On the Membrane Computing Based on Splicing

Andrei PĂUN, Mihaela PĂUN
Department of Computer Science
University of Western Ontario
London, Ontario, Canada N6A 5B7
apaun@csd.uwo.ca

Abstract. This paper is a direct continuation of [11]. Characterizations of recursively enumerable languages are given, by means of splicing P systems having splicing rules of small size (that is, involving short context strings). Also it is shown that with only two membranes we can generate all the recursively enumerable languages; this improves a result from [11], where three membranes are used.

1 Introduction

The P systems are a class of distributed parallel computing devices of a biochemical inspiration (they can be considered as a possible branch of natural/molecular computing), introduced in [7]. The reader can find results in this area in [2], [10], [12], etc.

In short, one considers a *membrane structure* consisting of several cell-membranes which are hierarchically embedded in a main membrane, called the *skin* membrane. The membranes delimit *regions*, where we place *objects*. The objects evolve according to given *evolution rules*, which are associated with the regions. In the present paper we deal with the case when the objevts are represented by strings and the evolution rules are splicing rules, as formalized in [3].

Starting from an initial configuration (identified by the membrane structure, and the objects placed in its regions) and using the evolution rules in parallel, we get a *computation*. A computation identifies a language, the set of all terminal strings which leave the system. P systems of this form were investigated in [11], where several characterizations of recursively enumerable languages were obtained. We improve here one of these results in what concerns the number of membranes and we also consider the size of the splicing rules – in the sense of [4], we bound the length of the contexts used in the splicing rules. We find that P systems with splicing rules of a rather small complexity suffice.

2 Splicing P Systems

We will first remind the splicing operation introduced in [3] as a formal model of the DNA recombination under the influence of restriction enzymes and ligases.

A *splicing rule* (over an alphabet V) is a string $r = u_1\#u_2\$u_3\#u_4$, where $u_1, u_2, u_3, u_4 \in V^*$ and $\#, \$$ are two special symbols not in V. (V^* is the free monoid generated by the alphabet V under the operation of concatenation; the empty string is denoted by λ; the length of $x \in V^*$ is denoted by $|x|$.)

For $x, y, w, z \in V^*$ and r as above we write

$$\begin{aligned}(x,y) \vdash_r (w,z) \quad \text{iff} \quad & x = x_1u_1u_2x_2,\ y = y_1u_3u_4y_2,\\ & w = x_1u_1u_4y_2,\ z = y_1u_3u_2x_2,\\ & \text{for some } x_1, x_2, y_1, y_2 \in V^*.\end{aligned}$$

We say that we splice x, y at the *sites* $u_1u_2,\ u_3u_4$. These sites encode the patterns recognized by restrictions enzymes able to cut the DNA sequences between u_1, u_2, respectively between u_3, u_4.

When r is understood, we write $\vdash$ instead of $\vdash_r$. For clarity, we usually indicate by a vertical bar the place of splicing: $(x_1u_1|u_2x_2, y_1u_3|u_4y_2) \vdash (x_1u_1u_4y_2, y_1u_3u_2x_2)$.

The *radius* of a splicing rule $u_1\#u_2\$u_3\#u_4$ is the length of the longest string u_1, u_2, u_3, u_4.

A pair $\sigma = (V, R)$, where V is an alphabet and R is a set of splicing rules over V, is called an *H scheme*. With respect to an H scheme $\sigma = (V, R)$ and a language $L \subseteq V^*$ we define

$$\begin{aligned}&\sigma(L) = \{w \in V^* \mid (x,y) \vdash_r (w,z) \text{ or } (x,y) \vdash_r (z,w), \text{ for some } x, y \in L, r \in R, z \in V^*\},\\ &\sigma^*(L) = \bigcup_{i \geq 0} \sigma^i(L), \text{ for}\\ &\sigma^0(L) = L \text{ and } \sigma^{i+1}(L) = \sigma^i(L) \cup \sigma(\sigma^i(L)),\ i \geq 0.\end{aligned}$$

An *extended H system* is a construct $\gamma = (V, T, A, R)$, where V is an alphabet, $T \subseteq V$, $A \subseteq V^*$, and $R \subseteq V^*\#V^*\$V^*\#V^*$. ($T$ is the *terminal* alphabet, A is the set of *axioms*, and R is the set of *splicing rules*.) When $T = V$, the system is said to be *non-extended*. The pair $\sigma = (V, R)$ is the *underlying H scheme* of γ.

