A Matrix Representation for Computations on EC P Systems with Energy

Richelle Ann B. Juayong  
Algorithms and Complexity Laboratory  
University of the Philippines Diliman  
Quezon City, Philippines  
rbjuayong@up.edu.ph

Henry N. Adorna  
Algorithms and Complexity Laboratory  
University of the Philippines Diliman  
Quezon City, Philippines  
ha@dcs.upd.edu.ph

ABSTRACT
In this paper, we extend the works in previous literatures to obtain a matrix representation for a P System variant called Evolution-Communication P Systems with Energy (ECP-E). Through this representation, computations in ECP-E can be done through matrix operations. The problem of determining the combination of each rule’s number of applications which satisfies maximal parallelism is also shown to be answerable by satisfying conditions involving a system of linear inequalities.

1. INTRODUCTION
Membrane Computing is a nonconventional computing paradigm that provides a framework for compartmentalized computation [4]. This field is inspired by how living cells compute; protected reactors are delimited by compartments called membranes and these reactors co-operate and communicate with each other for a cell to accomplish its work correctly. The computing model used for membrane computing is called a P System. Its main ingredient is a collection of membranes that function as delimiters and passageways of objects. Though the field is relatively young compared to other nonconventional computing models, it has captured the interest of computer scientists because of its contribution to natural computing with some variants being able to solve computationally hard problems in polynomial time. Numerous papers on membrane computing can be found in their site [6].

It has been proven that obtaining algebraic representations are helpful in simulating P Systems. In 2009, a matrix representation for cell-like P Systems without dissolution and without cooperation is introduced in order to derive a general method for finding all configurations that produce a given configuration [3]. Last year, matrix representations are also employed to simulate computations for spiking neural P Systems [2]. In this paper, we extend the work in both papers to come up with a matrix representation for Evolution-Communication P Systems with Energy (ECP-E) [1]. ECP-E is a cell-like P System variant which uses rewriting rules to evolve objects and symport and antiport rules to communicate objects. It was presented to provide a measure for communication over Evolution-Communication P Systems. To do this, a special object called energy is introduced. The object may be produced through evolution rules and consumed through transportation rules.

Our work is motivated by questions of how ECP-E can be implemented. Note that there is not much work on this variant since it was only introduced last year. Through implementations, analysis on ECP-E can be eased and speeded up. Although developing a dedicated hardware for membrane systems is a longstanding problem in this area, recently, efforts have been made to implement these models in Graphics Processing Units (GPUs). A matrix representation provides a means for efficient simulation since there already exists a number of literatures (like [7]) on how to efficiently simulate matrix operations on parallel devices.

The outline of this paper proceeds as follows: We define Evolution-Communication P Systems with Energy in Section 2. In Section 3, we give the necessary definitions needed for the representation, and in Section 4, we show how we can use matrix operations to simulate computations in ECP-E. Determining the correct number of application of each rule which satisfies maximal parallelism is discussed in Section 5. Finally, our conclusions are stated in Section 6.

2. EC P SYSTEMS WITH ENERGY (ECP-E)
Before we introduce ECP-E as a variant of P System, we shall describe first how a typical P System works.
An introduction detailing how P Systems compute can be found in [4].

As mentioned in Section 1, a P System has a set of membranes. The regions delimited by these membranes serve as placeholders of a multiset of objects. Aside from being transported (through transport rules), these objects may evolve through multiset-rewriting rules called evolution rules. When an object is involved in an evolution rule, we say that this object is ‘consumed’. Rules in P Systems are applied in a nondeterministic and maximally parallel manner. The nondeterminism is manifested when several rules can be applied to a single object. Since there can only be one rule consuming this object, the system picks out one among all applicable rules and evolves the object through this chosen one. Maximal parallelism, on the other hand, requires that at a single unit of time, everything that can evolve should evolve.

A **configuration** of a P System at any time $i$, denoted by $C_i$, corresponds to the multiset of objects present in each membrane at a certain time step. A transition from a configuration $C_i$ to configuration $C_{i+1}$, denoted by $C_i \Rightarrow C_{i+1}$, is an evolution from $C_i$ to $C_{i+1}$ through application of a set of rules. A computation is a set of valid transitions wherein rules applied satisfy the condition of nondeterminism and maximal parallelism.

