3-Col problem modelling using simple kernel P systems

Marian Gheorghe[a,c], Florentin Ipate[b,c], Raluca Lefticaru[b,c], Mario J. Pérez-Jiménez[d], Adrian Ţurcanu[c], Luis Valencia Cabrera[d], Manuel García-Quismondo[d] and Laurenţiu Mierlă[c]

[a]Department of Computer Science, University of Sheffield, Portobello Street, Regent Court, Sheffield, S1 4DP, UK; [b]Faculty of Mathematics and Computer Science, University of Bucharest, Academiei 14, Bucharest, Romania; [c]Faculty of Mathematics and Computer Science, University of Pitesti, Targu din Vale 1, Pitesti, Romania; [d] Research Group on Natural Computing, Department of Computer Science and Artificial Intelligence, University of Sevilla, Avda. Reina Mercedes s/n. 41012, Sevilla, Spain

This paper presents the newly introduced class of (*simple*) *kernel P systems* ((*s*)*kP systems*) and investigates through a 3-colouring problem case study the expressive power and efficiency of kernel P systems. It describes two skP systems that model the problem and analyses them in terms of efficiency and complexity. The skP models prove to be more succinct (in terms of number of rules, objects, number of cells and execution steps) than the corresponding tissue P system, available in the literature, that solves the same problem, at the expense of a greater length of the rules.

Keywords: membrane computing; tissue P systems; kernel P systems; 3-colour problem; formal verification

1. Introduction

Inspired by the behaviour and structure of the living cell, P systems have emerged in recent years as powerful computational tools [15]. Many variants of P systems have been introduced and a number of theoretical aspects have been studied intensely: the computational power of different variants, their capabilities to solve hard problems, such as nondeterministic polynomial time (NP)-complete ones, decidability, complexity aspects and hierarchies of classes of languages generated and/or recognized by these devices [17]. In the last few years, there have also been significant developments in using the P system paradigm to model, simulate and formally verify various systems [3,17], including a number of well-known distributed algorithms and problems [14]. In many cases, however, the specifications produced required additional features or constraints compared with the original definition of the P system variant used; such additional features added expressiveness to the specification and clarified complex aspects of the system involved. Although extremely useful for the actual modelling, the *ad hoc* addition of such extra features is meant to lead to an adverse effect on the capability of P systems to provide a coherent analysis and verification framework.

To alleviate this problem, *kernel P systems* (*kP systems*, for short) have been defined recently [7]. This is a low-level specification language that uses the established features of existing P system variants and also includes some new elements. Most importantly, kP systems offer a coherent way of integrating these elements into the same formalism. On a longer term, it is envisaged that a kP system simulator will be developed and integrated into the P-Lingua platform [6,18] and model-checking facilities will also be available, thus providing a coherent platform for system analysis and verification.

In this paper, a particular type of kP system, called *simple kernel P system* (*skP system*, for short), is introduced. Like a kP system, this uses a graph-like structure, but rules of only two types:

- *membrane division rules*, which divide an existing compartment into two or more compartments of the same type, and
- *rewriting and communication rules*, which evolve objects inside a compartment and may also send some of the resulting objects to compartments linked with the current compartment.

The execution of a rule is conditioned by a guard, defined in a general manner using activators and inhibitors. The execution strategy of kP systems is defined in a more general way than in traditional P systems. However, skP systems, the particular case considered in this paper, use only the well-established maximal parallelism mode.

This paper illustrates the modelling power of skP systems by using a well-known NP-complete problem, the 3-colouring (3-Col) problem. A tissue P system with cell division that models the 3-Col problem is available in the literature [5] and will be used as a term of reference for our models. A kP system that models the 3-Col problem has been produced recently in [12]; however, this system uses division rules in which a membrane is divided into three membranes, which make the performance comparison between the two systems unrealistic. To facilitate a realistic comparison, an skP system using membrane division rules in which a membrane is always divided only into two membranes is produced here. A second skP system which only uses membrane division rules with no objects rewritten is also considered. For each of the two skP systems, it is proved that it solves the 3-Col problem and, in each case, a comparison between the original tissue P system and the new skP system model is performed. The skP systems are implemented and simulated using P-Lingua and MeCoSim [19,20]. Furthermore, a number of case studies are presented in order to illustrate the behaviour and performances of various skP systems introduced here and the potential benefit of using a formal verification for checking their different properties.

The results show that skP systems are flexible specification mechanisms – different distinct solutions are produced for the same problem, and given their rich set of features, whereby more complex rewriting and communicating rules and powerful guard expressions are utilized, succinct, but still easy-to-understand, solutions are obtained. The skP system models provide better solutions, with respect to execution steps and the number of cells utilized, compared with the tissue P systems [5], but the rules applied, as it has been mentioned earlier, are more complex. It is worth mentioning that the topics discussed in this paper, such as kP systems, complexity aspects and formal verification, are some of the research areas of membrane computing envisaged to be studied in the near future [8].

The paper is structured as follows. Section 2 introduces skP systems. The two skP system models for the 3-Col problem are presented in Section 3 together with some particular cases. Section 4 describes the MeCoSim platform and presents the implementation and simulation of these skP system models using P-Lingua and MeCoSim and shows how the systems can be analysed using these tools or model-checking approaches. Finally, conclusions are drawn and future work is outlined in Section 5.

2. Background

This section presents a simplified version of a kP system (for more details about this formalism, see [7]). The model of a kP system presented in [7] includes features dealing with object transformation and communication, aspects related to ways of dynamically changing the structure of the system and strategies of running it. In this paper, apart from operations regarding object transformation and communication, cell division is considered – as a mechanism to generate new components of the system – and it is assumed that its execution is handled according to the principle of maximal parallelism, widely used in membrane computing [16,17]. This formalism is called *skP system*. In the following, the necessary concepts to formalize this model will be introduced.

An *alphabet*, A, is a non-empty set whose elements are called *symbols*. A finite sequence of symbols is a *string* or *a word*. If u and v are strings over A, then so is their *concatenation* uv, obtained by juxtaposition, that is, writing u and v one after the other. The number of symbols in a string u is the *length* of the string and it is denoted by $|u|$. As usual, the empty string (with length zero) will be denoted by λ. The set of all strings over an alphabet A is denoted by A^*. When the empty string is not considered, the set is denoted by A^+. Subsets, finite or infinite, of A^* are referred to as *languages* over A. If $u \in A^*$ and $a \in A$, then $|u|_a$ denotes the number of occurrences of symbol a in string u.

A *multiset* m over a set A is a pair (A,f) where $f : A \to \mathbb{N} \cup \{\emptyset\}$ is a mapping. The *support* of multiset m is defined as $\text{supp}(m) = \{x \in A \mid f(x) > 0\}$. A multiset, which in general might be infinite, is empty (respectively, finite) if its support is the empty set (respectively, a finite set). If $m = (A,f)$ is a finite multiset over A and $\text{supp}(m) = \{a_1, \ldots, a_k\}$, then it will be denoted as $m = \{a_1^{f(a_1)}, \ldots, a_k^{f(a_k)}\}$. It can also be represented by the string $a_1^{f(a_1)} \cdots a_k^{f(a_k)}$ over the alphabet $\{a_1, \ldots, a_k\}$. Nevertheless, all permutations of this string identify the same multiset m precisely. Throughout this paper, only 'the finite multiset m' is considered, where m is a string, meaning 'the finite multiset represented by the string m'.

