Computational living systems based on an Abstract chemical system

Yasuhiro Suzuki
Bio-informatics, Medical Research Institute,
Tokyo Medical and Dental University,
1-5-45 Yushima
Bunkyo, Tokyo 113-8501
suzuki@gentzen.mri.tmd.ac.jp

Hiroshi Tanaka
Bio-informatics, Medical Research Institute,
Tokyo Medical and Dental University,
1-5-45 Yushima
Bunkyo, Tokyo 113-8501
tanaka@cim.tmd.ac.jp

Abstract- We propose a new computing model, *Computational living systems*. We use the chemical system in our model as it is the basic system of living things. Since the real biochemical system is so complicated that it is hard to deal with. We use an abstract chemical system by using the multiset rewriting system, Abstract Rewriting System on Multisets (ARMS) that is also a multiset transform system .Considering that a membrane is an important structure for living thing to separate 'self' from its environment as well as many organelle inside are composed of membranes, we introduce membrane structure in ARMS. We further develop Artificial Cell System (ACS) and investigate the behavior of ACS under various environments by introducing genetic method and using genetic programming.

1 Introduction

$$\boxed{\text{Environment}} + \boxed{\text{Computational living systems}} = \underset{\text{Computation}}{}$$

Computational living systems

The aim of this study is to create a biological inspired computing model, *Computational living systems*. In this system, in order to obtain the result, we observe their behaviors and change the condition, without stopping their computations. In other words, we steer them to the right direction by changing the environment and lead them to our settled goal. Although this system may not be suitable to make optimizer, it may be suitable to create a system like living systems. This contribution is a preliminary research for composing the system.

How we will compose it? Since one of the basic system of a living system is chemical system, we also use it as the basic system of the new computing model. However real bio-chemical system is so complex that it is difficult to understand its precise property. Hence, we use an abstract chemical system by using the multiset rewriting system (Abstract Rewriting System on Multisets (ARMS)) that is multiset transform system. It is consists of a multiset of symbols and a set of rewriting rules.

Because of *ARMS* is not a strings rewriting system but a (multi)-*set* rewriting system, it can deal with many degrees of freedom such as a *chemical solution* and the *concentration* of *chemical compounds*.

A membrane is an important structure for living systems. It distinguishes "self" from its environment and hierarchical structures inside the system (like cells, organs and so on) are composed by membranes. Membranes change their structure dynamically and constitute a system. Thus, we introduce membrane structure in ARMS and try to create a computing model "*Artificial Cell System*"(*ACS*) by using membranes.

In the model, a membrane is composed of "chemical compounds (denoted by symbols) " which are generated through chemical reactions in the cell. In each cell there is some chemical compounds and these chemical compounds interact with each other according to the rewriting rule (*reaction rules*).

2 ARMS

Extending the concepts of the abstract rewriting system, we introduce an abstract rewriting system on multi-sets (ARMS). Intuitively, ARMS is like a chemical solution in which floating *molecules* can interact with each other according to reaction rules. Technically, a chemical solution is a finite multi-set of elements denoted $A^k = \{a, b, \ldots, \}$; these elements correspond to *molecules*, and reaction rules are specified in terms of rewrite rules. As to the intuitive meaning of an ARMS, we refer to the study of chemical abstract machines [Berry92]. In fact, rewrite rule systems can be thought of as reflecting an underlying *algorithmic chemistry* [Fontana94].

We denote the empty set by ϕ, and the *base number* of a multi-set (size of a multi-set) by $|S|$ (where S is a multi-set), respectively. Then we define:

Definition 1 (Multi-set) *A "multi-set" is an element $t \in A^k$ $(1 \leq k \leq n) \in \Sigma$, where n is a finite number, and A^k is a Cartesian product $A_1 \ldots A_k$. $A^k = A_1 \times A_2 \ldots \times A_k$, Σ denotes the set of multi-sets, and n is called the "maximal multi-set size."*

The multi-sets correspond to possible states of *chemical*

solution. The set of multi-sets corresponds to the space of transitions of an ARMS.