We define the *diameter* of γ, [4], by $dia(\gamma) = (n_1, n_2, n_3, n_4)$, where

$$n_i = \max\{|u_i| \mid u_1\#u_2\$u_3\#u_4 \in R\},\ 1 \leq i \leq 4.$$

For any $L \subseteq V^*$ and $\gamma = (V, T, A, R)$ we define

$$\begin{aligned}&\sigma(L) = \{w \mid (x,y) \vdash_r (w,z) \text{ or } (x,y) \vdash_r (z,w), \text{ for } x, y \in L, r \in R\},\\ &\sigma^*(L) = \bigcup_{i \geq 0} \sigma^i(L), \text{ for}\\ &\sigma^0(L) = L, \text{ and } \sigma^{i+1}(L) = \sigma^i(L) \cup \sigma(\sigma^i(L)),\ i \geq 0.\end{aligned}$$

Then, the language generated by γ is $L(\gamma) = \sigma^*(A) \cap T^*$. (We iterate the splicing operation according to rules in R, starting from strings in A, and we keep only the strings composed of terminal symbols.)

It is known that extended H systems with finite sets of axioms and of splicing rules characterize the regular languages, [1], [13], while H systems with regular sets of rules characterize the recursively enumerable languages, [6].

Let us now pass to splicing P systems, the object of our investigation.

We identify a membrane structure with a string of correctly matching parentheses, placed in a unique pair of matching parentheses; each pair of matching parentheses corresponds to a membrane. Graphically, a membrane structure is represented by a Venn diagram.

A *P system* is a membrane structure with multisets of *objects* placed in its regions and provided with *evolution rules* for these objects. We define here only the splicing P systems, in the variant we investigate in this paper.

A *splicing P system* (of degree $m, m \geq 1$) is a construct

$$\Pi = (V, T, \mu, L_1, \ldots, L_m, R_1, \ldots, R_m),$$

where:

(i) V is an alphabet; its elements are called *objects*;

(ii) $T \subseteq V$ (the *output* alphabet);

(iv) μ is a membrane structure consisting of m membranes (labeled with $1, 2, \ldots, m$);

(v) $L_i, 1 \leq i \leq m$, are languages over V associated with the regions $1, 2, \ldots, m$ of μ;

(vi) $R_i, 1 \leq i \leq m$, are finite sets of *evolution rules* associated with the regions $1, 2, \ldots, m$ of μ, given in the following form: $(r = u_1 \# u_2 \$ u_3 \# u_4; tar_1, tar_2)$, where $r = u_1 \# u_2 \$ u_3 \# u_4$ is a usual splicing rule over V and $tar_1, tar_2 \in \{here, out\} \cup \{in_j \mid 1 \leq j \leq m\}$.

Note that, as usual in H systems, when a string is present in a region of our system, it is assumed to appear in arbitrarily many copies (any number of copies of a DNA molecule can be obtained by amplification). Thus, we do not use here multisets, as in basic P systems.

Any m-tuple $(M_1, \ldots, M_m)$ of languages over V is called a *configuration* of Π. For two configurations $(M_1, \ldots, M_m), (M'_1, \ldots, M'_m)$ of Π we write $(M_1, \ldots, M_m) \Longrightarrow (M'_1, \ldots, M'_m)$ if we can pass from $(M_1, \ldots, M_m)$ to $(M'_1, \ldots, M'_m)$ by applying the splicing rules from each region of μ, in parallel, to all possible strings from the corresponding regions, and following the target indications associated with the rules. More specifically, if $x, y \in M_i$ and $(r = u_1 \# u_2 \$ u_3 \# u_4, tar_1, tar_2) \in R_i$ such that we can have $(x, y) \vdash_r (w, z)$, then w and z will go to the regions indicated by tar_1, tar_2, respectively. If $tar_j = here$, then the string remains in M_i, if $tar_j = out$, then the string is moved to the region immediately outside the membrane i (maybe, in this way the string leaves the system), if $tar_j = in_k$, then the string is moved to the region k, providing that this is immediately below; if not, then the rule cannot be applied. Note that the strings x, y are still available in region M_i, because we have supposed that they appear in arbitrarily many copies (an arbitrarily large number of them were spliced, arbitrarily many remain), but if a string w, z is sent out of region i, then no copy of it remains here.

A sequence of transitions between configurations of a given P system Π, starting from the initial configuration $(L_1, \ldots, L_m)$, is called a *computation* with respect to Π. The result of a computation consists of all strings over T which are sent out of the system at any time during the computation. We denote by $L(\Pi)$ the language of all strings of this type. We say that $L(\Pi)$ is *generated* by Π.

Note two important facts: if a string leaves the system but it is not terminal, then it is ignored; if a string remains in the system, even if it is terminal, then it does not contribute to the language $L(\Pi)$. It is also worth mentioning that we do not consider here halting computations. We leave the process to continue forever and we just observe it from outside and collect the terminal strings leaving it.

We denote by $SPL(tar, m, p)$ the family of languages $L(\Pi)$ generated by splicing P systems as above, of degree at most $m, m \geq 1$, and of depth at most $p, p \geq 1$. (The depth of a P system is equal to the height of the tree describing its membrane structure.) If all target indications tar_1, tar_2 in the evolution rules of a P system are of the form *here, out, in*, then we say that Π is of the *i/o type*; the strings produced by splicing and having associated the indication *in* are

moved into any lower region immediately below the region where the rule is used. The family of languages generated by P systems with this weaker target indication and of degree at most m and depth at most p is denoted by $SPL(i/o, m, p)$.

