ROMANIAN JOURNAL OF INFORMATION SCIENCE AND TECHNOLOGY
Volume 19, Number 4, 2016, 369–384

Locally Evolving Splicing Systems

Kalpana Mahalingam, Prithwineel Paul

Department of Mathematics,
Indian Institute of Technology, Madras, Chennai-36, India
Email: kmahalingam@iitm.ac.in, prithwineelpaul@gmail.com

Abstract. An extended H system with locally evolving rules is a model of splicing computation with a special feature that the splicing rules evolve at each step. It was originally proved by Paun et al. that all RE languages can be generated by such systems with a finite set of rules of radius at most 4 and with contexts of length at most 7 for the insertion-deletion rules by which the splicing rules evolve. We improve this result by reducing the length of the contexts to at most 2 and we also show that such systems with radius at most 2 and contexts of length at most 3 can generate all RE languages. In both cases, the insertion-deletion rules will insert or delete only one symbol. We also show that if the inserted strings are of length at most 2 and the deleted strings are of length 1, then the system can generate all RE languages with splicing rules of radius 3 and insertion-deletion rules with contexts of length at most 2. Finally, we show that the length of contexts and of the strings inserted/deleted can be reduced to 1 if the rules are applied in a matrix controlled manner.

Keywords: Extended H system, splicing, evolving rules, recursively enumerable.

1 Introduction

The splicing operation is a formal model of the recombinant behaviour of the DNA molecules in the presence of restriction enzymes. A splicing system, also called H system, consists of an alphabet, a set of splicing rules modeling the role of enzymes and a set of axioms. Several variants have been discussed, depending on the nature of the axioms, the set of rules etc. A detailed study can be found in [1, 3, 9]. Of course, it is desirable to have a finite set of rules and a finite set of axioms. Some important variants can be found in [4, 5, 7, 10, 12, 13, 15].

Splicing systems with a finite set of axioms and rules can only generate regular languages [2, 11]. Different controlled versions of finite splicing systems are able of generating all recursively enumerable languages [1, 9].

One such version is that of restricted locally evolving splicing systems, introduced in [8]. Locally evolving splicing systems are extended H systems where the splicing rules constantly evolve. The entire set of splicing rules is changed from one step to another by means of insertion/deletion/substitution operations with respect to given contexts. These systems are shown to generate all recursively enumerable languages. In [6], the authors have defined a splicing system that is "non-reflexively evolving" and proved that such systems are computationally complete.

In this paper, we investigate locally evolving splicing systems as introduced in [8], where it was shown that these systems can generate all RE languages when the radius is 4 and the length of the context of insertion-deletion rules is at most 7. It was conjectured that the radius as well as the length of the context can be reduced to 2. In this paper we provide characterizations of RE languages by such systems in two different improved versions: the radius at most 2 and the length of the contexts at most 3, respectively when the radius is at most 4 and the length of the context is at most 2. We also show that instead of using point mutation rules if we use insertion-deletion rules which can insert two symbols and delete one symbol, then the radius can be decreased to 3 and the length of the contexts of insertion-deletion rules can be at most 2. Finally, we show that, if we apply the rules in a matrix controlled way, then the length of the contexts as well as the inserted/deleted strings is 1 while the radius of the splicing rules is 3.

This paper is organized as follows: Section 2 recalls the necessary definitions. Section 3 illustrates the characterizations of RE languages with these systems with reduced radius and contexts of the rewriting rules. We end the section with a characterization of RE languages where the insertion-deletion rules are applied in a matrix controlled way. Section 4 contains some concluding remarks.

2 Definitions

The reader is referred to [14] for basic elements of formal language theory. We use the convention that V is a finite alphabet, λ denotes the empty string, V^* represents the set of all strings over V and $V^+ = V^* \setminus \{\lambda\}$. We recall the following from [9]. Let $\#, \$$ be two symbols not in V. A splicing rule over V is a string $u_1\#u_2\$u_3\#u_4$, where $u_i \in V^*$ for $1 \leq i \leq 4$. For a splicing rule $r = u_1\#u_2\$u_3\#u_4$ and for $x, y, w, z \in V^*$ we write, $(x, y) \vDash_r (w, z)$ iff $x = x_1u_1u_2x_2$, $y = y_1u_3u_4y_2$, $w = x_1u_1u_4y_2$, $z = y_1u_3u_2x_2$ for $x_1, x_2, y_1, y_2 \in V^*$.

An extended H system with locally evolving splicing rules is a construct

$$\gamma = (V, T, A_0, A_c, E, C_0, P)$$

where

1. V is a finite alphabet,
2. $T \subseteq V$ is the terminal alphabet,
3. A_0 is a finite set of strings from V^*,
4. $A_c \subseteq V^*$ is the finite set of current axioms,
5. E is an alphabet such that $E \cap V \neq \emptyset$,
6. C_0 is an initial sequence of splicing rules, $C_0 = (r_1, r_2, \cdots, r_k)$, $r_i \in E^*\#E^*\$E^*\#E^*$, $1 \leq i \leq k$,
7. P is a finite set of rewriting rules of the form $(u, \alpha/\beta, v)$ with $u, v \in (E \cup \{\#, \$\})^*$, and $\alpha, \beta \in E \cup \{\lambda\}$, $\alpha \neq \beta$.

The rules in P are called point mutation rules if the rules are of the form $\alpha \in E, \beta = \lambda$ (insertion rules) and $\alpha = \lambda, \beta \in E$ (deletion rules). The splicing rules are constructed from the template splicing rules in C_0 using the point mutation rules. The strings inside the system are modified by the splicing rules constructed as above. The splicing rules, if applicable, must be applied. Initially, strings from A_0 and A_c are spliced. Later, the strings produced inside the system are spliced with the strings in A_c using the rules constructed by applying the insertion-deletion rules to the current set of splicing rule.

The notation $\Longrightarrow_P$ denotes the usual derivation relation with respect to rules in P. For a splicing rule $r \in E^*\#E^*\$E^*\#E^*$, we define

$$P(r) = \{r' \mid r \Longrightarrow_P r'\}.$$

We extend the relation $\Longrightarrow_P$ to k-tuples of splicing rules by

$$(r_1, r_2, \cdots, r_k) \Longrightarrow_P (r_1', r_2', \cdots, r_k') \text{ iff } r_j' \in P(r_j), 1 \le j \le k.$$

Starting from C_0, at the time $i \ge 1$ we can obtain in this fashion a sequence $C_i = (r_{i,1}, \cdots, r_{i,k})$ to which the following sequence of splicing rules are associated.

$$R_i = \{r \mid r = r_{i,j} \text{ for some } 1 \le j \le k\}.$$

The set R_i, contains exactly one descendant of every rule in C_0; out of the possible variants which can be obtained due to the possible non-determinism of using the rules in P, only one is actually chosen. The sets A_i for $i \ge 0$ are defined as follows. The initial set A_0 is given. For $x \in V^*$ and a given set R of splicing rules and for $i \ge 0$, define

$$\delta_i(x, R) = \begin{cases} 1 & \text{if } B \\ 0 & \text{otherwise} \end{cases}$$

where B is the condition that "there exists $y \in A_i \cup A_c$ and $r \in R$ such that $(x, y) \vDash_r (w, z)$ or $(y, x) \vDash_r (w, z)$, for some $w, z \in V^*$".

Moreover, $R_i(A_i) = \{w \in V^* \mid (x, y) \vDash_r (w, z) \text{ or } (x, y) \vDash_r (z, w) \text{ for } r \in R_i,$ $x, y \in A_i \cup A_c, \{x, y\} \cap A_i \neq \emptyset\}$, $i \ge 0$. Then, $A_i = \{x \in A_{i-1} \mid \delta_{i-1}(x, R_{i-1}) = 0\} \cup R_{i-1}(A_{i-1})$, $i \ge 1$ and the language generated by γ is defined by

$$L(\gamma) = (\textstyle\bigcup_{i \ge 0} A_i) \cap T^*.$$

An extended H system with locally evolving splicing rules $\gamma = (V, T, A_0, A_c, E, C_0, P)$ is called restricted if $card(A_0) = 1, card(C_0) = 1$. The system γ is said to be of radius at most m if for each $i \ge 1$ if $R_i = u_1\#u_2\$u_3\#u_4$, then $|u_j| \le m$, $1 \le j \le 4$. Note that in a restricted locally evolving system we have exactly one splicing rule at each time. The family of languages generated by the restricted locally evolving H systems is denoted as $EH_{rle}(m, c, id)$, where m is the maximal radius of the splicing rules, c is the maximal length of the contexts in the insertion/deletion rules, and id is the maximal length of the string inserted/deleted. In [8], the notation $REHE([m])$ and in [9] the notation $EH_2(FIN, rle([m]))$ have been used to denote such systems. For more clarifications, the reader is referred to [8,9].

