A Small Universal Splicing P System

Artiom Alhazov[1,2], Yurii Rogozhin[1], Sergey Verlan[3,1]

[1] Institute of Mathematics and Computer Science
Academy of Sciences of Moldova
Academiei 5, Chişinău MD-2028 Moldova
E-mail: {artiom,rogozhin}@math.md
[2] IEC, Department of Information Engineering
Graduate School of Engineering, Hiroshima University
Higashi-Hiroshima 739-8527 Japan
[3] LACL, Département Informatique, Université Paris Est
61, av. gén de Gaulle, 94010, Créteil, France
E-mail: verlan@univ-paris12.fr

Abstract. In this article we present a universal splicing P system with 6 rules. Thus we improve the previous result that used 8 rules and lower the possible value for the boundary between the universality and non-universality for such systems.

1 Introduction

Head splicing systems (H systems) were one of the first theoretical model of bio-molecular computing (DNA-computing) and they were introduced by T.Head [5]. The molecules from biology are replaced by words over a finite alphabet and the chemical reactions are replaced by a *splicing* operation. An H system specifies a set of rules used to perform a splicing and a set of initial words or axioms. The computation is done by applying iteratively the rules to the set of words until no more new words can be generated. This corresponds to a bio-chemical experiment where one has enzymes (splicing rules) and initial molecules (axioms) which are put together in a tube and one waits until the reaction stops.

H systems are not very powerful, so, a lot of other models introducing additional control elements were proposed (see [9] for an overview).

Another extension of H systems was done using the framework of P systems [8], see also [4] and [10]. In a formal way, splicing P systems can be considered like a graph, whose nodes contain sets of strings and sets of splicing rules. Every rule permits to perform a splicing and to send the result to some other node.

Since splicing P systems generate any recursively enumerable languages, it is clear that there are universal splicing P systems. Like for small universal Turing machines, we are interested by such universal systems that have a small number of rules. A first result was obtained in [11] where a universal splicing P system with 8 rules was shown. Similar investigations for P systems with symbol-objects were done in [3, 1] and the latter article constructs a universal antiport P system with 23 rules.

In this article we provide a new construction for splicing P systems and prove the remarkable fact that 6 splicing rules are powerful enough for the universality.

2 Definitions

We do not present here definitions concerning the concepts of the theory of formal languages. We refer to [6] and [12] for more details. We only remark that we denote the empty word by ε.

A *tag system* of degree $m > 0$, see [2] and [7], is the triplet $T = (m, V, P)$, where $V = \{a_1, \ldots, a_{n+1}\}$ is an alphabet and where P is a set of productions of form $a_i \to P_i, 1 \leq i \leq n$, $P_i \in V^*$. We remark that for every a_i, $1 \leq i \leq n$, there is exactly one production in P. The symbol a_{n+1} is called the halting symbol. A configuration of the system T is a word w. One passes from a configuration $w = a_{i_1} \ldots a_{i_m} w'$ to the next configuration z by erasing the first m symbols of w and by adding P_{i_1} to the end of the word: $w \Rightarrow z$, if $z = w' P_{i_1}$.

The computation of T over the word $x \in V^*$ is a sequence of configurations $x \Rightarrow \ldots \Rightarrow y$, where either $y = a_{n+1} a_{i_1} \ldots a_{i_{m-1}} y'$, or $y' = y$ and $|y'| < m$. In this case we say that T halts on x and that y is the result of the computation of T over x. We say that T recognizes the language L if there exists a coding φ such that for all $x \in L$, T halts on $\varphi(x)$, and T halts only on words from $\varphi(L)$.

We note that tag systems of degree 2 are able to recognize the family of recursively enumerable languages. Moreover, systems constructed in [2] and [7] have non-empty productions and halt only by reaching the symbol a_{n+1} in the first position.

2.1 Splicing operations

A *splicing rule* (over an alphabet V) is a 4-tuple (u_1, u_2, u_3, u_4) where u_1, u_2, u_3, $u_4 \in V^*$. It is frequently written as $u_1 \# u_2 \$ u_3 \# u_4$, $\{\$, \#\} \notin V$ or in two dimensions: $\begin{array}{c|c} u_1 & u_2 \\ \hline u_3 & u_4 \end{array}$. Strings $u_1 u_2$ and $u_3 u_4$ are called splicing *sites*.