We define the *diameter* of a splicing P system $\Pi = (V, T, \mu, L_1, \ldots, L_m, R_1, \ldots, R_m)$, in a similar way as we defined in the case of extended H systems ([4]), by $dia(\Pi) = (n_1, n_2, n_3, n_4)$, where

$$n_i = \max\{|u_i| \mid u_1\#u_2\$u_3\#u_4 \in R_1 \cup \ldots \cup R_m\},\ 1 \leq i \leq 4.$$

We denote the family of languages generated by P systems with the weaker target indication and of degree at most m and depth at most p and with diameter (n_1, n_2, n_3, n_4) by $SPL(i/o, m, p, (n_1, n_2, n_3, n_4))$.

By RE we denote the family of recursively enumerable languages.

3 The Power of Splicing P Systems

We start by improving a result from [11], where it is proved that $RE = SPL(i/o, 3, 3)$. No care is paid in the proof of this result to the diameter of the used system.

Theorem 1. $SPL(i/o, 2, 2) = SPL(tar, 2, 2) = RE$.

Proof. Let $G = (N, T, S, P)$ be a type-0 Chomsky grammar and let B be a new symbol. Assume that $N \cup T \cup \{B\} = \{\alpha_1, \ldots, \alpha_n\}$ and that P contains m rules, $u_i \to v_i, 1 \leq i \leq m$. Consider also the rules $u_{m+j} \to v_{m+j}, 1 \leq j \leq n$, for $u_{m+j} = v_{m+j} = \alpha_j$.

We construct the splicing P system (of degree 2):

$$\begin{aligned}
\Pi &= (V, T, \mu, L_1, L_2, R_1, R_2),\\
V &= N \cup T \cup \{B, X, X', Y, Y', Z, Z'\} \cup \{X_i \mid 0 \leq i \leq n+m\} \cup \{Y_i \mid 0 \leq i \leq n+m\},\\
\mu &= [_1[_2]_2]_1,\\
L_1 &= \{XSBY, Z', X'Z, ZY'\} \cup \{ZY_i \mid 0 \leq i \leq n+m\} \cup \{X_iv_iZ \mid 1 \leq i \leq n+m\},\\
R_1 &= \{(X_iv_i\#Z\$X\#;\ in,\ out),\ (\#Y_i\$Z\#Y_{i-1};\ in,\ out) \mid 1 \leq i \leq m+n\}\\
&\cup\ \{(\#Y_0\$Z\#Y';\ in,\ out),\ (X_0\#\$X'\#Z;\ out,\ here)\}\\
&\cup\ \{(\#BY\$Z'\#;\ here,\ out),\ (X\#\$\#Z';\ out,\ out)\}.\\
L_2 &= \{XZ, ZY\} \cup \{ZY_i \mid 1 \leq i \leq m+n\} \cup \{X_iZ \mid 0 \leq i \leq m+n-1\},\\
R_2 &= \{(\#u_iY\$Z\#Y_i;\ out,\ here),\ (X_i\#\$X_{i-1}\#Z;\ here,\ out) \mid 1 \leq i \leq m+n\}\\
&\cup\ \{(X'\#\$X\#Z;\ here,\ here),\ (\#Y'\$Z\#Y;\ out,\ here)\}.
\end{aligned}$$

The idea of this proof is the "rotate-and-simulate" procedure, as used in many proofs in the H systems theory (first in [6]). Here both the simulation of the rules in G and the circular permutation of strings are performed in Π in the same way: a suffix u of the current string is removed and the corresponding string, v, is added in the left end of the string. For $u \to v$ a rule in P, we simulate a derivation step in G. For $u = v$ a symbol in $N \cup T \cup \{B\}$ we have one symbol "rotation" of the current string.

In the end of a computation we want to have the word in the right permutation, so we have to mark the beginning of the word. For this purpose we use the new letter B which marks the beginning of the right sentential word.

Let us see in more detail the work of the system.

The "main" axiom is $XSBY$; we will always process a string of the form Xw_1Bw_2Y (the axiom is of that form). We replace a suffix u_iY of this word with Y_i (in region 2) and the prefix X with X_jv_j (in region 1). Then we will repeatedly decrease the subscripts of Y_i and X_j by one. In the end we will replace Y_0 with Y and X_0 with X (this means that $i = j$; so we simulated the production $u_i \to v_i$). In this way we can simulate the productions from P and rotate the string.

In the end we cut in membrane 1 the markers B and Y together (to be sure that we have the right permutation of the word) and finally we cut the marker X and send the string out. Thus, we get $L(G) \subseteq L(\Pi)$.

Now we will prove the converse inclusion.

First we can observe that we send out strings from the first membrane, and that we send out strings with at least a marker Z, X, Y (or variants of them with subscripts or primed). The only exception is the following splicing rule: $(X\#\$\#Z'; \ out, \ out)$. Of course, the first string produced by this splicing rule contains the markers X and Z', but the second string can be a terminal one.

