

Eilenberg P Systems

Tudor Bălănescu¹, Marian Gheorghe², Mike Holcombe², and Florentin Ipate¹

¹ Faculty of Sciences, Pitești University
Str. Targu din Vale 1, 0300 Pitești, Romania
f.ipate@ifsoft.ro

² Department of Computer Science, Sheffield University
Regent Court, Portobello Street, Sheffield, S1 4DP, UK
{m.gheorghe,m.holcombe}@dcs.shef.ac.uk

Abstract. A class of P systems, called EP systems, with string objects processed by evolution rules distributed alongside the transitions of an Eilenberg machine, is introduced. A parallel variant of EP systems, called EPP systems, is also defined and the power of both EP and EPP systems is investigated in relationship with three parameters: number of membranes, states and set of distributed rules. It is shown that EPP systems represent a promising framework for solving NP-complete problems. In particular linear time solutions are provided for the SAT problem.

1 Introduction

P systems were introduced in the history making paper [21] by Gh. Păun. One of the main classes of P systems is that of string objects where the evolution rules are defined as string rewriting operations (see [22,21,6,18,19,20]) (an up-to-date bibliography of the whole area may be found at the web address <http://psystems.disco.unimib.it>). Because rewriting alone even in the context of a highly parallel environment of a membrane structure is not enough to lead to characterizations of recursively enumerable languages, various other features have been considered, such as a *priority relationship* over the set of rules, *permitting* or *forbidding* conditions associated with rules, *restrictions on the derivation mode*, the possibility to control the *membrane permeability* [6] etc (for more details see [22]). In general the most used priority relationship on the set of rewriting rules is a partial order relationship, well studied in the context of generative mechanisms with restrictions in derivation [4]. In this paper the priority relationship will be replaced by a transition diagram associated with an Eilenberg machine giving birth to two classes of Eilenberg systems, a sequential version and a parallel one, called *EP systems* and *EPP systems*, respectively. In both variants, each transition has a specific set of evolution rules acting upon the string objects contained in different regions of the membrane system. The system will start in a given state and with an initial set of string objects. Given a state and a current set of string objects, in the case of EP systems, the machine will evolve by applying rules associated with one of the transitions going out from the current state. The system will resume from the destination state of the

current transition. In the parallel variant, instead of one state and a single set of string objects we may have a number of states, called *active states*, that are able to trigger outgoing transitions and such that each state hosts a different set of string objects; all the transitions emerging from every active state may be triggered once the rules associated with them may be applied; then the system will resume from the next states, which then become active states. EP systems are models of cells evolving under various conditions when certain factors may inhibit some evolution rules or some catalysts may activate other rules. EPP systems introduce a parallel behaviour of the system in respect of the transitions emerging from active states, model cellular division and parallel development of the new born cells as well as cell collision when multiple transitions join a target state. The EP model has some similarities with the grammar systems controlled by graphs [3], replacing a one-level structure, which is the current sentential form, with a hierarchical structure defined by a membrane element. It is also close to the state based model defined by Ji [15] in relation to the living cell, called molecular machine. Whereas Ji's molecular machine is a sort of Mealy machine [17], the proposed model penetrates inside of the cell as usually P systems do. On the other hand, this P system variant may be viewed as an Eilenberg machine [5] having sets of evolution rules as basic processing relationships. EP systems are also similar to the Eilenberg machines based on distributed grammar systems [8]. Eilenberg machines, generally known under the name of X machines [5], have been initially used as a software specification language [10], further on intensively studied in connection with software testing [13], but also utilized as a model of metabolic pathways [11], bee colony behaviour [9], or reactive agents [16]; a survey of the whole area at the end of 2000 is given by [12]. EPP systems present similarities with Petri nets [14], but also with communicating X-machine systems (see [1], [16]). The main difference between EPP systems and the other models consists in a structured framework which is a cell like structure, where at every step, each string object in each region of the system may be transformed by applying some evolution rules. The style of triggering transitions in parallel recalls replicated rewriting derivation mode studied for some classes of P systems [19].

In this paper it is investigated the power of both systems in connection with three parameters: number of membranes, states and set of distributed rules. On the other hand it is shown that EPP systems represent a promising framework for solving NP-complete problems; in particular linear time solutions are provided for SAT problem. The last result relies heavily on similarities between EPP systems and P systems with replicated rewriting [19], showing that more connections with these types of P systems might be further investigated.

2 Definitions

Definition 1. A stream Eilenberg machine is a tuple

$$X = (\Sigma, \Gamma, Q, M, \Phi, F, I, T, m_0),$$

where:

- Σ and Γ are finite sets called the input and the output alphabets, respectively;
- Q is the finite set of states;
- M is a (possibly infinite) set of memory symbols;
- Φ is a set of basic partial relations on $\Sigma \times M \times M \times \Gamma^*$;
- F is the next state function $F : Q \times \Phi \rightarrow 2^Q$;
- I and T are the sets of initial and final states;
- m_0 is the initial memory value.

Definition 2. An EP system is a construct $E\Pi = (\mu, X)$, where μ is a membrane structure consisting of m membranes, with the membranes and the regions labelled in a one to one manner with the elements $1, \dots, m$ and an Eilenberg machine whose memory is defined by the regions $1, \dots, m$ of μ . The Eilenberg machine is a system

$$X = (V, \Gamma, Q, M_1, \dots, M_m, \Phi, F, I),$$

having the following properties

- V is the alphabet of the system;
- Γ, Q, F are as in Definition 1; $\Gamma \subseteq V$, is called now terminal alphabet;
- M_1, \dots, M_m are finite languages over V and represent the initial values occurring in the regions $1, \dots, m$ of the system;
- $\Phi = \{\Phi_1, \dots, \Phi_p\}$, $\Phi_i = (R_{i,1}, \dots, R_{i,m})$, $1 \leq i \leq p$ and $R_{i,j}$ is a set of evolution rules (possibly empty) associated with region j , of the form $X \rightarrow (u, tar)$, with $X \in V$, $u \in V^*$, $tar \in \{\text{here, out, in}\}$; the indication here will be omitted and the rule will be written $X \rightarrow u$;
- $I = \{q_0\}$, $q_0 \in Q$ is the initial state; all the states are final states (equivalent to $Q = T$).