3 Results

It was shown in [8] that locally evolving splicing systems with radius 4 and the length of the context of insertion-deletion rules is less than 7 can generate all RE languages. It was conjectured that the radius as well as the length of the context can be reduced to 2. In this section we provide two results which improve the result in [8].

In the next theorem, we prove that the family of languages generated by restricted locally evolving H systems with splicing rules of radius ≤ 2 is equal to the family of recursively enumerable languages when the insertion-deletion rules used are point mutation rules with contexts of length ≤ 3.

Theorem 3.1. $EH_{rle}(2, 3, 1) = RE$.

Proof: The inclusion $EH_{rle}(2, 3, 1) \subseteq RE$ can be proved in a straightforward but cumbersome way, or we can invoke for it the Church-Turing thesis. So we only prove the other inclusion. Let $G = (N, T, P_0, S)$ be a grammar in Kuroda normal form. The rules in G are of the form $A \to BC, AB \to CD, A \to a, A \to \lambda$ where $A, B, C, D \in N$ and $a \in T$. The rules of P_0 are labeled in a one-to-one manner. We construct a restricted extended H system $\gamma = (V, T, A_0, A_c, E, C_0, P)$ with locally evolving splicing rules such that $L(G) = L(\gamma)$ as follows:

1. $V = N \cup T \cup \{X, Y, Z, Y_r^1, Y_r^2, Z_r, B_0\}$,
2. $A_0 = \{XB_0SY\}$,
3. $A_c = \{ZY\} \cup \{X\alpha Z \mid \alpha \in N \cup T \cup \{B_0\}\} \cup \{ZY_r^1, ZCY_r^2, Z_rDY \mid r : AB \to CD\}$ $\cup \{ZY_r^1, ZBY_r^2, Z_rCY \mid r : A \to BC\} \cup \{Z_raY \mid r : A \to a\} \cup \{Z_rY \mid r : A \to \lambda\}$,
4. $E = N \cup T \cup \{X, Y, Z, Y_r^1, Y_r^2, Z_r\} \cup \{[r, 1], [r, 2], [r, 3], \ldots, [r, 13] \mid r \in P_0\}$ $\cup \{d_1, d_2, d_3, d_4\} \cup \{e_1, e_2\} \cup \{f_1, f_2, \ldots, f_8\} \cup \{g_1, g_2, g_3\} \cup \{h_1, h_2, h_3, h_4\}$,
5. $C_0 = (\#c_1\$Z\#Y)$.

The rule of C_0 is called the template splicing rule and it evolves using the insertion and deletion rules (i.e., point mutation rules) present in P. Initially, only the splicing rule $\#c_1\$Z\#Y$ is present in the system. No string of A_0 and A_c of the system γ can be spliced using this rule. Using the point mutation rules in P, the rule of C_0 is modified so that the obtained rules can simulate the rules of P_0. In the constructions below, the point mutation rules are placed on the left-hand side and the splicing rules obtained after applying the point mutation rules to the splicing rule obtained in the previous step are placed in the corresponding right-hand side of the table. The proof is based on the rotate-and-simulate technique introduced in [7].

The splicing rules $(A\#BY\$Z\#Y_r^1), (\#AY_r^1\$Z\#CY_r^2), (C\#Y_r^2\$Z_r\#DY)$ (corresponding to rules $SA1-16$, $SA2-18$ and $SA3-20$ respectively) if applied sequentially can simulate the rule $r : AB \to CD$ in the following manner:

$$(XwA \mid BY, Z \mid Y_r^1) \vdash (XwAY_r^1, ZBY); \text{(Rule } SA1-16)$$
$$(Xw \mid AY_r^1, Z \mid CY_r^2) \vdash (XwCY_r^2, ZAY_r^1); \text{(Rule } SA2-18)$$
$$(XwC \mid Y_r^2, Z_r \mid DY) \vdash (XwCDY, Z_rY_r^2); \text{(Rule } SA3-20).$$

The above mentioned splicing rules are constructed in $SA1, SA2, SA3$. The rule $r : AB \to CD$, can be simulated by the application of the splicing rules constructed in $(SA1), (SA2), (SA3)$.
$(SA1)$: Starting with the template rule, we construct the rule $A\#BY\$Z\#Y_r^1$.

0.	—	$\#c_1\$Z\#Y$
1.	$(\#c_1, \lambda/[r,1], \$)$	$\#c_1[r,1]\$Z\#Y$
2.	$(\#, c_1/\lambda, [r,1])$	$\#[r,1]\$Z\#Y$
3.	$(\lambda, \lambda/A, \#[r,1])$	$A\#[r,1]\$Z\#Y$
4.	$(A\#, \lambda/B, [r,1])$	$A\#B[r,1]\$Z\#Y$
5.	$(B[r,1]\$, \lambda/[r,2], Z)$	$A\#B[r,1]\$[r,2]Z\#Y$
6.	$(B, [r,1]/\lambda, \$[r,2])$	$A\#B\$[r,2]Z\#Y$
7.	$(B, \lambda/Y, \$[r,2])$	$A\#BY\$[r,2]Z\#Y$
8.	$(Y\$[r,2], Z/\lambda, \#)$	$A\#BY\$[r,2]\#Y$
9.	$(Y\$[r,2], \lambda/[r,3], \#Y)$	$A\#BY\$[r,2][r,3]\#Y$
10.	$(\$, [r,2]/\lambda, [r,3]\#)$	$A\#BY\$[r,3]\#Y$
11.	$([r,3]\#Y, \lambda/[r,4], \lambda)$	$A\#BY\$[r,3]\#Y[r,4]$
12.	$(\$, [r,3]/\lambda, \#Y[r,4])$	$A\#BY\$\#Y[r,4]$
13.	$(\$\#, Y/\lambda, [r,4])$	$A\#BY\$\#[r,4]$
14.	$(\$, \lambda/Z, \#[r,4])$	$A\#BY\$Z\#[r,4]$
15.	$(\$Z\#, \lambda/Y_r^1, [r,4])$	$A\#BY\$Z\#Y_r^1[r,4]$
16.	$(Z\#Y_r^1, [r,4]/\lambda, \lambda)$	$A\#BY\$Z\#Y_r^1$

Only the last splicing rule $A\#BY\$Z\#Y_r^1$ can be applied to the strings from A_0 and A_c. Other rules are not applicable, since the symbols c_1, $[r,1]$, $[r,2]$, $[r,3]$ and $[r,4]$ are present in the splicing rules. After the application of the rule $A\#BY\$Z\#Y_r^1$, the strings $XwAY_r^1$ and ZBY are produced. The first string again can be spliced with the string ZY_r^2 using the splicing rule $\#AY_r^1\$Z\#Y_r^2$, but the string ZBY cannot generate terminal strings.

