Implementation of Evolution-Communication P Systems with energy in P-Lingua

Algina Castillo, Joni Marie E. Jimenez,
Nestine Hope S. Hernandez, Richelle Ann B. Juayong, Henry N. Adorna
Algorithms and Complexity Laboratory, Department of Computer Science
College of Engineering, University of the Philippines-Diliman
Diliman, Quezon City, Philippines
{ancastillo,jejimenez,rbjuayong}@up.edu.ph
{nshernandez,hnadorna}@dcs.upd.edu.ph

ABSTRACT

In this proposal, we present our initial research regarding the implementation of Evolution-Communication P system with energy (ECPe systems) in P-Lingua ECPe Systems has been proposed as an extension to Evolution-Communication P Systems. The rules in ECPe works exactly as in ECP Systems except for the additional consideration for energy. Energy is a property of the regions and is represented by an integer greater than or equal to 0. Initial configuration does not permit having an energy in the regions. Instead it is produced during evolution and is consumed during communication. A construct for ECPe had been proposed already. ECP Systems had been implemented in P-Lingua. This paper will propose an implementation of ECPe Systems in P-Lingua by extending ECP Systems in P-Lingua.

Key words: Membrane computing; Evolution-Communication P System with energy; PLingua

1. INTRODUCTION

Membrane computing[1] is a branch of natural computing aiming to abstract computing ideas and models from the structure and the functioning of living cells. Membrane Computing deals with distributed and parallel computing models called P Systems which processes multisets of symbol objects in a localized manner with an essential role played by communication between compartments. Many different types of P Systems exist.

ECP Systems with energy[2] is a variant of P Systems that originates from Evolution-Communication P Systems. Evolution-Communication P Systems, on the other hand, is a hybrid of Transition P Systems and P Systems with Symport and Antiport rules. ECPe Systems has a special object called 'energy' that is produced in evolution rules and consumed in communication rules.

P-Lingua [3] is a programming language used to compile and simulate P systems. As of this time of writing, P-Lingua and its library- pLinguaCore is in its version 4.0 and already has a compiler and simulator for Spiking Neural P Systems, Kernel P Systems, Tissue-Like P Systems with Cell separation rules and with division rules, Population Dynamics P systems , Active Membrane P Systems with division rules, Transition P Systems, and many other cell-like P systems.

2. EVOLUTION COMMUNICATION P-SYSTEMS WITH ENERGY (ECPE)

ECPe systems have also been implemented in GPU in a form of matrix.

Evolution Communication P systems with energy has the following construct: [2]

$$\prod = (O, \mathrm{e}, \mu, \omega_i, ..., \omega_m, R_1, R'_1, ..., R_m, R'_m, i_{out})$$

where:

- m - the total number of membranes
- O - alphabet of objects
- μ - membrane structure
- $\omega_1, ..., \omega_m$ - strings over O^*, where ω_i denotes the multiset of object present in the region bounded by membrane i.
- i_{out} - output membrane
- $R_1, ..., R_m$ - set of evolution rules. Object e can be produced, but should never be in the initial configuration and is not allowed to evolve.
- $R'_1, ..., R'_m$ - set of communication rules. Symport has rules (ae^i, in) or (ae^i, out), where $a \in 0$, $i \geq 1$. Antiport has rule $(ae^i, out; be^j, in)$, $i, j \geq 1$, where $a, b \in O$, $i, j \geq 1$. Copies of object e are lost after application.

3. P-LINGUA

P-Lingua is a programming language for Membrane Computing [3]. Its library *pLinguaCore*, which is written in Java and now on its version 4.0, can be downloaded and edited under GPL license from their website[3] .
At this time of writing, P-Lingua can already compile and simulate Spiking Neural P Systems, Kernel P Systems, Tissue-Like P Systems with Cell separation rules and with division rules, Population Dynamics P systems , Active Membrane P Systems with division rules, and many other cell-like P systems.
For this paper, let's look into the models (originally presented in [1]) of P-Lingua that might be helpful.

