Computación y Sistemas Vol. 5 No. 2 pp. 120 - 127
© 2001, CIC - IPN. *ISSN 1405-5546* Impreso en México

On the Power of P Systems With Valuations

Sobre la Potencia de los Sistemas P con Valoraciones

Carlos Martín Vide[1], **Victor Mitrana**[2] and **Gheorghe Păun**[3]
[1]Research Group on Mathematical Linguistics, Rovira i Virgili University
Pl. Imperial Tárraco 1, 43005 Tarragona, Spain
[2]Faculty of Mathematics, University of Bucharest
Str. Academiei 14, 70109 Bucuresti, Romania
[3]Institute of Mathematics of the Romanian Academy
PO Box 1-176, 70600 Bucuresti, Romania
e-mail : cmv@astor.urv.es, mitrana@funinf.math.unibuc.ro, gpaun@imar.ro

Article received on February 12, 2001; accepted on October 31, 2001

Abstract

We consider some slight variants of P systems with valuation as introduced in Martín-Vide and Mitrana (2000) and we study their generative power and computational efficiency. When rewriting takes place only in the ends of the strings, each recursively enumerable language can be generated by a system with only two membranes. If the string replication is allowed (when the valuation of a string allows the string to go to several membranes, then copies of the string are sent to all these membranes), then NP-complete problems can be solved in linear time. We prove this for HPP (the existence of a Hamiltonian path in a directed graph) and give some informal explanations on a similar solution for SAT.

Keywords: P System, Membrane, Valuation, Polarization, Valence Grammar, Hamiltonian Path Problem.

Resumen

Consideramos pequeñas variantes de sistemas P con valoración tal como se presentan en Martín-Vide y Mitrana (2000), y estudiamos su capacidad generativa y su eficiencia computacional. Cuando la reescritura tiene lugar solamente en las partes finales de las cadenas, todo lenguaje recursivamente enumerable puede ser generado por medio de un sistema con sólo dos membranas. Si se permite la replicación de la cadena (cuando la valoración de una cadena permite a ésta ir a varias membranas, se envían copias de la cadena a todas ellas), entoces se pueden resolver el problemas NP-completos en tiempo lineal. Probamos esto para HPP (la existencia de un camino hamiltoniano en un grafo orientado) y ofrecemos algunos comentarios informales sobre una solución similar para el SAT.

Palabras clave: Sistema P, Membrana, Valoración, Polarización, Gramática de Valencias, Problema de Camino Hamiltoniano.

1 Introduction

P systems are a class of computing models abstracting from the way in which the alive cell works. Informally, in the regions delimited by a membrane structure (a membrane is understood as a three dimensional vesicle), certain objects evolve according to given rules; both the objects and the rules are associated with the regions. The objects can also pass through membranes, sometimes the membranes can dissolve or divide. In this way, transitions between configurations of the system are obtained; a sequence of transitions is a computation, and the result of a (halting) computation consists of all objects which leave the system during the computation.

Two main classes of P systems can be considered: with objects described by symbols (then we work with multisets of symbols placed in regions), or with objects described by strings of symbols (then we can work with sets or with multisets of strings placed in regions). There are many types of systems dealing with string-objects; see, e.g., (Bottoni et al., 2000; Calude and Păun, 2000; Păun, 2000; Freund et al., 2000; Krishna and Rama, 2000; Martín-Vide and Păun (RIMS), 2000; Martín -Vide and Păun (EATCS), 2000; Zandron et al., 2000). An up-to-date bibliography of the area can be found at the web address `http://bioinformatics.bio.disco.unimib.it/psystems`. We consider here the variant introduced in Martín-Vide and Mitrana (2000), where the strings are processed by point mutations (including insertion rules of the form $\lambda \to a$) applied only in the end of the string, and communicated from a region to another one according to their ***valuation***: a morphism from the alphabet of the strings into the set of integer numbers. We say that a string has a positive, negative, or neutral "charge" if its valuation is positive, negative, or zero, respectively; the membranes also have "charges" given in the initial configuration; a string which is positively or negatively charged will go to a membrane of the opposite charge, nondeterministically chosen in the case of multiple targets, while a neutral object will go to a membrane having a neutral charge.

Such systems were proven in the aforementioned paper to be able to solve SAT and HPP in linear time, providing that we start from a given initial configuration with an exponentially large number of membranes. The language generating power of systems from (Martin-Vide and Mitrana, 2000) is not known. In some sense, the problem is of a limited interest for the variant considered there, because of the many restrictions imposed on the functioning of the systems: the allowed rules are point mutation rules, and the deletion and insertion rules are always used in the right hand end of the strings.

We slightly modify here the definition from (Martin-Vide and Mitrana, 2000), in two ways.