It may be observed that the set Σ and m_0 from Definition 1 are no longer used in the context of EP systems. In fact, these concepts have been replaced by V and M_1, \dots, M_m , respectively.

A P system has m sets of evolution rules, each one associated with a region. An EP system has the evolution rules distributed among p components Φ_i , $1 \leq i \leq p$, each one containing m sets of evolution rules.

A computation in $E\Pi$ is defined as follows: it starts from the initial state q_0 and an initial configuration of the memory defined by M_1, \dots, M_m and proceeds iteratively by applying in parallel rules in all regions, processing in each one all strings that can be rewritten; in a given state q , each string is processed by a single rule following the target indication of that rule (for instance, when rewriting xXv by a rule $X \rightarrow (u, tar)$, the string xuv obtained will be sent to the region indicated by tar , with the usual meaning in P systems (see [2], [22], [6])); if several rules may be applied to a string, then one rule and one symbol to which it is applied are randomly chosen; the rules are from a component Φ_i which is associated with one of the transitions emerging from the current state q and

the resulting strings constitute the new configuration of the membrane structure with the associated regions; the next state, belonging to $F(q, \Phi_i)$, will be the target state of the selected transition. The result (a set of strings containing only symbols from Γ) is collected outside of the system at the end of a halting computation.

EPP systems have the same underlying construct (μ, X) , with the only difference that instead of one single membrane structure, it deals with a set of instances having the same organization (μ) , but being distributed across the system. More precisely, these instances are associated with states called *active states*; these instances can divide up giving birth to more instances or collide into single elements depending on the current configuration of the active states and the general topology of the underlying machine. Initially only q_0 is an active state and the membrane configuration associated with q_0 is M_1, \dots, M_m . All active states are processed in parallel in one step: all emerging transitions from these states are processed in parallel (and every single transition processes in parallel each string object in each region, if evolution rules match them).

Cell division: if q_j is one of the active states, $M_{j,1}, \dots, M_{j,m}$ is its associated membrane configuration instance, and $\Phi_{j,1}, \dots, \Phi_{j,t}$ are Φ 's components associated with the emerging transitions from q_j , then the rules occurring in $\Phi_{j,i}$, $1 \leq i \leq t$, are applied to the string objects from $M_{j,1}, \dots, M_{j,m}$, the control passes onto $q_{j,1}, \dots, q_{j,t}$, which are the target states of the transitions earlier nominated, with $M_{j,1,1}, \dots, M_{j,m,1}, \dots, M_{j,1,t}, \dots, M_{j,m,t}$, their associated membrane configuration instances, obtained from $M_{j,1}, \dots, M_{j,m}$, by applying rules of $\Phi_{j,1}, \dots, \Phi_{j,t}$; the target states become active states, q is deactivated and $M_{j,1}, \dots, M_{j,m}$ vanish. Only $\Phi_{j,i}$ components that have rules matching the string objects of $M_{j,1}, \dots, M_{j,m}$, are triggered and consequently only their target states become active and associated with memory instances $M_{j,1,i}, \dots, M_{j,m,i}$. If none of $\Phi_{j,i}$ is triggered then in the next step q is deactivated and $M_{j,1}, \dots, M_{j,m}$ vanish too. If some of $\Phi_{j,i}$ are indicating the same component of Φ then the corresponding memory configurations $M_{j,1,i}, \dots, M_{j,m,i}$ are the same as well; this means that always identical transitions emerging from a state yield the same result.

Cell collision: if Φ_1, \dots, Φ_t enter the same state r and some or all of them emerge from active states, then the result associated with r is the union of membrane instances produced by those Φ_i 's emerging from active states and matching string objects from their membrane instances.

A computation of an EP (EPP) system halts when none of the rules associated with the transitions emerging from the current states (active states) may be applied.

The language computed by a system EII is denoted by $L(EII)$ and consists of all strings over Γ that are sent out of the system during a halting computation.

The family of languages generated by EP (EPP) systems with at most m membranes, at most s states and using at most p sets of rules is denoted by $EP_{m,s,p}$ ($EPP_{m,s,p}$). If one of these parameters is not bounded, then the corresponding subscript is replaced by $*$. An EP (EPP) system is called pure when

$V = \Gamma$ and the family of languages generated by such systems with the above considered parameters is denoted by $PEP_{m,s,p}$ ($PEPP_{m,s,p}$).

A matrix grammar with appearance checking in the *binary normal form* (for more details see [3]) is a construct $G = (N, T, S, M, F)$ where $N = N_1 \cup N_2 \cup \{S, \#\}$, with these three sets mutually disjoint, and the matrices in M with one of the following forms:

- 1. $(S \rightarrow ZB)$, with $Z \in N_1, B \in N_2$,
- 2. $(X \rightarrow Y, A \rightarrow x)$, with $X, Y \in N_1, A \in N_2, x \in (N_2 \cup T)^*$,
- 3. $(X \rightarrow Y, A \rightarrow \#)$, with $X, Y \in N_1, A \in N_2$,
- 4. $(X \rightarrow \lambda, A \rightarrow x)$, with $X \in N_1, A \in N_2, x \in T^*$.

The set F consists only of rules $A \rightarrow \#$ appearing in matrices of type 3. The family of languages produced by these devices is denoted by MAT_{ac} . When rules of type 3 are not used, then the corresponding family of languages is denoted by MAT . With RE denoting the set of recursively enumerable languages, the following relations hold $MAT \subset MAT_{ac} = RE$, where the inclusion is proper. It is also known that two nonterminals used in rules $A \rightarrow \#$ suffice to generate all RE languages [7].