EC P Systems, as introduced in [5], are combinations of two variants of P Systems called Transition P Systems [4] and P Systems with Symport and Antiport [6]. A new variant of these model have been introduced in [1] to evaluate communication that are dependent on some energy produced from evolution rules. To do this, a special object $e$ is introduced to the system to represent a quantum of energy.

We use the definition for EC P System with Energy (ECP-E) from [1], as follows,

**Definition 1.** An EC P System with Energy is a construct of the form

$\Pi = (O, e, \mu, w_1, \ldots, w_m, R_1, R'_1, \ldots, R_m, R'_m, i_{out})$

where:

(i) $m$ pertains to the total number of membranes;
(ii) $O$ is the alphabet of objects;
(iii) $\mu$ is the membrane structure which can be denoted by a set of paired square brackets with labels. We say that membrane $i$ is the parent membrane of a membrane $j$ if the paired squared bracket representing membrane $j$ is located inside the paired squared bracket representing membrane $i$, i.e., $[i] \subset [j]$. On the other hand, we say that membrane $j$ is a child membrane of membrane $i$.
(iv) $w_1, \ldots, w_m$ are strings over $O^*$ denoting the multisets of objects present in the regions bounded by membranes;
(v) $R_1, \ldots, R_m$ are sets of evolution rules, each associated with a region delimited by a membrane in $\mu$:

- An evolution rule is of the form $a \rightarrow v$ where $a \in O$, $v \in (O \cup \{e\}^*)$. In the event that this type of rule is applied, the object $a$ transforms into a multiset of objects $v$, in the next time step. Through evolution rules, object $e$ can be produced, but $e$ should never be in the initial configuration and object $e$ is not allowed to evolve.
(vi) $R'_1, \ldots, R'_m$ are sets of communication rules, each associated with a membrane in $\mu$: A communication rule can either be a symport or an antiport rule:

- A symport rule can be of the form $(ae^i, \text{in})$ or $(ae^j, \text{out})$, where $a \in O$, $i \geq 1$. By using this rule, $i$ copy of $e$ objects are consumed to transport object $a$ inside (denoted by in) or outside (denoted by out) the membrane where the rule is defined. To consume copies of object $e$ means that upon completion of the transportation of object involved in the rule, the occurrences of $e$ are lost, they do not pass from a region to another one. We say that $i$ is the energy of this rule.
- An antiport rule is of the form $(ae^i, \text{out}; be^j, \text{in})$ where $a, b \in O$ and $i, j \geq 1$; By using this rule, we know that there exists an object $a$ in the region immediately outside the membrane where the rule is declared, and an object $b$ inside the region bounded by the membrane. In the application of this rule, object $a$ and object $b$ are swapped using $i$ and $j$ copies of object $e$ in the different regions, respectively. As in symport rules, the copies of object $e$ are lost after the application. We, then, say that the number $i + j$ is the energy of this rule.

Note that no communication can be applied without the utilization of object $e$.

(vii) $i_{out} \in \{0, 1, \ldots, m\}$ is the output membrane. If $i_{out} = 0$, this means that the environment shall be the placeholder of the output.

We denote by $N(\Pi)$ the set of numbers generated by a given ECP-E II.

**2.1 An Example**

To show how ECP-E works, we shall give an example of an ECP-E with two membranes adapted from [1]:
Figure 1: Graphical representation of an ECP-E \( \Pi \) where \( N(\Pi) = \{3(2^n) \mid n \geq 0\} \). Adapted from [1].

\[
\Pi = \{(b,c), e, [1[2]2], 1, \{cb^2\}, \emptyset, \{r_{11}, r_{12}, \bar{r}_{11}, \bar{r}_{21}, r_{21}\}, \emptyset, 1\}
\]

where

- \( r_{11} : c \rightarrow bbcc \)
- \( r_{12} : c \rightarrow e \)
- \( \bar{r}_{11} : (ce, in) \)
- \( r_{21} : c \rightarrow c \)