Suppose that we start in the first membrane.

The rule $(\#Y_i\$Z\#Y_{i-1}; \ in, \ out)$ can only be applied to two axioms ZY_i and ZY_{i-1} and we produce the same strings; the string ZY_{i-1} being sent to membrane 2. In membrane 2 we have these axioms but ZY_0 which cannot enter any splicing here.

The rule $(\#Y_0\$Z\#Y'; \ in, \ out)$ also uses two axioms, namely ZY_0 and ZY', so we produce the same strings; the string ZY' will be sent to membrane 2 where it can be spliced using $(\#Y'\$Z\#Y; \ out, \ here)$ and then nothing can be produced by the word ZY which gets in membrane 1.

The rule $(X_0\#\$X'\#Z; \ out, \ here)$ cannot be applied (we don't have X_0 now in the first membrane).

If we apply the rule $(\#BY\$Z'\#; \ here, \ out)$ then we still have the string XS in membrane 1. This string can enter splicing using the rule $(X_iv_i\#Z\$X\#; \ in, \ out)$ and the string X_iv_iS gets in the second membrane where the subscript of X is decreased by one and then the string gets in the first membrane. Here the string cannot enter any splicing.

Thus, we have to use first a rule $(X_iv_i\#Z\$X\#; \ in, \ out)$.

If we start in the second membrane, then the last two rules cannot be applied (we don't have X' or Y' here yet), while the outputs of the first two rules are the same strings that have entered the splicing, or strings which cannot enter any new splicing.

Consequently, we have to replace X by X_iv_i in membrane 1 and then to cut u_jY and replace it with Y_j in membrane 2. The string $X_iv_iwY_j$ gets in membrane 1, where the only possibility is to apply the rule $(\#Y_j\$Z\#Y_{j-1}; \ in, \ out)$. The string $X_iv_iwY_{j-1}$ gets in membrane 2, where the only possibility is to apply the rule $(X_i\#\$X_{i-1}\#Z; \ here, \ out)$, so the string $X_{i-1}v_iwY_{j-1}$ gets in membrane 1. We iterate the process until at least one subscript of X or Y is 0.

If we got X_0, then we decreased the subscript of X in membrane 2 and sent the string $X_0v_iY_{j-i}$ in membrane 1. Here we have two possibilities: $j \neq i$ or $j = i$.

If $j \neq i$ then $j - i \neq 0$, so we can decrease the subscript of Y and send the string in membrane 2. But here the string $X_0v_iwY_{j-i-1}$ can enter no further splicings. Before decreasing the subscript of Y, in membrane 1 we can also apply the splicing rule $(X_0\#\$X'\#Z; \ out, \ here)$; the string $X'v_iwY_{j-i}$ is sent to membrane 1 and we continue as before. In this case in membrane 2 we can replace X' by X using the rule $(X'\#\$X\#Z; \ here, \ here)$ and the string Xv_iY_{j-i-1} cannot enter any other splicings so it remains in membrane 2.

If $j = i$, then the only productions from region 1 that can be applied are $(\#Y_0\$Z\#Y'; \ in, \ out)$

and $(X_0\#\$X'\#Z;\ out,\ here)$. If we apply the first one, then the string X_0v_iwY' is sent to membrane 2, here we can only apply the rule that replaces Y' with Y, so the string X_0v_iwY gets in membrane 1. This string will never lead to a terminal string because we cannot delete the left marker (we can replace X_0 with X' but the string remains in this membrane and that marker cannot be deleted).

If we start with the rule $(X_0\#\$X'\#Z;\ out,\ here)$, then we get the string $X'v_iwY_0$. The only possibility to continue is to apply $(\#Y_0\$Z\#Y';\ in,\ out)$ and the string $X'v_iwY'$ gets in membrane 2. If we don't replace here X' with X, then again the string that gets into membrane 1 cannot lead to a terminal string. So first we replace X' with X (using the rule $X'\#\$X\#Z$) and then we replace Y' by Y by using the rule $(\#Y'\$Z\#Y;\ out,\ here)$. In this way we send the string Xv_iwY in membrane 1 and we can perform another step of rotating the word or simulating the rules from P.

Let us consider the case of Y_0. This means that so we decreased the subscript of Y in membrane 1 and sent the string $X_{i-j+1}v_iwY_0$ in membrane 2. If the subscript of X is 0, then we cannot apply any splicing rule and the computation stops. Suppose now that the subscript of X is not 0: then the only splicing that we can apply is $(X_{i-j+1}\#\$X_{i-j}\#Z;\ here,\ out)$ so we send the string $X_{i-j}v_iwY_0$ in membrane 1. Suppose now that $i \neq j$ (we already treated the case when they are equal). The only splicing rule that we can apply is $(\#Y_0\$Z\#Y';\ in,\ out)$ so $X_{i-j}v_iwY'$ gets in membrane 2. In this moment we can apply the following two splicing rules: $(X_{i-j}\#\$X_{i-j-1}\#Z;\ here,\ out)$ and $(\#Y'\$Z\#Y;\ out,\ here)$. If we apply the first one, then the string $X_{i-j-1}v_iwY'$ gets in membrane 1 and doesn't lead to terminal words because we cannot delete the marker Y'. If we apply the second splicing rule, then the word $X_{i-j}v_iwY$ enters membrane 1 and we cannot apply splicing rules that send away this word, so it doesn't count to the language (we also cannot delete the X_k marker).