We say that a word x *matches* rule r if x contains an occurrence of one of the two sites of r. We also say that x and y are *complementary* with respect to a rule r if x contains one site of r and y contains the other one. In this case we also say that x or y may *enter* rule r. When x and y can enter a rule $r = u_1 \# u_2 \$ u_3 \# u_4$, *i.e.*, one has $x = x_1 u_1 u_2 x_2$ and $y = y_1 u_3 u_4 y_2$, it is possible to define the application of r to the couple x, y. The result of this application is w and z where $w = x_1 u_1 u_4 y_2$ and $z = y_1 u_3 u_2 x_2$. We also say that x and y are spliced and w and z are the result of this splicing. We write this as follows: $(x,\ y) \vdash_r (w,\ z)$.

The pair $\sigma = (V, R)$ where V is an alphabet and R is a set of splicing rules is called a *splicing scheme* or an H-scheme.

For a splicing scheme $\sigma = (V, R)$ and for a language $L \subseteq V^*$ we define:

$\sigma(L) \stackrel{\text{def}}{=} \{w, z \in V^* \mid \exists x, y \in L, \exists r \in R : (x,\ y) \vdash_r (w,\ z)\}$.

We now can introduce the iteration of the splicing operation.

$\sigma^0(L) = L$,
$\sigma^{i+1}(L) = \sigma^i(L) \cup \sigma(\sigma^i(L)),\ i \geq 0$,
$\sigma^*(L) = \cup_{i \geq 0} \sigma^i(L)$.
The iterated splicing preserves the regularity of a language:

Theorem 1. *[9] Let $L \subseteq T^*$ be a regular language and let $\sigma = (T, R)$ be a splicing scheme. Then language $\sigma^*(L)$ is regular.*

An *extended H system* is a quadruple $\gamma = (V, T, A, R)$, where V is an alphabet, $T \subseteq V$, $A \subseteq V^*$, and R is a set of splicing rules over V. We call V the alphabet of γ, T is the *terminal* alphabet, A is the set of *axioms*. The *language generated* by γ is defined by $L(\gamma) = \sigma^*(A) \cap T^*$, where $\sigma = (V, R)$ is the underlying H scheme of γ.

We say that $\gamma = (V, T, A, R)$ *computes* $L \subseteq V^*$ on input w if $L = L(\gamma')$, where $\gamma' = (V, T, A \cup \{w\}, R)$.

2.2 Splicing (tissue) P systems

A *splicing tissue P system* of degree $m \geq 1$ is a construct

$$\Pi = (V, T, G, A_1, \ldots, A_m, R_1, \ldots, R_m),$$

where V is a finite alphabet, $T \subseteq V$ is the terminal alphabet and G is the underlying directed labeled graph of the system. The graph G has m nodes (cells) numbered from 1 to m. Each node i contains a set of strings (a language) A_i over V. Symbols R_i, $1 \leq i \leq m$ are finite sets of rules (associated to nodes) of the form $(r; tar_1, tar_2)$, where r is a splicing rule: $r = u_1 \# u_2 \$ u_3 \# u_4$ and $tar_1, tar_2 \in \{here, out\} \cup \{go_j \mid 1 \leq j \leq m\}$ are target indicators. We remark that the communication graph G can be deduced from the sets of rules. More precisely, G contains an edge (i, j), iff there is a rule $(r; tar_1, tar_2) \in R_i$ with $tar_k = go_j$, $k \in \{1, 2\}$. If one of tar_k is equal to *here*, then G contains the loop (i, i).