A graphical illustration of \( \Pi \) is shown in Figure 1. Its output is \( N(\Pi) = \{3(2^n) \mid n \geq 0\} \). The computation to generate this proceeds as follows:

Initially, we can use either rule \( r_{11} \) or \( r_{12} \) in order to consume the only copy of object \( c \) in membrane 1. Using only \( r_{11} \), the number of objects \( b \) will always be twice the number of objects \( c \). If at any step \( i \), \( i \geq 1 \), we were able to use both \( r_{11} \) and \( r_{12} \), then in the next step, some copies of object \( c \) may be transported to membrane 2 through rule \( r_{11} \) leading the system to a nonhalting state. Thus, we either use only the rule \( c \rightarrow bbcc \) or only the rule \( c \rightarrow e \). After a finite number of steps, the computation will stop with \( 3(2^n) \) objects in region 1, for some \( n \geq 0 \) (the value \( n = 0 \) is obtained if we use the rule \( c \rightarrow e \) in the first step).

3. REPRESENTATION FOR ECP-E COMPUTATIONS

For the purpose of definitions we shall use here and in the succeeding sections, we add the following notations given an ECP-E \( \Pi \):

- We let \( \Gamma \) be the union of all rules in \( \Pi \) such that \( R = \bigcup_{1 \leq i \leq m} R_i \cup R'_i \). Also, we let \( n = |R| \).
- We let \( d \) be the cardinality of the set \( \Gamma \times H \) such that \( d = |\Gamma \times H| \).

In [2], the concept of configuration vector, spiking vector and spiking transition matrix were introduced to represent in matrix operation how a certain configuration can lead to the next configuration in the case of SN P Systems without delay. In this paper, we also adapt the concepts of these vectors and matrix, but we use the representation of elementary objects as in [3]. To accomplish this, we adapt the concept of total order over the set \( \Gamma \times H \) and over the rules in \( R \):

- We let \( O_{\text{objects}} \) be the total order on the set \( \Gamma \times H \), \( O_{\text{objects}} : \{1, \ldots, d\} \rightarrow \Gamma \times H \). Using this order, \( \Gamma \times H \) is represented as the finite sequence \( \langle \gamma_1, \ldots, \gamma_d \rangle \) where \( \gamma_k \in \Gamma \times H \) corresponds to the \( k^{\text{th}} \) pair of \( \Gamma \times H \) in the order \( O_{\text{objects}} \).
- We let \( O_{\text{rules}} \) be the total order over the rules in \( R \), \( O_{\text{rules}} : \{1, \ldots, n\} \rightarrow R \). We represent the order \( O_{\text{rules}} \) as the finite sequence \( \langle \beta_1, \ldots, \beta_n \rangle \) where \( \beta_k \in R \), \( 1 \leq k \leq n \), is the \( k^{\text{th}} \) rule of \( R \) in the order \( O_{\text{rules}} \).

We represent a configuration \( C_i \) as vector \( \vec{C}_i \) whose elements are numbers representing the multiplicity of objects in each membrane at time \( i \).

**Definition 2. Configuration Vector**

Given an ECP-E \( \Pi \) having order \( O_{\text{objects}} = \langle \gamma_1, \ldots, \gamma_d \rangle \) over \( \Gamma \times H \), a configuration vector is a representation of a configuration \( C_i \), denoted \( \vec{C}_i = (\bar{C}_i(\gamma_1), \ldots, \bar{C}_i(\gamma_d)) \) where \( \bar{C}_i : \Gamma \times H \rightarrow \mathbb{N} \), and for all \( \gamma_k = (\alpha, j) \in \Gamma \times H \), \( 1 \leq k \leq d \), \( \bar{C}_i((\alpha, j)) \) corresponds to the multiplicity of object \( \alpha \) at membrane \( j \) in configuration \( C_i \).

We use an application vector in order to keep track of the rules chosen and applied at a certain transition.