$(SA2)$: The splicing rule $A\#BY\$Z\#Y_r^1$ is now modified into the splicing rule $\#AY_r^1\$Z\#Y_r^2$ as follows:

0.	—	$A\#BY\$Z\#Y_r^1$
1.	$(\lambda, \lambda/[r,5], A\#)$	$[r,5]A\#BY\$Z\#Y_r^1$
2.	$([r,5], A/\lambda, \#B)$	$[r,5]\#BY\$Z\#Y_r^1$
3.	$([r,5]\#, B/\lambda, Y)$	$[r,5]\#Y\$Z\#Y_r^1$
4.	$([r,5]\#, Y/\lambda, \$)$	$[r,5]\#\$Z\#Y_r^1$
5.	$([r,5]\#, \lambda/A, \$)$	$[r,5]\#A\$Z\#Y_r^1$
6.	$([r,5]\#A, \lambda/[r,6], \$)$	$[r,5]\#A[r,6]\$Z\#Y_r^1$
7.	$(\lambda, [r,5]/\lambda, \#A[r,6])$	$\#A[r,6]\$Z\#Y_r^1$
8.	$([r,6]\$, \lambda/[r,7], Z)$	$\#A[r,6]\$[r,7]Z\#Y_r^1$
9.	$(A, [r,6]/\lambda, \$[r,7])$	$\#A\$[r,7]Z\#Y_r^1$
10.	$(\#A, \lambda/Y_r^1, \$[r,7])$	$\#AY_r^1\$[r,7]Z\#Y_r^1$
11.	$(Y_r^1\$[r,7], Z/\lambda, \#)$	$\#AY_r^1\$[r,7]\#Y_r^1$
12.	$([r,7]\#Y_r^1, \lambda/[r,8], \lambda)$	$\#AY_r^1\$[r,7]\#Y_r^1[r,8]$
13.	$(\$, [r,7]/\lambda, \#Y_r^1[r,8])$	$\#AY_r^1\$\#Y_r^1[r,8]$
14.	$(\$, \lambda/Z, \#Y_r^1[r,8])$	$\#AY_r^1\$Z\#Y_r^1[r,8]$
15.	$(Z\#, Y_r^1/\lambda, [r,8])$	$\#AY_r^1\$Z\#[r,8]$
16.	$(\$Z\#, \lambda/Y_r^2, [r,8])$	$\#AY_r^1\$Z\#Y_r^2[r,8]$
17.	$(Z\#, \lambda/C, Y_r^2[r,8])$	$\#AY_r^1\$Z\#CY_r^2[r,8]$
18.	$(\#CY_r^2, [r,8]/\lambda, \lambda)$	$\#AY_r^1\$Z\#CY_r^2$

After splicing the strings $XwAY_r^1$ and ZCY_r^2 using the rule $\#AY_r^1\$Z\#CY_r^2$, the strings $XwCY_r^2$ and ZAY_r^1 are produced.

$(SA3)$: Now, $\#AY_r^1\$Z\#CY_r^2$ is modified into $C\#Y_r^2\$Z_r\#DY$:

0.	—	$\#AY_r^1\$Z\#CY_r^2$
1.	$(\lambda, \lambda/[r,9], \#A)$	$[r,9]\#AY_r^1\$Z\#CY_r^2$
2.	$([r,9]\#, A/\lambda, Y_r^1)$	$[r,9]\#Y_r^1\$Z\#CY_r^2$
3.	$([r,9]\#, \lambda/[r,10], Y_r^1)$	$[r,9]\#[r,10]Y_r^1\$Z\#Y_r^2$
4.	$([r,9], \lambda/C, \#[r,10])$	$[r,9]C\#[r,10]Y_r^1\$Z\#CY_r^2$
5.	$(\lambda, [r,9]/\lambda, C\#[r,10])$	$C\#[r,10]Y_r^1\$Z\#CY_r^2$
6.	$([r,10], Y_r^1/\lambda, \$)$	$C\#[r,10]\$Z\#CY_r^2$
7.	$([r,10], \lambda/Y_r^2, \$)$	$C\#[r,10]Y_r^2\$Z\#CY_r^2$
8.	$([r,10]Y_r^2\$, \lambda/[r,11], Z)$	$C\#[r,10]Y_r^2\$[r,11]Z\#CY_r^2$
9.	$(\#, [r,10]/\lambda, Y_r^2\$[r,11])$	$C\#Y_r^2\$[r,11]Z\#CY_r^2$
10.	$([r,11], Z/\lambda, \#C)$	$C\#Y_r^2\$[r,11]\#CY_r^2$
11.	$([r,11]\#, C/\lambda, Y_r^2)$	$C\#Y_r^2\$[r,11]\#Y_r^2$
12.	$([r,11], \lambda/Z_r, \#Y_r^2)$	$C\#Y_r^2\$[r,11]Z_r\#Y_r^2$
13.	$([r,11]Z_r\#, \lambda/[r,12], Y_r^2)$	$C\#Y_r^2\$[r,11]Z_r\#[r,12]Y_r^2$
14.	$(\$, [r,11]/\lambda, Z_r\#[r,12])$	$C\#Y_r^2\$Z_r\#[r,12]Y_r^2$
15.	$(\$Z_r, \lambda/[r,13], \#[r,12])$	$C\#Y_r^2\$Z_r[r,13]\#[r,12]Y_r^2$
16.	$([r,13]\#, [r,12]/\lambda, Y_r^2)$	$C\#Y_r^2\$Z_r[r,13]\#Y_r^2$
17.	$([r,13]\#, Y_r^2/\lambda, \lambda)$	$C\#Y_r^2\$Z_r[r,13]\#$
18.	$([r,13]\#, \lambda/D, \lambda)$	$C\#Y_r^2\$Z_r[r,13]\#D$
19.	$([r,13]\#D, \lambda/Y, \lambda)$	$C\#Y_r^2\$Z_r[r,13]\#DY$
20.	$(Z_r, [r,13]/\lambda, \#DY)$	$C\#Y_r^2\$Z_r\#DY$

Now the string $XwCY_r^2$ can be spliced further with Z_rDY using the rule $SA3-20$ to obtain $XwCDY$ and $Z_rY_r^2$. Note that $Z_rY_r^2$ cannot be used further to produce terminal strings.

Since there exists only one splicing rule at any time in the system, after completely simulating one rule in P_0, either the system simulates another rule of P_0 or will terminate the computation. The simulation of rules in P_0 is done at the end of the strings, hence the strings should be circularly rotated. But, to proceed further the system must go back to the template splicing rule. Hence, the splicing rule $C\#Y_r^2\$Z_r\#DY$ is transformed into $(\#c_1\$Z\#Y)$.

$(T1)$: In the next step the rule $C\#Y_r^2\$Z_r\#DY$ is transformed into the template rule in C_0.

0.	—	$C\#Y_r^2\$Z_r\#DY$
1.	$(\lambda, \lambda/d_1, C\#Y_r^2)$	$d_1C\#Y_r^2\$Z_r\#DY$
2.	$(d_1C\#, Y_r^2/\lambda, \$)$	$d_1C\#\$Z_r\#DY$
3.	$(d_1, C/\lambda, \#\$)$	$d_1\#\$Z_r\#DY$
4.	$(d_1\#\$, Z_r/\lambda, \#DY)$	$d_1\#\$\#DY$
5.	$(d_1\#, \lambda/c_1, \$\#)$	$d_1\#c_1\$\#DY$
6.	$(d_1\#c_1, \lambda/d_2, \$\#)$	$d_1\#c_1d_2\$\#DY$
7.	$(\lambda, d_1/\lambda, \#c_1d_2)$	$\#c_1d_2\$\#DY$
8.	$(c_1d_2\$, \lambda/d_3, \#D)$	$\#c_1d_2\$d_3\#DY$
9.	$(c_1, d_2/\lambda, \$d_3)$	$\#c_1\$d_3\#DY$
10.	$(d_3\#, D/\lambda, Y)$	$\#c_1\$d_3\#Y$
11.	$(\$d_3, \lambda/Z, \#Y)$	$\#c_1\$d_3Z\#Y$
12.	$(\$, d_3/\lambda, Z\#Y)$	$\#c_1\$Z\#Y$

The construction of splicing rule to simulate the rule $r : A \to BC$ is similar and hence we omit it.