First, we allow arbitrary context-free rules to be used, but only for rewriting the first or the last symbol of a string. This simple and natural change is already sufficient in order to obtain computational completeness: a characterization of recursively enumerable languages is obtained by means of systems with only two membranes. Also, the polarization of membranes is very restricted, it can be only positive or negative; however, an object can have neutral charge only in the skin membrane, and then it has to leave the system, in any other membrane it is destroyed. It is also worth mentioning that we obtain the completeness in the *non-extended case*, that is, without using a terminal alphabet for squeezing the generated language.

Second, we allow string replication: if a string is polarized and there are several neighboring membranes of the opposite polarity, then a copy of the string is sent to each of these membranes. As expected, this is an efficient way to get exponential working space, and NP-complete problems can be solved in polynomial time in this framework. Actually, a linear time suffices for SAT (satisfiability of propositional formula in the conjunctive normal form) and HPP (whether or not a directed graph contains a Hamiltonian path).

2 P Systems with Valuation

We introduce here only the class of P systems that we will investigate in this paper. As usual, a membrane structure is represented by a string of labeled parentheses, and with each membrane we associate a *region*, which is referred to by the label of the membrane, and an electrical charge $+$ or $-$. For an alphabet V we denote by V^* the free monoid generated by V under the operation of concatenation; λ is the empty string.

A *rewriting P system* (of degree $m \geq 1$) with *valuation* is a construct

$$\Pi = (V, \mu, M_1, \ldots, M_m, R_1, \ldots, R_m, \varphi),$$

where:

1. V is the alphabet of the system;
2. μ is a membrane structure with m membranes, having associated electrical charges $+$ and $-$ (always in this paper the membranes are injectively labeled by $1, 2, \ldots, m$);
3. $M_1, \ldots, M_m$ are finite languages over V, representing the strings initially present in the regions $1, 2, \ldots, m$ of the system;
4. $R_1, \ldots, R_m$ are finite sets of context-free rules over V associated with the regions of μ;
5. φ is a morphism from the monoid $(V^*, \cdot, \lambda)$ to the additive monoid $(\mathbf{Z}, +, 0)$ (called the *valuation mapping* of Π).

In a system as above, transitions are defined as usual in rewriting P systems (in each region, each string which can be rewritten by a rule from that region is rewritten), with the following two differences: (1) the rules are always applied only in the ends of the strings, rewriting either the first or the last symbol of each string; (2) the communication of strings from a region to another one is controlled by the charges of membranes and of strings, in the following way. If $\varphi(x) > 0$, then we say that the string is positively charged, if $\varphi(x) < 0$, then the string is negatively charged, if $\varphi(x) = 0$, then the string has no charge. If a non-charged string is produced in the skin membrane, then it leaves the system, in any other membranes such a string "vanishes". If in a given region, associated with a membrane of a given charge α, we produce a string of the opposite charge, then it remains in the same region (and it can be rewritten again). If we produce a string of the same charge α, then the string must go to any of the adjacent membranes of the opposite polarity, nondeterministically chosen. If there is no such an adjacent membrane, then the string "vanishes".

As usual, a sequence of transitions forms a computation and the result of a halting computation is the set of strings over V sent out of the system during the computation. A computation which never halts yields no result. A string which remains inside the system does not contribute to the generated language. The language generated in this way by a system Π is denoted by $L(\Pi)$.

The family of all languages $L(\Pi)$, computed as above by rewriting P systems Π of degree at most $m \geq 1$ with valuation is denoted by $VRP_m(left/right)$. When, in all regions rewriting is allowed in the left hand or in the right hand end of the strings only, we write $VRP_m(left)$ or $VRP_m(right)$, respectively. We denote by REG and RE the families of regular and recursively enumerable languages, respectively.

3 Computational Completeness

We start with a result stating the computational power of P systems with valuation working only in the right or only in the left hand end of strings. To this aim, we recall the definition of grammars with valences in an arbitrary group, as a generalization of that in Păun (1980).

A context-free/regular grammar with valences in the group $\mathbf{K} = (M, +, e)$ is a 6-tuple $G = (N, T, S, P, \mathbf{K}, v)$, where (N, T, S, P) is a context-free/regular grammar and v is a mapping from P into M. If $P = \{r_1, \ldots, r_n\}$, for some n, then the language generated by G, denoted by $L(G)$, consists of all words x such that there is a derivation

$$S \Longrightarrow_{i_1} x_1 \Longrightarrow_{i_2} x_2 \ldots \Longrightarrow_{i_p} x_p = x$$

with $\sum_{j=1}^{p} v(r_{i_j}) = e$. As one can easily see, there is a strong similarity between the generating way in a grammar with valences in the additive group of integers and the computational way in a P system with valuation. We denote by $REG(\mathbf{K})$ the family of languages generated by regular grammars with valences in the group $\mathbf{K}$.

Proposition 1. 1. $REG \subset VRP_1(\alpha), \alpha \in \{left, right\}$.
2. $VRP_k(\alpha) - REG(\mathbf{K}) \neq \emptyset$, $\alpha \in \{left, right\}$, for any $k \geq 2$ and any commutative group $\mathbf{K}$.