**Definition 3. Application Vector**

Given an ECP-E \( \Pi \) having order \( O_{\text{rules}} = \langle \beta_1, \ldots, \beta_n \rangle \) over \( R \), an application vector \( \vec{a}_i \) on a given transition \( C_{i-1} \Rightarrow C_i \) is defined as \( \vec{a}_i = (\vec{a}_i(\beta_1), \ldots, \vec{a}_i(\beta_n)) \) where \( \vec{a}_i(\beta_k), 1 \leq k \leq n \), gives the number of application of rule \( \beta_k \) during the transition \( C_{i-1} \Rightarrow C_i \). If the rule is not applied in this transition, the value of its \( \vec{a}_i \) will be zero.

We employ the concept of transition matrix to show the effect upon application of each rule in \( R \). The negative values correspond to objects that will be consumed or moved out of a given region and the positive values correspond to objects that will be produced or moved into a given region upon application of a rule.
A zero value may refer to objects that are either not involved in the rule or involved but the summed effect of its production and consumption results to zero.

**Definition 4. Transition Matrix**
Given an ECP-E Π with order \( O_{\text{objects}} = \langle \gamma_1, \ldots, \gamma_d \rangle \) over \( \Gamma \times H \), and order \( O_{\text{rules}} = \langle \beta_1, \ldots, \beta_n \rangle \), a transition matrix \( M_\Pi \) is an \( n \times d \) matrix defined as follows:

\[
M_\Pi = [\sigma_{ik}]_{n \times d}
\]

where \( 1 \leq i \leq n \), \( 1 \leq k \leq d \) and the rows correspond to elements in the order \( O_{\text{objects}} \) and columns correspond to elements in the order \( O_{\text{rules}} \). The value of \( \sigma_{ik} \) changes depending on what type of rule is \( \beta_i \). Let \( \gamma_k \) be the pair \((a, j)\).

- if \( \beta_i \) corresponds to an evolution rule of the form \( a \rightarrow v \) defined in membrane \( h \):

\[
\sigma_{ik} = \begin{cases} 
  -1 + c & \text{if } \alpha = a, j = h \\
  \text{where } c \text{ is the count of } \alpha \text{ at the right hand side} \\
  \text{of the rule } \beta_i \\
  c & \text{if } \alpha = v, j = h \\
  \text{where } c \text{ is the count of } \alpha \text{ at the right hand side} \\
  \text{of the rule } \beta_i \\
  1 & \text{if } \alpha = a, j = h \\
  0 & \text{otherwise}
\end{cases}
\]

- if \( \beta_i \) corresponds to a communication rule of the form \((ae^p, in)\) defined in membrane \( h \):

\[
\sigma_{ik} = \begin{cases} 
  -1 & \text{if } \alpha = a, j = h \\
  \text{is the parent membrane of } h \\
  -p & \text{if } \alpha = e, j = h \\
  \text{is the parent membrane of } h \\
  1 & \text{if } \alpha = a, j = h \\
  0 & \text{otherwise}
\end{cases}
\]

- if \( \beta_i \) corresponds to a communication rule of the form \((ae^p, out)\) defined in membrane \( h \):

\[
\sigma_{ik} = \begin{cases} 
  1 & \text{if } \alpha = a, j = h \\
  \text{is the parent membrane of } h \\
  -p & \text{if } \alpha = e, j = h \\
  -1 & \text{if } \alpha = a, j = h \\
  0 & \text{otherwise}
\end{cases}
\]

- if \( \beta_i \) corresponds to a communication rule of the form \((ae^p, in; be^q, out)\) defined in membrane \( h \):

\[
\sigma_{ik} = \begin{cases} 
  1 & \text{if } \alpha = a, j = h \\
  \text{or } \alpha = b, j = h \\
  \text{is the parent membrane of } h \\
  -1 & \text{if } \alpha = b, j = h \\
  \text{or } \alpha = a, j = h \\
  \text{is the parent membrane of } h \\
  -p & \text{if } \alpha = e, j = h \\
  \text{is the parent membrane of } h \\
  -q & \text{if } \alpha = e, j = h \\
  0 & \text{otherwise}
\end{cases}
\]

4. **ECP-E Computations Through Matrix Operations**

Using the matrices defined in Section 3, we now show how we can simulate computations in ECP-E. The next definition and style of proof are similar to that in [2].