$(SB1)$: Construction of splicing rules to simulate the rule $r : A \to a$:

0.	—	$\#c_1\$Z\#Y$
1.	$(\#, \lambda/[r,1], c_1)$	$\#[r,1]c_1\$Z\#Y$
2.	$(\#[r,1], c_1/\lambda, \$)$	$\#[r,1]\$Z\#Y$
3.	$(\#[r,1]\$, \lambda/[r,2], Z\#)$	$\#[r,1]\$[r,2]Z\#Y$
4.	$(\#, [r,1]/\lambda, \$[r,2])$	$\#\$[r,2]Z\#Y$
5.	$(\#, \lambda/A, \$[r,2])$	$\#A\$[r,2]Z\#Y$
6.	$(\#A, \lambda/Y, \$[r,2])$	$\#AY\$[r,2]Z\#Y$
7.	$(Y\$[r,2], Z/\lambda, \#Y)$	$\#AY\$[r,2]\#Y$
8.	$(Y\$[r,2], \lambda/[r,3], \#Y)$	$\#AY\$[r,2][r,3]\#Y$
9.	$([r,2][r,3]\#, \lambda/a, Y)$	$\#AY\$[r,2][r,3]\#aY$
10.	$(\$, [r,2]/\lambda, [r,3]\#a)$	$\#AY\$[r,3]\#aY$
11.	$(\$, \lambda/Z_r, [r,3]\#a)$	$\#AY\$Z_r[r,3]\#aY$
12.	$(\$Z_r, [r,3]/\lambda, \#aY)$	$\#AY\$Z_r\#aY$

Similarly, we can construct the splicing rule $\#AY\$Z_r\#Y$ which simulates the rule $r : A \to \lambda$.

$(T2)$: Now the rule $\#AY\$Z\#aY$ will be transformed to the template rule $\#c_1\$Z_r\#Y$.

0.	—	$\#AY\$Z\#aY$
1.	$(\lambda, \lambda/d_1, \#AY)$	$d_1\#AY\$Z\#aY$
2.	$(d_1\#, A/\lambda, Y\$)$	$d_1\#Y\$Z\#aY$
3.	$(d_1\#, Y/\lambda, \$)$	$d_1\#\$Z\#aY$
4.	$(d_1\#, \lambda/d_2, \$Z)$	$d_1\#d_2\$Z\#aY$
5.	$(d_1\#, \lambda/c_1, d_2\$)$	$d_1\#c_1d_2\$Z\#aY$
6.	$(\lambda, d_1/\lambda, \#c_1d_2)$	$\#c_1d_2\$Z\#aY$
7.	$(c_1d_2\$, \lambda/d_3, Z)$	$\#c_1d_2\$d_3Z\#aY$
8.	$(\#c_1, d_2/\lambda, \$d_3)$	$\#c_1\$d_3Z\#aY$
9.	$(d_3Z\#, a/\lambda, Y)$	$\#c_1\$d_3Z\#Y$
10.	$(\$, d_3/\lambda, Z\#Y)$	$\#c_1\$Z\#Y$

We now construct splicing rules necessary for rotating the current string.

$(ROT1)$: 0.	—	$\#c_1\$Z\#Y$
1.	$(\lambda, d_4/\lambda, \#c_1)$	$d_4\#c_1\$Z\#Y$
2.	$(d_4\#, c_1/\lambda, \$Z)$	$d_4\#\$Z\#Y$
3.	$(d_4\#, \lambda/\alpha, \$Z)$	$d_4\#\alpha\$Z\#Y$
4.	$(d_4\#\alpha, \lambda/Y, \$Z)$	$d_4\#\alpha Y\$Z\#Y$
5.	$(\lambda, d_4/\lambda, \#\alpha Y)$	$\#\alpha Y\$Z\#Y$

In $(ROT2)$ the splicing rule $X\alpha\#Z\$X\#$ is constructed, which adds the symbol $\alpha \in N \cup T \cup \{B_0\}$ to the right of the marker X.

$(ROT2)$: 0.	—	$\#\alpha Y\$Z\#Y$
1.	$(\lambda,\lambda/f_1,\#\alpha Y)$	$f_1\#\alpha Y\$Z\#Y$
2.	$(f_1\#,\alpha/\lambda,Y)$	$f_1\#Y\$Z\#Y$
3.	$(f_1\#,Y/\lambda,\$)$	$f_1\#\$Z\#Y$
4.	$(f_1,\lambda/X,\#\$)$	$f_1X\#\$Z\#Y$
5.	$(f_1X\#,\lambda/f_2,\$Z)$	$f_1X\#f_2\$Z\#Y$
6.	$(\lambda,f_1/\lambda,X\#f_2)$	$X\#f_2\$Z\#Y$
7.	$(X,\lambda/\alpha,\#f_2)$	$X\alpha\#f_2\$Z\#Y$
8.	$(X\alpha\#,\lambda/Z,f_2)$	$X\alpha\#Zf_2\$Z\#Y$
9.	$(Zf_2\$,\lambda/f_3,Z\#Y)$	$X\alpha\#Zf_2\$f_3Z\#Y$
10.	$(Z,f_2/\lambda,\$f_3)$	$X\alpha\#Z\$f_3Z\#Y$
11.	$(Z\$f_3,Z/\lambda,\#Y)$	$X\alpha\#Z\$f_3\#Y$
12.	$(f_3\#Y,\lambda/f_4,\lambda)$	$X\alpha\#Z\$f_3\#Yf_4$
13.	$(\$,f_3/\lambda,\#Yf_4)$	$X\alpha\#Z\$\#Yf_4$
14.	$(\$\#,Y/\lambda,f_4)$	$X\alpha\#Z\$\#f_4$
15.	$(Z\$,\lambda/X,\#f_4)$	$X\alpha\#Z\$X\#f_4$
16.	$(X\#,f_4/\lambda,\lambda)$	$X\alpha\#Z\$X\#$

Using rules $ROT1-5$ and $ROT2-16$ the symbol $\alpha \in N \cup T \cup \{B_0\}$ adjacent to the marker Y in $Xw\alpha Y$ can be rotated to obtain $X\alpha wY$.

$$(Xw|\alpha Y, Z|Y) \vdash (XwY, Z\alpha Y).$$

The string XwY can be spliced further using $ROT2-16$ but $Z\alpha Y$ will never lead to terminal strings.

$$(X\alpha|Z, X|wY) \vdash (X\alpha wY, XZ).$$

After rotation is complete, the rule $X\alpha\#Z\$X\#$ is transformed into the rule $\#c_1\$Z\#Y$.

$(T3)$:

0.	—	$X\alpha\#Z\$X\#$
1.	$(\alpha\#Z,\lambda/f_5,\$X)$	$X\alpha\#Zf_5\$X\#$
2.	$(X\alpha\#,Z/\lambda,f_5)$	$X\alpha\#f_5\$X\#$
3.	$(X,\alpha/\lambda,\#f_5)$	$X\#f_5\$X\#$
4.	$(\lambda,\lambda/f_6,X\#f_5)$	$f_6X\#f_5\$X\#$
5.	$(f_6,X/\lambda,\#f_5)$	$f_6\#f_5\$X\#$
6.	$(f_6\#,\lambda/c_1,f_5)$	$f_6\#c_1f_5\$X\#$
7.	$(\lambda,f_6/\lambda,\#c_1f_5)$	$\#c_1f_5\$X\#$
8.	$(c_1f_5\$,\lambda/f_7,X\#)$	$\#c_1f_5\$f_7X\#$
9.	$(c_1,f_5/\lambda,\$f_7)$	$\#c_1\$f_7X\#$
10.	$(f_7X\#,\lambda/f_8,\lambda)$	$\#c_1\$f_7X\#f_8$
11.	$(f_7,X/\lambda,\#f_8)$	$\#c_1\$f_7\#f_8$
12.	$(\$,f_7/\lambda,\#f_8)$	$\#c_1\$\#f_8$
13.	$(\$,\lambda/Z,\#f_8)$	$\#c_1\$Z\#f_8$
14.	$(Z\#,\lambda/Y,f_8)$	$\#c_1\$Z\#Yf_8$
15.	$(Z\#Y,f_8/\lambda,\lambda)$	$\#c_1\$Z\#Y$

Then either the process can be repeated by constructing all the splicing rules that simulate rules in P_0 as well as the rules which helps with rotation or the following terminating rules can be constructed.