A *configuration* of Π is the m-tuple $(N_1, \ldots, N_m)$, where $N_i \subseteq V^*$. A *transition* between two configurations $(N_1, \ldots, N_m) \Rightarrow (N'_1, \ldots, N'_m)$ is defined as follows. In order to pass from one configuration to another, splicing rules of each node are applied in parallel to all possible words that belong to that node. After that, the result of each splicing is distributed according to target indicators. More exactly, if there are x, y in N_i and $r = (u_1 \# u_2 \$ u_3 \# u_4; tar_1, tar_2)$ in R_i, such that $(x,\ y) \vdash_r (w,\ z)$, then words w and z are sent to nodes indicated by tar_1, respectively tar_2. We write this as follows $(x,\ y) \vdash_r (w,\ z)(tar_1, tar_2)$. If $tar_k = here$, $k = 1, 2$, then the word remains in node i (is added to N'_i); if $tar_k = go_j$, then the word is sent to node j (is added to N'_j); if $tar_k = out$, the word is sent outside of the system.

Since the words are present in an arbitrary number of copies, after the application of rule r in node i, words x and y are still present in the same node.

A *computation* in a splicing tissue P system Π is a sequence of transitions between configurations of Π which starts from the initial configuration

$(A_1, \ldots, A_m)$. The result of the computation consists of all words over terminal alphabet T which are sent outside the system at some moment of the computation. We denote by $L(\Pi)$ the language generated by system Π.

We also define the notion of an *input* for the system above. An input word for a system Π is simply a word w over the non-terminal alphabet of Π. The computation of Π on input w is obtained by adding w to the axioms of A_1 and after that by evolving Π as usual. We denote by $L(\Pi, w)$ the result of the computation of Π on w.

We consider the following restricted variant of splicing tissue P systems. A *restricted splicing tissue P system* is a subclass of splicing tissue P systems which has the property that for any rule $(r; tar_1, tar_2)$ either $tar_1 = tar_2 = go_j$, or $tar_1 = tar_2 = out$ or $tar_1 = tar_2 = here$. This means that both resulting strings are moved over the same connection. In this case, we may associate splicing rules to corresponding edges. If both targets are *out*, then we can associate the splicing rule with an edge going to the special node called *out*.

3 Universal restricted splicing tissue P system of small size

The universality proofs are based on a simulation of tag systems [2, 7] using the well known rotate-and-simulate method. We show that the functioning of any tag system can be simulated using restricted splicing tissue P system with 6 rules. The universality of the corresponding system follow from the existence of universal tag systems.

Let $V = \{a_1, \ldots, a_{n+1}\}$ be an alphabet and $\alpha, \beta \notin V$. Consider coding morphisms c and $\bar{c}$ defined as follows: $c(a_i) = \alpha^{i+1}\beta$, $\bar{c}(a_i) = \beta\alpha^{i+1}$.

Theorem 2. *Let $TS = (2, V, P)$ be a tag system and $w \in V^*$. Then, there is a restricted splicing tissue P system $\Pi = (V', T, G, A_1, A_2, A_3, R_1, R_2, R_3)$, having 6 rules, which given the word $X\beta\beta c(w)\beta Y$ as input simulates TS on input w,* i.e. *such that:*

1. *for any word w on which TS halts producing the result w', the system Π produces a unique result $X'c(w')Y'$,* i.e., $L(\Pi, w) = \{X'c(w')Y'\}$.
2. *for any word w on which TS does not halt, the system Π computes infinitely without producing a result,* i.e., $L(\Pi, w) = \emptyset$.

Proof. We construct the system Π as follows.

Let $|V| = n + 1$. We use the following alphabets V' and T.

$$V' = \{\alpha, \beta, X, X', Y, Y', Z, Z'\}, \qquad T = \{X', Y', \alpha, \beta\}.$$

The initial languages A_j, $j \in \{1, 2, 3\}$ are given as follows.

$$A_1 = \{Z'c(P_i)\bar{c}(a_i)Y \mid a_i \to P_i \in P, 1 \leq i \leq n\} \cup \{X\beta Z,\ ZY,\ Z'Y'\},$$
$$A_2 = \{XZ\},$$
$$A_3 = \{XZ,\ X'Z\}.$$

The set of rules R_j, $j \in \{1, 2, 3\}$ are given as follows.