**Definition 5. Transition Net Gain Vector**
Given an ECP-E Π with order \( O_{\text{objects}} = \langle \gamma_1, \ldots, \gamma_d \rangle \) over \( \Gamma \times H \), and order \( O_{\text{rules}} = \langle \beta_1, \ldots, \beta_n \rangle \) over \( R \), \( M_\Pi \) is the transition matrix of \( M_\Pi \), \( \vec{a}_k \) is the application vector at transition \( C_{k-1} \Rightarrow C_k \). Then the transition net gain vector at step \( k \) can be obtained by

\[
NG_k = \vec{C}_k - \vec{C}_{k-1}.
\]

**Lemma 4.1.** Given an ECP-E Π with \( O_{\text{objects}} = \langle \gamma_1, \ldots, \gamma_d \rangle \) over \( \Gamma \times H \), and order \( O_{\text{rules}} = \langle \beta_1, \ldots, \beta_n \rangle \) over \( R \), \( M_\Pi \) is the transition matrix of \( M_\Pi \), \( \vec{a}_k \) is the application vector at transition \( C_{k-1} \Rightarrow C_k \). Then the transition net gain vector at step \( k \) can be obtained by

\[
NG_k = \vec{a}_k \cdot M_\Pi.
\]

**Proof.** To prove this, we identify the role of the application vector \( \vec{a}_k \) in achieving \( NG_k \). From Definition 5, we know that \( NG_k = C_k - C_{k-1} \). Let \( NG_k = (g_1, g_2, \ldots, g_d) \). Each \( g_r \) is the summation of all consumed and produced \( \gamma_r \) subject to the total application, based on vector \( \vec{a}_k \) of all rules involving \( \gamma_r \). To get this summation, we need to identify the total effect of using \( \vec{a}_k(\beta_i) \) applications of rule \( \beta_i \), \( 1 \leq i \leq n \). We know that \( M_\Pi \) keeps track of the effect upon a single application of a rule \( \beta_i \). By multiplying the number of applications of rule \( \beta_i \), i.e. \( \vec{a}_k(\beta_i) \), to the effect of the application of a single rule, we get the desired effect of applying rule \( \beta_i \) exactly \( \vec{a}_k(\beta_i) \) times. In equation, for all \( g_r, g_r = \sum_{r=1}^{n} \vec{a}_k(\beta_i) \) for \( r \), \( 1 \leq r \leq d \). Thus, \( NG_k = \vec{a}_k \cdot M_\Pi \).

**Theorem 4.1.** Given an ECP-E Π with \( O_{\text{objects}} = \langle \gamma_1, \ldots, \gamma_d \rangle \) over \( \Gamma \times H \), and order \( O_{\text{rules}} = \langle \beta_1, \ldots, \beta_n \rangle \) over \( R \), \( M_\Pi \) is the transition matrix of \( M_\Pi \), \( \vec{C}_k \) is the configuration vector at time \( k \) and \( \vec{a}_k \) is the application vector at transition \( C_{k-1} \Rightarrow C_k \), then, every configuration \( C_k \) of \( \Pi \) can be obtained by

\[
C_k = C_{k-1} + \vec{a}_{k-1} \cdot M_\Pi
\]

**Proof.** This proof follows directly from the preceding lemma.

Given an ECP-E Π, we can already build a transition matrix \( M_\Pi \) and a configuration vector \( \vec{C}_0 \) for the initial configuration following their definitions in Section 3. These can be accomplished after fixing a total order over \( \Gamma \times H \) and over the set of rules \( R \) for the ECP-E Π. To illustrate computation, we shall show a
sample configuration and computation for the example given in Section 2. We fixed order $\mathcal{O}_{\text{objects}}$ to be \((b,1),(c,1),(e,1),(b,2),(c,2),(e,2)\) and $\mathcal{O}_{\text{rules}}$ to be \((r_{11},r_{12},r_{11}^*,r_{21})\). The corresponding $C_0$ and $M_\Pi$ for this ECP-E are shown below:

$$C_0 = (2,1,0,0,0,0)$$

<table>
<thead>
<tr>
<th>$M_\Pi$</th>
<th>(b,1)</th>
<th>(c,1)</th>
<th>(e,1)</th>
<th>(b,2)</th>
<th>(c,2)</th>
<th>(e,2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_{11}$</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$r_{12}$</td>
<td>0</td>
<td>−1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$r_{11}^*$</td>
<td>0</td>
<td>−1</td>
<td>−1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$r_{21}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