Definition 2 (Rewriting rule) *A "rewriting rule" is a relation $l \mathcal{R} r$ ($l, r \in \Sigma$), $|l|, |r| \leq$ maximal multi-set size, n. A rewriting rule $l \mathcal{R} r$ is denoted as $l \rightarrow r$.*

Definition 3 (ARMS) *An "Abstract Rewriting System on Multi-sets" (ARMS) is a pair (T, R) consisting of a multi-set T and a set R of rewriting rules.*

Definition 4 (Rewriting on ARMS) *Let (T, R) be an ARMS. We write $s \stackrel{Ru}{\rightarrow} t$ if there exists a rewriting rule $l \rightarrow r \in R$ such that $l \subseteq s$ and $t = (\mathbf{s} - l) \cup r$.*

The ARMS can construct input by such a rule, for example, $\phi \rightarrow a$.

The reader will notice that the method of rewriting of ARMS is different from that of the strings rewriting system. Since ARMS is a multi-set replacing system, the system regards a,b and b,a as multi-sets of symbols, $\{ab\}$ and $\{ba\}$. Thus, they are treated the same. Hence, e.g., strings rewriting systems cannot rewrite a,b using the rule $ba \rightarrow c$, while ARMS, however, can rewrite a,b into c using this rule.

2.1 How ARMS works

In ARMS, we assume that one randomly selected rule is applied in each rewriting step, unless no input is allowed. An algorithm for rewriting steps in ARMS is described in Figure 1.

```
procedure ARMS (Rewriting Step)
begin
  count-step ← 0;
  while count-step ≠ n do
    begin
      Select a rule;
      if the rule can rewrite the multi-set then
        Rewrite the multi-set;
        count-step := count-step + 1;
      end if
    end
  end while
end.
```

Figure 1: An algorithm for ARMS (for the first n steps)

Example In this example, we assume that a will be inputted on each rewriting step, the maximal multi-set size is 4 and the initial state is given by $\{a, a, f, a\}$. The set of the rewriting rules, R_1 is $\{r_1, r_2, r_3, r_4\}$, where each rule is described by the following:

$$aaa \rightarrow b : r_1,\ b \rightarrow a : r_2,\ b \rightarrow c : r_3,\ a \rightarrow bb : r_4.$$

In this example, we assume that rules are selected as following the order $\{r_4 \Rightarrow r_1 \Rightarrow r_3 \Rightarrow r_2\}$. Then, each rule is applied in the following way. First, r_4 is applied. Next, as steps 2 and 3, r_1 and r_3 are applied, respectively. Finally, as step 4, r_2 is applied.

$\{aafa\}$ $\subseteq a$ (the left hand side of r_4)
$\downarrow$ .... can not input a and can not apply r_4,
$\{aafa\}$ $\subseteq a,a,a$ (the left hand side of r_1)
$\downarrow$ .... can not input a but can apply r_1
$\{ba\}$

Figure 2: Example of rewriting steps of ARMS

Figure 2 illustrates two rewriting steps of the calculation from the initial state.

As the first step, since the base number of the multi-set is 4, the system can not input a. On the left hand side of r_4, a is included in $\{aafa\}$, however, r_4 can not be used. If a is replaced with b,b, the base number of the multi-set becomes 5 and it exceeds the maximal multi-set size, 4.

In the next step, the system can not input a, however, r_1 can apply to the multi-set and $\{aafa\}$ is rewritten into $\{ba\}$ (because if a,a,a is replaced with b, the base number of the multi-set does not exceed the maximal multi-set size, see Figure 2).

In step 3, ARMS inputs a to the multi-set and transforms it to $\{c, a, a\}$ with r_3.

$$\text{Step } 3 : \{c, a, a\}.$$

In step 4, the system inputs a, but r_2 can not apply to it. Thus $\{c, a, a\}$ becomes $\{c, a, a, a\}$.

$$\text{Step } 4 : \{c, a, a, a\}.$$

Examples of state transition of an ARMS In this paragraph, we shall present two examples. Let us assume a set of rewriting rule R_1 and a maximal multi-set size of 4. The first example is a case where ARMS generates two cycles. This example has the following rule order:

$$\{r_4 \Rightarrow r_3 \Rightarrow r_2 \Rightarrow r_4 \Rightarrow r_1 \Rightarrow r_2 \Rightarrow r_1 \Rightarrow r_3 \Rightarrow r_4\},$$

whose state transition is shown in Figure 3. After 8 steps, the system forms two cycles, whose periods are of 3 steps.

