systems with polarizations on membranes we will use a cell-like structure with $n = 6$ regions defined below with the initial multisets included and initial polarizations; the implementation of the static sorting with P systems with polarization on membranes is using priorities over the sets of rules.

$$\mu = [[[[[[[a^3x_1]_1^0 a^6x_1]_2^+ a^9x_1]_3^0 a^5x_1]_4^+ a^7x_1]_5^0 a^8x_1]_6^+]_{aux}^0$$

Rules:
"Comparator" rules:
$r_1 : a[]_j^0 \to [b]_j^0, 1 \leq j \leq n;$
$r_2 : [ab]_j^0 \to a[a]_j^+, 1 \leq j \leq n;$
$r_3 : [b \to a]_j^0, 1 \leq j \leq n;$

Rules for switching polarities between adjacent membranes:
$r_4 : [x_1 \to x_2]_j^i, 1 \leq j \leq n;$
$r_5 : [x_2]_j^0 \to y_1[]_j^+, 1 \leq j \leq n;$
$r_6 : [x_2]_j^+ \to y_1[]_j^0, 1 \leq j \leq n;$
$r_7 : [y_1 \to y_2]_j^i, 1 < j \leq n + 1;$

$r_8 : y_2[]_j^0 \to [x_1]_j^+, 1 \leq j \leq n;$
$r_9 : y_2[]_j^+ \to [x_1]_j^0, 1 \leq j \leq n;$
where $i \in \{0, +\}$; and the order relationship $r_1, r_2, r_4, r_5, r_6, r_7, r_8, r_9 > r_3$.

M/S	$[]_1$	$[]_2$	$[]_3$	$[]_4$	$[]_5$	$[]_6$	$[]_{aux}$
0	$[a^3x_1]^0$	$[a^6x_1]^+$	$[a_9x_1]^0$	$[a^5x_1]^+$	$[a^7x_1]^0$	$[a^8x_1]^+$	$[\_]^0$
1	$[a^3b^6x_2]^0$	$[x_2]^+$	$[a^9b^5x_2]^0$	$[x_2]^+$	$[a^7b^8x_2]^0$	$[x_2]^+$	$[\_]^0$
2	$[a^6]^+$	$[a^3y_1]^0$	$[a^9y_1]^+$	$[a^5y_1]^0$	$[a^8y_1]^+$	$[a^7y_1]^0$	$[y_1]^0$
3	$[a^6]^+$	$[a^3b^9y_2]^0$	$[y_2]^+$	$[a^5b^8y_2]^0$	$[y_2]^+$	$[a^7y_2]^0$	$[y_2]^0$
4	$[a^6x_1]^0$	$[a^9x_1]^+$	$[a_3x_1]^0$	$[a^8x_1]^+$	$[a^5x_1]^0$	$[a^7x_1]^+$	$[\_]^0$
5	$[a^6b^9x_2]^0$	$[x_2]^+$	$[a_3b^8x_2]^0$	$[x_2]^+$	$[a^5b^7x_2]^0$	$[x_2]^+$	$[\_]^0$

There are no additional requirements in the case where $n = 2k + 1$, however we always entail an extra auxiliary membrane to enable *out* communication of the $n-$th membrane, therefore allowing it to switch polarity.

A similar implementation of the static sorting algorithm can be obtained by using P systems with labels on membranes. As illustrated in [1], we can encode electrical charges in strings of the membrane labels, in order to differentiate between the two necessary states. For each membrane h_i we synthesise its complementary label h'_i, which is changed to by a communication rule. We leave this as an exercise to the reader.

A number of (preliminary) conclusions can be drawn from the above case study:

- kP systems are conceptually closer to tissue P systems than cell-like P systems; in our case studies, this is reflected by the similarity between the specifications using kP systems and tissue P systems, respectively. On the other hand, the model realized using the cell-like P system variant is significantly more complex.
- In terms of complexity, the three implementations are roughly equivalent. The kP system executes in each step one more rule then the P system with states; this rule is either r_2 or r_3 (dealing with p). On the other hand, the number of rules applied in each compartment for every step by cell-like P systems is similar to the case of kP systems.

5 Conclusions

The kP system introduced in this work represents a low level specification language. Its syntax and informal semantics and some examples have been introduced and discussed. A case study based around a simple sorting algorithm has allowed us to compare different specifications of this using various types of P systems. In the next stage formal semantics will be defined and an implementation using model checkers (SPIN, Maude) is also expected. Several extensions can be considered for kP systems that may lead to a more flexible and higher level specification language. A first set of extensions refer to ways of defining objects, rules and compartments using indexes over some specified domains. Modules can also be introduced using a syntactic approach, rather than considering additional semantic features [8]. In order to prove the expressive power of kP systems, a more systematic study of simulating important classes of P systems with kP systems will be produced in a forthcoming paper.

Acknowledgement. The work of MG and FI was partially supported by project MuVet, Romanian National Authority for Scientific Research (CNCS – UEFISCDI) grant number PN-II-ID-PCE-2011-3-0688.

References

1. A. Alhazov, L. Pan, Gh. Păun, Trading Polarizations for Labels in P Systems with Active Membranes, *Acta Informatica*, 41, 111-144, 2004.
2. A. Alhazov, D. Sburlan, Static Sorting P Systems. In [5], 215 – 252, 2006.
3. M. Ben-Ari, *Principles of the SPIN Model Checker*, Springer, 2008.
4. R. Ceterchi, C. Martín-Vide, P Systems with Communication for Static Sorting. In *Pre-Proceedings of Brainstorming Week on Membrane Computing, Tarragona, February 2003*, M. Cavaliere, C. Martín-Vide, Gh. Păun, eds., Technical Report no 26, Rovira i Virgili Univ., Tarragona, 101–117, 2003.
5. G. Ciobanu, Gh. Păun, M. J. Pérez-Jiménez, eds., *Applications of Membrane Computing*, Springer, 2006.
6. M. Clavel, F.J. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, Maude: Specification and Programming in Rewriting Logic, *Theoretical Computer Science*, 285, 187 – 243, 2002.
7. D. Díaz-Pernil, M. A. Gutiérrez-Naranjo, M. J. Pérez-Jiménez, A Uniform Family of Tissue P Systems with Cell Division Solving 3-COL in a Linear Time, *Theoretical Computer Science*, 404, 76 – 87, 2008.
8. M. Gheorghe, V. Manca, F.J. Romero-Campero, Deterministic and Stochastic P Ssytems for Modelling Cellular Processes, *Natural Computing*, 9, 457–473, 2010.
9. R. Nicolescu, M. J. Dinneen, Y.-B. Kim, Structured Modelling with Hyperdag P Systems: Part A. In *Membrane Computing, Seventh Brainstorming Week, BWMC 2009, Sevilla, Spain, February 2009*, R. Gutiérrez-Escudero, M. A Gutiérrez-Naranjo, Gh. Păun, I. Pérez-Hurtado, eds., Universidad de Sevilla, 85 – 107, 2009.
10. Gh. Păun, *Membrane Computing: An Introduction*, Springer, 2002.
11. Gh. Păun, G. Rozenberg, A. Salomaa, eds., *Handbook of Membrane Computing*, Oxford University Press, 2010.