Proof. 1. Let $G = (N, T, S, P)$ be a right-linear grammar. Consider the P system with only one membrane, positively charged, containing the string S and all the rules in P. The valuation mapping φ associates the value 0 with each terminal, and the value -1 with each nonterminal. Clearly, the language computed by this P system is the language generated by the grammar G, hence we have the inclusion $REG \subseteq VRP_1(right)$. If we start from a left-linear grammar G, then we get the inclusion $REG \subseteq VRP_1(left)$.

In order to prove the properness of these inclusions, we define the P systems with one membrane positively charged, containing the string A and the rules $A \to aA$, $A \to aB$, $B \to bB$, $B \to \lambda$, in the case of $VRP_1(right)$, and $A \to Aa, A \to Ba, B \to Bb, B \to \lambda$, in the case of $VRP_1(left)$. We define the valuation mapping φ by $\varphi(a) = -2$, $\varphi(b) = 2$, $\varphi(A) = \varphi(B) = -1$. It is easy to note that both systems compute the language $\{a^n b^n \mid n \geq 1\}$, which is not regular.

2. Consider the P system

$$\Pi = (\{a, b, A, B\}, [_1 [_2 \,]_2^-]_1^+, \{A\}, \emptyset, R_1, R_2, \varphi),$$

where

$$R_1 = \{A \to aA, A \to aB, B \to bB, B \to b\},$$
$$R_2 = \{B \to A\},$$

and

$$\varphi(a) = -2,\ \varphi(b) = 2,$$
$$\varphi(A) = -1,\ \varphi(B) = 1.$$

The computational process starts in the skin membrane, where a string of the form $a^n b^m B$ is produced. This string can go to membrane 2 if and only if $n = m$. In membrane 2, the rightmost symbol B is replaced by A and the string migrates to the skin membrane where a new suffix of the same form is generated. By a repeated migration between the two membranes, a string in the language $\{a^n b^n \mid n \geq 1\}^+$ is finally sent out of the system. Moreover, all the strings in this language can be computed. From (Mitrana and Stiebe, 2001) it is known that no regular grammar with valences in any commutative group can generate this language.

By reversing the rules, we can get a system Π' such that $L(\Pi') = \{a^n b^n \mid n \geq 1\}^+$, which also proves that $VRP_k(left) - REG(\mathbf{K}) \neq \emptyset$. $\square$

It is an *open problem* whether or not the families $VRP_k(left), VRP_k(right)$ are equal, $k \geq 1$.

We continue by giving the main result of this paper, which shows that the systems above defined, able to work in both ends of strings, are computationally complete.

Theorem 1. *$RE = VRP_2(left/right)$.*

Proof. We only have to prove the inclusion $RE \subseteq VRP_2(left/right)$ since the converse inclusion follows from the Church-Turing thesis. Let us consider a type-0 Chomsky grammar $G = (N, T, S, P)$ in Kuroda normal form (Rozenberg and Salomaa, 1997), that is, with rules of two forms: $A \to x$ (context-free rules), $A \in N$, $x \in (N \cup T)^*$, and $AB \to CD$, for $A, B, C, D \in N$. Assume that $N \cup T = \{\alpha_1, \alpha_2, \ldots, \alpha_{n-1}\}$, and take one more symbol, $\alpha_n = \$$. Also, assume that the non-context-free rules from P are injectively labelled, $r : AB \to CD$, with $r \in H$.

We construct the P system (of degree 2)

$$\Pi = (V, [_1 [_2 \,]_2^+]_1^-, \emptyset, \{X\$S\}, R_1, R_2, \varphi),$$

with the alphabet

$$\begin{aligned} V \;=\;& N \cup T \cup \{X, X', Z, \$\} \cup \{X^{(r)} \mid r \in H\} \\ \cup\;& \{(X_i, j), (X'_i, j), (X_i^{(r)}, j), \overline{(X_i, j)}, \overline{(X'_i, j)}, \\ & \overline{(X_i^{(r)}, j)} \mid 1 \leq j \leq i \leq n, r \in H\} \\ \cup\;& \{(\alpha_i, j), \overline{(\alpha_i, j)}, \alpha'_i \mid 1 \leq j \leq i \leq n\}, \end{aligned}$$

the valuation mapping

$$\varphi(\alpha_i) = 0,\ 1 \leq i \leq n,$$
$$\varphi(X) = \varphi(X') = \varphi(Z) = -1,$$
$$\varphi(X^{(r)}) = -1,\ r \in H,$$

$$\varphi((X_i,j)) = \varphi((X_i',j)) = \varphi(\overline{(X_i,j)}) =$$
$$\varphi(\overline{(X_i',j)}) = -2j-1, 1 \leq j \leq i \leq n,$$

$$\varphi((X_i^{(r)},j)) = \varphi(\overline{(X_i^{(r)},j)}) = -2j-1,$$
$$1 \leq j \leq i \leq n, r \in H,$$

$$\varphi((\alpha_i,j)) = \varphi(\overline{(\alpha_i,j)}) = 2j, 1 \leq j \leq i \leq n,$$
$$\varphi(\alpha_i') = 2(i+1), 1 \leq i \leq n,$$

and with the following sets of rules:

R_1 :
- $X \to (X_i,1)$, for all $1 \leq i \leq n$,
- $X' \to (X_i',1)$, for all $1 \leq i \leq n$,
- $X^{(r)} \to (X_i^{(r)},1)$, for all $1 \leq i \leq n, r \in H$,
- $\overline{(X_i,j)} \to (X_i,j+1)$, and
- $\overline{(X_i',j)} \to (X_i',j+1)$,
  for all $1 \leq j < i \leq n$,
- $\overline{(X_i^{(r)},j)} \to (X_i^{(r)},j+1)$,
  for all $1 \leq j < i \leq n, r \in H$,
- $(\alpha_i,j) \to \overline{(\alpha_i,j)}$, for all $1 \leq j \leq i \leq n$,
- $\alpha_i' \to \lambda$, for all $1 \leq i \leq n$,
- $\overline{(X_i,i)} \to X\alpha_i$, for all $1 \leq i \leq n$,
- $\overline{(X_n,n)} \to X'\$$,
- $\overline{(X_i,i)} \to X^{(r)}$, if $r : AB \to CD$
  and $B = \alpha_i$,
- $\overline{(X_i^{(r)},i)} \to XCD$, if $r : AB \to CD$
  and $A = \alpha_i$,
- $\overline{(X_i',i)} \to X'\alpha_i$, for all $1 \leq i \leq n$
  such that $\alpha_i \in T$,
- $\overline{(X_i',i)} \to Z$, for $\alpha_i \in N, 1 \leq i \leq n$,
- $\overline{(X_n',n)} \to \lambda$.

R_2 :
- $A \to x$, if such a rule exists in P,
- $\alpha_i \to (\alpha_i,1)$, for all $1 \leq i \leq n$,
- $\overline{(\alpha_i,j)} \to (\alpha_i,j+1)$,
  for all $1 \leq j < i \leq n$,
- $(X_i,j) \to \overline{(X_i,j)}$, and
- $(X_i',j) \to \overline{(X_i',j)}$, for all $1 \leq j \leq i \leq n$,
- $(X_i^{(r)},j) \to \overline{(X_i^{(r)},j)}$,
  for all $1 \leq j \leq i \leq n, r \in H$,
- $\overline{(\alpha_i,i)} \to \alpha_i'$, for all $1 \leq i \leq n$.

Let us examine the work of this system. Assume that in membrane 2 we have a string Xw; initially, we have here $X\$S$ (with $\varphi(X\$S) = -1$). In any moment, only one string is present in the system.

If we use a rule of the form $A \to x$ (this is possible only in the rightmost position of the string), then the polarity of the string remains the same, hence the string remains in this membrane. If we use a rule of the form $\alpha_i \to (\alpha_i,1)$ (again, in the rightmost position), then the polarity of the string becomes +, hence the string goes to the skin membrane. Here we have to apply two rules: $(\alpha_i,1) \to \overline{(\alpha_i,1)}$ and $X \to (X_j,1)$, for some j, in this order. The application of the former rule does not change the polarity of the string, hence it has to be followed by the application of the latter rule. If we first apply the rule $X \to (X_j,1)$, then the string goes to membrane 2 (it has a negative charge) and remains there forever after applying the rule $(X_j,1) \to \overline{(X_j,1)}$. In this case no output is produced.

In the second membrane we have to use the rules $(X_j,1) \to \overline{(X_j,1)}$ and $\overline{(\alpha_i,1)} \to (\alpha_i,2)$ in this order. In a similar way as above, if we first apply the rule $\overline{(\alpha_i,1)} \to (\alpha_i,2)$, then the string goes to the skin membrane where the only possibility to continue is to apply the rule $(\alpha_i,2) \to \overline{(\alpha_i,2)}$. Now, no rule can be applied anymore; since the string polarity is still positive, the string is blocked in this membrane.

In this way, the string will go back and forth between membranes 1 and 2, at each passing towards the skin membrane increasing the second component of the symbol of the form (α_i,k) and at each passing towards the second membrane increasing the second component of the symbol of the form (X_j,k).

We now distinguish three cases:

Case 1: $i < j$. At some point we reach the situation where in the second membrane we have the string $(X_j,i)w(\alpha_i,i)$. We cannot first apply the rule $\overline{(\alpha_i,i)} \to \alpha_i'$ because the string would go to the skin membrane, where its rightmost symbol α_i' would be deleted (this is the only possibility). The string now has a negative polarity and goes to the second membrane where a rule of the form $\alpha_l \to (\alpha_l,1)$ is applied. Since the string polarity is still negative and no rule can be further applied, no output is observed.