The next example is a case where ARMS terminates. Although the ARMS applies the same rules, the obtained result is completely different (Figure 4). This example has the following rule order:

$$\{r_4 \Rightarrow r_1 \Rightarrow r_2 \Rightarrow r_4 \Rightarrow r_3 \Rightarrow r_1 \Rightarrow r_2\}.$$

The state transition is shown in Figure 4:

3 Artificial Cell System (ACS)

In this section, we introduce the basic structural ingredients of ARMS, membrane structures and how ACS

0.	$\{f\}$	
1.	$\{a, f\}$	
2.	$\{a, a, f\}$	
3.	$\{a, a, a, f\}$	
4.	$\{b, f\}$	$\uparrow$
5.	$\{a, b, f\}$	a cycle
8.	$\{a, a, a, f\}$	$\downarrow$
9.	$\{b, f\}$	$\uparrow$
10.	$\{a, b, f\}$	a cycle
11.	$\{a, a, a, f\}$	$\downarrow$
12.	$\{b, f\}$	

Figure 3: Example of a system that generates cycles

1.	$\{a, f\}$
2.	$\{a, a, f\}$
3.	$\{a, a, a, f\}$
4.	$\{b, f\}$
5.	$\{a, b, f\}$
6.	$\{a, a, b, f\}$
7.	$\{a, a, c, f\}$.

Figure 4: Example of a system that halts

works. A membrane is composed of a "membrane compound (MC)" which is in fact a symbol. To maintain a membrane, it needs to have a certain minimal volume. A membrane disappears if the volume of membrane compounds decreases below the needed volume to maintain the membrane.

3.1 ACS and ACSE

Currently, we have developing two types of ACS;

(1) ACS and

(2) ACS with an Elementary membrane (ACSE).

ACSE is different only in the way of dissolving and dividing from ACS.

3.2 Descriptions of ACS

A transition ACS is a construct

$$\Gamma = (A, \mu, M_1, ..., M_n, R, MC, \delta, \sigma),$$

where:

(1) A is a set of objects;

(2) μ is a membrane structure (it can be changed throughout a computation);

(3) $M_1, ..., M_n$, are multisets associated with the regions 1,2, ... n of μ;

(4) R is a finite set of multiset evolution rules over A.

(5) MC is a set of membrane compounds;

(6) δ is the threshold value of dissolving a membrane;

(7) σ is the threshold value of dividing a membrane;

μ is a membrane structure of degree n, n $\geq$ 1, with the membranes labeled in a one-to-one manner, for instance, with the numbers from 1 to n. In this way, also the regions of μ are identified by the numbers from 1 to n.

Rewriting rules are applied in following manner:

(1) The same rules are applied to every membrane. There are no rules specific to a membrane.

(2) All the rules are applied in parallel. In every step, all the rules are applied to all objects in every membrane that can be applied. If there are more than two rules that can apply to an object then one rule is selected randomly.

(3) If a membrane dissolves, then all the objects in its region are left free in the region immediately above it.

(4) All objects and membranes not specified in a rule and which do not evolve are passed unchanged to the next step.

Rewriting rule R is a finite set of multiset rewriting rules over A. Both the left and the right side of a rule are obtained by sampling with replacement of symbols. A set of reaction rules is constructed as the overall permutation of both sides of the rules.

Membrane structure Each matching pair of parentheses [,] appearing in a membrane structure is called a *membrane*. The number of membranes in a membrane structure μ is called the *degree* of μ and is denoted by deg(μ). The external membrane of a membrane structure μ, M_0 is called the *skin* membrane of μ. When a membrane which has the form [ ] and no other membrane appears inside the two parentheses, then it is called an *elementary* membrane.

Input and Output Chemical compounds are supported from outside of the system to M_0 and some compounds are exhausted from M_0. All chemical compounds are transformed among cells, a randomly selected chemical compound is transformed into the membrane just above or below it. Although a membrane does not allow specificity of transport across the membrane, a cell can control its chemical environment by chemical reaction.

Evolution of Cells When a cell grows and the cell exceeds the threshold value for dividing, it divides into parts of random sizes. This can be seen as a kind of

mutation. If a divided cell does not have any membrane compounds, it must disappear soon.

Furthermore, to maintain the membrane through chemical reactions inside the cell can be seen as *natural selection*. If cells can not maintain the membrane compounds, it must disappear soon.

Thus, both dividing membranes and dissolving membranes produce evolutionary dynamics. These correspondences are summarized into;

Natural Selection	Dissolving a membrane,
Mutation	Dividing a cell into parts of random size.