3.0.1 Model.

Since P-Lingua supports many P systems, it is necessary to identify what particular P system is our file defined.This is important because not each type of rule is allowed in every model, for example, membrane creation rules are not permitted in P systems with symport/antiport rules. The built-in compiler of P-Lingua detects such error. Models are specified by using $@model < modelname >$ and at this stage, the allowed models are:
$@model < membrane\_division >$
$@model < membrane\_creation >$
$@model < transition >$
$@model < probabilistic >$
$@model < evolution\_communication >$
$@model < ev\_symport\_antiport >$
$@model < symport\_antiport >$
$@model < TSCS >$
$@model < spiking\_psystems >$
$@model < tissue\_psystems >$

3.0.2 Rules.

For relevance and simplicity, let's have a look on the rules in the model for ECP Systems which is $@model < evolution\_communication >$.

Evolution Rule.
$[a - - > b]'h$
where a and $b \in O$ and h is the label of the membrane wherein this evolution rule is applicable.

Send-in Rule.
$a[]'h - - > [a]$
where $b \in O$, and h is the label of the child membrane of the membrane containing a.a will be moved to membrane with label h after rule application.

Send-out Rule.
$[b]'h - - > b[]$
where $b \in O$, and h is the label of the membrane wherein a is located.a is then moved to the parent membrane of membrane h after rule application.

Antiport Rule.
$a[b]'h - - > b[a]'h$
where a and $b \in O$,and h is the label of the membrane where b is contained before the rule is applied.

3.1 Model for ECPe Systems

ECPe Systems have been extended in ECP, thus, also has a model name of "*evolution_communication*".

ECPe copies the same syntax as ECP Systems for function definition, membrane configuration initialization, and multiset initialization for each membrane. It can do so because of the absence of the energy e during the initialization. ECP and ECPe syntax will differ during the rule implementation, which we will now show:
Evolution rules

$$[\ a\ ]'h\ \text{- ->}\ [\ b\ ]\ :\ e$$

where:

- a and $b \in O$.
- h is a label of the membrane where the evolution rule applies
- e is an integer ≥ 0. It is represented by the energy produced during evolution.

Communication rules: Symport
For send-in:

$$a[\ ]'h{:}e\ \text{- ->}\ [a]$$

where:

- $a \in O$.
- h is the membrane label where a will be sent in. a will always be in the membrane immediately outside membrane h. After the rule is applied, a will be inside membrane h.
- e is an integer ≥ 1. It is represented by the energy consumed during communication. The rule cannot be applied without the sufficient number of e in the membrane directly outside of h.

For send out:

$$[b]'h{:}e\ \text{- - >}\ b[\ ]$$

where:

- $b \in O$.

- h is the label of the membrane where b is contained initially. After the rule is applied, a will be transferred the membrane immediately outside h .
- e is an integer ≥ 1. It is represented by the energy consumed during communication. The rule cannot be applied without the sufficient number of e in membrane h.

Communication rules: Antiport

$$b[a]'h{:}e_1 \text{ - - } > a[b]'h{:}e_2$$

where:

- a and $b \in O$.
- h is the label of the membrane where a is contained initially. After the rule is applied, a will be transferred the membrane immediately outside h. Consequently, b, which is in the membrane outside the membrane h, will be transferred inside the membrane h.
- $e_1, e_2 \geq 1$. e_1 is associated to the membrane where a is initially contained (membrane h) while e_2 is associated to the membrane where b is initially contained (membrane outside h). The rule cannot be applied without sufficient number of e_1 in the region of membrane h and without sufficient number e_2 in the region directly outside membrane h.

4. P-LINGUA MODIFICATION

Since ECPe is an extension of ECP, we decided to extend the model for ECP which is $< evolution\_communication >$ in P-Lingua to include the implementation of energy e. Due to this, we didn't add any additional classes and directories in the P-Lingua Java. Firstly, in order to recognize e in P-Lingua input (indicated by a colon), we modified the Parser to incorporate the use of energy. Though it was successful, this resulted in the lost of the use of ranges. Ranges are another component in P-Lingua that uses colon as an indicator.
After taking care of the input, we made the energy an attribute of a membrane - both of the configuration and the rule. Then we added the energy attribute in the compiled output.
As for the simulation, we modified the classes responsible for executing cellLike simulation to incorporate the use of energy. For example, adding energies during implementation of evolution rule and subracting energies during communication rule from involved membranes.