It follows that we first have to apply the rule $(X_j,i) \to \overline{(X_j,i)}$ and, because the string is still negatively charged, then the rule $\overline{(\alpha_i,i)} \to \alpha_i'$. Now, the string becomes positively charged and goes to the skin membrane. Here we have two possibilities. If we start by deleting the rightmost symbol α_i', then the string returns to the second membrane where a new rule of the form $\alpha_l \to (\alpha_l,1)$ can be applied, but after that it remains here forever.

If we start by using the rule $\overline{(X_j,i)} \to (X_j,i+1)$, then the string goes to the second membrane where it is blocked after replacing $(X_j,i+1)$ by $\overline{(X_j,i+1)}$.

In conclusion, if $i < j$, then no output is produced.

Case 2: $i > j$. At some moment, in the second membrane we shall have a string of the form

$(X_j, j)w\overline{(\alpha_i, j)}$. As we have already seen, in two rewriting steps we obtain the string $\overline{(X_j, j)}w(\alpha_i, j+1)$ which goes to the skin membrane where, no matter the order, the symbol $\overline{(X_j, j)}$ is replaced by one of the rules with this symbol in its left hand side. Since the new string cannot get a negative polarity, even after using a rule of the form $X \to (X_l, 1)$, it remain in this membrane forever. Therefore, no output is produced when $i > j$.

Case 3: $i = j$. After receiving the string $(X_i, i)w\overline{(\alpha_i, i)}$ in the second membrane, we use here the rules $(X_i, i) \to \overline{(X_i, i)}$ and $\overline{(\alpha_i, i)} \to \alpha_i'$, in this order. Indeed, if we first use the rule $\overline{(\alpha_i, i)} \to \alpha_i'$, then the string goes to the skin membrane, where α_i' is deleted, the string returns to the second membrane where it remains forever no matter which rules are used.

The new string, $\overline{(X_i, i)}w\alpha_i'$, is sent to the skin membrane where first $\overline{(X_i, i)}$ is replaced by $X\alpha_i$ and then α_i' is deleted. If this order is inverted, the string will be blocked again in the second membrane.

Another wrong possibility is to first replace $\overline{(X_i, i)}$ by $X\alpha_i$, then X by $(X_l, 1)$, for some l, and then to delete α_i'. Also in this case the string will go to the second membrane where it will remain forever, no polarity change being possible.

In this way, the symbol α_i was cut from the right hand end of the string and introduced in the left hand end of the string. That is, the string is circularly permuted with a symbol.

The previous procedure can be repeated, hence the string can be circularly permuted with any number of symbols. Because the symbol \$ is treated as any other symbol, we always know when the string is completely permuted.

In this way, we can simulate the context-free rules of P in any position of the sentential forms of G.

If we want to simulate a rule $r : AB \to CD$ from P we proceed in the following manner. As above, we cut the symbol $\alpha_i = B$ from the right hand end of the string. When the procedure is completed, instead of using the rule $\overline{(X_i, i)} \to X\alpha_i$, we use the rule $\overline{(X_i, i)} \to X^{(r)}$. The process continues in the presence of $X^{(r)}$ as in the presence of X, but after cutting one more symbol, say α_j, from the right hand end of the string, we will introduce the symbols CD in the left hand end of the string, by using the rule $\overline{(X_j^{(r)}, j)} \to XCD$, providing that $\alpha_j = A$.

Since the process above can be resumed now in the aim of simulating another rule of G, all derivations in G can be simulated in Π, working in the ends of the strings.

In order to terminate the computation and send out a string, we will use the symbol X'. It is introduced by the symbol $\overline{(X_n, n)}$, hence after completely rotating the string. The same rotation procedure is repeated in the presence of X' and its paired variants (X_i', j) and $\overline{(X_i', j)}$, but this time only terminal symbols can be moved from an end of the string to the other end; if a nonterminal symbol is met, then the computation is blocked, and we get no output. When we complete a circular permutation, hence we have cut \$ from the right end of the string, we remove the symbol $\overline{(X_n', n)}$, which leads to a string of neutral polarity in the skin membrane. It leaves the system. From the way the last permutation was done, we know that this string is composed only of terminal symbols.

In conclusion, $L(\Pi)$ contains exactly the terminal strings which can be generated by G, that is, $L(\Pi) = L(G)$. □

We close this section with a few *open problems*:

1. Is the previous result optimal, or the number of used membranes can be decreased?

2. Characterize the family $VRP_1(left/right)$. Are there non-context-free languages which do not lie in $VRP_1(left/right)$?

3. Are there languages generated by regular grammars with valences in the additive group of integers which do not belong to $VRP_k(right)$ for any k?