From the configuration, we can either use any one of rule $r_{11}$ or $r_{12}$, so that in the next step, the vector representation of their corresponding configuration will be $C_1 = (4,2,0,0,0,0)$ and $C_1 = (2,0,1,0,0,0)$, respectively. Note that only one application must be applied. Thus, at this step, our application vector can either be $\vec{a}_i = (1,0,0,0)$ if rule $r_{11}$ is used or $\vec{a}_i = (0,1,0,0)$ if rule $r_{12}$ is chosen to be applied. If we use $\vec{a}_i = (1,0,0,0)$, and we have established that

$$C_0 + \vec{a}_1 \cdot M_\Pi = C_1,$$

the sample computation to achieve the next configuration will be:

$$(2 1 0 0 0 0) + (1 0 0 0) = (4 2 0 0 0 0)$$

Finally, the following Corollary is a direct consequence of the preceding Theorem.

**Corollary 4.1.** Given an ECP-E $\Pi$ with order $\mathcal{O}_{\text{objects}} = \langle \gamma_1, \ldots, \gamma_d \rangle$ over $\Gamma \times H$, and order $\mathcal{O}_{\text{rules}} = \langle \beta_1, \ldots, \beta_n \rangle$ over $R$, $M_\Pi$ is the transition matrix of $M_\Pi$, $C_k$ is the configuration vector at time $k$ and $\vec{a}_k$ is the application vector at step $k$, then, the previous configuration $C_{k-1}$ of $\Pi$ can be obtained by

$$C_{k-1} = C_k - \vec{a}_k \cdot M_\Pi$$

### 5. Finding a Valid Application Vector

As mentioned in Section 4, the application vector represents the number of applications of each rule at a step. Given the current configuration vector, we multiply the application vector with the transition matrix and add it to the current configuration vector to get to the vector representation of the next configuration. However, it is important to remember that the application of each rule must satisfy nondeterminism and maximal parallelism. Due to the many possible application vector that may be multiplied to the transition matrix, we know that the nondeterministic property for rule application is satisfied. However, the idea of finding an application vector that will also satisfy maximal parallelism is not an easy task.

**Definition 6.** A valid application vector is an application vector that satisfies the requirement of maximal parallelism, i.e., all objects that may evolve (or be communicated) should evolve (or be communicated).

Note that, if the addition of the net gain with the current configuration leads to a vector having at least one negative value, this will already mean that the application vector is not valid because we cannot have negative copies of objects. On the other hand, arriving with a resulting configuration vector with nonnegative elements does not assure us of the validity of the application vector.

As a way to check for the validity of the application vector, we will need a trigger matrix $T_\Pi$ that contains all the needed objects in order to activate a rule.

**Definition 7. Trigger Matrix**

Given an ECP-E $\Pi$ with order $\mathcal{O}_{\text{objects}} = \langle \gamma_1, \ldots, \gamma_d \rangle$ over $\Gamma \times H$, and order $\mathcal{O}_{\text{rules}} = \langle \beta_1, \ldots, \beta_n \rangle$, a transition matrix $T_\Pi$ is a $d \times n$ matrix defined as follows:

$$T_\Pi = [\rho_{ki}]_{d \times n}$$

where $1 \leq k \leq d$, $1 \leq i \leq n$, the rows correspond to elements in the order $\mathcal{O}_{\text{objects}}$ and columns correspond to elements in the order $\mathcal{O}_{\text{rules}}$. The value of $\rho_{ki}$ changes depending on what type of rule is $\beta_i$. Let $\gamma_h$ be the pair $(a, j)$.