The terminating rules are constructed in $(TE1)$ and $(TE2)$. The rule $XB_0\#\$\#ZY$, constructed from the template splicing rule in $(TE1)$, removes XB_0 from the left hand side of the string XB_0wY and the rule constructed in $(TE2)$ removes the marker Y.
$(TE1)$:

0.	—	$\#c_1\$Z\#Y$
1.	$(\lambda, \lambda/g_1, \#c_1)$	$g_1\#c_1\$Z\#Y$
2.	$(g_1\#, c_1/\lambda, \$Z)$	$g_1\#\$Z\#Y$
3.	$(g_1, \lambda/X, \#\$)$	$g_1X\#\$Z\#Y$
4.	$(g_1X\#, \lambda/g_2, \$Z\#)$	$g_1X\#g_2\$Z\#Y$
5.	$(\lambda, g_1/\lambda, X\#g_2)$	$X\#g_2\$Z\#Y$
6.	$(X, \lambda/B_0, \#g_2\$)$	$XB_0\#g_2\$Z\#Y$
7.	$(g_2\$, \lambda/g_3, Z\#Y)$	$XB_0\#g_2\$g_3Z\#Y$
8.	$(XB_0\#, g_2/\lambda, \$g_3)$	$XB_0\#\$g_3Z\#Y$
9.	$(g_3, Z/\lambda, \#Y)$	$XB_0\#\$g_3\#Y$
10.	$(g_3\#, \lambda/Z, Y)$	$XB_0\#\$g_3\#ZY$
11.	$(\#\$, g_3/\lambda, \#ZY)$	$XB_0\#\$\#ZY$

The produced splicing rule is $XB_0\#\$\#ZY$. This rule is applicable to strings XB_0wY and ZY.

$$(XB_0 \mid wY, |ZY) \vdash (XB_0ZY, wY)$$

i.e., XB_0 is removed and wY is spliced further.

$(TE2)$: The rule $\#Y\$ZY\#$ is constructed from $XB_0\#\$\#ZY$. This rule removes the marker Y.

0.	—	$XB_0\#\$\#ZY$
1.	$(XB_0\#, \lambda/h_1, \$\#)$	$XB_0\#h_1\$\#ZY$
2.	$(X, B_0/\lambda, \#h_1)$	$X\#h_1\$\#ZY$
3.	$(\lambda, \lambda/h_2, X\#h_1)$	$h_2X\#h_1\$\#ZY$
4.	$(h_2, X/\lambda, \#h_1)$	$h_2\#h_1\$\#ZY$
5.	$(h_2\#, \lambda/Y, h_1)$	$h_2\#Yh_1\$\#ZY$
6.	$(\lambda, h_2/\lambda, \#Yh_1)$	$\#Yh_1\$\#ZY$
7.	$(Yh_1\$, \lambda/h_3, \#ZY)$	$\#Yh_1\$h_3\#ZY$
8.	$(\#Y, h_1/\lambda, \$h_3)$	$\#Y\$h_3\#ZY$
9.	$(h_3\#, Z/\lambda, Y)$	$\#Y\$h_3\#Y$
10.	$(h_3\#Y, \lambda/h_4, \lambda)$	$\#Y\$h_3\#Yh_4$
11.	$(h_3\#, Y/\lambda, h_4)$	$\#Y\$h_3\#h_4$
12.	$(\$, h_3/\lambda, \#h_4)$	$\#Y\$\#h_4$
13.	$(\$, \lambda/Z, \#h_4)$	$\#Y\$Z\#h_4$
14.	$(Z, \lambda/Y, \#h_4)$	$\#Y\$ZY\#h_4$
15.	$(ZY\#, h_4/\lambda, \lambda)$	$\#Y\$ZY\#$

The rule $ROT2 - 15$ can be applied to strings wY and ZY to obtain w and ZYY.

$$(w \mid Y, ZY \mid) \vdash (w, ZYY).$$

By splicing the string ZYY with strings from A_c, terminal strings can never be reached. If the string w is in T^* then $w \in L(G)$. Since w does not contain any markers, no rule is applicable to w even if it is not a terminal string. Also the wrong application of the insertion-deletion rules cannot generate splicing rules which lead to terminal strings. The computation stops after the construction of rule $TE2 - 15$ is complete, since the rule $\#Y\$ZY\#$ cannot be modified further. Thus we can conclude that $L(G) = L(\gamma)$. This implies $RE \subseteq EH_{rle2}(2, 3, 1)$. □

The next result is a variation of Theorem 3.1 in which we give a characterization of RE languages when the radius of the system is at most 4 and the length of the context of the insertion-deletion rules is at most 2 with point mutation rules. Thus we have the following.

Theorem 3.2. $RE = EH_{rle}(4, 2, 1)$.

Proof: We only prove the inclusion $RE \subseteq EH_{rle}(4, 2, 1)$. Let $L \in RE$ and $G = (N, T, P_0, S)$ be a type-0 grammar in Kuroda normal form generating L.

We construct a restricted extended H system with locally evolving splicing rules $\gamma = (V, T, A_0, A_c, E, C_0, P)$ where

1. $V = \{X, Y, Z, B_0\} \cup N \cup T$,
2. $A_0 = \{XB_0SY\}$,
3. $A_c = \{ZY\} \cup \{ZvY \mid u \to v \in P_0\} \cup \{X\alpha Z \mid \alpha \in N \cup T \cup \{B_0\}\}$,
4. $E = N \cup T \cup \{X, Y, Z, B_0, c_1, e_1, e_2, \ldots, e_6, d_2 \ldots, d_6, f_1, f_2, \ldots, f_5, g_1, g_2, \ldots, g_6, h_1, h_2, \ldots, h_6, k_1, k_2, \ldots, k_7\} \cup \{[r, 1], [r, 2], [r, 3], [r, 4], [r, 5], [r, 6], [r, 7], [r, 8] \mid r : u \to v \in P_0\}$,
5. $C_0 = (c_1\#Y\$Z\#)$.

We only construct the splicing rule which simulates the rule $r : AB \to CD$ and omit the constructions of rules that simulate $A \to BC$, $A \to a$ and $A \to \lambda$ as they are similar.

The rule $\#ABY\$Z\#CDY$ simulates the rule $r : AB \to CD \in P_0$. The following insertion-deletion rules with context ≤ 2 are constructed which transforms the splicing rule $c_1\#Y\$Z\#$ into $\#ABY\$Z\#CDY$.

$(SA1)$: 0.	——	$c_1\#Y\$Z\#$
1.	$(c_1\#, \lambda/[r, 1], Y)$	$c_1\#[r, 1]Y\$Z\#$
2.	$(\lambda, c_1/\lambda, \#[r, 1])$	$\#[r, 1]Y\$Z\#$
3.	$(\#, \lambda/A, [r, 1]Y)$	$\#A[r, 1]Y\$Z\#$
4.	$([r, 1]Y, \lambda/[r, 2], \$Z)$	$\#A[r, 1]Y[r, 2]\$Z\#$
5.	$(\#A, [r, 1]/\lambda, Y[r, 2])$	$\#AY[r, 2]\$Z\#$
6.	$(\#A, \lambda/B, Y[r, 2])$	$\#ABY[r, 2]\$Z\#$
7.	$([r, 2]\$, \lambda/[r, 3], Z\#)$	$\#ABY[r, 2]\$[r, 3]Z\#$
8.	$(Y, [r, 2]/\lambda, \$[r, 3])$	$\#ABY\$[r, 3]Z\#$
9.	$([r, 3]Z, \lambda/[r, 4], \#)$	$\#ABY\$[r, 3]Z[r, 4]\#$
10.	$(\$, [r, 3]/\lambda, Z[r, 4])$	$\#ABY\$Z[r, 4]\#$
11.	$([r, 4]\#, \lambda/C, \lambda)$	$\#ABY\$Z[r, 4]\#C$
12.	$([r, 4]\#, \lambda/[r, 5], C)$	$\#ABY\$Z[r, 4]\#[r, 5]C$
13.	$(Z, [r, 4]/\lambda, \#[r, 5])$	$\#ABY\$Z\#[r, 5]C$
14.	$([r, 5]C, \lambda/[r, 6], \lambda)$	$\#ABY\$Z\#[r, 5]C[r, 6]$
15.	$(Z\#, [r, 5]/\lambda, C[r, 6])$	$\#ABY\$Z\#C[r, 6]$
16.	$(\#C, \lambda/D, [r, 6])$	$\#ABY\$Z\#CD[r, 6]$
17.	$(D, \lambda/Y, [r, 6])$	$\#ABY\$Z\#CDY[r, 6]$
18.	$(DY, [r, 6]/\lambda, \lambda)$	$\#ABY\$Z\#CDY$

Thus the obtained splicing rules are of the form $\#uY\$Z\#vY$ which simulates the rules $r : u \to v \in P_0$. Therefore the strings $XwuY$ and ZvY can be spliced to obtain $XwvY$ and ZuY.

$$(Xw \mid uY, Z \mid vY) \vdash (XwvY, ZuY).$$

The first string $XwvY$ can be spliced further but ZuY will never lead to a terminal string.