Therefore, the computations in Π correctly cimulate rules in G or circularly permute the string. In the end we remove all markers from a string using $(\#BY\$Z'\#;\ here,\ out)$ and $(X\#\$\#Z';\ out,\ out)$.

In this way we get that $L(\Pi) \subseteq L(G)$. □

Now we will try to minimize the diameter of the splicing P systems of the i/o type. To be able to obtain the results we had to consider systems with 3 components. The results are similar to those obtained for the extended H systems with permitting/forbidding contexts, [4], [5]. The following auxiliary result is easy to be proved.

Lemma 1. $SPL(i/o, m, p, (n_1, n_2, n_3, n_4)) = SPL(i/o, m, p, (n_3, n_4, n_1, n_2))$, for all $m, p \geq 1$ and $n_i \geq 0$, $1 \leq i \leq 4$.

Now, we give the results anticipated.

Lemma 2. $SPL(i/o, 3, 3, (0, 2, 1, 0)) = SPL(i/o, 3, 3, (1, 0, 0, 2)) = RE$.

Proof. We will only prove that $SPL(i/o, 3, 3, (0, 2, 1, 0)) = RE$, the other equality follows from Lemma 1.

Let $G = (N, T, S, P)$ be a type-0 Chomsky grammar in the Kuroda normal form, and let B be a new symbol. Assume that $N \cup T \cup \{B\} = \{\alpha_1, \ldots, \alpha_n\}$ and that P contains m rules, $u_i \to v_i, 1 \leq i \leq m$. Consider also the rules $u_{m+j} \to v_{m+j}, 1 \leq j \leq n$, for $u_{m+j} = v_{m+j} = \alpha_j$.

We denote by P_1 the set of context-free rules considered above, and with P_2 the rest of the rules. One can see that the rules $u_{m+j} \to v_{m+j}, 1 \leq j \leq n$ are in P_1.

We construct the splicing P system (of degree 3):

$$
\begin{aligned}
\Pi &= (V, T, \mu, L_1, L_2, L_3, R_1, R_2, R_3),\\
V &= N \cup T \cup \{B, X, X', Y, Z_X, Z_{X'}, Z_Y, Z_\lambda, Z'_\lambda\} \cup \{Y_i, Z_{Y_i} \mid 0 \leq i \leq n+m\}\\
&\cup \{X_i, Z_{X_i}, Z_i, Y'_i, Z_{Y'_i} \mid 1 \leq i \leq m+n\},\\
\mu &= [_1[_2[_3]_3]_2]_1,\\
L_1 &= \{XSBY, Z_\lambda, Z'_\lambda\} \cup \{Z_{Y_i}Y_i \mid 0 \leq i \leq n+m\} \cup \{Z_{Y'_i}Y'_i \mid 1 \leq i \leq n+m\},\\
R_1 &= \{(\#u_iY\$Z_{Y_i}\#; in, out), \mid u_i \to v_i \in P_1\}\\
&\cup \{(\#DY\$Z_{Y'_i}\#; here, out), (\#CY'_i\$Z_{Y_i}\#; in, out) \mid u_i = CD \to v_i \in P_2\}\\
&\cup \{(\#Y_i\$Z_{Y_{i-1}}\#; in, out) \mid 1 \leq i \leq n+m\}\\
&\cup \{(\#BY\$Z_\lambda\#; here, out),\ (\#Z'_\lambda\$X\#; out, out)\},\\
L_2 &= \{XZ_X, X'Z_{X'}\} \cup \{X_iv_iZ_i \mid 1 \leq i \leq m+n\} \cup \{X_iZ_{X_i} \mid 1 \leq i \leq m+n-1\},\\
R_2 &= \{(\#Z_i\$X\#; out, in) \mid 1 \leq i \leq n+m\}\\
&\cup \{(\#Z_{X_{i-1}}\$X_i\#; out, in) \mid 2 \leq i \leq n+m\}\\
&\cup \{(\#Z_{X'}\$X_1\#; in, in),\ (\#Z_X\$X'\#; out, in)\},\\
L_3 &= \{Z_YY\},\\
R_3 &= \{(\#Y_0\$Z_Y\#; out, out)\}.
\end{aligned}
$$

This proof closely follows the proof of Theorem 1 from [11], with a special attention paid to the diameter of the splicing rules.