3 Computational Power of EP and EPP Systems

It has been noted that rewriting alone even in a highly structured and parallel environment of a membrane, does not suffice for characterizing the set of recursively enumerable languages [6]. Thus various priority relationships have been considered. In this paper, instead of a partial order relationship on the set of rewriting rules, the rules are first distributed among sets Φ_i and then controlled by the next state function which selects the set of rules to be applied in the current state. Let us first consider an example to illustrate how an EP system works. Let $E\Pi = ([1]_1, X)$, where X contains the following elements:

- $V = \{A, A', B, B', \#, a, b, c\}$;
- $\Gamma = \{a, b, c\}$;
- $\Phi = \{\Phi_1, \Phi_2, \Phi_3, \Phi_4, \Phi_5\}$, where
 - $\Phi_1 = (\{A \rightarrow aAb, A' \rightarrow B', B' \rightarrow \#\})$,
 - $\Phi_2 = (\{B \rightarrow Bc, B' \rightarrow A', A' \rightarrow \#\})$,
 - $\Phi_3 = (\{A \rightarrow ab, A' \rightarrow B', B' \rightarrow \#\})$,
 - $\Phi_4 = (\{B \rightarrow (c, out), B' \rightarrow \lambda, A' \rightarrow \#\})$,
 - $\Phi_5 = \{\# \rightarrow \#\}$;
- $Q = \{1\}, I = \{1\}$;
- $F(\Phi_i, 1) = \{1\}, 1 \leq i \leq 5$;
- $M_1 = \{aAbBc, A'\}$.

The region 1 always has one of the following forms:

1. $\{a^n Ab^n Bc^{n+k}, A'\}, n \geq 1, k \geq 0$,
2. $\{a^{n+1} Ab^{n+1} Bc^{n+k}, B'\}, n \geq 1, k \geq 0$,

3. $\{a^{n+1}b^{n+1}Bc^{n+k}, B'\}, n \geq 1, k \geq 0,$
4. $\{a^{n+1}b^{n+1}Bc^{n+1+k}, A'\}, n \geq 1, k \geq 0,$
5. $\{y, \#\}, y \in V^*,$
6. $\{\lambda\}.$

This may be proved by induction on the number of the computation steps. Initially it has the form 1, with $n = 1, k = 0$. Next, it may be seen that:

- from 1 it results 5 (applying Φ_2 or Φ_4), or 2 (applying Φ_1), or 3 (applying Φ_3);
- from 2 it results 5 (applying Φ_1 or Φ_3), or 1 (applying Φ_2), or 6 (applying Φ_4);
- from 3 it results 5 (applying Φ_1 or Φ_3), or 4 (applying Φ_2), or 6 (applying Φ_4);
- from 4 it results 5 (applying Φ_2 or Φ_4), or 3 (applying Φ_3);
- from 5 it results 5 (applying any set that matches y ; Φ_5 may be always applied);
- in the form 6 the computation halts.

The only way to obtain terminal strings outside the system is to apply Φ_4 on the form 3 and to send out a string of the form $a^{n+1}b^{n+1}c^{n+1+k}, n \geq 1, k \geq 0$. The language of all strings of this form is not context-free. Some lessons may be learned from this simple example: the computation process of a set of words cannot be split down into independent computations of the individual elements (this property will be used further on in some more general proofs); using only one membrane and one state (i.e., no control mechanism imposed by the next state function F), non-context-free languages may be generated.

If we consider the above specification as an EPP system denoted by $EIII$ then $L(EIII) = \emptyset$, because all first four components are triggered, $\#$ is introduced, and the computation never halts. The following $EIII'$ using the above defined V and Γ , but redefining Φ, Q, F and M_1

- $\Phi = \{\Phi_1, \Phi_2, \Phi_3, \Phi_4, \Phi_5\}$, where
 - $\Phi_1 = (\{A \rightarrow aAb, A \rightarrow aA'b\})$,
 - $\Phi_2 = (\{B \rightarrow Bc, B \rightarrow B'c\})$,
 - $\Phi_3 = (\{A' \rightarrow ab\})$,
 - $\Phi_4 = (\{B' \rightarrow B'c, B' \rightarrow Bc\})$,
 - $\Phi_5 = \{B \rightarrow (c, out)\}$;
- $Q = \{1, 2, 3, 4\}, I = \{1\}$;
- $F(\Phi_1, 1) = \{2\},$
 $F(\Phi_2, 2) = \{1\},$
 $F(\Phi_3, 1) = \{3\},$
 $F(\Phi_4, 3) = \{3\},$
 $F(\Phi_5, 3) = \{4\}$;
- $M_1 = \{AB\}$;

leads to $L(EIII') = L(EII)$. The underlying system computes the same language when it is to be considered an EP system as well.

Some preliminary results follow directly from definitions ([P] means that the involved relationships hold for both P being present in both its members or none of them).

- Lemma 1.** (i) $PEP_{m,s,p} \subseteq EP_{m,s,p}$, $m, s, p \geq 1$;
(ii) $[P]EP_{m,s,p} \subseteq [P]EP_{m+k_1, s+k_2, p+k_3}$, $k_1, k_2, k_3 \geq 0$.

The next result is similar to Lemma 2 in [6], giving also details about the number of states and rules used by the proof.

- Lemma 2.** $EP_{m,s,p} \subseteq PEP_{m+1, s+1, p+2}$, $m, s, p \geq 1$.

Proof. Let us consider an EP system $E\Pi = (\mu, X)$, with μ a membrane structure with m regions and X a machine given by

$$X = (V, \Gamma, Q, M_1, \dots, M_m, \Phi, F, I),$$

according to Definition 2. We construct now the pure EP system $E\Pi' = (\mu', X')$, where $\mu' = [0\mu]_0$, and

$$X' = (V', V', Q', M'_0, M'_1, \dots, M'_m, \Phi', F', I),$$

with

- $V' = V \cup \{f\}$, $f \notin V$;
- $Q' = Q \cup \{q_0\}$, $q_0 \notin Q$;
- $M'_i = \{fw \mid w \in M_i\}$, $1 \leq i \leq m$; $M'_0 = \emptyset$;
- $\Phi' = \{\Phi'_i \mid \Phi_i \in \Phi, 1 \leq i \leq p\} \cup \{\Phi'_{p+1}, \Phi'_{p+2}\}$, where
 $\Phi'_i = (\emptyset, R_{i,1}, \dots, R_{i,m})$, for $\Phi_i = (R_{i,1}, \dots, R_{i,m})$,
 $\Phi'_{p+1} = (\{a \rightarrow (a, in) \mid a \in V \setminus \Gamma\}, \emptyset, \dots, \emptyset)$ and
 $\Phi'_{p+2} = (\{f \rightarrow (\lambda, out)\}, \emptyset, \dots, \emptyset)$;
- $F'(q, \Phi'_i) = F(q, \Phi_i)$, $q \in Q$, $\Phi_i \in \Phi$, $1 \leq i \leq p$, and
 $F'(q, \Phi'_{p+1}) = \{q_0\}$, $F'(q_0, \Phi'_{p+2}) = \{q_0\}$.