5. TESTING AND RESULTS

5.1 Sample Input

$@model < evolution\_communication >$
def SAMP() {
$@mu = [[[[]'4]'3]'2]'1;$
$@ms(4) = ab, s, c;$
$@ms(3) = bad, c;$
$@ms(2) = fg, ab;$
$@ms(1) = bad, ab * 2;$

$[bad]'1 - - > [cd] : 1;$
$[ab]'1 - - > [fg] : 1;$
$cd[]'2 : 4 - - > [cd]'2;$
$fg[]'3 : 2 - - > [fg];$

$[ab]'3 - - > [fg] : 3;$
$c[s]'3 : 2 - - > s[c]'3 : 1;$

$[bad]'3 - - > [s] : 3;$
$[s]'4 - - > [bd] : 5;$
$[bd]'4 : 1 - - > bd[];$
}

def main() {
call SAMP();
}

5.1.1 Simulation Output

****

CONFIGURATION: 0
TIME: 0.0 s.
MEMORY USED: 62976
FREE MEMORY: 57060
TOTAL MEMORY: 932352

MEMBRANE ID: 3, Label: 4, Charge: 0, Energy: 0
Multiset: ab, s, c
Parent membrane ID: 2

MEMBRANE ID: 2, Label: 3, Charge: 0, Energy: 0
Multiset: bad, c
Internal membranes count: 1
Parent membrane ID: 1

MEMBRANE ID: 1, Label: 2, Charge: 0, Energy: 0
Multiset: fg, ab
Internal membranes count: 1
Parent membrane ID: 0

SKIN MEMBRANE ID: 0, Label: 1, Charge: 0, Energy: 0
Multiset: bad, ab∗2
Internal membranes count: 1

STEP: 1

Rules selected for MEMBRANE ID: 3, Label: 4, Charge: 0, Energy: 5
1 ∗ #−1 [s −−> bd]'4

Rules selected for MEMBRANE ID: 2, Label: 3, Charge: 0, Energy: 3
1 ∗ #−1 [bad −−> s]'3

Rules selected for SKIN MEMBRANE ID: 0, Label: 1, Charge: 0, Energy: 2
2 ∗ #−1 [ab −−> fg]'1
1 ∗ #−1 [bad −−> cd]'1

****

CONFIGURATION: 1
TIME: 0.162 s.
MEMORY USED: 62976
FREE MEMORY: 56405
TOTAL MEMORY: 932352

MEMBRANE ID: 3, Label: 4, Charge: 0, Energy: 5
Multiset: ab, c, bd
Parent membrane ID: 2

MEMBRANE ID: 2, Label: 3, Charge: 0, Energy: 3
Multiset: c, s
Internal membranes count: 1
Parent membrane ID: 1

MEMBRANE ID: 1, Label: 2, Charge: 0, Energy: 0
Multiset: fg, ab
Internal membranes count: 1
Parent membrane ID: 0

SKIN MEMBRANE ID: 0, Label: 1, Charge: 0, Energy: 2
Multiset: fg∗2, cd
Internal membranes count: 1

STEP: 2

Rules selected for MEMBRANE ID: 3, Label: 4, Charge: 0, Energy: 4
1 ∗ #−1 [bd]'4:1 −−> []bd:0

****

CONFIGURATION: 2
TIME: 0.174 s.
MEMORY USED: 62976
FREE MEMORY: 56405
TOTAL MEMORY: 932352

MEMBRANE ID: 3, Label: 4, Charge: 0, Energy: 4
Multiset: ab, c
Parent membrane ID: 2

MEMBRANE ID: 2, Label: 3, Charge: 0, Energy: 3
Multiset: c, s, bd
Internal membranes count: 1
Parent membrane ID: 1

MEMBRANE ID: 1, Label: 2, Charge: 0, Energy: 0
Multiset: fg, ab
Internal membranes count: 1
Parent membrane ID: 0

SKIN MEMBRANE ID: 0, Label: 1, Charge: 0, Energy: 2
Multiset: fg∗2, cd
Internal membranes count: 1

Halting configuration
(No rule can be selected to be executed in the next step)

5.1.2 Analysis

The time it took is around 1.79E-4 s.