- if $\beta_i$ corresponds to an evolution rule of the form $a \rightarrow v$ defined in membrane $h$:

  $$\rho_{ki} = \begin{cases} 
  1 & \text{if } a = a, j = h \\
  0 & \text{otherwise}
  \end{cases}$$

- if $\beta_i$ corresponds to a communication rule of the form $(ae^p, \text{in})$ defined in membrane $h$:

  $$\rho_{ki} = \begin{cases} 
  1 & \text{if } a = a, j = \text{the parent membrane of } h \\
  p & \text{if } a = e, j = \text{the parent membrane of } h \\
  0 & \text{otherwise}
  \end{cases}$$

- if $\beta_i$ corresponds to a communication rule of the form $(ae^p, \text{out})$ defined in membrane $h$:

  $$\rho_{ki} = \begin{cases} 
  1 & \text{if } a = a, j = h \\
  p & \text{if } a = e, j = h \\
  0 & \text{otherwise}
  \end{cases}$$

- if $\beta_i$ corresponds to a communication rule of the form $(ae^p, \text{in})$ defined in membrane $h$:

  $$\rho_{ki} = \begin{cases} 
  1 & \text{if } a = a, j = \text{the parent membrane of } h \\
  p & \text{if } a = e, j = \text{the parent membrane of } h \\
  0 & \text{otherwise}
  \end{cases}$$

- if $\beta_i$ corresponds to a communication rule of the form $(ae^p, \text{out})$ defined in membrane $h$:

  $$\rho_{ki} = \begin{cases} 
  1 & \text{if } a = a, j = h \\
  p & \text{if } a = e, j = h \\
  0 & \text{otherwise}
  \end{cases}$$
form (ae^p, in; be^q, out) defined in membrane h:
\[
\rho_{ki} = \begin{cases} 
1 & \text{if } \alpha = b, j = h \\
\text{or } \alpha = a, j \text{ is the parent membrane of } h \\
\rho & \text{if } \alpha = e, j \text{ is the parent membrane of } h \\
0 & \text{otherwise}
\end{cases}
\]

We shall use the trigger matrix \( T_H \) to prove the following theorem:

**Theorem 5.1.** Given an ECP-E \( P \) with a trigger matrix \( T_H \) and a configuration \( C_{k-1} \), the problem of finding a valid application vector \( \vec{a}_k \) for the transition \( C_{k-1} \Rightarrow C_k \) is reducible to finding a set of natural numbers that satisfy conditions involving a system of linear inequalities.

**Proof.** To prove this, we shall construct from \( T_H \) and \( C_{k-1} \) the inequalities that needs to be solved. Any set of natural numbers solving the system can also be used as elements of a valid application vector which when multiplied to the transition matrix \( M_H \) assures correct evolution from \( C_{k-1} \) to \( C_k \).

We need the trigger matrix \( T_H \) to identify the rules involving a particular object in a membrane. Since we know that the summation of all rules triggered by an object, say \( \gamma_\ell \in \Gamma \times H \), should not exceed \( \gamma_i \)'s total number of copies in configuration \( C_{k-1} \), we assure that this condition is satisfied for all elementary object \( \gamma_i \) by multiplying \( T_H \) with \( \vec{a}_k \) transposed and restricting that the resulting vector should be less than or equal to \( \vec{C}_k^T \) (\( \vec{C}_k \) transposed) such that:
\[
T_H \cdot \vec{a}_k \leq \vec{C}_k^T
\]

From the equation above, we can arrive with at most \( d \) initial conditions for determining validity of an application vector. Violation of any one of the condition would result to having negative total copies of an object in the next configuration.