DEFINITION 2.1 *Let* $\text{Rel} = \{<, \leqslant, =, \neq, \geqslant, >\}$ *be a set of relational operators. Let A be a non-empty finite set and denote $\bar{A} = \{\bar{a} \mid a \in A\}$. A multiset over $A \cup \bar{A}$ with relational operators from Rel is an expression* $w = \theta_1 a_1^{n_1} \cdots \theta_k a_k^{n_k}$, *where* $a_1^{n_1} \cdots a_k^{n_k}$ *is a string over* $A \cup \bar{A}$ (*in which* a_i *and* a_j *are not necessarily distinct*, $1 \leq i < j \leq k$), *and* $\theta_j \in \text{Rel}$, *for each* j, $1 \leq j \leq k$.

DEFINITION 2.2 *A guard g is a finite disjunction of expressions w introduced by Definition* 2.1.

OBSERVATION 2.1 *$|g|$, g as in Definition* 2.2, *denotes the number of w expressions occurring in g; $|g|$ is called the length of the guard g.*

A guard is a finite disjunction of expressions involving multisets over A – activators (or permitting symbols), or $\bar{A}$ – inhibitors (or forbidding symbols).

A particular case of a guard g is the empty set. In this case, the guard is omitted.

DEFINITION 2.3 *Given two non-empty finite sets A (alphabet), L (labels) and $l \in L$, the set of rules associated with l is denoted by R_l. A rule from R_l has one of the following two forms*:

(a) $[x]_l \to [y_1]_{l_1} \cdots [y_h]_{l_h}\{g\}$, *where* $x \in A$, $y_j \in A^*$, $l_j \in L$ *for all* $1 \leq j \leq h$, *and g is a guard* (h-membrane division *rule*). *The length of the rule is defined as* $1 + |y_1| + \cdots + |y_h|$ ($|g|$ *can also be considered*).

(b) $x \to y\{g\}$, *where* $x \in A^+$, *y is a string over* $A \times L$, $y = (a_1, t_1) \cdots (a_h, t_h)$, *with* $a_j \in A$ *and* $t_j \in L$, *and g is a guard* (rewriting and communication *rule*). *The length of the rule is defined as* $|x| + |y|$ (*the complexity of the guard g, denoted $|g|$, can also be considered for this rule*).

A particular case of type (a) rules is that of an *h-membrane division rule with no objects rewritten*, that is,

(a') $[]_l \to []_{l_1} \cdots []_{l_h}\{g\}$. This rule is applicable if and only if the guard is true.

An example of a guard which is true is $g =< a^2 = \bar{b}$ with the current multiset containing at most one a and no b, for instance, ac – see Definition 2.5 for a more general setting.

DEFINITION 2.4 *Given two non-empty finite sets A, L, a compartment, C, is a tuple (l, w_0, R_l), where $l \in L$, w_0 is a multiset over A and R_l is a set of rules associated with l.*

A compartment can be viewed as a region labelled by an element l of L which initially contains a multiset w_0 of objects (elements of A) and such that it is associated a set R_l of rules.

DEFINITION 2.5 *Let r be a rule from R_l, a set of rules associated with a compartment $C = (l, w_0, R_l)$, such that the guard of r is g, then g is considered true at an instant when the current multiset of compartment C is z if and only if the following happens:*

(a) *If $g = \theta_1 a_1^{n_1} \cdots \theta_k a_k^{n_k}$, then for every j, $1 \leq j \leq k$, either (i) if $a_j \in A$, then $|z|_{a_j}\ \theta_j\ n_j$ holds, or (ii) if $a_j \in \bar{A}$, then $|z|_{a_j} \theta_j n_j$ does not hold.*
(b) *If g is a finite non-empty disjunction of multisets over $A \cup \bar{A}$ with relational operators from* Rel, *$g = w_1 | \cdots | w_p$, then there exists j, $1 \leq j \leq p$, such that w_j is true, according to (a).*
(c) *If g is the empty set, then it is always evaluated true.*

DEFINITION 2.6 *An skP system of degree $n \geq 1$ is a tuple*

$$\mathrm{sk}\Pi = (A, L, IO, C_1, \ldots, C_n, \mu, i_0),$$

where

- *A and L are non-empty finite sets,*
- *IO is a finite alphabet $IO \subseteq A$,*
- *$C_1, \ldots, C_n$ are compartments,*
- *$\mu = (V, E)$ is an undirected graph, where $V \subseteq L$ are vertices and E the edges, and*
- *$i_0 \in L$.*

An skP system, $\mathrm{sk}\Pi = (A, L, IO, C_1, \ldots, C_n, \mu, i_0)$, can be viewed as a set of n compartments, $C_1, \ldots, C_n$, interconnected by edges from E, of an undirected graph μ. The elements of the set A are called *objects* and the elements of L are called *labels*. IO is the alphabet of the *environment objects*. Each compartment is identified by a label of L, has initially a multiset over A, and a finite set of rules. The compartment receiving the result of a computation is denoted by i_0; in the sequel, this will always be the *environment*.

An h-membrane division rule $[x]_l \to [y_1]_{l_1} \cdots [y_h]_{l_h}\ \{g\}$ associated with a compartment $C = (l, w_0, R_l)$ is applicable at a given instant to the current multiset z if the guard g is evaluated true with respect to z and the object x is in the multiset z. When applying such a rule to x, the compartment labelled l will be replaced by h compartments labelled $l_1, \ldots, l_h$ and x is replaced by multiset y_j in compartment l_j; the content of l, but x, after using all the applicable rewriting and communication rules is copied in each of these compartments; all the links of l are inherited by each of the newly created compartments.

A rewriting and communication rule $x \to (a_1, t_1) \cdots (a_h, t_h)\{g\}$ associated with a set of rules, R_l, of a compartment $C = (l, w_0, R_l)$, is applicable at a given moment to the current multiset z if the guard g is evaluated true, x is contained in z and the target $t_j \in L$, $1 \leq j \leq h$, must be either

the label of the current compartment, l, or the label of its existing neighbour ($\{l, t_j\} \in E$). When applying such a rule, objects a_j are sent to the compartment labelled by t_j, for each j, $1 \leq j \leq h$. 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. When t_j indicates the label of the environment, the corresponding object a_j is sent to the environment.

In an skP system, the rules are applied in a maximally parallel way with the usual restriction that at most one rule of type (a) (membrane division rule) can be applied per membrane in each step.

3. Results

In order to illustrate the modelling and expressive power of skP systems and their efficiency, the 3-Col problem [5] is considered and, below, it is shown how this can be specified within this formalism and also how its complexity aspects with respect to the number of objects, rules and execution steps can be estimated.

