ELSEVIER

Theoretical Computer Science 168 (1996) 321–336

Theoretical Computer Science

Computing by splicing[1]

Gheorghe Păun[a,2], Grzegorz Rozenberg[b,c,3], Arto Salomaa[d,*]

[a] *Institute of Mathematics of the Romanian Academy, P.O. Box 1-764, 70700 București, Romania*
[b] *Department of Computer Science, Leiden University, P.O. Box 9512, 2300 RA Leiden, The Netherlands*
[c] *Department of Computer Science, University of Colorado at Boulder, Boulder, CO 80309, USA*
[d] *Academy of Finland and University of Turku, Department of Mathematics, Yliopistonmäki, SF-20500 Turku, Finland*

Abstract

Computing by splicing is a new powerful tool stemming originally from molecular genetics. This new model of computing, splicing systems, is investigated here. Several variants, resulting from the use of the rules in different ways, are considered. The power of such systems with very weak structure imposed on rules turns out to be very large. Characterizations of recursively enumerable languages are obtained for many variants. In this way our study is analogous to the early studies concerning variations of Turing machines. Other classes of such splicing systems generate only regular or context-free languages (giving, in fact, characterizations of these families). With a few exceptions, we are able to obtain precise characterizations for all resulting families.

1. Introduction

Splicing systems were introduced in [6], as a formal language model of the recombinant behavior of DNA sequences. Basically, one gives an alphabet V, an initial language A over V, and a (finite) set of splicing rules, quadruples (u_1, u_2, u_3, u_4). Using such a rule, from two strings of the forms $x = x_1u_1u_2x_2$, $y = y_1u_3u_4y_2$, we produce, by splicing, the string $z = x_1u_1u_4y_2$. (Also the string $z' = y_1u_3u_2x_2$ is sometimes considered, but this amounts to considering also the symmetric rule, (u_3, u_4, u_1, u_2), as being present.) The language consisting of all strings in A and of all strings obtained by iterated splicing, starting from strings in A, is said to be *generated* by our splicing system.

* Corresponding author. Tel.: +358 (21) 633 5; fax: +358 (21) 633 6; e-mail: asalomaa@kontu.utu.fi.
[1] Research supported by the Academy of Finland, project 11281, and the ESPRIT Basic Research Working Group ASMICS II.
[2] E-mail: gpaun@imar.ro.
[3] E-mail: rozenber@wi.leidenuniv.nl.

PII S0304-3975(96)00082-5

Several papers are devoted to the study of splicing systems, where several variants/ generalizations of the basic operation and of the splicing systems were considered (see the references). We follow here the style of [12], allowing the set of rules to be infinite. Writing them in the form $u_1\#u_2\$u_3\#u_4$, where $\#, \$$ are new symbols, we can impose conditions on the *language* of rules (for instance, we can suppose that it is regular).

We add here a further component to a splicing system, an alphabet of *terminal* symbols, like in Chomsky grammars and in extended Lindenmayer systems. Moreover, we consider *modes* of using the splicing rules, as usual in language theory: leftmost, prefix, rightmost, etc. When splicing the strings x and y by a given rule, we can consider a mode of applying this rule to x different from the mode of applying it to y. The combination of all these possibilities – in choosing the type of the initial language, the type of the language of rules, and the modes of applying the rules to the two terms of a splicing – leads to several hundred of different classes of splicing systems. Fortunately, the associated families of languages collapse to a much smaller number of different families: in many cases we obtain exactly the family of regular languages, for many other classes we get exactly the family of recursively enumerable languages (hence the corresponding splicing systems have the computing power of Turing machines); some other families are equal to the family of context-free languages. Such results often exhibit amazing capabilities of one splicing mode to simulate other modes. A few families remain to be placed in a more precise way in the Chomsky hierarchy.