5.2 Sample Input 2

```
@model<evolution_communication>
def SAMP() {
@mu = [ [ ]'2]'1;
@ms(2) = S;


[S]'2−−> [Sp,a]:2;
[Sp]'2−−> [Spp,b];

[a]'2:1−−>a[];
[b]'2:1−−>b[];
[Spp]'2:1−−>Spp[];


[a]'1−−> [ap]:1;
[b]'1−−> [bp]:1;

[ap]'1:1−−>ap[];
[bp]'1:1−−>bp[];

}

def main(){
        call SAMP();
}
```

5.2.1 Simulated Output

****

CONFIGURATION: 0
TIME: 0.0 s.
MEMORY USED: 62976
FREE MEMORY: 57060
TOTAL MEMORY: 932352

MEMBRANE ID: 1, Label: 2, Charge: 0, Energy: 0
Multiset: S
Parent membrane ID: 0

SKIN MEMBRANE ID: 0, Label: 1, Charge: 0, Energy: 0
Multiset: #
Internal membranes count: 1

STEP: 1

Rules selected for MEMBRANE ID: 1, Label: 2, Charge: 0, Energy: 2
1 ∗ #−1 [S −−> Sp, a]'2

****

CONFIGURATION: 1
TIME: 0.095 s.
MEMORY USED: 62976
FREE MEMORY: 56732
TOTAL MEMORY: 932352

MEMBRANE ID: 1, Label: 2, Charge: 0, Energy: 2
Multiset: Sp, a
Parent membrane ID: 0

SKIN MEMBRANE ID: 0, Label: 1, Charge: 0, Energy: 0
Multiset: #
Internal membranes count: 1

STEP: 2

Rules selected for MEMBRANE ID: 1, Label: 2, Charge: 0, Energy: 1
1 * #−1 [Sp −−> Spp, b]'2
1 * #−1 [a]'2:1 −−> []a:0

****

CONFIGURATION: 2
TIME: 0.109 s.
MEMORY USED: 62976
FREE MEMORY: 56405
TOTAL MEMORY: 932352

MEMBRANE ID: 1, Label: 2, Charge: 0, Energy: 1
Multiset: Spp, b
Parent membrane ID: 0

SKIN MEMBRANE ID: 0, Label: 1, Charge: 0, Energy: 0
Multiset: a
Internal membranes count: 1

STEP: 3

Rules selected for MEMBRANE ID: 1, Label: 2, Charge: 0, Energy: 0
1 * #−1 [Spp]'2:1 −−> []Spp:0
1 * #−1 [b]'2:1 −−> []b:0

Rules selected for SKIN MEMBRANE ID: 0, Label: 1, Charge: 0, Energy: 1
1 * #−1 [a −−> ap]'1

****

CONFIGURATION: 3
TIME: 0.118 s.
MEMORY USED: 62976
FREE MEMORY: 56405
TOTAL MEMORY: 932352

MEMBRANE ID: 1, Label: 2, Charge: 0, Energy: 0
Multiset: b
Parent membrane ID: 0

SKIN MEMBRANE ID: 0, Label: 1, Charge: 0, Energy: 1
Multiset: Spp, ap
Internal membranes count: 1

STEP: 4

Rules selected for SKIN MEMBRANE ID: 0, Label: 1, Charge: 0, Energy: 1
1 * #−1 [ap]'1:1 −−> []ap:0

****

CONFIGURATION: 4
TIME: 0.125 s.
MEMORY USED: 62976
FREE MEMORY: 56077
TOTAL MEMORY: 932352

MEMBRANE ID: 1, Label: 2, Charge: 0, Energy: 0
Multiset: b
Parent membrane ID: 0

SKIN MEMBRANE ID: 0, Label: 1, Charge: 0, Energy: 1
Multiset: Spp
Internal membranes count: 1

ENVIRONMENT: ap

Halting configuration
(No rule can be selected to be executed in the next step)

5.3 Sample Input 3

This is modified version of Sample Input 2. A rule is deleted to always end with a halting configuration.

```
@model<evolution_communication>
def SAMP() {
@mu = [ [ ]'2]'1;
@ms(2) = S;

[S]'2−−> [S,a]:2;
[S]'2−−> [Sp];
[Sp]'2−−> [Spp,b];
[Sp]'2−−> [Spp];

[a]'2:1−−>a[];
[b]'2:1−−>b[];
[Spp]'2:1−−>Spp[];


[a]'1−−> [ap]:1;
[b]'1−−> [bp]:1;

[ap]'1:1−−>ap[];
[bp]'1:1−−>bp[];
```

```
}

def main(){
        call SAMP();
}
```

