A Note on P Systems with Activators

Artiom ALHAZOV

Research Group on Mathematical Linguistics
Rovira i Virgili University
Pl. Imperial Tárraco 1, 43005 Tarragona, Spain
E-mail: `artiome.alhazov@estudiants.urv.es`

Institute of Mathematics and Computer Science
Academy of Sciences of Moldova
Str. Academiei 5, Chişinău, MD 2028, Moldova
E-mail: `artiom@math.md`

Abstract. The usual assumption in P systems behavior is that of *maximal parallelism*, however in living cells it is not the case because they have a limited number of *enzymes*. The aim of this paper is to try to merge these ideas by introducing a notion of *activator* - a formal model of enzyme as a usual symbol-object, more or less a middle notion between a catalyst and a promoter. Each activator executes one (context-free) rule, and can evolve in the same step. The rules will need activators to be applied, so the parallelism of each rule is maximal, but limited to the number of its activators.

Such systems can generate any recursively enumerable language or deterministically accept any recursively enumerable set of vectors of nonnegative integers. It is open what is the power of P systems with uniport rules and activators.

1 Introduction: Meet the Activator

The idea behind this paper came as an attempt to have a universal theory of objects and rules: let us associate one (non-cooperative) rule to every object, called the *activator* of that rule, think of *enzyme* (a biological protein, acting as a catalyst) as its prototype. For a reaction to occur, rewriting one copy of an object into 0 or more objects, one copy of the activator must be present in the region. The reaction is then performed nondeterministically, in the maximally parallel manner.

Example: If $R_1 = \{(a : b \to aa), (b : a \to \lambda)\}$, then $[_1 a^3 b^5]_1 \Rightarrow [_1 a^6 b^2]_1 \Rightarrow [_1 a^8]_1$.

The important properties are the following:

1. Each object activates one rule (only one context-free rule can be associated to each object).
2. Each rule needs an activator to be performed.
3. The reactant (the object in the left-hand side of the rule) is, clearly, also needed for the rule to be performed.

4. The presence of each activator allows one object to be rewritten by the rule it activates.

5. The activator can be itself rewritten (by some activated rule) while itself activating some rule (or while not activating any, if there are no reactants available for it).

6. The parallelism of the activated rules is maximal.

The difference of the new notion from the existing ones is that the catalysts cannot evolve, that the promoters promote rewriting of many rules, not just one, and, finally, that a catalyst/promoter can enforce rules from a fixed set, while for the activators this set is restricted to singletons.

The biological counterpart of the inspiration for the activators model comes from the following observations:

- Each enzyme only performs a certain reaction (or a group of similar actions).
- The speed of the reactions depends on the number of enzymes present.
- An enzyme can react while performing a reaction.
- An enzyme can act on a protein molecule, producing two protein molecules.
- An enzyme is itself a protein molecule.

2 Definition

We assume the reader to be familiar with the basic notions of membrane computing. We now give a more formal description of the systems with activators.

Such a system is written as

$$\Pi = (O, \mu, w_1, \cdots, w_m, R_1, \cdots, R_m, i_0),$$

where O is a finite set of objects, μ is the membrane structure with m membranes, injectively labeled as $1, \cdots, m$ and $(w_i)_{1 \le i \le m}$ are the initial multisets. i_0 is the output region (0 if it is the environment). Finally, for every region i, R_i is a set of rules of the form $(a : b \to x)$, where x is a string of objects with target indications from $T = \{here, out\} \cup \{in_j \mid 1 \le i \le m\}$ (a is called the activator of such a rule), such that $Card\{(f : b \to x) \in R_i \mid b \in O, x \in (O \times T)^*\} = 1$ for every fixed $f \in O$. (The last restriction comes from the idea that the enzymes usually can only activate a single reaction.)

The transition step (in each region i with a multiset w) is defined as follows: the multiset $A = \{r^{|w|_a} \mid r = (a : b \to x) \in R\}$ of activated rules is applied in the maximally parallel manner to w. For every $b \in O$, $\min\{|w|_b, \sum\limits_{r \text{ rewrites } b} A(r)\}$ occurrences of b are rewritten. From now on we only consider systems with one region and with external output (hence the result of a computation is a string).