$$R_1 = \{1.1 : (\varepsilon\#\beta Y\$Z'\#\varepsilon; go_3, go_3);\ 1.2 : (\varepsilon\#\alpha Y\$Z\#Y; go_2, go_2);$$
$$1.3 : (X\beta\alpha\#\varepsilon\$X\beta\#Z; here, here)\};$$
$$R_2 = \{2.1 : (X\alpha\#\varepsilon\$X\#Z; go_1, go_1)\};$$
$$R_3 = \{3.1 : (X\beta\beta\#\alpha\alpha\$X\#Z; go_1, go_1);\ 3.2 : (X\beta\beta\#\alpha\beta\$X'\#Z; out, out)\}.$$

The graph G can be deduced from the rules above and it is represented in Figure 1. We observe that this graph is induced by a tree structure with duplex communication and loops.

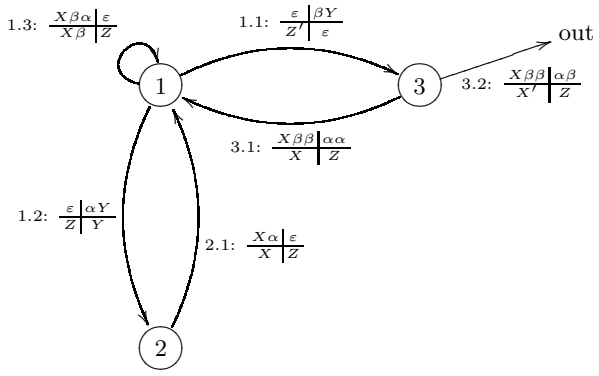

Fig. 1. The communication graph G associated to the construction from Theorem 2.

The simulation of TS is performed as follows. For every step of the derivation in TS there is a sequence of several derivation steps in Π. The current configuration w of TS is encoded by a string $X\beta\beta c(w)\beta Y$ present in node 1 of Π (the initial configuration of Π satisfies this property). The simulation of a production $a_i \to P_i$, $1 \leq i \leq n$ is performed using the rotate-and-simulate method used for many proofs in the area of splicing systems. We use this method as follows. First, suffixes $c(P_j)\bar{c}(a_j)$, $1 \leq j \leq n$ are attached to the string producing $X\alpha^i\beta c(a_k w')c(P_j)\beta\alpha^j Y$. After that one symbol α is removed at the left end of the string (rule 2.1) and one symbol α is removed at the right end of the string (rule 1.2). A process of deleting of symbols α continues in the same

manner. Hence, only the string for which $j = i$ will remain at the end, producing $X\beta c(a_k w')\beta Y$. After that the symbol a_k is removed (by removing corresponding α's) and a new round begins. The simulation stops when the first symbol is a_{n+1}.

Now we consider a simulation of one step of the derivation in TS. Notice, that in our construction every string that cannot evolve during one step of the simulation cannot evolve anymore. Let $X\beta\beta c(w)\beta Y$ be present in node 1 and $1 \leq j \leq n$. Then, there are two cases with respect to the first letter of w.

1. If $w = a_i a_k w'$, with $1 \leq i \leq n$ and $1 \leq k \leq n+1$ (hence the corresponding configuration in Π will be $X\beta\beta\alpha^i\beta\alpha^k\beta c(w')\beta Y$), then the only possibility is to use the rule 1.1:

   $$(X\beta\beta\alpha^i\beta\alpha^k\beta c(w')\beta Y,\ \ Z'c(P_j)\beta\alpha^j Y) \vdash_{1.1} (X\beta\beta\alpha^i\beta\alpha^k\beta c(w')c(P_j)\beta\alpha^j Y,\ Z'\beta Y),$$

   or

   $$(X\beta\beta\alpha^i\beta\alpha^k\beta c(w')\beta Y,\ Z'Y') \vdash_{1.1} (X\beta\beta\alpha^i\beta\alpha^k\beta c(w')Y',\ Z'\beta Y).$$