In general, the k-colouring problem is formulated as follows: given an undirected graph $G = (V, E)$, decide whether or not G is k-colourable; that is, if there exists a labelling of the vertices of G, using k different labels (colours) such that for every edge $\{u, v\} \in E$, the colours of u and v are different.

OBSERVATION 3.1 *Obviously, the 3-Col problem has no solutions for* (*a*) *a complete graph with* $n > 3$ *vertices and* (*b*) *a graph that contains a subgraph with at least four vertices, any two connected by an edge.*

As shown in [5], the 3-Col problem can be solved in linear time by a recognizer tissue P system with cell division and symport/antiport rules. In what follows, an skP system model that solves the same problem is presented and the two approaches are compared. The proof of this result is based on the idea presented in [12], but it is adapted for an skP system with 2-membrane division rules.

Notation. Denote by $C\mathrm{sk}\Pi(t, s, o, m_1, l_1, \gamma_1, m_2, l_2, \gamma_2)$ the class of skP systems with t types of compartments, s initial compartments, using an alphabet with o objects, having m_1 2-membrane division rules (type (a), according to Definition 2.3) – with length at most l_1 and guards of length at most γ_1 – and m_2 rewriting and communication rules (type (b), according to Definition 2.3) – with length at most l_2 and guards of length at most γ_2.

THEOREM 3.1 *The 3-Col problem for a graph with* n, $n \geq 2$, *nodes can be solved by an skP system of type* $C\mathrm{sk}\Pi(2, 2, n(n-1)/2 + 7n + 10, 2n, 5, 1, 2n + 7, 3, 3n(n-1)/2)$ *and an answer to whether a solution exists or not is obtained in at most* $2n + 3$ *steps using maximum* $3^n + 1$ *compartments.*

Proof For a given undirected graph $G = (V, E)$, where V contains $n \geq 2$ vertices, we build the following skP system (which depends on n):

$$\mathrm{sk}\Pi(n) = (A, L, IO, \mu, C_1, C_2, 0),$$

where

- $A = \{A_1, \ldots, A_n\} \cup \{A_{i,j} \mid 1 \leq i < j \leq n\} \cup \{T_1, \ldots, T_n\} \cup \{B_1, \ldots, B_n\} \cup \{R_1, \ldots, R_n\} \cup \{G_1, \ldots, G_n\} \cup \{a, s, X, Y, T, \mathrm{yes}, \mathrm{no}\} \cup \{X_1, \ldots, X_{2n+3}\}$, where A_i, $1 \leq i \leq n$, stand for the n vertices, $A_{i,j}$, $1 \leq i < j \leq n$, are for all possible edges between the n vertices, T_i, $1 \leq i \leq n$, are used in the division process of C_2 compartments, B_i, R_i, G_i, $1 \leq i \leq n$, stand for the three

colours, blue, red and green, respectively, that can be associated with the n vertices; the object a is used only in compartment C_1 to select one single answer to be sent to the environment; s, X, Y are used in compartments C_2, T is sent to compartment C_1, yes, no are the two possible answers – with one of them being sent from C_1 to the environment at the end of the computation; $X_1, \ldots, X_{2n+3}$ are used to count the maximum number of steps, $2n+2$, requested for the last possible input from C_2;

- $L = \{0, 1, 2\}$; 0 is the label of the environment and 1 and 2 are the labels of the two types of compartments;
- IO consists of yes, no;
- $C_1 = (1, w_{1,0}, R_1), C_2 = (2, w_{2,0}, R_2)$, where $w_{1,0} = aX_1$, $w_{2,0} = A_1 s\text{code}(n)$, with code$(n)$ being the multiset of edges of the graph to be coloured; an edge $\{i, j\}$ is codified as $A_{i,j}$, $1 \leq i < j \leq n$;
- μ is given by the graph with edge $\{1, 2\}$;
- R_1 and R_2 are given below:

(1) R_1 contains

$r_{1,i} : X_i \to X_{i+1}, 1 \leq i \leq 2n+2$,
$r_{1,2n+3} : aT \to (\text{yes}, 0)$,
$r_{1,2n+4} : aX_{2n+3} \to (no, 0)\{\geq \bar{T}\}$;
rules $r_{1,i}, 1 \leq i \leq 2n+2$, are for counting the first $2n+2$ steps; in the first $2n+2$ steps, for each solution found, an object T will be sent from C_2 to C_1; when one or more T's are received from compartments C_2, that is, there is at least one solution, compartment C_1 releases yes into the environment; otherwise, when no T's are received, after $2n+3$ steps, a no is sent out instead;

(2) R_2 contains

membrane division rules:
$r_{2,2i-1} : [A_i]_2 \to [R_i A_{i+1}]_2 [T_i]_2 \{= s\}$,
$r_{2,2i} : [T_i]_2 \to [B_i A_{i+1}]_2 [G_i A_{i+1}]_2$, $1 \leq i \leq n-1$, and
$r_{2,2n-1} : [A_n]_2 \to [R_n X]_2 [T_n]_2 \{= s\}$,
$r_{2,2n} : [T_n]_2 \to [B_n X]_2 [G_n X]_2$;
these are applied in at most $2n$ steps and all the possible combinations of colouring n vertices with three colours are obtained, given that the guards are true;
rewriting and communication rules:
$r_{2,2n+1} : s \to \lambda\{= A_{1,2} = B_1 = B_2 | = A_{1,2} = G_1 = G_2 | = A_{1,2} = R_1 = R_2 | \cdots | = A_{n-1,n} = B_{n-1} = B_n | = A_{n-1,n} = G_{n-1} = G_n | = A_{n-1,n} = R_{n-1} = R_n\}$ (one rule with a guard containing $3n(n-1)/2$ terms); it checks for any pair $1 \leq i < j \leq n$ that the colour of the nodes i and j is the same and s is available; if so, s is erased; when s disappears, no further calculations are performed in the corresponding compartment;
$r_{2,2n+2} : X \to Y$; when all the calculations are completed, X is transformed into Y;
$r_{2,2n+3} : Ys \to (T, 1)$; this rule is applied when there is a solution in the current compartment C_2, and T is sent to compartment C_1.

The skP system sk$\Pi(n)$ works as follows: a solution is sought in a compartment C_2 as a multiset $x_1 \cdots x_n$, where $x_i \in \{B_i, R_i, G_i\}$, $1 \leq i \leq n$, such that x_i and x_j, $1 \leq i < j \leq n$, do not indicate the same colour (B_i, B_j or R_i, R_j or G_i, G_j). The skP system starts with two compartments, C_1 and C_2, containing their initial multisets, $w_{1,0}$ and $w_{2,0}$; the presence of s in compartment C_2 means that a potential solution can be developed in that compartment; when s is no longer present, this shows that two adjacent nodes of the graph have the same colour. As long as s is present in a compartment C_2, a membrane division rule, $r_{2,2i-1}$, $1 \leq i \leq n$, can be applied and two new compartments C_2 replace the current C_2, with one of them having colour R_i for node i and the other being divided in the next step when two new compartments will appear with colours B_i and G_i for the same node

Table 1. Tissue P systems versus skP systems.