As additional condition to assure us that each rule’s number of application is maximal, we have to restrict that for all \( \vec{a}_k(\beta_\ell) \) defined on a transition \( C_{k-1} \Rightarrow C_k \), applying a rule once more will be impossible because there is not enough trigger object to enable its application. This means that for all \( \vec{a}_k(\beta_\ell) \), \( \vec{a}_k(\beta_\ell) + 1 \) will violate any one of the initial conditions. To do this, we first label each condition as \( Cond_i \), \( 1 \leq i \leq d \) where:
\[
Cond_i : \sum_{j=1}^{n} \rho_{ij}(\vec{a}_k(\beta_j)) \leq \vec{C}_{k-1}(\gamma_i)
\]

Then, maximal parallelism requires that for all rule \( \beta_\ell \), there should exists at least one \( Cond_i \) involving \( \beta_\ell \) that satisfies the condition:
\[
\sum_{j=1, j\neq \ell}^{n} \rho_{ij}(\vec{a}_k(\beta_j)) + \rho_{ij}(\vec{a}_k(\beta_\ell)) + 1 > \vec{C}_{k-1}(\gamma_i)
\]

If there exists a rule \( \beta_\ell \) such that \( \vec{a}_k(\beta_\ell) + 1 \) does not violate any one condition, this would mean that the rule \( \beta_\ell \) can be applied one more time. Thus, there still exist some objects that can be evolved or transported through the rule \( \beta_\ell \). Clearly, this would imply that the current value of \( \vec{a}_k(\beta_\ell) \) is not maximal.

Any set of natural number satisfying all of these conditions is a valid application vector.

We shall illustrate how the conditions for maximal parallelism can be achieved on our example ECP-E whose graphical illustration is shown in Figure 1. Following \( T_H \cdot \vec{a}_k^T \leq \vec{C}_k^T \):
\[
\begin{pmatrix}
0 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0
\end{pmatrix}
\begin{pmatrix}
\vec{a}_1(r_{11}) \\
\vec{a}_1(r_{12}) \\
\vec{a}_1(r_{11}) \\
\vec{a}_1(r_{21})
\end{pmatrix} \leq \begin{pmatrix}
2 \\
1 \\
0 \\
0
\end{pmatrix}
\]

Yields the conditions:
\[
\begin{align*}
\vec{a}_1(r_{11}) + \vec{a}_1(r_{12}) + \vec{a}_1(r_{11}) & \leq 1 \\
\vec{a}_1(r_{11}) & \leq 0 \\
\vec{a}_1(r_{21}) & \leq 0
\end{align*}
\]

All of the conditions above must be satisfied. The additional conditions required per rule are the following:

- For rule \( r_{11} \) and \( r_{12} \):
  \[
  \vec{a}_1(r_{11}) + \vec{a}_1(r_{12}) + \vec{a}_1(r_{11}) + 1 > 1
  \]

- For rule \( r_{11} \), either of these conditions must be true:
  \[
  \begin{align*}
  \vec{a}_1(r_{11}) + \vec{a}_1(r_{12}) + \vec{a}_1(r_{11}) + 1 & > 1 \\
  \vec{a}_1(r_{11}) + 1 & > 0
  \end{align*}
  \]

- For rule \( r_{21} \):
  \[
  \vec{a}_1(r_{21}) + 1 > 0
  \]

As can be traced, any one of the application vectors \( \vec{a}_1 = (1, 0, 0, 0) \) and \( \vec{a}_1 = (0, 1, 0, 0) \) satisfies all posed conditions.

6. CONCLUSION

In this paper, we have shown how we can simulate computations in ECP-E through matrix operations similar to a previous literature. This will be helpful not only in analyzing computations in ECP-E but also in the implementation of this variant especially in parallel devices like Graphics Processing Units. The contribution of this paper may also be extended to other variants of P System especially those that do not involve dynamic increase or decrease on the total number of membranes. We leave as an open problem the extension of this representation to other P
System variants involving membrane operations like membrane division, creation or dissolution.

We also included in this work insights on determining the combination of rule application that satisfies maximal parallelism. We show that this problem can be answerable by satisfying conditions involving system of linear inequalities.

Acknowledgements
The authors would like to express their gratitude for the full support and unconditional assistance of people from Algorithms and Complexity Laboratory.

R.A. Juayong is partially funded by the Engineering Research and Development (ERDT) Scholarship Program. H. Adorna is funded by the Alexan Professorial Chair of the College of Engineering, UP Diliman.

7. REFERENCES