4 Solving NP-Complete Problems

In this section we shall provide a P decision algorithm for solving the HPP in linear time. By a P decision algorithm for a problem we mean a procedure to construct, for a given instance of the problem, a P system which produces an output if and only if that instance has a solution. Before starting our construction, we remember that the P systems used here allow string replication: if a string is charged and there are several neighboring membranes of the opposite polarity, then a copy of the string is sent to each of these membranes.

Theorem 2. *There is a P decision algorithm which solves HPP in linear time.*

Proof. Let $G = (\{c_1, c_2, \dots, c_n\}, E)$ be a directed graph such that, for each $1 \le i \le n$, $c_{i_1}, c_{i_2}, \dots, c_{i_{j_i}}$ are all the descendents of the node c_i. Assume that $k = \max\{j_1, j_2, \dots, j_n\}$ (the maximal outdegree of the graph nodes). We construct the P system

$$\Pi = (V, \mu, \emptyset, \{X^{(\emptyset)}a_i^{(0)} \mid 1 \le i \le n\}, \emptyset, \dots, \emptyset, (R_s)_{1 \le s \le k+2n+2}, \varphi),$$

where the membrane structure is given by

$$\mu = [_1[_2[_3\ ]_3^-[_4\ ]_4^- \cdots [_{k+2}\ ]_{k+2}^-[_{k+3}[_{k+n+3}\ ]_{k+n+3}^+\ ]_{k+3}^- \cdots [_{k+n+2}[_{k+2n+2}\ ]_{k+2n+2}^+\ ]_{k+n+2}^-]_2^+]_1^-,$$

and schematically represented in Figure 1,

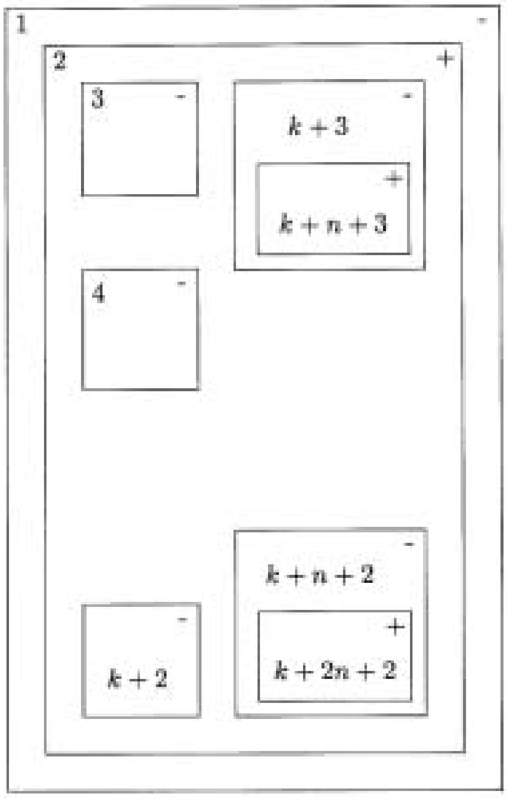


Figure 1: The membrane structure

the sets of rules are defined by

$$\begin{aligned}
R_1 &= \{c \to \lambda, Y \to \lambda\}, \\
R_2 &= \{X^{(\{1,2,\ldots,n\})} \to Y\} \\
&\cup \ \{a_i^{(t)} \to b_i^{(t)}, a_i^{(n+1)} \to c, c_i \to c_i', \\
&\quad \bar{c}_i \to \lambda \mid 1 \le i \le n, 0 \le t \le n\}, \\
R_{j+2} &= \{b_i^{(t)} \to c_i a_{i_p}^{(t+1)} \mid 1 \le i \le n, \\
&\quad 0 \le t \le n, (c_i, c_{i_p}) \in E\}, \\
&\quad 1 \le j \le k, \\
R_{k+j+2} &= \{c_j' \to \bar{c}_j, \bar{X}^{(M)} \to X^{(M)}\}, \\
&\quad 1 \le j \le n, M \subseteq \{1,2,\ldots,n\} \\
R_{k+n+j+2} &= \{X^{(M)} \to \bar{X}^{(M \cup \{j\})} \mid j \notin M\}, \\
&\quad 1 \le j \le n, M \subseteq \{1,2,\ldots,n\},
\end{aligned}$$

and the valuation mapping is defined by

$$\begin{aligned}
\varphi(X) &= \varphi(X^{(M)}) = -1, \\
&\quad \text{for any set } M \subseteq \{1,2,\ldots,n\}, \\
\varphi(Y) &= \varphi(\bar{X}^{(M)}) = 1, \\
&\quad \text{for any set } M \subseteq \{1,2,\ldots,n\}, \\
\varphi(c_i) &= \varphi(\bar{c}_i) = \varphi(a_i^{(t)}) = 0, \ 1 \le i \le n, \\
&\quad 0 \le t \le n+1, \\
\varphi(c) &= \varphi(b_i^{(t)}) = \varphi(c_i') = 2, \ 1 \le i \le n, \\
&\quad 0 \le t \le n.
\end{aligned}$$

The P system starts with all the strings $X^{(\emptyset)}a_i^{(0)}$, $1 \le i \le n$, in the second membrane. In parallel, each symbol $a_i^{(0)}$ is replaced by $b_i^{(0)}$. Since all the strings are positively charged, one copy of each is sent to all membranes $3, 4, \ldots, k+n+2$. The strings which arrive in membranes $k+3, k+4, \ldots, k+n+2$ remain there forever since no rewriting is possible, hence their polarity remain unchanged.