   The string $Z'\beta Y$ cannot evolve anymore. The other strings can be used in rule 3.1:

   $$(X\beta\beta\alpha^i\beta\alpha^k\beta c(w')c(P_j)\beta\alpha^j Y,\ XZ) \vdash_{3.1} (Xa^i\beta\alpha^k\beta c(w')c(P_j)\beta\alpha^j Y,\ X\beta\beta Z),$$
   $$(X\beta\beta\alpha^i\beta\alpha^k\beta c(w')Y',\ XZ) \vdash_{3.1} (X\alpha^i\beta\alpha^k\beta c(w')Y',\ X\beta\beta Z).$$

   The strings $X\beta\beta Z$ and $X\alpha^i\beta\alpha^k\beta c(w')Y'$ cannot evolve anymore. The remaining strings are of the form $X\alpha^i\beta\alpha^k\beta c(w')c(P_j)\beta\alpha^j Y$. There are 4 cases with respect to values of i and j.

   Case $\mathbf{i > 0}$, $\mathbf{j > 0}$. Then only rule 1.2 is applicable, followed by the application of rule 2.1:

   $$(Xa^i\beta\alpha^k\beta c(w')c(P_j)\beta\alpha^j Y,\ ZY) \vdash_{1.2} (Xa^i\beta\alpha^k\beta c(w')c(P_j)\beta\alpha^{j-1} Y,\ Z\alpha Y),$$
   $$(Xa^i\beta\alpha^k\beta c(w')c(P_j)\beta\alpha^{j-1} Y,\ XZ) \vdash_{2.1} (Xa^{i-1}\beta\alpha^k\beta c(w')c(P_j)\beta\alpha^{j-1} Y,\ X\alpha Z).$$

   Hence indices i and j are decremented simultaneously. The strings $Z\alpha Y$ and $X\alpha Z$ cannot evolve anymore.

   Case $\mathbf{i > 0}$, $\mathbf{j = 0}$. Then rule 1.1 is applicable:

   $$(X\alpha^i\beta\alpha^k\beta c(w')c(P_j)\beta Y,\ Z'c(P_t)\beta\alpha^t Y) \vdash_{1.1}$$
   $$(X\alpha^i\beta\alpha^k\beta c(w')c(P_j)c(P_t)\beta\alpha^t Y,\ Z'\beta Y),$$
   $$(X\alpha^i\beta\alpha^k\beta c(w')c(P_j)\beta Y,\ Z'Y') \vdash_{1.1} (X\alpha^i\beta\alpha^k\beta c(w')c(P_j)Y',\ Z'\beta Y).$$

   All the strings cannot evolve anymore in this case.

Case $\mathbf{i = 0,\ j > 0}$. Then rule 1.3 can be applied followed by an application of rule 1.2:

$$(X\beta\alpha^k\beta c(w')c(P_j)\beta\alpha^jY,\ X\beta Z) \vdash_{1.3} (X\beta\alpha^{k-1}\beta c(w')c(P_j)\beta\alpha^jY,\ X\beta\alpha Z),$$
$$(X\alpha^t\beta c(w')c(P_j)\beta\alpha^jY,\ ZY) \vdash_{1.2} (X\alpha^t\beta c(w')c(P_j)\beta\alpha^{j-1}Y,\ Z\alpha Y),$$
for some $t \geq 0$.

In this case, the strings cannot evolve anymore.

Case $\mathbf{i = 0,\ j = 0}$. We observe that in this case we had $i = j$. Then either rule 1.3 can be iteratively applied until $k = 0$, or rule 1.1 is applied when $k > 0$.

$$(X\beta\alpha^k\beta c(w')c(P_i)\beta Y,\ X\beta Z) \vdash_{1.3^*} (X\beta\beta c(w')c(P_i)\beta Y,\ X\beta\alpha Z),$$
$$(X\beta\alpha^k\beta c(w')c(P_i)\beta Y,\ Z'c(P_t)\beta\alpha^tY) \vdash_{1.1} (X\beta a^k\beta c(w')c(P_i)c(P_t)$$
$$\beta\alpha^tY,\ Z'\beta Y),$$
$$(X\beta\alpha^k\beta c(w')c(P_i)\beta Y,\ Z'Y') \vdash_{1.1} (X\beta a^k\beta c(w')c(P_i)Y',\ Z'\beta Y).$$

In the first case the string $X\beta\beta c(w'P_i)\beta Y$ is obtained, which corresponds to the string $w'P_i$ of TS. Hence the corresponding production is simulated. In the latter two cases, the resulting strings cannot evolve anymore.