The sentential forms generated by G are simulated in Π in a circular permutation: Xw_1Bw_2Y, maybe with variants of X, Y, will be present in a region of Π if and only if w_2w_1 is a sentential form of G. Note that we can remove the nonterminal symbol Y only together with B from a string of the form $XwBY$. In this way, we ensure that the string is in the right permutation.

The simulation of rules in P and the rotation are done in the same way. Assume that some string Xwu_iY is present in region 1. We have now two cases: if $u_i \to v_i \in P_1$, then we simulate right away the rule $u_i \to v_i$ with a splicing rule of the form $(\#u_iY\$Z_{Y_i}, in, here)$. If $u_i = CD \to v_i \in P_2$, then the simulation requires two steps: first we replace DY with Y'_i and then we replace CY'_i with Y_i.

So in both cases we replace the suffix u_iY with Y_i, $1 \leq i \leq m+n$: initially we have here the string $XSBY$. We can perform

$$(Xw|u_iY, Z_{Y_i}|Y_i) \vdash (XwY_i, Z_{Y_i}u_iY) \text{ for } u_i \to v_i \in P_1,$$

or else,

$$(XwC|DY, Z_{Y'_i}|Y'_i) \vdash (XwCY'_i, Z_{Y'_i}DY) \text{ and } (Xw|CY'_i, Z_{Y_i}|Y_i) \vdash (XwY_i, Z_{Y_i}CY'_i).$$

The string XwY_i is sent to region 2, the "by products" are sent out and don't enter the language generated by Π because they contain at least a nonterminal. In region 2 we can only perform a splicing of the form $(X|wY_i, X_jv_j|Z_j) \vdash (XZ_j, X_jv_jwY_i)$, for some $1 \leq j \leq n+m$. The string $X_jv_jwY_i$ is sent back to region 1, XZ_j is sent to membrane 1 and cannot enter other splicings. Now, in region 1 the only splicing which can be applied to the string $X_jv_jwY_i$ is $(X_jv_jw|Y_i, Z_{Y_{i-1}}|Y_{i-1}) \vdash (X_jv_jwY_{i-1}, Z_{Y_{i-1}}Y_i)$. The string $X_jv_jwY_{i-1}$ is sent to region 2, while

$Z_{Y_{i-1}}Y_{i-1}$ is sent out and don't enter the generated language. In region 2 we now decrease by one the subscript of X_j, using the rule $(X_j|v_jwY_{i-1}, X_{j-1}|Z_{X_{j-1}}) \vdash (X_jZ, X_{j-1}v_jwY_{i-1})$. We iterate this process of decreasing the subscripts until either the subscript of X reaches 1 or the subscript of Y becomes 0.

If at some moment we reach X_1, hence in region 2 we have a string $X_1v_jwY_k$, then we perform $(X_1|v_jwY_k, X'|Z_{X'}) \vdash (X_1Z_{X'}, X'v_jwY_k)$ and $X'v_jwY_k$ is sent to membrane 3. If $k \neq 0$, then nothing can be done, the string is "lost". Otherwise, Y_0 is replaced with Y and the string $X'v_jwY$ is sent to region 2; X' is replaced here by X and the string Xv_jwY is sent to the skin membrane.

If at some moment in region 2 we get a string $X_kv_jwY_0$, for $k \geq 2$, this string cannot be processed in the skin membrane, hence it is "lost". Thus, we can correctly continue only when $i = j$, hence we have passed from Xwu_iY to Xv_iwY; in this way we have either correctly simulated a rule from P or we have circularly permuted the string with one symbol. This is true because we cannot have "illegal" splicings: "by product" strings generated in membrane 1 are sent out, the ones produced in membrane 2 are sent to membrane 3 (none of them containing either Y_0 or Z_Y, hence cannot enter in splicings in membrane 3) and the "garbage" Z_YY_0 from membrane 3 is sent to membrane 2, where it cannot enter any splicing. Because of this fact, we know that when we have Z with a subscript in a word entering a splicing, then that word is a axiom (and one can see that there are not two different axioms containing Z with the same subscript).

The process of simulating a rule or of rotating the string with one symbol can be iterated. Therefore, all derivations in G can be simulated in Π and, conversely, all correct computations in Π correspond to correct derivations in G. Because we collect only terminal strings which leave the system, we have the equality $L(G)) = L(\Pi)$. It is easy to see that the diameter of the P system is $(0, 2, 1, 0)$. □

In the proof of Lemma 2, the application of rules of the grammar G was simulated in the equivalent P system in the right hand end of the strings. It is easy to see that we can perform this simulation also in the left hand end of the strings. Also the rotation can be done in the reverse direction to that in the proof of Lemma 2: cut a symbol from the left end of the string and add it in the right hand end, repeatedly. A counterpart of the result in Lemma 2 can be obtained in this way.

Lemma 3. $SPL(i/o, 3, 3, (2, 0, 0, 1)) = SPL(i/o, 3, 3, (0, 1, 2, 0)) = RE$.