Let us examine what happens with the copies of one string, say $X^{(\emptyset)}b_i^{(0)}$, which arrive in membranes $3, 4, \ldots, k+2$, respectively. They will produce the negatively charged strings $X^{(\emptyset)}c_i a_{i_t}^{(1)}$, $1 \le t \le \delta^+(c_i)$, which return to membrane 2. Here $\delta^+(c)$ delivers the outdegree of the node c. This process continues for exactly $2n+1$ steps, when in the second membrane we have all the strings $X^{(\emptyset)}wa_s^{(n+1)}$, for some $1 \le s \le n$, but with w a string of length n over $\{c_1, c_2, \ldots, c_n\}$. Note that a symbol may appear repeated several times in w. We now apply the rules $a_s^{(n+1)} \to c$, in parallel, to all of these strings. The obtained strings are replicated and sent to all the negatively charged membranes, but only in the outermost membrane there exists a rule $c \to \lambda$ which makes possible to continue the computation by changing the strings polarity to a negative one. Therefore, all these strings, which do not end by c anymore, are sent back to the second membrane.

Now, the computation continues as follows. The rightmost symbol of each of these strings is replaced by its primed copy, the string is replicated and sent to all membranes $3, 4, \ldots, k+n+2$, but these copies will be blocked in membranes $3, 4, \ldots, k+2$. In each of the other membranes, at most one string will be further rewritten, by replacing its last primed symbol with a barred copy. Let us consider such a membrane, say $k+j+2$, where the string is duplicated and one copy is sent to the second membrane, while the other copy goes to the inner membrane, that is $k+n+j+2$. In the second membrane, the rightmost symbol is removed and the process continues. In membrane $k+n+j+2$, the string is blocked if and only if the superscript of the leftmost symbol, M, does satisfy $j \in M$, otherwise this symbol is replaced by $\bar{X}^{(M \cup \{j\})}$ and the string returns to membrane $k+j+2$. Here the first symbol is further replaced by $X^{(M \cup \{j\})}$ and one of its copy is sent back to membrane $k+n+j+2$, where it is blocked, and the other copy is sent to the second membrane where the rightmost symbol $\bar{c}_j$ is removed. By the following explanations, one can state that if this string represented initially a solution, then one of its copies would arrive in the second membrane and produce an output. Moreover, if this string represented initially no solution, no copy of it received by the second membrane would produce any output.

In this way, each string $X^{(M)}wc_j$ existing in the second membrane produces, in four steps, either two strings $X^{(M\cup\{j\})}w$ and $X^{(M)}w$, if $j \notin M$, or one string $X^{(M)}$, if $j \in M$. After $4n$ such steps in the second membrane we have the string $X^{(\{1,2,\ldots,n\})}$ if and only if the given instance of the HPP has a solution. Now, this string is replaced by Y, one copy of it goes to the skin membrane where it becomes λ and exits the system. Therefore, the given instance of the HPP has a solution if and only if the system produces a string, more precisely the empty string.

Consequently, we have a P decision algorithm which solves the HPP in linear time. It is worth mentioning that the number of membranes and the number of initial strings in regions are linearly bounded with respect to the size of the input graph. However, the cardinality of the alphabet V does not have this property. □

It is easy to see that we can solve in a similar way the HPP problem with given initial and final nodes: two nodes c_{in}, c_{out} in the graph are given and one asks whether or not a path, starting in c_{in}, ending in c_{out}, and passing exactly once through all nodes of the graph, exists.

Now, we can proceed as in Lipton (1995), reducing SAT to a problem related to HPP with given initial and final nodes (find all paths from a given node to another one, not visiting twice any node), hence a linear time solution to SAT can be obtained by means of a P decision algorithm.

5 Final Remarks

We have slightly modified the P systems with valuation introduced in (Martin-Vide and Mitrana, 2000) by allowing arbitrary context-free rules to be used (in the ends of strings). Such P systems are proven to be computationally complete (systems with only two membranes suffice in order to get a characterization of recursively enumerable languages), and also able to solve NP-complete problems in linear time (to this aim, an exponential space is created by replicating the strings: if a string has a negative or positive charge – given by the valuation – and there are several adjacent membranes of the opposite polarity, then copies of the string are sent to each of these membranes). The last result is proven for HPP, and indicated how to be extended to SAT. Some open problems are also formulated.