Type/specification	Tissue P systems	skP systems
Alphabet	$6n^2 + 12n + 2m + 2[\log m] + 29$	$n(n-1)/2 + 7n + 10$
Rules	$2n$ & $6n(n-1)/2 + 8n + 2m + 3[\log m] + 25$	$2n$ & $2n + 7$
Maximum number of compartments	$3^n + 1$	$3^n + 1$
Number of steps	$2n + m + [\log m] + 11$	$2n + 3$

– this way, all the three possible colours for node i are generated in three distinct compartments C_2. In each computation step, the rule $r_{2,2n+1}$ checks whether there are two adjacent nodes with the same colour, and if yes, then s is removed from the compartment. The process of generating the compartments C_2 for all possible colourings and their validation through rule $r_{2,2n+1}$ last for $2n$ steps. In the step $2n + 1$, the object X introduced after the last membrane division step is transformed into Y in all compartments C_2 that might contain solutions. If s will remain in these compartments C_2 after this step, then in the next one, rule $r_{2,2n+3}$ is sending T to compartment C_1. If at least a T is received in compartment C_1 from one of the compartments C_2, then a yes is sent out to the environment, using rule $r_{1,2n+3}$, signalling that there is a solution; otherwise a no is sent out instead. ■

OBSERVATION 3.2 *Table 1 summarizes some comparative data concerning the specification of the 3-Col problem with the model from* [5] *using symport/antiport rules and the skP system solution. In this table, n is the number of vertices and m the number of edges of the graph. The table shows that the number of symbols and rules is less than that of the tissue P system model. However, the reduction of the number of rules is achieved at the expense of their complexity; indeed, the length of the rewriting and communication rules is bounded by 3, with a guard length of maximum* $3n(n-1)/2$*, for skP systems, whereas in the case of the tissue P systems, the corresponding rules have their length up bounded by 4 and no guards* [5]*. The division rules have similar complexity values: rule length up bounded by 5 and guard length by 1 for skP systems and rule length up bounded by 3, with no guards, for the tissue P systems* [5]*. A further discussion on the trade-off between the number of rules and the guard complexity is addressed by Observation* 3.3*. The actual number of compartments labelled* 2*, produced by the skP system, may in general be* (*significantly*) *less than* 3^n *since division rules are only performed when no two adjacent nodes have the same colour – see some examples given in the next section. In the case of the tissue P system,* 3^n *compartments labelled 2 are always produced.*

OBSERVATION 3.3 *The rule* $r_{2,2n+1}$ *can be replaced either by m rules* $s \to \lambda$ *with a guard, g, of the form* $= A_{i,j} = B_i = B_j| = A_{i,j} = G_i = G_j| = A_{i,j} = R_i = R_j$ (|g|=*3 – see Observation* 2.1) *or by* 3*m rules of the same form, but with simpler guards,* $= A_{i,j} = x_i = x_j$ (*of length 1*)*, where x is one of* R, B, G*; so, in this case, the number of rules increases with either m or* 3*m, but the guard length goes down to 3 or 1, respectively. Even in these cases, the number of rules of the skP system is less than the number of rules of the tissue model.*

If we now consider 2-membrane division rules with no objects rewritten (type (a') rules), then Theorem 3.1 can be reformulated as follows.

THEOREM 3.2 *The 3-Col problem for a graph with* n, $n \geq 2$, *nodes can be solved by an skP system of type* $C\mathrm{sk}\Pi(6, 2, n(n-1)/2 + 11n + 8, 5, 0, n, 9n + 6, 3, 3n(n-1)/2)$ *and an answer to whether a solution exists or not is obtained in at most* $5n + 2$ *steps using maximum* $3^n + 1$ *compartments.*

Proof The construction used in the proof of Theorem 3.1 will be used here with some changes mentioned in the sequel. Only the rules will be described; the alphabet and the structure of the skP system will follow from them and Theorem 3.1.

First, let us observe that we have six types of compartments, C_i, $1 \leq i \leq 6$; these are labelled $1, 2, 20, 21, 22$ and 23, respectively.

Initially, the skP system consists of two compartments, C_1 and C_2, with initial multisets $w_{1,0} = aX_1$, $w_{2,0} = A_1 s\ \text{code}(n)$, as in Theorem 3.1.

The set of rules from compartment C_2, denoted R_2, consists of the following:

- membrane division rules:
  $r_{2,1} : []_2 \to []_{21}[]_{20}\ \{= A_1 = s| \cdots | = A_n = s\}$;
- rewriting rules:
  $r_{2,i} : A'_i \to A_i, 2 \leq i \leq n$,
  $r_{2,n+1} : s \to \lambda \qquad \{= A_{1,2} = B_1 = B_2| = A_{1,2} = G_1 = G_2| = A_{1,2} = R_1 = R_2| \cdots | = A_{n-1,n} = B_{n-1} = B_n| = A_{n-1,n} = G_{n-1} = G_n| = A_{n-1,n} = R_{n-1} = R_n\}$,
  $r_{2,n+2} : X \to Y$;
- communication rule:
  $r_{2,n+3} : Ys \to (T, 1)$.

Set R_{20} in compartment C_3, with label 20, contains the membrane division rule $r_{20,1} : []_{20} \to []_{22}[]_{23}$.

Compartments C_4, C_5 and C_6, with labels 21, 22 and 23, respectively, have the following sets of rules:

- R_{21}
  $r_{21,1} : []_{21} \to []_2\ \{= A'_2| \cdots | = A'_n| = X\}$,
  $r_{21,i+1} : A_i \to B_i A'_{i+1}, 1 \leq i \leq n-1$, $r_{21,n+1} : A_n \to B_n X$;
- R_{22}
  $r_{22,1} : []_{22} \to []_2\ \{= A'_2| \cdots | = A'_n| = X\}$,
  $r_{22,i+1} : A_i \to G_i A'_{i+1}, 1 \leq i \leq n-1$, $r_{22,n+1} : A_n \to G_n X$;
- R_{23}
  $r_{23,1} : []_{23} \to []_2\ \{= A'_2| \cdots | = A'_n| = X\}$;
  $r_{23,i+1} : A_i \to R_i A'_{i+1}, 1 \leq i \leq n-1$, $r_{23,n+1} : A_n \to R_n X$.

Set R_1 contains $r_{1,i} : X_i \to X_{i+1}, 1 \leq i \leq 5n+1$, and the last two rules from set R_1 occurring in the proof of Theorem 3.1, relabelled according to the current set of labels:
$r_{1,5n+3} : aT \to (yes, 0)$,
$r_{1,5n+4} : aX_{5n+3} \to (no, 0)\ \{\geq \bar{T}\}$.

The division rules $r_{2,i}$, $1 \leq i \leq 2n$, from the proof of Theorem 3.1 are replaced by two rules, $r_{2,1}$ and $r_{20,1}$; $r_{2,1}$ splits a compartment with label 2 into one with label 21 and another one labelled 20 and $r_{20,1}$ splits a compartment labelled 20 into two compartments with labels 22 and 23, respectively. The rule $r_{2,1}$ is applied only when an A_i, $1 \leq i \leq n$, and an s appear in the current compartment with label 2.