$(T1)$: We now proceed further to construct the template splicing rule $c_1\#Y\$Z\#Y$ from the rule $SA1-18$.

0.	—	$\#ABY\$Z\#CDY$
1.	$(Z\#, \lambda/e_1, CD)$	$\#ABY\$Z\#e_1CDY$
2.	$(\#e_1, C/\lambda, DY)$	$\#ABY\$Z\#e_1DY$
3.	$(\#e_1, D/\lambda, Y)$	$\#ABY\$Z\#e_1Y$
4.	$(Z\#, \lambda/e_2, e_1Y)$	$\#ABY\$Z\#e_2e_1Y$
5.	$(\#e_2, e_1/\lambda, Y)$	$\#ABY\$Z\#e_2Y$
6.	$(\$Z, \lambda/e_3, \#e_2)$	$\#ABY\$Ze_3\#e_2Y$
7.	$(e_3\#, e_2/\lambda, Y)$	$\#ABY\$Ze_3\#Y$
8.	$(Y\$, \lambda/e_4, Ze_3)$	$\#ABY\$e_4Ze_3\#Y$
9.	$(e_4Z, e_3/\lambda, \#Y)$	$\#ABY\$e_4Z\#Y$
10.	$(Y, \lambda/e_5, \$e_4)$	$\#ABYe_5\$e_4Z\#Y$
11.	$(e_5\$, e_4/\lambda, Z\#)$	$\#ABYe_5\$Z\#Y$
12.	$(\#A, B/\lambda, Ye_5)$	$\#AYe_5\$Z\#Y$
13.	$(\#, A/\lambda, Ye_5)$	$\#Ye_5\$Z\#Y$
14.	$(\#, \lambda/e_6, Ye_5)$	$\#e_6Ye_5\$Z\#Y$
15.	$(e_6Y, e_5/\lambda, \$)$	$\#e_6Y\$Z\#Y$
16.	$(\lambda, \lambda/c_1, \#e_6)$	$c_1\#e_6Y\$Z\#Y$
17.	$(c_1\#, e_6/\lambda, Y)$	$c_1\#Y\$Z\#Y$

We now construct splicing rules which rotate the symbols between the markers X and Y of the string XwY where $w \in (N \cup T \cup \{B_0\})^*$.
At first, $\alpha \in N \cup T \cup \{B_0\}$ is cut from the right hand side of the string $Xw\alpha Y$. This operation can be done by application of the splicing rule $\#\alpha Y\$Z\#Y$ in $ROT1-15$.

$ROT1 : 0.$	—	$c_1\#Y\$Z\#$
1.	$(\lambda, \lambda/d_1, c_1\#)$	$d_1c_1\#Y\$Z\#$
2.	$(d_1, c_1/\lambda, \#Y)$	$d_1\#Y\$Z\#$
3.	$(d_1\#, \lambda/\alpha, Y)$	$d_1\#\alpha Y\$Z\#$
4.	$(d_1\#, \lambda/d_2, \alpha Y)$	$d_1\#d_2\alpha Y\$Z\#$
5.	$(\lambda, d_1/\lambda, \#d_2)$	$\#d_2\alpha Y\$Z\#$
6.	$(d_2\alpha, \lambda/d_3, Y)$	$\#d_2\alpha d_3Y\$Z\#$
7.	$(\#, d_2/\lambda, \alpha d_3)$	$\#\alpha d_3Y\$Z\#$
8.	$(d_3Y, \lambda/d_4, \$)$	$\#\alpha d_3Yd_4\$Z\#$
9.	$(\#\alpha, d_3/\lambda, Yd_4)$	$\#\alpha Yd_4\$Z\#$
10.	$(d_4\$, \lambda/d_5, Z)$	$\#\alpha Yd_4\$d_5Z\#$
11.	$(\alpha Y, d_4/\lambda, \$d_5)$	$\#\alpha Y\$d_5Z\#$
12.	$(d_5Z, \lambda/d_6, \#)$	$\#\alpha Y\$d_5Zd_6\#$
13.	$(Y\$, d_5/\lambda, Zd_6)$	$\#\alpha Y\$Zd_6\#$
14.	$(d_6\#, \lambda/Y, \lambda)$	$\#\alpha Y\$Zd_6\#Y$
15.	$(\$Z, d_6/\lambda, \#Y)$	$\#\alpha Y\$Z\#Y$

After application of the rule $\#\alpha Y\$Z\#Y$ to strings $Xw\alpha Y$ and ZY, the strings XwY and $Z\alpha Y$ are produced.

Using the rule $ROT2-22$ the string XwY is spliced with $X\alpha Z$ to obtain $X\alpha wY$ and XZ.

$(ROT2):0.$	—	$\#\alpha Y\$Z\#Y$
1.	$(\lambda, \lambda/k_1, \#\alpha)$	$k_1\#\alpha Y\$Z\#Y$
2.	$(k_1\#, \alpha/\lambda, Y)$	$k_1\#Y\$Z\#Y$
3.	$(k_1\#, Y/\lambda, \$)$	$k_1\#\$Z\#Y$
4.	$(k_1, \lambda/X, \#\$)$	$k_1X\#\$Z\#Y$
5.	$(k_1X, \lambda/\alpha, \#\$)$	$k_1X\alpha\#\$Z\#Y$
6.	$(k_1X, \lambda/k_2, \alpha\#)$	$k_1Xk_2\alpha\#\$Z\#Y$
7.	$(\lambda, k_1/\lambda, Xk_2)$	$Xk_2\alpha\#\$Z\#Y$
8.	$(k_2\alpha, \lambda/k_3, \#\$)$	$Xk_2\alpha k_3\#\$Z\#Y$
9.	$(X, k_2/\lambda, \alpha k_3)$	$X\alpha k_3\#\$Z\#Y$
10.	$(k_3\#, \lambda/k_4, \$)$	$X\alpha k_3\#k_4\$Z\#Y$
11.	$(X\alpha, k_3/\lambda, \#k_4)$	$X\alpha\#k_4\$Z\#Y$
12.	$(\alpha\#, \lambda/Z, k_4)$	$X\alpha\#Zk_4\$Z\#Y$
13.	$(k_4\$, \lambda/k_5, Z)$	$X\alpha\#Zk_4\$k_5Z\#Y$
14.	$(\#Z, k_4/\lambda, \$k_5)$	$X\alpha\#Z\$k_5Z\#Y$
15.	$(k_5, Z/\lambda, \#Y)$	$X\alpha\#Z\$k_5\#Y$
16.	$(k_5\#, \lambda/k_6, Y)$	$X\alpha\#Z\$k_5\#k_6Y$
17.	$(k_6Y, \lambda/k_7, \lambda)$	$X\alpha\#Z\$k_5X\#k_6Yk_7$
18.	$(k_6, Y/\lambda, k_7)$	$X\alpha\#Z\$k_5\#k_6k_7$
19.	$(k_5\#, k_6/\lambda, k_7)$	$X\alpha\#Z\$k_5\#k_7$
20.	$(\$, k_5/\lambda, \#k_7)$	$X\alpha\#Z\$\#k_7$
21.	$(\$, \lambda/X, \#k_7)$	$X\alpha\#Z\$X\#k_7$
22.	$(X\#, k_7/\lambda, \lambda)$	$X\alpha\#Z\$X\#$

The splicing rules constructed at the end of $(ROT1)$ and $(ROT2)$ rotate one symbol inside the markers X and Y. After completely rotating one symbol, the rule $X\alpha\#Z\$X\#$ will again be transformed into the rule $c_1\#Y\$Z\#$. The constructions of the template rule from the rotation rule and that of the terminating rule from the template rule are similar to that of the construction in Theorem 3.1 and hence we omit it.