2. If $w = a_{n+1}w'$, (hence the corresponding configuration in Π will be $X\beta\beta\alpha\beta c(w')\beta Y$), then the only possibility is to use rule 1.1:

$$(X\beta\beta\alpha\beta c(w')\beta Y,\ Z'c(P_j)\beta\alpha^jY) \vdash_{1.1} (X\beta\beta\alpha\beta c(w')c(P_j)\beta\alpha^jY,\ Z'\beta Y),$$
$$(X\beta\beta\alpha\beta c(w')\beta Y,\ Z'Y') \vdash_{1.1} (X\beta\beta\alpha\beta c(w')Y',\ Z'\beta Y).$$

Then the only applicable rule is rule 3.2.

$$(X\beta\beta\alpha\beta c(w')c(P_j)\beta\alpha^jY,\ X'Z) \vdash_{3.2} (X'\alpha\beta c(w')c(P_j)\beta\alpha^jY,\ X\beta\beta Z),$$
$$(X\beta\beta\alpha\beta c(w')Y',\ X'Z) \vdash_{3.2} (X'c(w')Y',\ X\beta\beta Z).$$

Only the string $X'c(w')Y'$ is considered as a result because other strings contain nonterminal symbols.

So, Π correctly simulates one step of the derivation in TS, thus Π correctly simulate whole derivation in TS. It is also clear that a successful computation in TS may be reconstructed from a successful computation in Π. This concludes the proof.

4 Conclusions

In this paper we investigated splicing tissue P systems and we constructed a universal splicing P system with 6 rules. This result is quite remarkable, because the usual implementation of the rotation method for splicing systems needs at least 4 rules. An open problem raised by this result is if the above number is minimal. Other open problems concern the minimal number of rules in the case of other variants of splicing systems, like TVDH systems or splicing test tube systems [9].

Acknowledgments A.Alhazov gratefully acknowledges the support of the Japan Society for the Promotion of Science and the Grant-in-Aid for Scientific Research, project 20·08364. All authors acknowledge the support by the Science and Technology Center in Ukraine, project 4032.

References

1. Alhazov, A., Verlan, S.: Minimization strategies for maximally parallel multiset rewriting systems. Technical Report 862, TUCS Report No. 862 (2008)
2. Cocke, J., Minsky, M.: Universality of Tag Systems with P=2. Journal of the ACM, vol.11, no.1, 15–20 (1964)
3. Csuhaj-Varjú, E., Margenstern, M., Vaszil, G., Verlan, S.: Small Computationally Complete Symport/Antiport P systems. Theoretical Computer Science, vol. 372, no. 2–3, 152–164 (2007)
4. Frisco, P.: Computing with Cells: Advances in Membrane Computing. Oxford University press (2009)
5. Head, T.: Formal Language Theory and DNA: an Analysis of the Generative Capacity of Specific Recombinant Behaviors. Bulletin of Mathematical Biology, vol. 49, no. 6, 737–759 (1987)
6. Hopcroft, J.E., Motwani, R., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley, Reading, Mass., 2nd edition (2001)
7. Minsky, M.: Computations: Finite and Infinite Machines. Prentice Hall, Englewood Cliffts, NJ (1967)
8. Păun, Gh.: Membrane Computing. An Introduction. Springer (2002)
9. Păun, Gh., Rozenberg, G., Salomaa, A.: DNA Computing: New Computing Paradigms. Springer (1998)
10. Păun, Gh., Rozenberg, G., Salomaa A. (eds.): The Oxford Handbook of Membrane Computing. Oxford University Press (2010)
11. Rogozhin, Yu., Verlan, S: On the Rule Complexity of Universal Tissue P Systems. LNCS, vol. 3850, 356–362, Springer (2006)
12. Rozenberg, G., Salomaa A. (eds.): Handbook of Formal Languages, 3 volumes. Springer (1997)