References

Bottoni P., A. Labella, C. Martin-Vide, and **Gh. Păun,** "Rewriting P systems with conditional communication", submitted, 2000.

Calude C., and **Gh. Păun,** *Computing with Cells and Atoms*, Taylor and Francis, London, 2000.

Freund R., C. Martin-Vide, and **Gh. Păun,** "Computing with membranes: Three more collapsing hierarchies", submitted, 2000.

Krishna S.N., and **R. Rama,** "On the power of P systems with sequential and parallel rewriting", *Intern. J. Computer Math.*, Vol. 77, No.1-2, 2000, pp.1–14.

Lipton R.J., "Using DNA to solve NP-complete problems", *Science*, Vol. 268 , April 1995, pp. 542–545.

Martin-Vide C., and **V. Mitrana,** "P systems with valuation". In Antoniou I, C. S. Calude, and M. J. Dinneen, editors *Unconventional Models of Computing. Proc. Second Conf. Unconventional Models of Computing*, Springer-Verlag, 2000, pp.154–166.

Martin-Vide C., and **Gh. Păun,** "String objects in P systems". *Proc. of Algebraic Systems, Formal Languages and Computations Workshop*, Kyoto, 2000, RIMS Kokyuroku, Kyoto Univ., 2000, pp. 161–169.

Martin-Vide C., and **Gh. Păun,** "Computing with membranes. One more collapsing hierarchy", *Bulletin of the EATCS*, Vol. 72, 2000, pp. 183–187.

Mitrana V., and **R. Stiebe,** "Extended finite automata over groups", *Discrete Applied Mathematics*, Vol. 108, No. 3, 2001, pp. 247–260.

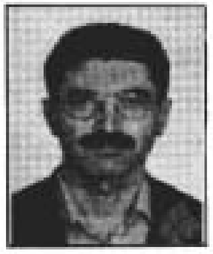
Păun Gh., "A new generative device: valence grammars", *Rev. Roum. Math. Pures et Appl.*, Vol. 25, No. 6, 1980, pp. 911–924.

Păun Gh., "Computing with membranes", *Journal of Computer and System Sciences*, Vol. 61, No.1, 2000, pp. 108–143 (see also *Turku Center for Computer Science-TUCS Report* No 208, 1998, www.tucs.fi).

Rozenberg G., and **A. Salomaa,** editors, *Handbook of Formal Languages*, Springer-Verlag, Heidelberg, 1997.

Zandron Cl., G. Mauri, and **Cl. Ferretti,** "Universality and normal forms on membrane systems. In Freund R., and A. Kelemenova, editori, *Proc. Intern. Workshop Grammar Systems 2000* , Bad Ischl, Austria, July 2000, 61–74.

Carlos Martin-Vide *(1955) is Professor and Head of the Research Group on Mathematical Linguistics at Rovira i Virgili University, Tarragona, Spain. His specialities are formal language theory and mathematical linguistics. His last volume edited is Where Mathematics, Computer Science, Linguistics and Biology Meet (Kluwer, 2001, with V. Mitrana). He published 125 papers in conference proceedings and journals such as: Acta Informatica, BioSystems, Computational Linguistics, Computers and Artificial Intelligence, Information Processing Letters, Information Sciences, International Journal of Computer Mathematics, Publicationes Mathematicae Debrecen, and Theoretical Computer Science. He is the editor-in-chief of the journal Grammars (Kluwer), and the chairman of the 1st International PhD School in Formal Languages and Applications (2001-2003).*



Victor Mitrana *received his PhD at the Faculty of Mathematics, University of Bucharest in 1994. He is now a full professor at the Department of Fundamentals of Computer Science at the Faculty of mathematics, University of Bucharest. He published three books, co-edited two books and published more than 100 papers in journals, books, and conference proceedings. His main interests are: automata and formal languages, molecular computing, ombinatorial algorithms.*

Gheorghe Păun *(born on December 6, 1950) graduated at the Faculty of Mathematics of Bucharest University in 1974, and got his PhD at the same faculty in 1977. He has won many scholarships, in Germany (from Alexander von Humboldt Foundation, in 1992-93), Finland, The Netherlands, Spain, etc. Presently he is a senior researcher in the Institute of Mathematics of the Romanian Academy, Bucharest, and a Corresponding Member of the Romanian Academy. His main research fields are formal language theory (regulated rewriting, contextual grammars, grammar systems), automata theory, combinatorics on words, computational linguistics, DNA computing, membrane computing (this last area was initiated by him in 1998). He has (co)authored and (co)edited more than two dozens of books in these areas, and he has (co)authored more than 350 research papers. In the last decade he has visited many universities from Europe, USA, Canada, Japan, also participating to many international conferences, several times as an invited speaker. He is a member of the editorial board of several computer science journals and professional associations. Hobbies: logical games and literature (he has published several books also in these fields, in Romanian).*