According to the above construction any string x processed by $E\Pi$ exits the system iff fx arrives in region 0 of $E\Pi'$, in a state $q \in Q$. Indeed any application of a set of rules Φ_i in $E\Pi$ is simulated in $E\Pi'$ by Φ'_i . Any string in region 0 is checked to contain only elements from Γ , by applying Φ'_{p+1} . All strings in region 0 which contain some symbols in $V \setminus \Gamma$ fall down into region 1. Otherwise the rule $f \rightarrow (\lambda, out)$ occurring in Φ'_{p+2} pops out strings containing only symbols from Γ . Hence $L(E\Pi) = L(E\Pi')$. \square

- Theorem 1.** $MAT = EP_{4,1,1} \subseteq PEP_{5,2,3}$.

Proof. An EP system with m membranes, a single state and only one set of rules is an extended rewriting system [6], and for these systems it has been shown that 4 membranes suffice to generate MAT , and consequently the result holds. \square

It is also known that a graph controlling the derivation, = 1 style, in grammar systems, without appearance checking feature, produces exactly *MAT* (Theorem 4.7 in [3]). What about an EP system with only one membrane? The only difference between such a system and a grammar system as described above is that in an EP system all strings are kept in one membrane and are rewritten in parallel in one step. From Theorem 1 we have learned that without spreading the rules and using only one state we do not get more than *MAT* languages.

Theorem 2. $EP_{1,1,*} = PEP_{2,2,*} = RE$.

Proof. According to Turing-Church thesis and Lemma 2, it is only to prove the inclusion $RE \subseteq EP_{1,1,*}$. As usually in such cases a matrix grammar with appearance checking in the binary normal form $G = (N_1 \cup N_2 \cup \{S, \#\}, T, S, M, F)$ is considered. It is also assumed that the rules are labelled in a one to one manner with m_1, \dots, m_{k_1} (the matrices of type 2), $m_{k_1+1}, \dots, m_{k_2}$ (the matrices of type 3), and $m_{k_2+1}, \dots, m_{k_3}$ (the matrices of type 4). The following EP system is constructed $EII = (\mu, X)$, where μ is a membrane structure consisting of a single membrane, and X an Eilenberg machine

$$X = (N_1 \cup N_2 \cup \{S, \#\} \cup T, T, \{q_0\}, M_1, \Phi, F, \{q_0\}),$$

with

- M_1 containing Z and B , the symbols occurring in the right hand side of the matrix of type 1;
- $F(q_0, \Phi_i) = \{q_0\}$, for any $\Phi_i \in \Phi$;
- the set Φ containing
 - for each matrix $m_i = (X \rightarrow Y, A \rightarrow x)$, $1 \leq i \leq k_1$, of type 2, a set of rules
 $\Phi_i = (\{A \rightarrow x, X \rightarrow Y\} \cup \{U \rightarrow \# \mid U \in N_1 \cup N_2, U \neq X, U \neq A\})$;
 - for each matrix $m_i = (X \rightarrow Y, A \rightarrow \#)$, $k_1 + 1 \leq i \leq k_2$, of type 3, a set of rules
 $\Phi_i = (\{A \rightarrow \#, X \rightarrow Y\} \cup \{U \rightarrow \# \mid U \in N_1, U \neq X\})$;
 - for each matrix $m_i = (X \rightarrow \lambda, A \rightarrow x)$, $k_2 + 1 \leq i \leq k_3$, of type 4, a set of rules
 $\Phi_i = (\{A \rightarrow x, X \rightarrow (\lambda, out)\} \cup \{U \rightarrow \# \mid U \in N_1 \cup N_2, U \neq X, U \neq A\})$
 - a new rule is considered in the set $\Phi_0 = (\{\# \rightarrow \#\})$;

The computation will start with $\{Z, B\}$ in the main region defined by the skin (main) membrane. If $\{X, uAv\}$ is the current content of the region, then this corresponds to the sential form $XuAv$ associated with the grammar G . A matrix m_i , $1 \leq i \leq k_1$ is succesfully applied to $XuAv$ iff the corresponding rules of Φ_i are applied in one step in parallel to both X , and uAv . A failure in correctly applying this matrix leads to blocking the derivation process in G and accordingly the introduction of $\#$ symbol in the current membrane. Similarly, the set of rules associated with matrices of types 3 and 4, simulate the use of these matrices in G . Any blocking derivation in G is simulated in EII by introducing ' $\#$ ' symbol which leads to an endless computation (set Φ_0) or to the situation when a terminal sequence is never sent out of the system. \square

The proof of Theorem 2 says that only one membrane and one state suffice to compute all *RE* languages, but with an unbounded number of rules. On the other hand the simulation is very efficient and natural, the number of steps involved in a computation in *EII* is exactly equal to the number of matrices required by the corresponding equivalent derivation of *G*. What can be said when all three parameters are bounded? The next result shows that by increasing either the number of membranes or the number of states, EP systems with a bounded number of functions that compute all *RE* languages may be found.

Theorem 3. (i) $EP_{1,3,8} = RE$; (ii) $EP_{2,1,7} = RE$.

Proof. Again we consider a matrix grammar with appearance checking in the binary normal form $G = (N_1 \cup N_2 \cup \{S, \#\} \cup T, T, S, M, F)$. It is assumed that the rules are labelled in a one to one manner with m_1, \dots, m_{k_1} (matrices of type 2), $m_{k_1+1}, \dots, m_{k_2}$ (matrices of type 3), and $m_{k_2+1}, \dots, m_{k_3}$ (matrices of type 4). Also we assume that type 3 matrices (with appearance checking) utilize only two nonterminals A_1, A_2 [7]. We denote by Dom_i the set of nonterminal symbols occurring in the left hand side of rules of matrices of type i , $1 \leq i \leq 4$.