3.2.1 Dissolving and dividing a membrane of ACS

A membrane is composed of a "membrane compound" which is in fact a symbol. To maintain a membrane, it needs to have a certain minimal volume. A membrane disappears if the volume of membrane compounds decreases below the needed volume to maintain the membrane. Dissolving the membrane is defined as follows:

$$[_h a, ...[_i b, ...]_i]_h \rightarrow [_h a, b, ...]_h,$$

where the ellipsis {...} illustrates chemical compounds inside the membrane. Dissolving takes place when

$$\frac{|w_i|_{MC}}{|M_i|} < \delta$$

where δ is a threshold value for dissolving the membrane. All chemical compounds in its region are then set free and they are merged into the region immediately above it.

On the other hand, when the volume of membrane compounds increases to a certain extent, then a membrane is divided. Dividing a membrane is realized by dividing it in multisets random sizes. The frequency at which a membrane is divided is decided in proportion to its size. As the size of a multiset becomes larger, the cell is divided more frequently. Technically, this is defined as follows;

$$[_h a, b, ...]_h \rightarrow [_h a, ...[_i b, ...]_i]_h$$

Dividing takes place when

$$\frac{|w_h|_{MC}}{|M_h|} > \sigma$$

where σ is a threshold for dividing the membrane. All chemical compounds in its region are then set free and they are separated randomly by new membranes.

3.3 Description of ACSE

ACSE is different only in the way of dividing and dissolving cells from ACS. Dissolving the membrane is defined as follows:

$$[_h a, b, ...]_h \rightarrow [_0 a, b, ...,]_0$$

Dissolving takes place when

$$\frac{|w_h|_{MC}}{|M_h} < \delta$$

where δ is a threshold value for dissolving the membrane. All chemical compounds in its region are then set free and they are merged into the region of M_0.

Dividing is defined as follows;

$$[_h a, b, ...]_h \rightarrow [_h a, ...]_h [_i b, ...]_i.$$

Dividing takes place when

$$\frac{|w_h|_{MC}}{|M_h|} > \sigma,$$

where σ is a threshold for dividing the membrane. All chemical compounds in its region are then set free and they are separated randomly in the old and new membranes. Hence, in ACSE, a structured cell such as $[a, b[c, [d, e]]]$ does not appear.

4 Simulations

We have investigated behaviors of these two models through simulations in order to compose the computational living system. We will report some of results.

4.1 ACSE

In this subsection, we will address the case when following *ACSE* was simulated;

$\Gamma = (A = \{a, b, c\}, \mu = \{ [,]_0, ...[,]_{100}\} M_0 = \{[a^{10}, b^{10}, c^{10}]^{100}\}, R, MC = \{b\}, \delta = 0.4, \sigma = 0.2)$,

where:

(1) R, the length of the left- or right-hand-side of a rule is between one and three. Both sides of the rules are obtained by sampling with replacement of the three symbols a, b and c;

(2) Membrane structures are assumed to be $(\mu = \{[_1]_1 ... [_{100}]_{100}\})$.

Through the simulation we discovered that the strength of a membrane affects the behavior of cells. The strength of a membrane is defined as the frequency of decreasing membrane compounds.

When a membrane is strong When a membrane is strong, the most stable cell consists of only one membrane, cells of this type become " mother" cells and they produce "daughter" cells.

In order to display a state of a cell we transform the state of a cell to a number by using the transformation function; $f(M(a), M(b), M(c)) = 10^2 \times M(a) + 10^1 \times$

$M(b) + 10^0 \times M(c)$. For example, the state $\{a, a, b, c, c\}$ is transformed into $10^2 \times 2 + 10^1 \times 1 + 10^0 \times 2 = 212$.

The figure 5 illustrates the evolution of cells when a membrane is strong. The cells that are close to the horizontal axis are mother cells. Some daughter cells depart from the group and evolve different types of cells, even though almost all cells are in the group. In this case, dissolving a membrane compound takes place per 100 steps.