If the string generated after application of the terminating rule is over terminals, then $w \in L(G)$. The splicing rules constructed from the template contains at least one of the markers X and/or Y. Hence, even if w is not a terminal string, it cannot be spliced further, since it does not contain any markers.

Since, the terminating rule cannot be modified further and the computation stops after application of this rule, $L(\gamma) = L(G)$. The splicing rules constructed have radius ≤ 4, so we can conclude that $RE \subseteq EH_{rle}(4, 3, 1)$. $\square$

In the next result, instead of using point mutation rules, we use insertion rules which can insert at most two symbols, but the deletion rules remove only one symbol. In Theorem 3.2 the radius of the splicing rules is ≤ 4, but the insertion-deletion rules are point mutation rules, having contexts of length ≤ 2. If insertion rules are allowed to insert at most two symbols, then the radius of the splicing rules can be reduced to 3. Since the construction in the proof is similar to that of the above results, we omit it.

Theorem 3.3. $RE = EH_{rle}(3, 2, 2)$.

In the next result, we investigate the computational power of restricted locally evolving splicing systems, where the insertion-deletion rules are applied in a matrix controlled manner. Matrix insertion-deletion systems are investigated in [10]. In the following result, we show that, if matrix controlled insertion-deletion rules are used, then locally evolving splicing systems can generate all RE languages, where the splicing rules are of radius ≤ 3, and insertion-deletion rules are point mutation rules with contexts of length 1.

In matrix insertion-deletion system, a set of rules are applied in a sequential manner, i.e., if the rules of the matrix $R_n = [r_{n1}, r_{n2}, \ldots, r_{nm}]$, $n, m \in \mathbb{N}$ is applied, it means that the rules $r_{n1}, r_{n2}, \ldots, r_{nm}$ are applied in sequence.
We denote by $Mat_k EH_{rle}(m, c, id)$ the family of languages generated by matrix restricted locally evolving splicing systems, where k represents the maximum number of insertion-deletion rules required to transform one splicing rule to another, white the other parameters are as above.

Theorem 3.4. $RE = Mat_{10}EH_{rle}(3, 1, 1)$.

Proof: We only prove the inclusion $RE \subseteq Mat_{10}EH_{rle}(3, 1, 1)$.
Let $G = (N, T, P_0, S)$ be a grammar in Kuroda normal form with the rules in P_0 labeled in one-to-one manner. We construct a restricted extended H system $\gamma = (V, T, A_0, A_c, E, C_0, P)$ with locally evolving splicing rules such that $L(G) = L(\gamma)$.

- $V = N \cup T \cup \{X, Y, Z, B_0\}$,
- $A_0 = \{XB_0SY\}$,
- $A_c = \{ZY\} \cup \{X\alpha Z | \alpha \in N \cup T \cup \{B_0\}\} \cup \{ZCDY | r_i : AB \to CD\}$
$\cup \{ZBCY | r_i : A \to BC\} \cup \{ZaY \mid r_i : A \to a\}$,
- $E = N \cup T \cup \{X, Y, Z\} \cup \{c_1, c_2\} \cup \{d_1, d_2\} \cup \{e_1, k_1\} \cup \{f_1, f_2, f_3\} \cup \{g_1, g_2\} \cup \{h_1\}$,
- $C_0 = (c_1\#\$Z\#Y)$.

In this proof, the insertion-deletion rules work in the matrix controlled manner. Thus, $P = \{R_i, R_i^{'} \mid r_i \in P\} \cup \{R_\alpha, R_\alpha^{'} \mid \alpha \in V\} \cup \{R_{B_0}, R_{B_0}^{'}\}$ is the set containing the matrices. We discuss the construction of $\#ABY\$Z\#CDY$ from $c_1\#Y\$Z\#$. Note that the rules of P_0 are labeled with r_i and for each rule there exist matrices R_i and $R_i^{'}$. The sequential application of the rules of R_i constructs the splicing rule, which simulates the rule $r_i \in P_0$ and the sequential application of rules from $R_i^{'}$ reconstructs the template splicing rule from the splicing rule constructed for simulating the r_i labeled rules.

$(R1)$: The $r_i : AB \to CD$ rule is simulated by the splicing rule $\#ABY\$Z\#CDY$. After applying the rules in $R_i = [r_{i1}, r_{i2}, \ldots, r_{i6}]$, the template splicing rule forms the splicing rule $\#ABY\$Z\#CDY$.

0.	—	$c_1\#\$Z\#Y$
1.	$r_{i1} : (\#, \lambda/C, Y)$	$c_1\#\$Z\#CY$
2.	$r_{i2} : (C, \lambda/D, Y)$	$c_1\#\$Z\#CDY$
3.	$r_{i3} : (\#, \lambda/A, \$)$	$c_1\#A\$Z\#CDY$
4.	$r_{i4} : (A, \lambda/B, \$)$	$c_1\#AB\$Z\#CDY$
5.	$r_{i5} : (B, \lambda/Y, \$)$	$c_1\#ABY\$Z\#CDY$
6.	$r_{i6} : (\lambda, c_1/\lambda, \#)$	$\#ABY\$Z\#CDY$

$(T1)$: After the complete simulation of the rule $r_i : AB \to CD$, the template splicing rule is constructed. If the rules of $R_i^{'} = [r_{i1}^{'}, r_{i2}^{'}, \ldots, r_{i8}^{'}]$ are applied, then the template splicing rules are produced from $\#ABY\$Z\#CDY$.

0.	——	$\#ABY\$Z\#CDY$
1.	$r'_{i1} : (Z, \lambda/d_1, \#)$	$\#ABY\$Zd_1\#CDY$
2.	$r'_{i2} : (B, Y/\lambda, \$)$	$\#AB\$Zd_1\#CDY$
3.	$r'_{i3} : (A, B/\lambda, \$)$	$\#A\$Zd_1\#CDY$
4.	$r'_{i4} : (\#, A/\lambda, \$)$	$\#\$Zd_1\#CDY$
5.	$r'_{i5} : (\#, C/\lambda, D)$	$\#\$Zd_1\#DY$
6.	$r'_{i6} : (\#, D/\lambda, Y)$	$\#\$Zd_1\#Y$
7.	$r'_{i7} : (\lambda, \lambda/c_1, \#)$	$c_1\#\$Zd_1\#Y$
8.	$r'_{i8} : (Z, d_1/\lambda, \#)$	$c_1\#\$Z\#Y$

We skip the construction for the rules $A \to BC$, $A \to a$ and $A \to \lambda$ as they are similar to those considered in previous proofs. We now construct the necessary rotation rules.
$(ROT1)$: For each $\alpha \in N \cup T \cup \{B_0\}$, we construct splicing rule $\#\alpha Y\$Z\#Y$ from the template splicing rule $c_1\#Y\$Z\#$ by application of the rules in $R_\alpha = [r_{\alpha 1}, r_{\alpha 2}, \ldots, r_{\alpha 5}]$.

0.	——	$c_1\#\$Z\#Y$
1.	$r_{\alpha 1} : (\#, \lambda/e_1, \$\#)$	$c_1\#e_1\$Z\#Y$
2.	$r_{\alpha 2} : (e_1, \lambda/\alpha, \$)$	$c_1\#e_1\alpha\$Z\#Y$
3.	$r_{\alpha 3} : (\alpha, \lambda/Y, \$)$	$c_1\#e_1\alpha Y\$Z\#Y$
4.	$r_{\alpha 4} : (\#, e_1/\lambda, \alpha)$	$c_1\#\alpha Y\$Z\#Y$
5.	$r_{\alpha 5} : (\lambda, c_1/\lambda, \#)$	$\#\alpha Y\$Z\#Y$

$(ROT2)$: Now the rule $X\alpha\#Z\$X\#$ is constructed from $\#\alpha Y\$Z\#Y$ to complete the rotation of the symbol $\alpha \in N \cup T \cup \{B_0\}$. This is possible only after the application of rules in $R'_\alpha = [r'_{\alpha 1}, r'_{\alpha 2}, \ldots, r'_{\alpha 10}]$.