In each of the compartments labelled 21, 22 and 23, the current A_i, $1 \leq i \leq n-1$, is transformed into $B_i A'_{i+1}$ in 21, $G_i A'_{i+1}$ in 22 and $R_i A'_{i+1}$ in 23, using $r_{21,i+1}$, $r_{22,i+1}$ and $r_{23,i+1}$, respectively. When object A_n arrives in one of the compartments 21, 22 and 23, it is transformed into $B_n X$ in 21, $G_n X$ in 22 and $R_n X$ in 23 (one of $r_{21,n+1}$, $r_{22,n+1}$, $r_{23,n+1}$ is used). Each of such compartments is then transformed into a compartment labelled 2, by using one of the rules $r_{x,1}$, $x \in \{21, 22, 23\}$. This transformation is allowed only when either A'_i, $2 \leq i \leq n$, or X appear in one of the above-mentioned compartments (see the guard of $r_{x,1}$); this happens only after applying one of $r_{x,i+1}$, $x \in \{21, 22, 23\}$, $1 \leq i \leq n$. Now in compartment 2, A'_i, $2 \leq i \leq n$, is transformed into A_i (rule $r_{2,i}$)

or X into Y (rule $r_{2,n+2}$). When A_i, $1 \leq i \leq n$, and s appear in the current compartment with label 2, the division process restarts; otherwise if only A_i is available, it stops. When Y is obtained, a solution exists in the current compartment labelled 2, and in the next step, the rule $r_{2,n+3}$ sends T to compartment C_1.

The complexity values stated by this theorem can be obtained by comparing the above proof with the values provided in Table 1. Indeed, the size of the alphabet of the skP system from the proof of Theorem 3.1 is $n(n-1)/2 + 7n + 10$, and in the proof given above, there are $3n - 1$ more X_i objects and $n - 1$ new A'_i objects; hence, the size of the alphabet given above is $n(n - 1)/2 + 11n + 8$. Note that the maximum number of steps is $5n + 2$ ($3n - 1$ more steps than those for the system in Theorem 3.1 – this is because $r_{2,i}$ is applied only $n - 1$ times). The number of rules is as follows: five 2-membrane division rules of type (a') ($r_{2,1}$, $r_{20,1}$, $r_{21,1}$, $r_{22,1}$, $r_{23,1}$); $9n + 6$ rewriting and communication rules ($n + 2$ in R_2; $5n + 4$ in R_1; and n in each of R_{21}, R_{22} and R_{23}). ■

Some better values for the complexity parameters stated by Theorems 3.1 and 3.2 can be obtained further.

OBSERVATION 3.4 *For Theorem* 3.1, *it can be observed that an edge* $\{i,j\}$, $1 \leq i < j \leq n$, *is checked when node* j *is introduced, after node* i; *this is between* $(j+1)$*th and* $(2j+1)$*th steps* (*similar values can be obtained for Theorem* 3.2); *compartments for which there exists an edge* $\{i,j\}$, $1 \leq i < j \leq n$, *and* i *and* j *have the same colour, will no longer divide.*

The particular cases of graphs $G(n,k) = (V,E)$, $n \geq 2$, $1 \leq k \leq n$, can be considered – the graphs with these two parameters, n, k, are introduced in order to differentiate them, where $V = \{1,\ldots,n\}$, and E has edges of types (i,k) and (k,j), where k is a fixed vertex, $1 \leq k \leq n$ and $i,j \in V\backslash\{k\}$, $i < k, j > k$. Two specific cases are $G(n,1)$, where edges are of type $(1,i)$, with $i \in V\backslash\{1\}$, and $G(n,n)$ where edges are of type (i,n), with $i \in V\backslash\{n\}$.

The graphs $G(n,k)$, $n \geq 2$, $1 \leq k \leq n$, are of interest as they are 2-colourable, so easier to handle than the general undirected graphs with n vertices, provide the worst case for the division process and will also appear in some of the examples analysed in the next section.

Before proceeding any further, it can be observed that for all $G(n,k)$, $n \geq 2$, $1 \leq k \leq n$, the following results hold (the proofs are immediate and are only sketched).

OBSERVATION 3.5

(a) *The answer to the question 'is the graph* $G(n,k)$ *3-colourable' is 'yes' and it is obtained after* $n + 4$ *steps.*
(b) *When the computation stops, after at most* $2n + 3$ *steps, all the* $3 \cdot 2^{n-1}$ *valid solutions of the problem are obtained.*

(a) Indeed, let k be the fixed vertex of the graph (the more general case of $1 < k < n$ is considered, but the other two, $k = 1$ and $k = n$, are similar to this one). It is easy to observe that the first valid solution of the problem found by our algorithm is of the form $R_1 \cdots R_{k-1} B_k R_{k+1} \cdots R_n$ or $R_1 \cdots R_{k-1} G_k R_{k+1} \cdots R_n$ and it is obtained after $n + 4$ steps.

(b) Choosing randomly one of the three colours for the fixed vertex, each of the other vertices can be coloured with any of the other two colours, so there are 2^{n-1} possible combinations in each of the three cases corresponding to the possible colour chosen for the fixed vertex.

The maximum number of compartments obtained at the end of the computation is $3^n + 1$ – as stated by Theorem 3.1. This is obtained for a graph $G(n,n)$, using the above-introduced notation.

PROPOSITION 3.1 *For a graph* $G(n,n) = (V,E)$, $V = \{1,\ldots,n\}$, *with edges of the form* (i,n), $i \in V\backslash\{n\}$, *the number of compartments labelled 2 is*

(a) 2^h, *after* h *steps, with* $1 \leq h \leq n$;
(b) $2^{n+1} - 1$, *after* $n+1$ *steps*;
(c) 3^n, *at the end of the division process* (*after* $2n$ *steps*).

Proof The rule $r_{2,2n+1}$ can be applied only when there appear two vertices i,j having an edge between them and coloured the same. This particular kind of graph has only edges of type (i,n), $1 \leq i < n$; consequently, the object s might be eventually deleted only after colouring the node n, which happens after colouring all the previous nodes $1,\ldots,n-1$. The computational process involves in the first n steps the division of each compartment labelled 2; consequently, their number is 2^h, for each $1 \leq h \leq n$.

After n steps, the only compartment labelled 2 which will not be further divided is the one containing $R_1 \cdots R_n$. All the others will divide; consequently, in configuration $n+1$, $2^{n+1} - 1$ compartments are obtained.

After $2n$ steps, all the possible combinations of three colours are generated, each in a distinct compartment; hence, their number is 3^n. ■

4. Modelling, simulation and verification framework

In this section, a methodology for modelling, simulation, analysis and formal verification is presented, by using model checking, for skP systems (for more details, see [13]). The methodology, outlined in Figure 1, is supported by a software framework called MeCoSim [20], which contains a specification language, P-Lingua [18,19], which covers various variants of P systems, including skP systems. Finally, this methodology is utilized for the simulation, analysis and formal verification of some instances of the 3-Col problem.