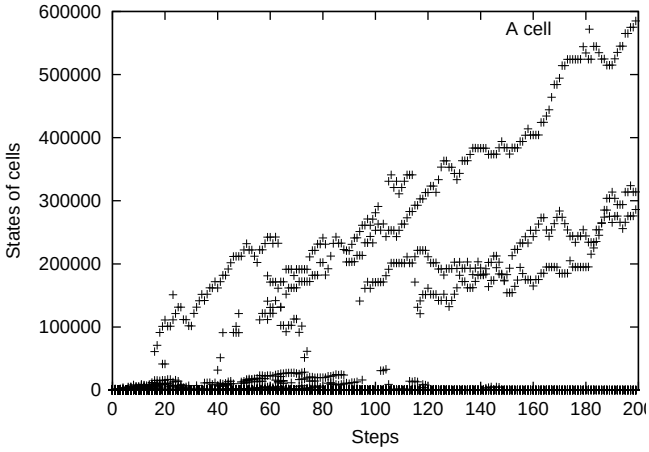

Figure 5: When a membrane is strong. The lines illustrate the regions where cells exist and points correspond to the state of cells.

The figure 6 is focused to the mother cells. At first there are about ten groups, and some of them become extinct: after 200 steps there remain about four groups.

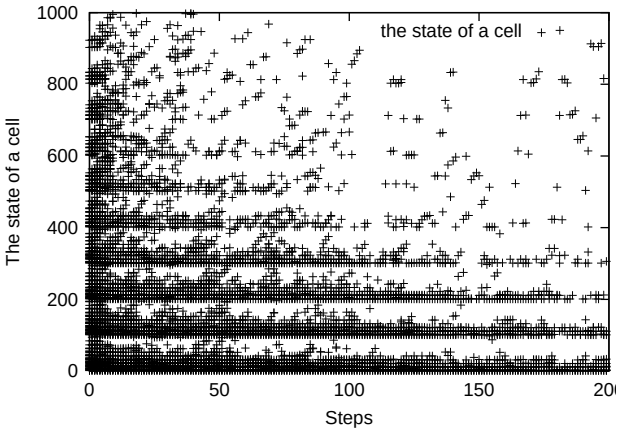

Figure 6: Evolution of mother cells. The points correspond to cells.

When a membrane is weak Figure 7 illustrates the case when a membrane is weak, a membrane dissolves every 3 steps. In this case, the system can not form a group of mother cells such as in the previous case. Since the group of cells drifts to more stable cells, the cells grow larger. Even if a large cell divides into parts of random sizes, the probability of including enough membrane compounds to maintain its membrane are larger than a small cell.

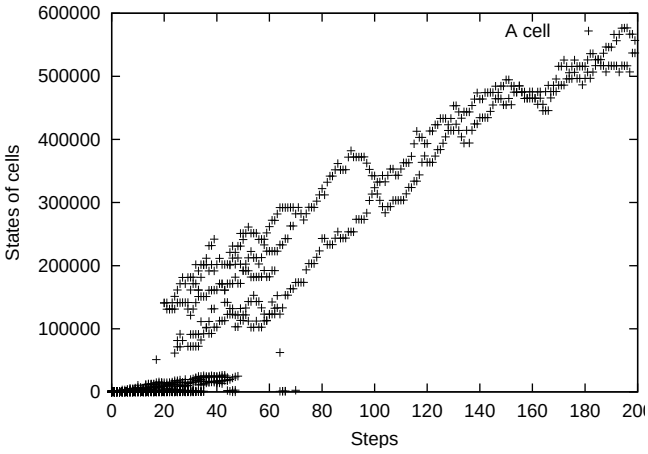

Figure 7: When a membrane is weak

We believe that this behaviors of evolution is similar to the evolution of viruses [Tanaka99]. The settings of this simulation are so rough, however, that the possibility remains open that chemical evolution in origin of life is similar to virus evolution. This will be addressed in future research.

4.2 Genetic ACS (GACS)

Since a rewriting rule promotes a reaction, it can be regarded as an enzyme. Here we extend ACS with evolutionary mechanism We called the system Genetic ACS (GACS).

4.3 Descriptions of *GACS*

A transition in *GACS* is a construct

$$\Gamma = (A, \mu, M_1, ..., M_n, R, \delta, \sigma),$$

where:

(1) A is a set of objects;

(2) μ is a membrane structure (it can be changed throughout a computation);

(3) $M_1, ..., M_n$, are multisets associated with the regions 1,2, ... n of μ;

(4) R is a finite set of multiset evolution rules over A.

(5) δ is the threshold of dissolving membrane;

(6) σ is the threshold of dividing membrane.

The way of applying rewriting rules, the way of dissolving and dividing and input and output are the same as ACS and ACSE.