(i) Let $EII = (\mu, X)$ be an EP system with one membrane and three states, where μ is a membrane structure, and X an Eilenberg machine. The underlying machine is given by

$$X = (N_1 \cup N_2 \cup \{S, \#, f\} \cup \{i_j \mid 0_j \leq i_j \leq (k_3)_j, 1 \leq j \leq 4\} \cup T, \\ T, Q, M_1, \Phi, F, \{q_1\}),$$

where f and i_j are new symbols and

- $Q = \{q_1, q_2, q_3\}$;
- M_1 contains Zf and Bf , with Z and B the symbols occurring in the right hand side of the rule of type 1, and f is the new symbol introduced by X ;
- $\Phi = \{\Phi_1, \dots, \Phi_8\}$, where
 - $\Phi_1 = (\{X \rightarrow Yi_1 \mid 1_1 \leq i_1 \leq (k_1)_1, \text{ and there is } i_1 : (X \rightarrow Y, A \rightarrow x) \text{ a matrix of type 2}\} \cup \{X \rightarrow \lambda i_3 \mid 1_3 \leq i_3 \leq (k_3)_3, \text{ and there is } i_3 : (X \rightarrow \lambda, A \rightarrow x) \text{ a matrix of type 4}\} \cup \{A \rightarrow xi_2 \mid 1_2 \leq i_2 \leq (k_1)_2, \text{ and there is } i_2 : (X \rightarrow Y, A \rightarrow x) \text{ a matrix of type 2}\} \cup \{A \rightarrow xi_4 \mid 1_4 \leq i_4 \leq (k_3)_4, \text{ and there is } i_4 : (X \rightarrow \lambda, A \rightarrow x) \text{ a matrix of type 4}\} \cup \{U \rightarrow \# \mid U \in N_1 \cup N_2, U \notin Dom_2 \cup Dom_4\})$,
 - $\Phi_2 = (\{i_j \rightarrow i_j - 1 \mid 1 \leq j \leq 4, 0_1 < i_1 \leq (k_1)_1, 0_2 < i_2 \leq (k_1)_2, 0_3 < i_3 \leq (k_3)_3, 0_4 < i_4 \leq (k_3)_4\} \cup \{f \rightarrow \#\})$,
 - $\Phi_3 = (\{0_1 \rightarrow \lambda, 0_2 \rightarrow \lambda\} \cup \{i_j \rightarrow \# \mid i_j > 0, 1 \leq j \leq 4\})$,
 - $\Phi_4 = (\{f \rightarrow \lambda\})$,
 - $\Phi_5 = (\{0_3 \rightarrow \lambda, 0_4 \rightarrow (\lambda, out)\} \cup \{i_j \rightarrow \# \mid i_j > 0, 1 \leq j \leq 4\})$,
 - $\Phi_6 = (\{X \rightarrow Y, A_1 \rightarrow \# \mid (X \rightarrow Y, A_1 \rightarrow \#) \text{ matrix of type 3}\} \cup \{U \rightarrow \# \mid U \in N_1 \setminus Dom_3\})$,

- $\Phi_7 = (\{X \rightarrow Y, A_2 \rightarrow \# \mid (X \rightarrow Y, A_2 \rightarrow \#)$ matrix of type 3 $\} \cup \{U \rightarrow \# \mid U \in N_1 \setminus Dom_3\})$,
 - $\Phi_8 = (\{\# \rightarrow \#\})$;
- $F(q_1, \Phi_i) = \{q_1\}$, $i \in \{6, 7, 8\}$, $F(q_1, \Phi_1) = \{q_2\}$, $F(q_2, \Phi_2) = \{q_2\}$,
 $F(q_2, \Phi_3) = \{q_1\}$, $F(q_2, \Phi_4) = \{q_3\}$, $F(q_3, \Phi_5) = \{q_1\}$.

A computation in EII develops as follows:

- the process starts in state q_1 with initial configuration $\{Zf, Bf\}$;
- given a configuration $\{Xf, \alpha A\beta f\}$, one of Φ_1, Φ_6 or Φ_7 may be applied; if Φ_1 is selected then it meant to simulate either matrices $i : (X \rightarrow Y, A \rightarrow x)$ of type 2 or $i : (X \rightarrow \lambda, A \rightarrow x)$ of type 4, by applying $X \rightarrow Yi_1, A \rightarrow xi_2$ or $X \rightarrow \lambda i_3, A \rightarrow xi_4$ and leading to $\{Yi_1f, \alpha xi_2\beta f\}$ or $\{i_3f, \alpha xi_4\beta f\}$ in state q_2 ; in both cases rules of Φ_2 are iteratively applied to check whether the previous rules came from the same matrix (either $i_1 = i_2$ or $i_3 = i_4$); if one index, i_j reaches 0_j and the other not then a $\#$ symbol is introduced by $f \rightarrow \#$; if Φ_2 is left before reaching 0_j then either Φ_3 or Φ_5 introduces $\#$ by one of the rules $i_j \rightarrow \#$; when i_1 and i_2 or i_3 and i_4 indicate the same value, then
 - for $\{Yi_1f, \alpha xi_2\beta f\}$, the two i 's become 0_1 and 0_2 , respectively, by applying Φ_2 ; then Φ_3 makes them λ and the process successfully simulates $i : (X \rightarrow Y, A \rightarrow x)$;
 - similarly for $\{i_3f, \alpha xi_4\beta f\}$, the two i 's become 0_3 and 0_4 , respectively, by applying Φ_2 ; then f is deleted from both strings (using Φ_4 and going to state q_3); Φ_5 removes both 0_3 and 0_4 and $0_4 \rightarrow (\lambda, out)$ sends out $\alpha\lambda\beta$;
- when either Φ_6 or Φ_7 is applied to $\{Xf, \alpha A\beta f\}$ this means that the use of either $(X \rightarrow Y, A_1 \rightarrow \#)$ or $(X \rightarrow Y, A_2 \rightarrow x)$ is simulated (using Φ_6 for the former or Φ_7 for the latter);
- once a symbol $\#$ is introduced the process never ends as Φ_8 may be always triggered.

The EP system EII computes exactly what G generates.