Notice that there is a one-to-one correspondence between the rules and objects, so

- the rules can be defined as (activating) functions of objects, or
- the objects can be defined as the activators of rules.

3 Universality

Theorem 3.1 *P systems with activators generate the family of recursively enumerable languages.*

Proof. Given a language $L \in RE$ over alphabet T, consider a counter machine G with set C of counters, the start instruction labeled by e_{init}, and the instruction set P, recognizing L. We construct the following P system

$$\begin{array}{rcl} \Pi & = & (O, [_1]_1, w_1, R_1, 0), \\ O & = & \{a, a' \mid a \in T\} \cup \{c, c_1, c' \mid c \in C\} \cup \{\#, q, q_1, q', r, r'\} \\ & \cup & \{e, e_1, e_2, e', e'' \mid e \in Lab(P)\}, \\ w_1 & = & e_{init}r'q'vyz, \text{ where } v \text{ consists of symbols } a' \text{ for all } a \in T, \\ & & y \text{ consists of symbols } c' \text{ for all } c \in C \text{ and} \\ & & z \text{ consists of symbols } e'e'' \text{ for all } e \in Lab(P). \end{array}$$

For $e, f, g \in Lab(P)$, $x \in T \cup \{\lambda\}$, $c \in C$, instructions $(e : read(x), add(c),\ f, g)$ are simulated in one step by rules

- $e' : e \to cxfrqq_1$,
- $e'' : e \to cxgrqq_1$.

Object e is replaced by e or f, x is produced if $x \neq \lambda$, and the value of counter c is incremented. The “garbage” objects r, q and q_1 are added for the uniformity, and will be erased immediately.

Instructions $(e : read(x), sub(c), f, g)$, are simulated in three steps by the rules below

- $e' : e \to e_1qx$,
  $e : c \to c_1$,
- $c' : c_1 \to \lambda$,
  $c_1 : q \to r$,
  $e'' : e_1 \to e_2$,
- $r : e_2 \to fqq_1$,
  $q : e_2 \to grq_1$.

Object x is produced if $x \neq \lambda$, e evolves into e_1q, in the same time checking if c is present in the region (in that case one copy of c evolves into c_1). Then, if c_1 was produced, then it is erased, in the same time changing q to r. In the meantime, e_1 changes to e_2. Next, if c was present in the beginning of the simulation of the instruction of the counter machine, then r changes e_2 to f, otherwise q changes e_2 to g.

We also need the following rules:

- $a' : a \to a_{out}$, $a \in T$,
- $r' : r \to \lambda$,
  $q' : q_1 \to \lambda$,
  $q_1 : q \to \lambda$,

- $X : \# \to \#,\ X \in \{\#, e, e_1, e_2, c, r\}$

to put the terminal symbols in the environment and to cleanup the extra symbols after the simulation of some instruction is finished. The last group, the dummy rules, is added to fulfill the condition that each object is an activator of some rule. $\square$

Remark: By a similar technique, simulating deterministic register machines, one can conclude that P systems with activators can deterministically recognize (by halting) any recursively enumerable set of vectors of nonnegative integers (the vectors are represented as multisets of objects, introduced in the membrane before the computation begins).

4 Conclusions and Open Problems

While trying to create a uniform system of objects and rules, we have introduced a new form of the cooperation between the symbol-objects (reactant – activator), inspired by the biological analog (chemical – enzyme).

These systems can generate RE, and only one membrane is enough. The following *open question* is of a special interest: what biologically inspired restrictions can be further placed on the general form of the rule $a : b \to x$? For instance, what is the power of such system if only *uniport* rules (of type $a : b \to b_{in_j}$, $a : b \to b_{out}$, $a : b_{come}$) are allowed?

Acknowledgements. The work is partially supported by IST-2001-32008 project "MolCoNet", as well as the Moldovan Research and Development Association (MRDA) and the U.S. Civilian Research and Development Foundation (CRDF), Award No. MM2-3034. The author would also like to thank Matteo Cavaliere and Ioan Ardelean for fruitful discussions.

References

[1] I.I. Ardelean, M. Cavaliere, Modelling biological processes by using a probabilistic P system software, *Natural Computing*, 2, 2 (2003), 173–197.

[2] Gh. Păun, *Membrane Computing. An Introduction*, Springer-Verlag, Berlin, 2002.