The above-mentioned methodology involves a number of steps. First, the problem is modelled as an skP system, which is then translated into a P-Lingua specification [6,18]. A custom set of interfaces, specifically designed in MeCoSim [19,20], allows us to handle input data for each problem, provides visualization results and extracts suitable output data. Simulations are performed within MeCoSim by using the simulator engine of pLinguaCore [6]. Properties and invariants can be detected from a pool of selected output data or formulated in accordance with

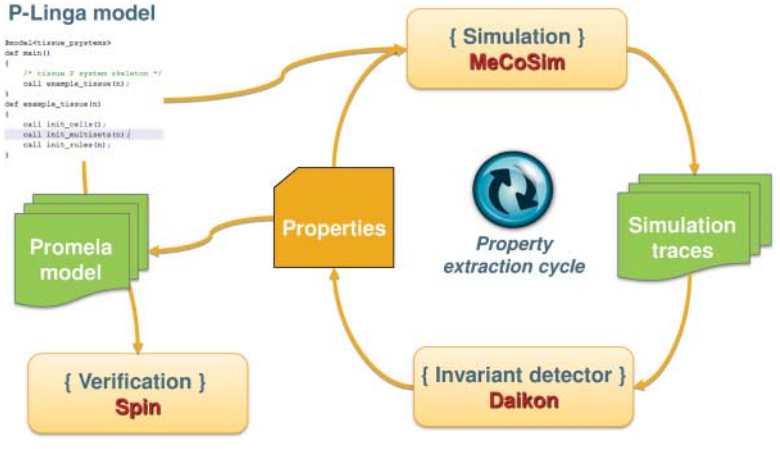

Figure 1. Specification, analysis and verification framework.

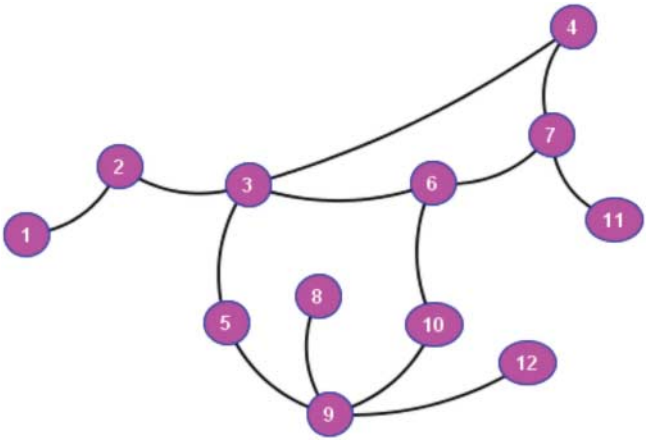

Figure 2. Example graph, n=12.

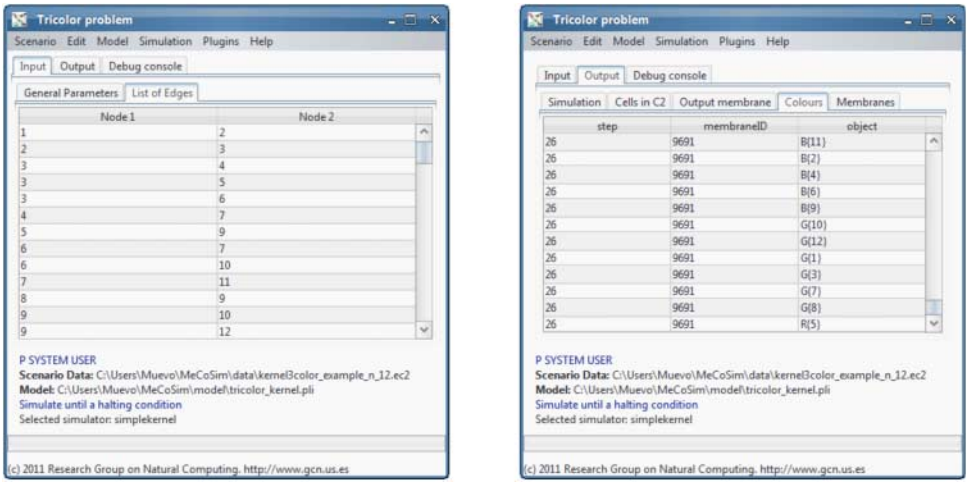

Figure 3. MeCoSim – input edges (left) and output objects.

the system behaviour, and the results can be verified by using model-checking techniques (e.g. using Spin for a model written in the Promela language [2]). For a more detailed description, see [13].

According to the above-mentioned methodology and its associated steps, the models described by Theorems 3.1 and 3.2 have been written in P-Lingua, as skP systems, including variable parameters configurable for different examples, as illustrated in Figure 3, left picture, for the graph given in Figure 2.

The simulation is performed starting from the initial configuration of the P system and the computation runs until a halting configuration is reached. Other options are available: the P system can be simulated step by step or for a given number of steps; a selection of objects and compartments can be produced or, alternatively, all of them can be generated. In addition, the output tables in the custom interface show the information selected to be generated. A solution to the example graph described by Figure 2 is provided in the output table from Figure 3, right picture. In order to help interpreting the result, a visualization tool, included in MeCoSim, has been used for representing the result, where each colour is represented by a specific symbol (the tool also allows us to use

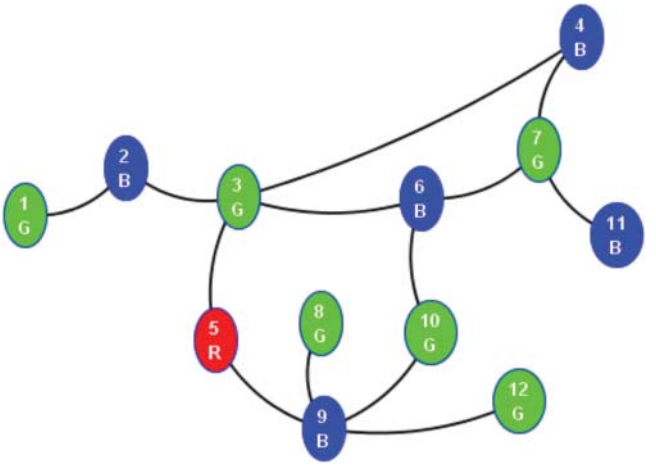

Figure 4. Visualization of a solution.

colours for vertices) – see Figure 4 (colour online). In this way, not only a dry answer to the 3-Col problem is provided, but a visual representation of the solutions found is also associated with it.

A number of simulations have been performed for different instances of the 3-Col problem by implementing the algorithms presented in the proofs of Theorems 3.1 and 3.2. These implementations aim to show the benefits of using the approach based on skP systems with respect to other formalisms using P systems. Note that these implementations cannot be compared with *ad hoc* implementations of the 3-Col problem, as these P system models describe a generic strategy of dynamically handling the compartments, which can be avoided in a direct implementation by specific programming techniques that do not represent this general process – for instance, by finding a recurrent process for generating candidate solutions.