An enzyme We denote a set of reaction rules as follows;

	a	b	c
a	x_{aa}	x_{ab}	x_{ac}
b	x_{ba}	x_{bb}	x_{bc}
c	x_{ca}	x_{cb}	x_{cc},

where x_{ij} means the number of compounds i which are transformed from j. For example, 2_{ab} means a rewriting rule, $b \longrightarrow a, a$. We call the table a transformation table.

Transmission of an enzyme When a membrane is divided, the enzyme which is inside the membrane is copied and passed down to a new divided cell. At that time, a point mutation occurs only in the copied enzyme and it is passed down to the new cell. The enzyme remain in the old membrane as well as the new one. Point mutations occur every time a membrane divides. When a membrane is dissolved the enzyme which is inside the membrane loses its activity. A point mutation is a rewriting of the number of x_{ij}. Thus, it changes the number of transforming compounds to i. We assumed $x_{ij} \in \{0, 1, 2\}$. In ACS and ACSE the system have only one rewriting rule, however, in GACS, each membrane has a set of rules.

4.4 An experimental result of a GACS

We will show experimental results of GACS. At first, the following $GACS$ was simulated;

$\Gamma = (A = \{a, b, c\}, \mu = \{/\ 0\} M_0 = \{a^{10}, b^{10}, c^{10}\}, R, MC = \{c\}, \delta = 0.4, \sigma = 0.2)$,

where the transformation table (R) is set as follows in the initial states;

	a	b	c
a	0_{aa}	0_{ab}	1_{ac}
b	1_{ba}	0_{bb}	0_{bc}
c	0_{ca}	1_{cb}	0_{cc}.

The productivity of membrane compounds p is defined as the ratio of the total number of non-membrane compounds to be produced to the total number of membrane compounds to be produced;

$$p = \frac{\Sigma_{j=a}^{j=b} x_{cj}}{\Sigma_{j=a}^{i=b} \Sigma_{j=a}^{j=c} x_{i,j}},$$

When $p = 0$ the enzyme does not produce any membrane compounds, when $p = 1$, it produces the same number of membrane compounds to the non- membrane compounds, and when $p > 1$, it produces more membrane compounds than non-membrane compounds. Figure 8 illustrates the time series of productivity, where the vertical axis illustrates the productivity, the horizontal axis illustrates the steps and each dot is an enzyme. It shows that at first almost enzymes evolve to $p > 1$. However, after 100 steps, the productivity of the enzymes decrease.

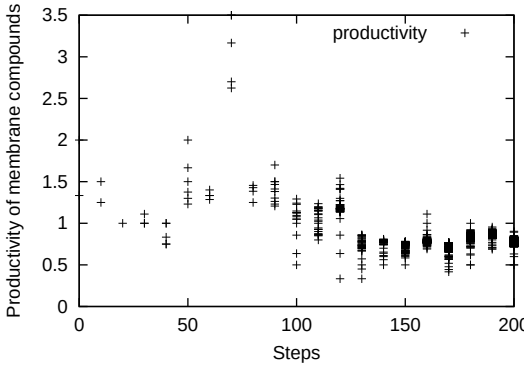

Figure 8: Productivity of enzymes.

After 100 steps, both the number of cells and the sizes of cells increase exponentially. Furthermore, the structure of cells becomes complicated. Figure 9 illustrates the correlation between the number of cells, the size of cells and the number of steps, where each dot corresponds to a cell.

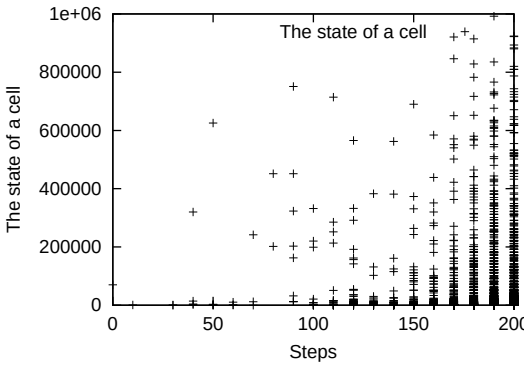

Figure 9: State transition of the system

Figure 10 illustrates the internal nodes of the whole system. If we regard M_0 as the root and other cells as internal nodes and leaves, we can regard the whole system as a tree. In order to indicate the complexity of the tree, we use the number of internal nodes in the tree. Figure 10 illustrates that the number of internal nodes increases exponentially, after 150 steps.

It is interesting that when a cell grows into a hierarchical cell, the enzyme evolves to a low productivity one.