0.	——	$\#\alpha Y\$Z\#Y$
1.	$r'_{\alpha 1} : (\lambda, \lambda/f_1, \#)$	$f_1\#\alpha Y\$Z\#Y$
2.	$r'_{\alpha 2} : (\#, \alpha/\lambda, Y)$	$f_1\#Y\$Z\#Y$
3.	$r'_{\alpha 3} : (\#, Y/\lambda, \$)$	$f_1\#\$Z\#Y$
4.	$r'_{\alpha 4} : (\$, Z/\lambda, \#\$)$	$f_1\#\$\#Y$
5.	$r'_{\alpha 5} : (\#, Y/\lambda, \lambda)$	$f_1\#\$\#$
6.	$r'_{\alpha 6} : (\$, \lambda/X, \#)$	$f_1\#\$X\#$
7.	$r'_{\alpha 7} : (\#, \lambda/Z, \$)$	$f_1\#Z\$X\#$
8.	$r'_{\alpha 8} : (f_1, \lambda/\alpha, Z)$	$f_1\alpha\#Z\$X\#$
9.	$r'_{\alpha 9} : (f_1, \lambda/X, \alpha)$	$f_1X\alpha\#Z\$X\#$
10.	$r'_{\alpha 10} : (\lambda, f_1/\lambda, X)$	$X\alpha\#Z\$X\#$

$(T2)$: The rule $c_1\#Y\$Z\#$ is constructed from $X\alpha\#Z\$X\#$ by applying $R''_\alpha = [r''_{\alpha 1}, r''_{\alpha 2}, \ldots, r''_{\alpha 9}]$.

0.	—–	$X\alpha\#Z\$X\#$
1.	$r''_{\alpha 1} : (\#, \lambda/k_1, Z)$	$X\alpha\#k_1Z\$X\#$
2.	$r''_{\alpha 2} : (\$, \lambda/X, \#)$	$X\alpha\#k_1Z\$\#$
3.	$r''_{\alpha 3} : (\$, \lambda/Z, \#)$	$X\alpha\#k_1Z\$Z\#$
4.	$r''_{\alpha 4} : (\#, \lambda/Y, \lambda)$	$X\alpha\#k_1Z\$Z\#Y$
5.	$r''_{\alpha 5} : (X, \alpha/\lambda, \#)$	$X\#k_1Z\$Z\#Y$
6.	$r''_{\alpha 6} : (X, \lambda/c_1, \#)$	$Xc_1\#k_1Z\$Z\#Y$
7.	$r''_{\alpha 7} : (\lambda, X/\lambda, c_1)$	$c_1\#k_1Z\$Z\#Y$
8.	$r''_{\alpha 8} : (\#, k_1/\lambda, Z)$	$c_1\#Z\$Z\#Y$
9.	$r''_{\alpha 9} : (\#, Z/\lambda, \$)$	$c_1\#\$Z\#Y$

Now we construct splicing rules that remove the markers from the given string and terminates the computation process.
$(TE1)$: The marker X is removed after application of the splicing rule $XB_0\#\$\#ZY$ which is constructed after application of $R_{B_0} = [r_{B_0 1}, \dots, r_{B_0 7}]$ from $c_1\#Y\$Z\#$.

0.	—–	$c_1\#\$Z\#Y$
1.	$r_{B_0 1} : (\#, \lambda/g_1, \$)$	$c_1\#g_1\$Z\#Y$
2.	$r_{B_0 2} : (\$, Z/\lambda, \#)$	$c_1\#g_1\$\#Y$
3.	$r_{B_0 3} : (\#, \lambda/Z, Y)$	$c_1\#g_1\$\#ZY$
4.	$r_{B_0 4} : (c_1, \lambda/B_0, \#)$	$c_1B_0\#g_1\$\#ZY$
5.	$r_{B_0 5} : (c_1, \lambda/X, B_0)$	$c_1XB_0\#g_1\$\#ZY$
6.	$r_{B_0 6} : (\lambda, c_1/\lambda, X)$	$XB_0\#g_1\$\#ZY$
7.	$r_{B_0 7} : (\#, g_1/\lambda, \$)$	$XB_0\#\$\#ZY$

$(TE2)$: Similarly, $\#Y\$ZY\#$ is constructed to remove the marker Y. The rules in $R'_{B_0} = [r'_{B_0 1}, \dots, r'_{B_0 9}]$ transforms $XB_0\#\$\#ZY$ into $\#Y\$ZY\#$.

0.	—–	$XB_0\#\$\#ZY$
1.	$r'_{B_0 1} : (Y, \lambda/h_1, \lambda)$	$XB_0\#\$\#ZYh_1$
2.	$r'_{B_0 2} : (Z, Y/\lambda, h_1)$	$XB_0\#\$\#Zh_1$
3.	$r'_{B_0 3} : (\#, Z/\lambda, h_1)$	$XB_0\#\$\#h_1$
4.	$r'_{B_0 4} : (\#, \lambda/Y, \$)$	$XB_0\#Y\$\#h_1$
5.	$r'_{B_0 5} : (\$, \lambda/Z, \#)$	$XB_0\#Y\$Z\#h_1$
6.	$r'_{B_0 6} : (Z, \lambda/Y, \#)$	$XB_0\#Y\$ZY\#h_1$
7.	$r'_{B_0 7} : (X, B_0/\lambda, \#)$	$X\#Y\$ZY\#h_1$
8.	$r'_{B_0 8} : (\lambda, X/\lambda, \#)$	$\#Y\$ZY\#h_1$
9.	$r'_{B_0 9} : (\#, h_1/\lambda, \lambda)$	$\#Y\$ZY\#$

Since the rules constructed at the end of each section are same as in the proof of Theorem 3.2 and A_0 contains the string XB_0SY, it follows from Theorem 3.2 that $L(\gamma) = L(G)$. □

4 Conclusions

In this paper, we have shown that recursively enumerable languages can be generated by restricted locally evolving splicing systems with reduced contexts of insertion-deletion rules and radius. It was conjectured [9] that the radius as well as the length of the context can be reduced to 2. Several results supporting this conjecture were given, without completely solving the conjecture, as the results mentioned in this paper may not be optimal.

References

[1] CALUDE C. and PAUN G.: *Computing with cells and atoms: An introduction to quantum, DNA and membrane computing*, *CRC Press*, (2000).

[2] CULIK K. II and HARJU T.: Splicing semigroups of dominoes and DNA, *Discrete Applied Mathematics*, 31 (1991), 261-277.

[3] FREUND R., KARI L. and PAUN G.: DNA computing based on splicing: the existence of universal computers, *Theory of Computing Systems*, 32 (1999), 69-112.

[4] JEGANATHAN L. and RAMA R.: Matrix splicing systems, *International Journal of Computer Mathematics*, 87 (2010), 278-309.

[5] KRITHIVASAN K., CHAKARAVARTHY V.T. and RAMA R.: Array splicing systems, *LNCS*, 1218 (1997), 346-365.

[6] LOOS R.: An alternative definition of splicing, *Theoretical Computer Science*, 358 (2006), 75-87.

[7] PAUN G.: Regular extended H systems are computationally universal, *Journal of Automata, Languages and Combinatorics*, 1(1996), 27-36.

[8] PAUN G., ROZENBERG G. and SALOMAA A.: Computing by splicing: Programmed and evolving splicing systems, *IEEE International Conference on Evolutionary Computing*, Indianapolis, (1997), 273-277.

[9] PAUN G., ROZENBERG G. and SALOMAA A.: *DNA computing. New computing paradigms*, *Springer*, (1998).

[10] PETRE I. and VERLAN S.: Matrix insertion-deletion systems, *Theoretical Computer Science*, 456 (2012), 80-88.

[11] PIXTON D. : Regularity of splicing languages, *Discrete Applied Mathematics*, 69 (1996), 101-124.

[12] RAMA R. and KRISHNA S.N.: Contextual array splicing systems, *SPIRE/CRIWG*, (1999), 168-175.

[13] RAMA R. and RAGHAVAN U.: Splicing array systems, *International Journal of Computer Mathematics*, 73 (1999), 167-182.

[14] SALOMAA A.: *Formal languages*, Academic Press, New York, (1973).

[15] SANTHANAM R. and KRITHIVASAN K.: Graph splicing systems, *Discrete Applied Mathematics*, 154 (2006), 1264-1278.