It has been shown that, on the one hand, when skP systems are used, the number of rules is smaller than that in the case of tissue systems, but, on the other hand, these rules can be more complex, so there is a trade-off between these two aspects. The examples given below show that the simulations produced utilize less compartments and exhibit a better execution time when skP systems are used. These examples envisage arbitrary graphs with n vertices and $G(n, 1)$, $G(n, n)$ graphs for $n = 4, 6, 8$. The results for arbitrary graphs are presented in Table 2 for the algorithms associated with Theorem 3.1 (kernel 1) and Theorem 3.2 (kernel 2) and for the tissue P systems. The difference in the number of compartments and execution time between the skP systems and the tissue variant can be observed. This is not surprising when taking into account that a smaller number of rules is utilized by these systems and the fact that the compartments where the condition for producing a solution is no longer true (there are already two vertices with the same colour) stop dividing.

Assuming that the division rules utilized in the proofs of Theorems 3.1 and 3.2 no longer take into account the guards, this leads to a division process that continues for $2n$ steps irrespective of the combination of colours – very similar to the case of tissue P systems. An illustration of the differences in execution time between skP systems with and without guards attached to division rules is shown in Figure 5 for $n = 4, 6, 8$.

In certain cases, n can be increased even more. For some examples of graphs for which there is no solution, which implies that the number of compartments might be smaller – for instance, for complete graphs with at least four vertices – the simulator has produced good results for $n = 25$ and 30.

The two specific graphs $G(n, 1)$ and $G(n, n)$ have been tested, by using the algorithm from the proof of Theorem 3.1; the results show that although for a smaller number of vertices ($n = 4$)

Table 2. Simulation results.

n	Model	Time	Membranes	Steps
4	kernel 1	49	49	10
4	kernel 2	53	34	22
4	tissue	1200	82	28
6	kernel 1	63	103	14
6	kernel 2	125	70	32
6	tissue	24,086	730	33
8	kernel 1	325	283	18
8	kernel 2	500	214	42
8	tissue	632,092	6562	41

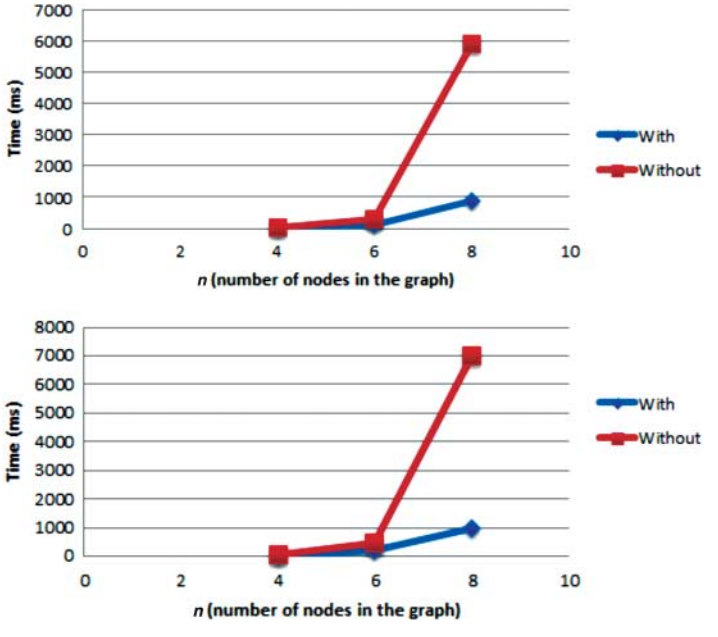

Figure 5. Times in models kernel 1 and kernel 2, with and without s in guards.

they are both executed within less than 0.03 ms, for $n = 10$ the algorithm for the first graph runs for 7 s, whereas the implementation for $G(n, n)$ requires more than 4 min.

A final remark regarding a further substantial improvement, with minimal implementation costs, refers to a mechanism for stopping the use of compartments which no longer produce a solution and no rules are applicable there. Although, in theory, a compartment where no rule is applicable should not count in the execution of a system, it turns out that simply checking that some rules are not applicable might introduce an overhead that becomes significant when the number of such compartments is large and the rules have complex guards or their number is large. In order to alleviate this, a mechanism to stop evaluating a compartment where no rule is applicable has been considered, and this has been implemented in the current version of P-Lingua. A consequence of this approach is the possibility of completely removing these compartments and consequently reducing the number of overall compartments. This approach has been tested for complete graphs, where for $n > 3$, there is no solution. The tests have been executed for values up to $n = 30$ and, without the stopping mechanism, the algorithm runs for 25 s, whereas with the stopping option being included, it takes only 5 s and it still produces an answer for $n = 70$ in 208 s, that is, a little more than 3 min.

Lastly, here it is shown how a Spin model checker is used to formally verify certain properties of the 3-Col problem. Formal verification based on model checking has been used for various classes of P systems in connection with Maude [1] or Spin [4]. The use of Spin has been also used in association with a Kripke structure attached to a P system [10], for generating test sets for P system specifications [9] or for verifying P systems with active membranes [11]. In this paper, Spin is used as part of the methodology presented above for a more systematic study of various systems. In this case, Spin specifications, written in Promela, are automatically generated from P-Lingua. Some of the properties investigated here have also been presented in [12], but in [12], the translation from P-Lingua was obtained manually and the verification was performed only for very small values of n, the number of vertices of the graph.

The properties verified can be grouped into two categories. One group consists of properties that hold at (or near) the end of the computation. For example, it is verified that 'eventually, there will be either a YES or a NO into the environment' or that 'if in a membrane labelled 1 objects a, T are present, then in the next step YES will be sent out into the environment'. Properties enumerated by Observation 3.5 or Proposition 3.1, which hold after a certain number of steps, have also been verified. The formulae involved in all these properties have been calculated for graphs with six vertices and 8–12 edges, whereas the relation stated by Observation 3.5(b) has been checked for $n = 12$. These results are superior to those reported so far in the literature related to the verification of P systems. In [4], the number of compartments is 3 and the number of steps is bounded by 6, and in [12], the graph has only three nodes. The P system reported in [4] is also verified for a P system with a fixed structure, whereas in our case, the structure changes and finally reaches $3 \times 2^{n-1}$ compartments C_2 for Observation 3.2(b).

The second group of properties verified consists of some 'invariants'. One of them can be formulated as follows: if at a certain step, in a compartment with the label 2, there exists an edge with the two vertices of the same colour, then at the next step, this compartment would not divide anymore. Another invariant, similar to the previous one, but a bit more subtle, states that for any $i, 2 \leq i \leq n$, if there is a compartment C_2, such that it contains $A_i s$ at step st, then, at $st + 1$, there is a compartment C_2 (derived from the previous one through membrane division), containing $R_i s$, iff the graph has the neighbours of the vertex i coloured differently from red (i is red, R_i). Another similar invariant can be written for the colours green and blue. These invariants have been verified for graphs with six vertices and the second one has also been written as an assertion in the Promela code.

Some more results and comparisons regarding the simulations described above and details concerning the format of the queries used in model checking and the corresponding output can be found on the MeCoSim website [20] on the page related to 3-Col problem solved with skP systems.