(ii) The EP systems $EII = (\mu, X)$ with one state and two membranes is built as follows: μ is a membrane structure $[_1[_2]_2]_1$ and X the Eilenberg machine:

$$X = (N_1 \cup N_2 \cup \{S, \#, f_1, f_2\} \cup \{i_j \mid 0_j \leq i_j \leq (k_3)_j, 1 \leq j \leq 4\} \\ \cup T, T, \{q_1\}, M_1, M_2, \Phi, F, \{q_1\}),$$

where f_1, f_2 and i_j are new symbols and:

- M_1 contains Zf_1 and Bf_2 , with Z and B the symbols occurring in the right hand side of the rule of type 1, and M_2 is empty;
- $\Phi = \{\Phi_1, \dots, \Phi_7\}$, is very similar with Φ built for the above proof (i), but makes use of the two membranes; Φ contains:
 - $\Phi_1 = (\{X \rightarrow (Yi_1, in) \mid 1_1 \leq i_1 \leq (k_1)_1$, and there is $i_1 : (X \rightarrow Y, A \rightarrow x)$ a matrix of type 2 $\} \cup$

- $\{X \rightarrow (\lambda i_3, in) \mid 1_3 \leq i_3 \leq (k_3)_3, \text{ and there is } i_3 : (X \rightarrow \lambda, A \rightarrow x)$
a matrix of type 4 $\} \cup$
- $\{A \rightarrow (xi_2, in) \mid 1_2 \leq i_2 \leq (k_1)_2, \text{ and there is } i_2 : (X \rightarrow Y, A \rightarrow x)$
a matrix of type 2 $\} \cup$
- $\{A \rightarrow (xi_4, in) \mid 1_4 \leq i_4 \leq (k_3)_4, \text{ and there is } i_4 : (X \rightarrow \lambda, A \rightarrow x)$
a matrix of type 4 $\} \cup$
- $\{U \rightarrow \# \mid U \in N_1 \cup N_2, U \notin Dom_2 \cup Dom_4\}, \emptyset),$
- $\Phi_2 = (\emptyset, \{i_j \rightarrow i_j - 1 \mid 1 \leq j \leq 4, 0_1 < i_1 \leq (k_1)_1, 0_2 < i_2 \leq (k_1)_2,$
 $0_3 < i_3 \leq (k_3)_3, 0_4 < i_4 < (k_3)_4\} \cup \{f_1 \rightarrow \#, f_2 \rightarrow \#\}),$
- $\Phi_3 = (\emptyset, \{0_j \rightarrow (\lambda, out)\} \cup \{i_j \rightarrow \# \mid i_j > 0, 1 \leq j \leq 4\}),$
- $\Phi_4 = (\{f_1 \rightarrow \lambda, f_2 \rightarrow (\lambda, out)\}, \emptyset),$
- $\Phi_5 = (\{X \rightarrow Y, A_1 \rightarrow \# \mid (X \rightarrow Y, A_1 \rightarrow \#)$ matrix of type 3 $\} \cup$
 $\{U \rightarrow \# \mid U \in N_1 \setminus Dom_3\}, \emptyset),$
- $\Phi_6 = (\{X \rightarrow Y, A_2 \rightarrow \# \mid (X \rightarrow Y, A_2 \rightarrow \#)$ matrix of type 3 $\} \cup$
 $\{U \rightarrow \# \mid U \in N_1 \setminus Dom_3\}, \emptyset),$
- $\Phi_7 = (\{\# \rightarrow \#\}, \{\# \rightarrow \#\}).$

In a manner very similar to the proof of (i) it may be shown that $E\Pi$ computes exactly what G generates. \square

EPP systems exhibit a parallel behaviour not only inside of the membrane structure but also at the underlying machine level. Potentially, all transitions emerging from active states may be triggered in one step giving birth to new cells or colliding others. One problem addressed in this case is also related to the power of these mechanisms. One may adopt the previous strategy by considering a RE language generated by a matrix grammar in binary normal form to find the EPP systems computing that language. Given that EP systems have been studied in this respect, another solution would be to compare them with EPP systems. More precisely, given an EP systems with m membranes, s states and p production rules associated via Φ components, is it possible to simulate it by an EPP system? If yes, what are the values for the number of membranes, states and rules? The next theorem gives an answer to this problem.

Lemma 3. *If $E\Pi$ is an EP system with m membranes, s states and p sets of rules then there exists $E\Pi\Pi$ an EPP systems with $m' \geq m$ membranes, $s' \geq s$ states and $p' \geq p$ rule transitions such that $L(E\Pi) = L(E\Pi\Pi)$.*

Proof. Let $E\Pi = (\mu, X)$, be an EP systems where μ is a membrane structure consisting of m membranes, and X an Eilenberg machine

$$X = (V, \Gamma, Q, M_1, \dots, M_m, \Phi, F, I),$$

where Q has s states and Φ contains p components. The following EPP system is built $E\Pi\Pi = (\mu', X')$, where $\mu' = [{}_0\mu]_0$ and

$$X' = (V', \Gamma, Q', M_0, M_1, \dots, M_m, \Phi', F', I),$$

with

- $V' = V \cup \{x\} \cup \{k \mid 1 \leq k \leq t\}$, where t is the maximum number of transitions going out from every state of X ;
- $Q' = Q \cup \{q_{j,0} \mid q_j \in Q\} \cup \{q_{j,k,h} \mid q_j \in Q, 1 \leq k \leq t, 1 \leq h \leq 4\}$;
- $M_0 = \{x\}$;
- $\Phi' = \Phi \cup \{\Phi_x, \Phi_{1x}, \dots, \Phi_{tx}, \Phi_\Gamma, \Phi_{V'}\}$, where
 - $\Phi_x = (\{x \rightarrow k \mid 1 \leq k \leq t\}, \emptyset, \dots, \emptyset)$,
 - $\Phi_{kx} = (\{k \rightarrow x\}, \emptyset, \dots, \emptyset, 1 \leq k \leq t)$,
 - $\Phi_\Gamma = (\{a \rightarrow (a, out) \mid a \in \Gamma\}, \emptyset, \dots, \emptyset)$,
 - $\Phi_{V'} = (\{X \rightarrow (X, in) \mid X \in V' \setminus \Gamma\}, \emptyset, \dots, \emptyset)$;
- for any $q_j \in Q$ if there are $1 \leq u \leq t$, transitions emerging from q_j and $F(q_j, \Phi_{j,k}) = \{q_{j,k}\}$, $1 \leq k \leq u$ (not all $\Phi_{j,k}$ are supposed to be distinct) then the following transitions are built in $EIII$:
 - $F'(q_j, \Phi_x) = \{q_{j,0}\}$, $F'(q_{j,0}, \Phi_{kx}) = \{q_{j,k,1}\}$, $1 \leq k \leq u$,
 - $F'(q_{j,k,1}, \Phi_{j,k}) = \{q_{j,k,2}, q_{j,k}\}$,
 - $F'(q_{j,k,2}, \Phi_{V'}) = \{q_{j,k,3}\}$, $F'(q_{j,k,3}, \Phi_\Gamma) = \{q_{j,k,4}\}$, $1 \leq k \leq u$.