5.3.1 Simulation Output

```
**********************************************

    CONFIGURATION: 0
    TIME: 0.0 s.
    MEMORY USED: 61440
    FREE MEMORY: 57887
    TOTAL MEMORY: 912064

    MEMBRANE ID: 1, Label: 2, Charge: 0, Energy: 0
    Multiset: S
    Parent membrane ID: 0

    SKIN MEMBRANE ID: 0, Label: 1, Charge: 0, Energy: 0
    Multiset: #
    Internal membranes count: 1

----------------------------------------------------

    STEP: 1

    Rules selected for MEMBRANE ID: 1, Label: 2, Charge: 0,
    Energy: 2
    1 * #1 [S -—> S, a]'2

**********************************************

    CONFIGURATION: 1
    TIME: 0.256 s.
    MEMORY USED: 61440
    FREE MEMORY: 57887
    TOTAL MEMORY: 912064

    MEMBRANE ID: 1, Label: 2, Charge: 0, Energy: 2
    Multiset: S, a
    Parent membrane ID: 0

    SKIN MEMBRANE ID: 0, Label: 1, Charge: 0, Energy: 0
    Multiset: #
    Internal membranes count: 1

------------------------------------
--------------

    STEP: 2

    Rules selected for MEMBRANE ID: 1, Label: 2, Charge: 0,
    Energy: 3
    1 * #1 [S -—> S, a]'2
    1 * #5 [a]'2:1 -—> []a:0

**********************************************

    CONFIGURATION: 2
    TIME: 0.529 s.
    MEMORY USED: 61440
    FREE MEMORY: 57887
    TOTAL MEMORY: 912064

    MEMBRANE ID: 1, Label: 2, Charge: 0, Energy: 3
    Multiset: a, S
    Parent membrane ID: 0

    SKIN MEMBRANE ID: 0, Label: 1, Charge: 0, Energy: 0
    Multiset: a
    Internal membranes count: 1

------------------------------------

    STEP: 3

    Rules selected for MEMBRANE ID: 1, Label: 2, Charge: 0,  Ener
    1 * #1 [S -—> S, a]'2
    1 * #5 [a]'2:1 -—> []a:0

    Rules selected for SKIN MEMBRANE ID: 0, Label: 1,
    Charge: 0, Energy: 1
    1 * #8 [a -—> ap]'1

**********************************************

    CONFIGURATION: 3
    TIME: 0.807 s.
    MEMORY USED: 61440
    FREE MEMORY: 57566
    TOTAL MEMORY: 912064

    MEMBRANE ID: 1, Label: 2, Charge: 0, Energy: 4
    Multiset: a, S
    Parent membrane ID: 0

    SKIN MEMBRANE ID: 0, Label: 1, Charge: 0, Energy: 1
    Multiset: a, ap
    Internal membranes count: 1

------------------------------------
--------

    STEP: 4

    Rules selected for MEMBRANE ID: 1, Label: 2, Charge: 0,
    Energy: 5
    1 * #5 [a]'2:1 -—> []a:0
    1 * #1 [S -—> S, a]'2

    Rules selected for SKIN MEMBRANE ID: 0, Label: 1,
    Charge: 0, Energy: 1
    1 * #8 [a -—> ap]'1
    1 * #10 [ap]'1:1 -—> []ap:0

**********************************************

    CONFIGURATION: 4
    TIME: 1.135 s.
    MEMORY USED: 61440
    FREE MEMORY: 57566
    TOTAL MEMORY: 912064
```

MEMBRANE ID: 1, Label: 2, Charge: 0, Energy: 5
Multiset: a, S
Parent membrane ID: 0

SKIN MEMBRANE ID: 0, Label: 1, Charge: 0, Energy: 1
Multiset: a, ap
Internal membranes count: 1

ENVIRONMENT: ap

STEP: 5

Rules selected for MEMBRANE ID: 1, Label: 2, Charge: 0, Energy: 4
1 * #5 [a]'2:1 −−> []a:0
1 * #2 [S −−> Sp]'2

Rules selected for SKIN MEMBRANE ID: 0, Label: 1, Charge: 0, Energy: 1
1 * #8 [a −−> ap]'1
1 * #10 [ap]'1:1 −−> []ap:0