The reason of this behavior can be considered as follows; the enzyme whose productivity is high always suffers from mutations, because it promotes membrane division thus it generates more mutations than low produc-

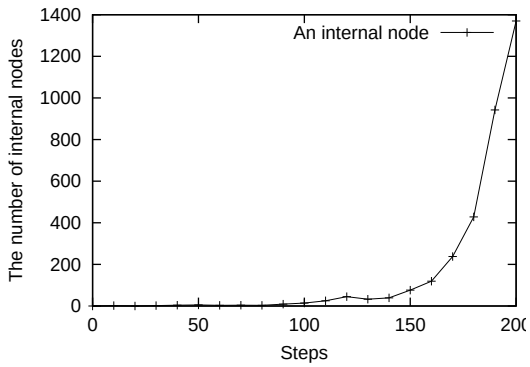

Figure 10: Internal nodes of the cell

tivity ones. If every cell is an elementary cell, an enzyme have to keep producing membrane compounds at a high rate. However, when the cell forms structure, it is not necessary one with high productivity, because, in a structured cell, if an inside cell dissolves, the cell that includes the dissolved cell obtains its membrane compounds.

Therefore, during evolution of a cell into a structured cell, the cell needs a high productivity enzyme. However, once it evolves a structured cell, high productivity enzymes are weeded out. This is the role of membranes in terms of computation.

5 Genetic Programming by using GACS

By using GACS, we attempted to generate a program by using a GACS. In the GACS, a program corresponds to an enzyme, thus we breed an enzyme which can solve a particular problem. We apply it to a simple problem *doubling*; calculate the double value of the number of a and b then show the result as the number of c ($c = 2(a+b)$).

Description of GACS

A *GACS* is defined as follows;

A transition *GACS* is a construct

$\Gamma = (A = \{a, b, c\}, \mu = \{[_1]_1 ... [_{100}]_{100}\}, M_0 = \{[a^2, b^2, c^0]^{100}\}, R)$. In the initial state, all transformation tables R are

	a	b	c
a	0_{aa}	0_{ab}	1_{ac}
b	1_{ba}	0_{bb}	0_{bc}
c	0_{ca}	1_{cb}	0_{cc},

and one hundred of elementary cells are assumed inside M_0. No compounds are transformed among cells and no input and output are assumed. Although we performed simulation by using different type of cells in the initial state, the experimental results are same, thus we will address only when each cell is $[a^2, b^2, c^0]$ in the initial state.

Dissolving and dividing a membrane The way of dissolving and dividing are the same as in ACSE. After n rewriting steps, if the number of c is smaller than 7, the membrane is dissolved and the enzyme inside it loses its activity, for example,

$$[_h a^3, b^5, c^8]_h [_i a^4, b^5, c^1]_i \rightarrow [_h a^3, b^5, c^8]_h.$$

In the above example, the number of c inside the membrane i is smaller than 7, so the membrane is dissolved.

After n rewriting steps, if the number of c is larger than 9, the membrane is divided and a point mutation takes place in its enzyme. Furthermore, a new enzyme is passed down to a new cell. When a cell divided, the inside multiset of the divided cell and its parent cell are set to $\{a^2, b^2, c^0\}$ again, so they try to solve the problem again. The results in:

$$[_h a^3, b^5, c^1 0]_h \rightarrow [_h a^2, b^2, c^0]_h [_i a^2, b^2, c^0]_i$$

In the example, because the number of c inside the membrane h is larger than 9, the membrane h is divided and a new membrane i emerges. Then compounds which are inside both membranes i and h are set to $\{a^2, b^2, c^0\}$. In this GACS, cells continue to solve the problem.

The fitness of the GACS The fitness of an enzyme is defined as the number of steps to reach the solution. By using this fitness, good enzymes are there that can solve the problem within a smaller number of steps than the others are selected.