A computation in system $EIII$ proceeds in the following way: at the beginning, only the initial state is active and the memory configuration in this state is M_0, M_1, \dots, M_m . If the EPP system EII is in a state q_j and the memory configuration is $M_{j,0}, M_{j,1}, \dots, M_{j,m}$, then $EIII$ must be in q_j as well. We will show that always $EIII$ has either an active state or at most two active states but in this case one of them is $q_{j,k,h}$, $2 \leq h \leq 4$, and from the last one ($q_{j,k,4}$) the membrane configuration will vanish and possibly the EPP system sends out a string. Indeed, if q_j is an active state in $EIII$ and $M_{j,0}, M_{j,1}, \dots, M_{j,m}$ are its associated membrane configuration, then in one step x from $M_{j,0}$ is changed by Φ_x into k , a value between 1 and t ; if u is the number of emerging transitions from q_j in EII , then $k > u$ implies that in the next step the current membrane configuration will vanish as no more continuation is then allowed from $q_{j,0}$; otherwise, when $1 \leq k \leq u$, only one transition may be triggered from $q_{j,0}$ and this is associated with Φ_{kx} which restores x back into $M_{j,0}$ (the other transitions emerging from $q_{j,0}$ cannot be triggered). Φ_{kx} leads the EPP system into $q_{j,k,1}$. From this state there are two transitions both associated with $\Phi_{j,k}$ that are triggered in the same time and consequently $M_{j,0}, M_{j,1}, \dots, M_{j,m}$ are processed by both $\Phi_{j,k}$'s and yield the same memory configuration $M'_{j,0}, M'_{j,1}, \dots, M'_{j,m}$ in both $q_{j,k,2}$ and $q_{j,k}$. These are then processed in parallel, being both active states. From the former the current configuration is checked for nonterminal symbols, $\Phi_{V'}$, and then in the next step only terminal strings are sent out by using Φ_Γ ; then this memory configuration vanishes as no further transition emerges from $q_{j,k,4}$. In fact this path introduced from every $q_{j,k,1}$ has the role of collecting terminal strings outside of the system. From $q_{j,k}$ the process resumes as from q_j , and in two steps at most one state will be active on the path from q_j . In this way EII and $EIII$ compute the same language, thus $L(EII) = L(EIII)$. \square

Note 1. From Lemma 3 it follows that if the EP system EII has m membranes, s states, p components of Φ , and the maximum number of transitions emerging from every state is t then the equivalent EPP system has $m' = m + 1$ membranes, at most $s' = (2 + 4t)s$ states, and at most $p' = p + t + 3$ sets of rules.

Theorem 4. $EPP_{2,54,15} = EPP_{3,30,17} = RE$.

Proof. By using Note 1 and the constructions from the proof of Theorem 3 the result follows. \square

Obviously lower bounds may be obtained for the above discussed parameters when the two constructions from the proof of Theorem 3 are used in order to get an EPP equivalent system. This is achieved by applying the procedure provided by the proof of Lemma 3, but it is left as an exercise for the reader.

4 Linear Solution to SAT Problem

SAT (satisfiability of propositional formulae in conjunctive normal form) is a well known NP-complete problem. This problem asks whether or not, for a given formula in the conjunctive normal form, there is a truth-assignment of the variables such that it becomes true. So far some methods to solve in polynomial or just linear time this problem have been indicated, but at the expense of using an exponential space of values.

Theorem 5. *The SAT problem can be solved in a time linear in the number of variables and the number of clauses by using an EPP system.*

Proof. Let γ be a formula in conjunctive normal form consisting of m clauses, C_1, \dots, C_m , each one being a disjunction, and the variables used are x_1, \dots, x_n . The following EPP system, $EIII = (\mu, X)$, may be then constructed:

$$\mu = [1[2 \dots [m+1]_{m+1} \dots]2]_1,$$

$$X = (V, \Gamma, Q, M_1, \dots, M_{m+1}, \Phi, F, I),$$

where:

- $V = \{a_k, t_k, f_k \mid 1 \leq k \leq n\}$;
- $\Gamma = \{t_k, f_k \mid 1 \leq k \leq n\}$;
- $Q = \{q_1, q_2\}$;
- $M_1 = \dots = M_m = \emptyset, M_{m+1} = \{a_1\}$;
- $\Phi = \{\Phi_1, \dots, \Phi_5\}$;
 - $\Phi_1 = (\emptyset, \dots, \emptyset, \{a_k \rightarrow f_k a_{k+1} \mid 1 \leq k \leq n-1\})$,
 - $\Phi_2 = (\emptyset, \dots, \emptyset, \{a_k \rightarrow t_k a_{k+1} \mid 1 \leq k \leq n-1\})$,
 - $\Phi_3 = (\emptyset, \dots, \emptyset, \{a_n \rightarrow (f_n, out)\})$,
 - $\Phi_4 = (\emptyset, \dots, \emptyset, \{a_n \rightarrow (t_n, out)\})$,
 - $\Phi_5 = (\{t_k \rightarrow (t_k, out) \mid x_k \text{ is present in } C_1, 1 \leq k \leq n\} \cup \{f_k \rightarrow (f_k, out) \mid \neg x_k \text{ is present in } C_1, 1 \leq k \leq n\},$
 $\dots,$
 $\{t_k \rightarrow (t_k, out) \mid x_k \text{ is present in } C_m, 1 \leq k \leq n\} \cup \{f_k \rightarrow (f_k, out) \mid \neg x_k \text{ is present in } C_m, 1 \leq k \leq n\}, \emptyset)$;
- $F(q_1, \Phi_k) = \{q_1\}, 1 \leq k \leq 2, F(q_1, \Phi_k) = \{q_2\}, 3 \leq k \leq 4, F(q_2, \Phi_5) = \{q_2\}$;
- $I = \{q_1\}$.