****

CONFIGURATION: 5
TIME: 1.525 s.
MEMORY USED: 61440
FREE MEMORY: 57244
TOTAL MEMORY: 912064

MEMBRANE ID: 1, Label: 2, Charge: 0, Energy: 4
Multiset: Sp
Parent membrane ID: 0

SKIN MEMBRANE ID: 0, Label: 1, Charge: 0, Energy: 1
Multiset: a, ap
Internal membranes count: 1

ENVIRONMENT: ap*2

STEP: 6

Rules selected for MEMBRANE ID: 1, Label: 2, Charge: 0, Energy: 4
1 * #3 [Sp −−> Spp, b]'2

Rules selected for SKIN MEMBRANE ID: 0, Label: 1, Charge: 0, Energy: 1
1 * #10 [ap]'1:1 −−> []ap:0
1 * #8 [a −−> ap]'1

****

CONFIGURATION: 6
TIME: 1.857 s.
MEMORY USED: 61440
FREE MEMORY: 57244
TOTAL MEMORY: 912064

MEMBRANE ID: 1, Label: 2, Charge: 0, Energy: 4
Multiset: Spp, b
Parent membrane ID: 0

SKIN MEMBRANE ID: 0, Label: 1, Charge: 0, Energy: 1
Multiset: ap
Internal membranes count: 1

ENVIRONMENT: ap*3

STEP: 7

Rules selected for MEMBRANE ID: 1, Label: 2, Charge: 0, Energy: 2
1 * #7 [Spp]'2:1 −−> []Spp:0
1 * #6 [b]'2:1 −−> []b:0

Rules selected for SKIN MEMBRANE ID: 0, Label: 1, Charge: 0, Energy: 0
1 * #10 [ap]'1:1 −−> []ap:0

****

CONFIGURATION: 7
TIME: 2.154 s.
MEMORY USED: 61440
FREE MEMORY: 57244
TOTAL MEMORY: 912064

MEMBRANE ID: 1, Label: 2, Charge: 0, Energy: 2
Multiset: #
Parent membrane ID: 0

SKIN MEMBRANE ID: 0, Label: 1, Charge: 0, Energy: 0
Multiset: Spp, b
Internal membranes count: 1

ENVIRONMENT: ap*4

STEP: 8

Rules selected for SKIN MEMBRANE ID: 0, Label: 1, Charge: 0, Energy: 1
1 * #9 [b −−> bp]'1

****

CONFIGURATION: 8
TIME: 2.373 s.
MEMORY USED: 61440
FREE MEMORY: 57244
TOTAL MEMORY: 912064

MEMBRANE ID: 1, Label: 2, Charge: 0, Energy: 2
Multiset: #
Parent membrane ID: 0

SKIN MEMBRANE ID: 0, Label: 1, Charge: 0, Energy: 1
Multiset: Spp, bp
Internal membranes count: 1

ENVIRONMENT: ap∗4

−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
−−−−−−−−−−−−

STEP: 9

Rules selected for SKIN MEMBRANE ID: 0, Label: 1, Charge: 0, Energy: 0
1 ∗ #11 [bp]'1:1 −−> []bp:0

****

CONFIGURATION: 9
TIME: 2.607 s.
MEMORY USED: 61440
FREE MEMORY: 56923
TOTAL MEMORY: 912064

MEMBRANE ID: 1, Label: 2, Charge: 0, Energy: 2
Multiset: #
Parent membrane ID: 0

SKIN MEMBRANE ID: 0, Label: 1, Charge: 0, Energy: 0
Multiset: Spp
Internal membranes count: 1

ENVIRONMENT: ap∗4, bp

Halting configuration
(No rule can be selected to be executed in the next step)

6. CONCLUSION AND FUTURE WORK

This paper tried to add energy in ECP in P-Lingua. After defining the syntax of ECPe, modification in P-Lingua were made. In the future, modification may be needed in the Parser to incorporate both ranges and energy.

7. REFERENCES

[1] Păun, G.: Introduction to membrane computing. Springer (2006)

[2] Juayong, R. A. B., e.a.: On the simulations of evolution-communication p systems with energy without antiport rules for gpus. 10th Brainstorming Week on Membrane Computing Proceeding (2012) 267–289

[3] : The p system website