Proof. Again we will prove only one equality, $SPL(i/o, 3, 3, 2001) = RE$, the second one following from Lemma 1.

We simply repeat the construction in the proof of Lemma 2, and make the changes so that the strings be rotated in the converse order. We give here only the construction of the system; its correctedness can be proved as in the previous proof. Let $G = (N, T, S, P)$ be a type-0 Chomsky grammar in the Kuroda normal form, and let B be a new symbol. Assume that $N \cup T \cup \{B\} = \{\alpha_1, \ldots, \alpha_n\}$ and that P contains m rules, $u_i \to v_i, 1 \leq i \leq m$. Consider also the rules $u_{m+j} \to v_{m+j}, 1 \leq j \leq n$, for $u_{m+j} = v_{m+j} = \alpha_j$. We denote by P_1 the set of context-free rules, and with P_2 the rest of the rules.

We construct the splicing P system :

$$\begin{aligned}
\Pi &= (V, T, \mu, L_1, L_2, L_3, R_1, R_2, R_3), \\
V &= N \cup T \cup \{B, X, Y, Y', Z_X, Z_Y, Z_{Y'}, Z_\lambda, Z'_\lambda\} \cup \{X_i, Z_{X_i} \mid 0 \leq i \leq n + m\} \\
&\cup \{Y_i, Z_{Y_i}, Z_i, X'_i, Z_{X'_i} \mid 1 \leq i \leq m + n\},
\end{aligned}$$

$$
\begin{aligned}
\mu &= [_1[_2[_3]_3]_2]_1,\\
L_1 &= \{XBSY, Z_\lambda, Z'_\lambda\} \cup \{X_iZ_{X_i} \mid 0 \le i \le n+m\} \cup \{X'_iZ_{X'_i} \mid 1 \le i \le n+m\},\\
R_1 &= \{(Xu_i\#\$\#Z_{X_i}; out, in), \mid u_i \to v_i \in P_1\}\\
&\cup \{(XC\#\$\#Z_{X'_i}; out, here), (X'_iD\#\$\#Z_{X_i}; out, in) \mid u_i = CD \to v_i \in P_2\}\\
&\cup \{(X_i\#\$\#Z_{X_{i-1}}; out, in) \mid 1 \le i \le n+m\}\\
&\cup \{(XB\#\$\#Z_\lambda; out, here),\ (Z'_\lambda\#\$\#Y; out, out)\},\\
L_2 &= \{Z_YY, Z_{Y'}Y'\} \cup \{Z_iv_iY_i \mid 1 \le i \le m+n\} \cup \{Z_{Y_i}Y_i \mid 1 \le i \le m+n-1\},\\
R_2 &= \{(Z_i\#\$\#Y; in, out) \mid 1 \le i \le n+m\}\\
&\cup \{(Z_{Y_{i-1}}\#\$\#Y_i; in, out) \mid 2 \le i \le n+m\}\\
&\cup \{(Z_{Y'}\#\$\#Y_1; in, in),\ (Z_Y\#\$\#Y'; in, out)\},\\
L_3 &= \{XZ_X\},\\
R_3 &= \{(X_0\#\$\#Z_X; out, out)\}.
\end{aligned}
$$

Again we obtain $L(\Pi) = RE$, and it is clear that the diameter of the system is $(2, 0, 0, 1)$. □

Lemma 4. $SPL(i/o, 3, 3, (1, 2, 0, 1)) = SPL(i/o, 3, 3, (0, 1, 1, 2)) = RE$.

Proof. Again we will prove only one equality, $SPL(i/o, 3, 3, 1201) = RE$, the second one following from Lemma 1. We will give again only the construction of the splicing P system, because the proof idea (and the construction) follows closely the previous proofs.

With the notations from the previous proofs we construct the following system:

$$
\begin{aligned}
\Pi &= (V, T, \mu, L_1, L_2, L_3, R_1, R_2, R_3),\\
V &= N \cup T \cup \{B, X, X', Y, Z_X, Z_{X'}, Z_Y, Z_\lambda, Z'_\lambda, Z_T\} \cup \{Y_i, Z_{Y_i} \mid 0 \le i \le n+m\}\\
&\cup \{X_i, Z_{X_i}, Z_i, Y'_i, Z_{Y'_i} \mid 1 \le i \le m+n\},\\
\mu &= [_1[_2[_3]_3]_2]_1,\\
L_1 &= \{XSBY, Z_\lambda, Z'_\lambda, Z_T\} \cup \{Z_{Y_i}Y_i \mid 0 \le i \le n+m\} \cup \{Z_{Y'_i}Y'_i \mid 1 \le i \le n+m\},\\
R_1 &= \{(\#u_iY\$\#Y_i; in, out), \mid u_i \to v_i \in P_1\}\\
&\cup \{(C\#DY\$\#Y'_i; here, out), (\#CY'_i\$\#Y_i; in, out) \mid u_i = CD \to v_i \in P_2\}\\
&\cup \{(Z_{Y_{i-1}}\#Y_{i-1}\$\#Y_i; out, in) \mid 1 \le i \le n+m\}\\
&\cup \{(\#BY\$\#Z_T; here, out),\ (Z_\lambda\#\$\#Z_T; out, here),\ (X\#\$\#Z'_\lambda; out, out)\},\\
L_2 &= \{XZ_X, X'Z_{X'}\} \cup \{X_iv_iZ_i \mid 1 \le i \le m+n\} \cup \{X_iZ_{X_i} \mid 1 \le i \le m+n-1\},\\
R_2 &= \{(X\#\$\#Z_i; in, out) \mid 1 \le i \le n+m\}\\
&\cup \{(X_i\#\$\#Z_{X_{i-1}}; in, out) \mid 2 \le i \le n+m\}\\
&\cup \{(X_1\#\$\#Z_{X'}; in, in),\ (X'\#\$\#Z_X; in, out)\},\\
L_3 &= \{Z_YY\},\\
R_3 &= \{(Z_Y\#\$\#Y_0; out, out)\}.
\end{aligned}
$$

It is easy to see that the diameter of this system is $(1, 2, 0, 1)$, and that $L(\Pi) = L(G)$. □

In a similar fashion as we proceeded in the case of Lemma 2 we can now get a new result using the construct in Lemma 4, but rotating the strings in the converse order:

Lemma 5. $SPL(i/o, 3, 3, (1, 0, 2, 1)) = SPL(i/o, 3, 3, (2, 1, 1, 0)) = RE$.

Synthesizing these results, we obtain the following characterizations of RE.

Theorem 2. *$RE = SPL(i/o, 3, 3, (n_1, n_2, n_3, n_4))$, for all (n_1, n_2, n_3, n_4) componentwise greater than or equal to any of the following four-tuples:* $(0, 2, 1, 0)$, $(1, 0, 0, 2)$, $(2, 0, 0, 1)$, $(0, 1, 2, 0)$, $(1, 2, 0, 1)$, $(0, 1, 1, 2)$, $(2, 1, 1, 0)$, $(1, 0, 2, 1)$.

4 Final Remarks

Several problems remain to be further investigated. For instance, in Theorem 1 we have $RE = SPL(i/o, 2, 2, (2, 2, 2, 2))$. Can this result be improved, by using systems of a diameter smaller than $(2, 2, 2, 2)$? Are the previous results optimal? What about the case of P systems with target indications of the form in_j? (The target feature is stronger than the in/out indication; can the previous result be improved in this case?)

Another research direction is to consider other classes of languages. For example, what is the diameter needed to generate the regular (or the context-free) languages?

References

[1] K. Culik II, T. Harju, Splicing semigroups of dominoes and DNA, *Discrete Appl. Math.*, **31** (1991), 261–277.

[2] J. Dassow, Gh. Păun, On the power of membrane computing, *J. Univ. Computer Sci.*, **5**, 2 (1999), 33–49 (www.iicm.edu/jucs).

[3] T. Head, Formal language theory and DNA: an analysis of the generative capacity of specific recombinant behaviors, *Bull. Math. Biology*, **49** (1987), 737–759.

[4] A. Păun, Controlled H systems of small radius, *Fundamenta Informaticae*, **31**, 2 (1997), 185 – 193.

[5] A. Păun, M. Păun, Controlled and distributed H systems of a small diameter. In: Gh. Păun, ed., *Computing with Bio-Molecules; Theory and Experiments*, Springer-Verlag, Singapore, 1998.

[6] Gh. Păun, Regular extended H systems are computationally universal, *J. Automata, Languages, Combinatorics*, **1**, 1 (1996), 27–36.

[7] Gh. Păun, Computing with membranes, *Journal of Computer and System Sciences*, to appear (see also *TUCS Research Report* No. 208, November 1998, www.tucs.fi).

[8] Gh. Păun, Computing with membranes. An introduction, *Bulletin of the EATCS*, **67** (Febr. 1999), 139–152.

[9] Gh. Păun, G. Rozenberg, A. Salomaa, *DNA Computing. New Computing Paradigms*, Springer-Verlag, Heidelberg, 1998.

[10] Gh. Păun, G. Rozenberg, A. Salomaa, Membrane computing with external output, submitted, 1999 (see also *TUCS Research Report* No. 218, December 1998, www.tucs.fi).

[11] Gh. Păun, T. Yokomori, *Membrane Computing Based on Splicing, Preliminary Proc. of Fifth Intern. Meeting on DNA Based Computers* (E. Winfree, D. Gifford, eds.), MIT, June 1999, 213–227.

[12] Gh. Păun, S. Yu, On synchronization in P systems, *Fundamenta Informaticae*, **38**, 4 (1999), 397–410

[13] D. Pixton, Regularity of splicing languages, *Discrete Appl. Math.*, **69** (1996), 101–124.