EP_{III} starts from state q_1 with $\emptyset, \dots, \emptyset, \{a_1\}$. By applying $n - 1$ times Φ_1 and Φ_2 in parallel and then Φ_3 and Φ_4 one generates all truth values for the n variables in the form of 2^n strings with t_k or f_k indicating that variable x_k is either *true* or *false*. All these combinations are obtained in n steps in state q_2 . In the next m steps Φ_5 checks whether or not at least one truth-assignment satisfies all clause; this, if exists, will exit the system. The SAT problem is solved in this way in $n + m$ steps. \square

5 Conclusions

In this paper two types of Eilenberg P systems, namely EP systems and EPP systems, have been introduced. They combine the control structure of an Eilenberg machine as a driven mechanism of the computation with a cell-like structure having a hierarchical organisation of the objects involved in the computational process. The computational power of EP systems is investigated in respect of three parameters: number of membranes, number of states, and number of sets of rules. It is proved that when only one state and one set of rules are used, four membranes suffice to compute *MAT* languages. It may be easily observed that in this case the number of states is irrelevant as with only one single set of rules, even distributed across many states, one cannot compute more than with a single state and the same set of rules, i.e., $EP_{m,s,1} = MAT$, $m \geq 4, s \geq 1$. When at least three states and eight sets of rules or two membranes and seven sets of rules are used, then the whole set of *RE* languages may be computed, i.e., $EP_{1,3,8} = RE$ or $EP_{2,1,7} = RE$. A number of questions regarding lower limits for the above parameters remain to be further addressed. It is possible to compute *RE* by using EP systems with less than three states and/or eight sets of rules, i.e., $EP_{1,s,p}$, where $s < 3$ and/or $p < 8$? Is it true that $EP_{1,1,7} = RE$, $p < 7$?

EPP systems represent the parallel counter-part of EP systems, allowing not only the rules inside of the cell-like structure to develop in parallel, but also the transitions emerging from the same state. More than this, all states that are reached during the computation process as target states, may trigger in the next step all transitions emerging from them. It is shown that a general method to simulate an EP system as an EPP system computing the same language may be stated. This result allows us to map all properties concerning computationally completeness properties of EP systems onto EPP systems. Apart from the fact that EPP systems might describe interesting biological phenomena like cell division and collision, it is also a computationally complete device and an effective mechanism for solving NP-complete problems, like SAT, in linear time.

References

1. T. Bălănescu, T. Cowling, H. Georgescu, M. Gheorghe, M. Holcombe, C. Ver-tan, Communicating stream X-machines systems are no more than X-machines, *J. Universal Comp. Sci.*, 5, 9 (1999), 494–507.

2. C. Calude, Gh. Păun, *Computing with Cells and Atoms*, Taylor and Francis, London, 2000.
3. E. Csuhaj-Varju, J. Dassow, J. Kelemen, Gh. Păun, *Grammar Systems. A Grammatical Approach to Distribution and Cooperation*, Gordon & Breach, London, 1994.
4. J. Dassow, Gh. Păun, *Regulated Rewriting in Formal Language Theory*, Springer Verlag, Berlin, 1989.
5. S. Eilenberg, *Automata, Languages and Machines*, Academic Press, 1974.
6. C. Ferretti, G. Mauri, Gh. Păun, C. Zandron, On Three Variants of Rewriting P Systems, *Pre-proceedings of Workshop on Membrane Computing (WMC-CdeA2001)*, (C. Martin-Vide, Gh. Păun, eds), Curtea de Argeş, Romania, August 2001, 63–76, and *Theor. Comp. Sci.*, to appear.
7. R. Freund, Gh. Păun, On the number of non-terminals in graph-controlled, programmed, and matrix grammars, *Proc. Conf. Universal Machines and Computations*, Chisinau, 2001 (M. Margenstern and Y. Rogozhin, eds.), Springer-Verlag, Berlin, 2001.
8. M. Gheorghe, Generalised stream X-machines and cooperating grammar systems, *Formal Aspects of Computing*, 12 (2000), 459–472.
9. M. Gheorghe, M. Holcombe, P. Kefalas, Computational models of collective foraging, *BioSystems*, 61 (2001), 133–141.
10. M. Holcombe, X-machines as a basis for dynamic system specification, *Software Engineering Journal*, 3, 2 (1988), 69–76.
11. M. Holcombe, Computational models of cells and tissues: Machines, agents and fungal infection, *Briefings in Bioinformatics*, 2 (2001), 271–278.
12. M. Holcombe, What are X-machines, *Formal Aspects of Computing*, 12 (2000), 418–422.
13. M. Holcombe, F. Ipate, *Correct Systems Building a Business Process Solution*, Springer, Applied Computing Series, 1998.
14. K. Jensen, *Coloured Petri Nets – Basic Concepts, Analysis Methods and Practical Use*, vol 1-3, Springer, Berlin, 1992, 1994, 1997.
15. S. Ji, The Bhopalator, An information/energy dual model of the living cell, *Pre-proceedings of Workshop on Membrane Computing (WMC-CdeA2001)* (C. Martin-Vide, Gh. Păun, eds), Curtea de Argeş, Romania, August 2001, 123–141, and *Fundamenta Informaticae*, 49 (2002), 147–165.
16. P. Kefalas, Formal modelling of reactive agents as an aggregation of simple behaviours, LNAI 2308 (I.P. Vlahavas, C.D. Syropoulos, eds.), Springer, 461–472, 2002.
17. Z. Kohavi, *Switching and Finite Automata Theory*, McGraw-Hill, 1978.
18. S.N. Krishna, R. Rama, On the power of P systems with sequential and parallel rewriting, *Intern. J. Computer Math.*, 77, 1-2 (2000), 1–14.
19. S.N. Krishna, R. Rama, P systems with replicated rewriting, *Journal of Automata, Languages and Combinatorics*, 6 (2001), 345–350.
20. A. Păun, P systems with string objects: Universality results, *Pre-proceedings of Workshop on Membrane Computing (WMC-CdeA2001)* (C. Martin-Vide, Gh. Păun, eds), Curtea de Argeş, Romania, August 2001, 229–241.
21. Gh. Păun, Computing with membranes, *Journal of Computer System Sciences*, 61, 1 (2000), 108–143 (see also *Turku Center for Computer Science, TUCS Report No 208*, 1998, <http://www.tucs.fi>).
22. Gh. Păun, *Membrane Computing. An Introduction*, Springer, Berlin, 2002.