5. Conclusions

The kP systems offer an unified and elegant way of integrating the established features of existing P system variants with new elements, valuable for formal modelling. This paper introduces a particular type of kP systems, called skP systems, and illustrates their expressive power and efficiency on the 3-Col problem. It presents two skP systems that model the problem and analyses them in terms of efficiency and complexity. The skP models prove to be more succinct (in terms of the number of rules, objects, number of cells and execution steps) than the corresponding tissue P system used in the current literature. The efficiency, in terms of execution time, of skP systems for modelling 3-Col problem is revealed through a number of examples that clearly show the benefits of using some of the characteristics of these systems. The experiments also suggest new features that further improve efficiency, such as a stopping mechanism associated with the execution of the system.

Overall, the paper takes a first step towards assessing the modelling power and efficiency of (s)kP systems; this will be continued in a future work on other, more complex, case studies. Future

work also involves the development of a platform for the simulation and formal verification of kP systems.

Acknowledgements

The authors thank the anonymous reviewers for their constructive comments and useful suggestions that significantly helped improving this paper. The work of FI, MG and RL was supported by a grant of the Romanian National Authority for Scientific Research, CNCS–UEFISCDI, project number PN-II-ID-PCE-2011-3-0688. The authors MPJ, LVC and MGQ acknowledge the support of the project TIN2009–13192 of the Ministerio de Ciencia e Innovación of Spain, cofinanced by FEDER funds, and the support of the Project of Excellence with Investigador de Reconocida Valía of the Junta de Andalucía, grant P08-TIC-04200. The author MGQ also acknowledges the support from the National FPU Grant Programme from the Spanish Ministry of Education.

References

[1] O. Andrei, G. Ciobanu, and D. Lucanu, *A rewriting logic framework for operational semantics of membrane systems*, Theoret. Comput. Sci. 373 (2007), pp. 163–181.
[2] M. Ben-Ari, *Principles of the Spin Model Checker*, Springer, London, 2008.
[3] G. Ciobanu, M.J. Pérez-Jiménez, and G. Păun (eds.), *Applications of Membrane Computing*, Natural Computing Series, Springer, Berlin, 2006.
[4] Z. Dang, O.H. Ibarra, C. Li, and G. Xie, *On the decidability of model-checking for P systems*, J. Autom. Lang. Comb. 11 (2006), pp. 279–298.
[5] D. Díaz-Pernil, M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, and A. Riscos-Núñez, *A uniform family of tissue P systems with cell division solving 3-COL in a linear time*, Theoret. Comput. Sci. 404 (2008), pp. 76–87.
[6] D. Díaz-Pernil, I. Pérez-Hurtado, M.J. Pérez-Jiménez, and A. Riscos-Núñez, *A P-Lingua programming environment for membrane computing*, in *Membrane Computing – 9th International Workshop, WMC 2008, Revised Selected and Invited Papers*, D.W. Corne, P. Frisco, Gh. Păun, G. Rozenberg, and A. Salomaa, eds., Lecture Notes in Computer Science Vol. 5391, Springer, Berlin, 2009, pp. 187–203.
[7] M. Gheorghe, F. Ipate, and C. Dragomir, *A kernel P system*, in *Proceedings of 10th Brainstorming Week on Membrane Computing*, M.A. Martínez-del-Amor, Gh. Păun, I. Pérez-Hurtado, and F.J. Romero-Campero, eds., Fénix Editora, Seville, Spain, 2012, pp. 153–170.
[8] M. Gheorghe, G. Păun, M.J. Pérez-Jiménez, and G. Rozenberg, *Research frontiers of membrane computing: Open problems and research topics*, Int. J. Found. Comput. Sci. (2012), submitted.
[9] F. Ipate, M. Gheorghe, and R. Lefticaru, *Test generation from P systems using model checking*, J. Log. Algebr. Program. 79 (2010), pp. 350–362.
[10] F. Ipate, R. Lefticaru, and C. Tudose, *Formal verification of P systems using Spin*, Int. J. Found. Comput. Sci. 22 (2011), pp. 133–142.
[11] F. Ipate, R. Lefticaru, I. Pérez-Hurtado, M.J. Pérez-Jiménez, and C. Tudose, *Formal verification of P systems with active membranes through model checking*, in *International Conference on Membrane Computing*, M. Gheorghe, Gh. Păun, G. Rozenberg, A. Salomaa, and S. Verlan, eds., Lecture Notes in Computer Science Vol. 7184, Springer, Berlin, 2011, pp. 215–225.
[12] F. Ipate, C. Dragomir, R. Lefticaru, L. Mierlă, and M.J. Pérez-Jiménez, *Using a kernel P system to solve the 3-Col problem*, in *Pre-Proceedings of the 13th International Conference on Membrane Computing*, E. Csuhaj-Varjú, M. Gheorghe, and G. Vaszil, eds., Computer and Automation Research Institute, Hungarian Academy of Sciences, 2012, pp. 243–258.
[13] R. Lefticaru, F. Ipate, L. Valencia Cabrera, A. Ţurcanu, C. Tudose, M. Gheorghe, M.J. Pérez-Jiménez, I.M. Niculescu, and C. Dragomir, *Towards an integrated approach for model simulation, property extraction and verification of P systems*, in *Proceedings of 10th Brainstorming Week on Membrane Computing*, M.A. Martínez-del-Amor, Gh. Păun, I. Pérez-Hurtado, and F.J. Romero-Campero, eds., Fénix Editora, Seville, Spain, 2012, pp. 291–318.
[14] R. Nicolescu, M.J. Dinneen, and Y.B. Kim, *Structured modelling with hyperdag P systems*, in *Membrane Computing, Seventh Brainstorming Week, BWMC 2009*, R. Gutiérrez-Escudero, M.A. Gutiérrez-Naranjo, Gh. Păun, I. Pérez-Hurtado, and A. Riscos-Núñez, eds., vol. Part A, Universidad de Sevilla, 2009, pp. 85–108.
[15] G. Păun, *Computing with membranes*, J. Comput. Syst. Sci. 61 (2000), pp. 108–143.
[16] G. Păun, *Membrane Computing: An Introduction*, Springer, Berlin, 2002.
[17] G. Păun, G. Rozenberg, and A. Salomaa (eds.), *The Oxford Handbook of Membrane Computing*, Oxford University Press, Oxford, 2010.
[18] I. Pérez-Hurtado, *P-Lingua webpage,* 2009. Available at http://www.p-lingua.org.
[19] I. Pérez-Hurtado, L. Valencia-Cabrera, M.J. Pérez-Jiménez, M.A. Colomer, and A. Riscos-Núñez, *MeCoSim: A general purpose software tool for simulating biological phenomena by means of P systems*, IEEE Fifth International Conference on Bio-inspired Computing: Theories and Applications (BIC-TA 2010), Hunan, China, I (2010), pp. 637–643.
[20] L. Valencia-Cabrera, *MeCoSim web site*, 2010. Available at http://www.p-lingua.org/mecosim.