5.0.1 Experimental result

At first, all enzymes are set to,

$$\begin{aligned} a &\rightarrow b, \\ b &\rightarrow c, \\ c &\rightarrow a. \end{aligned}$$

After 5000 rewriting steps, the enzymes that reach the solution within 8 steps are selected;

$$\begin{array}{lll} a \rightarrow a, c, & a \rightarrow a, b, c, c, \\ b \rightarrow a, b, c, c, & b \rightarrow a, b, c, c, \\ c \rightarrow c, c; & c \rightarrow c, c; \end{array}$$

$$\begin{aligned} a &\rightarrow a, c, c, \\ b &\rightarrow b, \\ c &\rightarrow c, c. \end{aligned}$$

For each rule, one hundred of elementary cells that are $[a^2, b^2, c^0]$ are set again and calculations performed again. Next, the enzymes that can solve the problem within 5 steps are selected. Then, there remains only one enzyme;

$$\begin{aligned} a &\rightarrow a, c, c, \\ b &\rightarrow c, c, \\ c &\rightarrow c, c. \end{aligned}$$

it is a solution of this simulation. In fact a solution of this problem is

$$
\begin{aligned}
a &\rightarrow c, c, \\
b &\rightarrow c, c, \\
c &\rightarrow c;
\end{aligned}
$$

Thus the survived enzyme evolved to a similar enzyme in the solution.

This model is a candidate of the computational living system. In the future we plan to create a GACS as an artificial living system of computation that can solve more complicated problems. In such a system, in order to obtain the result, we observe their output (behaviors) and change the condition, do not stop their computations. In other words, we steer them in the right direction by using the selection and mutation, and lead them to our settled goal.

6 Discussion

P systems

The above ACSes correspond to a class of P systems, a parallel molecular computing model proposed by G. Păun[Paun00] that is based on the processing of multisets of objects in cell-like membrane structures. A P system is a multiset transformation system. However, it is different from [Berry92], since it includes a "*membrane*" in its computing mechanism. In P systems *cells* are structured like living cells. The system itself is composed of several cells that are delimited from the neighboring cells.

The computing power of P systems is equal to that of a Turing machine, as proved in [Paun00], and an algorithm to compute the SAT problem in linear time [Paun99] was proposed. Various P systems have also been proposed and their mathematical properties have been investigated. On the computing power of ACSes that are treated in this paper is open problems.

7 Conclusion

It is obvious that living systems never do calculation only by using *pen* and *paper*. Thus, if we could abstract *computation* in terms of living systems then we might obtain a new computational world.

We have not found the new world yet, however, we have already obtained some *hints* throughout the observations on living systems, such as "parallelism" and "membranes." In a living system, every cell and every organ is acting in parallel and they are consisted of membranes.

To implement our system on a parallel computer and investigating its mathematical properties are our future work. We expect that these studies lead us to a new computing paradigm which goes beyond the Turing Machine based computing paradigm.

Acknowledgment

We thank for fruitful discussions with Dr. Steen Rasmussen, Prof. Dr. W. Banzaf gave us important comments and encouraged us, long discussions with P. Dittrich were useful. F. Pepper and SOMA research group gave us important suggestions. And the authors would like to express many thanks to Dr. Gheorghe Păun for his useful comments, discussions and mathematical refinements. This research is supported by Grants-in Aid for Scientific Research No.11837005 from the Ministry of Education, Science and Culture in Japan.

Bibliography

[Fontana94] Fontana, W. and L.W. Buss, (1994) The arrival of the fittest: Toward a theory of biological organization. *Bulletin of Mathematical Biology* 56: 1–64.

[Berry92] Berry, G. and G. Boudol, (1992) The chemical abstract machine. *Theoretical Computer Science* 96:, 217–248.

[Ganti75] Ganti, T., (1975) Organization of chemical reactions into dividing and metabolizing units: the chemotons. *Biosystems* 7: 189–195.

[Prigogine89] Nicolis, G. and I. Prigogine. (1989) *Exploring Complexity, An Introduction*. San Francisco: Freeman and Company.

[Paun00] Păun, G., (2000) Computing with Membranes, *J. of Computer and System Sciences*, (in press) (submitted, also on http://www.tucs.fi)

[Paun99] Păun, G., (1999) P Systems with Active Membranes: Attacking NP Complete Problems, *J. of Automata, Languages and Combinatorics*, (in press).

[Suzuki98] (1998) Suzuki, Y. and H. Tanaka. Order parameter for a Symbolic Chemical System, *Artificial Life VI*:130-139, MIT press.

[Suzuki96] (1996) Suzuki, Y. S., Tsumoto and H. Tanaka. Analysis of Cycles in Symbolic Chemical System based on Abstract Rewriting System on Multisets, *Artificial Life V*: 522-528. MIT press.

[Tanaka99] (1999) Tanaka, H., F. Ren, S. Ogishima, Evolutionary Analysis of Virus Based on Inhomogeneous Markov Model, ISMB'99, p 148, 1999.