In [1] it is stated that the actual DNA language is not context-free. Our approach answers the need "for a grammatical theory of gene regulation" able to handle non-context-free languages, in the very framework of the splicing operation, which is known from [2, 14] to lead, by iteration, to regular languages only, when starting from regular languages and using a finite set of splicing rules in the free mode (in [15] it is proved that also the context-freeness is preserved by the iterated use of finitely many splicing rules). In view of the claim in [1], our result, that splicing systems with non-context-free sets of rules can generate all recursively enumerable languages, leads to the interesting conclusion that the actual DNA language can be of an arbitrary complexity (in Chomsky hierarchy).

2. Definitions

We denote: V^* = the free monoid generated by the alphabet V, λ = the empty string, $V^+ = V^* - \{\lambda\}$, $|x|$ = the length of $x \in V^*$, FIN, REG, CF, CS, RE = the families of finite, regular, context-free, context-sensitive, and recursively enumerable languages, respectively, $\partial_x^l(L) = \{w \in V^* \mid xw \in L\}$ (the left derivative of $L \subseteq V^*$ with respect to $x \in V^*$), $\partial_x^r(L) = \{w \in V^* \mid wx \in L\}$ (the right derivative), $L_1/L_2 = \{w \in V^* \mid wx \in L_1$ for some $x \in L_2\}$ (the right quotient of $L_1 \subseteq V^*$ with respect to $L_2 \subseteq V^*$). For further elements of formal language theory we refer to [16].

An *extended splicing system* is a quadruple

$$H = (V, T, A, R),$$

where V is an alphabet, $T \subseteq V$, $A \subseteq V^*$, and $R \subseteq V^*\#V^*\$V^*\#V^*$, where #, \$ are special symbols not in V.

We call V the alphabet of H, T is the *terminal* alphabet, A is the set of *axioms*, and R is the set of *splicing rules*. As we have already said in the Introduction, a rule $u_1\#u_2\$u_3\#u_4$ in R is used as depicted in Fig. 1. This suggests to represent the splicing rules in the more readable form in Fig. 2. (The idea is that originally the quadruples (u_1, u_2, u_3, u_4) are arbitrary. Then one views the associations $u_1 \to u_3$, $u_2 \to u_4$.)

The correspondence between u_1 and u_3, as well as that between u_2 and u_4, is visible in Fig. 2. Because A and R are languages, we may consider for them various restrictions: to be finite, regular, context-free, etc. Moreover, Fig. 2 suggests to consider a mapping φ acting on the left column and a mapping ψ acting on the right column, as in Fig. 3.

However, even for very simple mappings φ, ψ, the corresponding language R can be non-context free. For instance, if φ, ψ are the identity, and u_1, u_2 can be arbitrary, we obtain $R = \{u_1\#u_2\$u_1\#u_2 \mid u_1, u_2 \in V^*\}$, which is not context-free. This suggests to consider only the "halfs" of R

$$R_{12} = \{u_1\#u_2 \mid u_1\#u_2\$u_3\#u_4 \in R\},$$
$$R_{34} = \{u_3\#u_4 \mid u_1\#u_2\$u_3\#u_4 \in R\},$$

or the "quarters"

$$R_i = \{u_i \mid u_1\#u_2\$u_3\#u_4 \in R\}, \quad i = 1, 2, 3, 4.$$

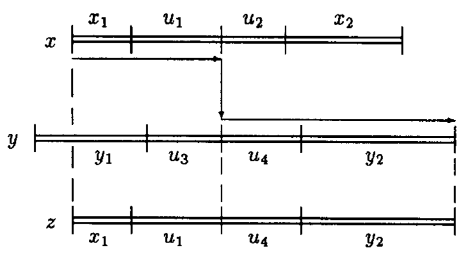

Fig. 1.

u_1	u_2
u_3	u_4

Fig. 2.

u_1	u_2
$\varphi(u_1)$	$\psi(u_2)$

Fig. 3.

In the example above, these languages are regular. Therefore, we can say that R is of type $REG/2$ if R_{12} and R_{34} are regular, and that it is of type $REG/4$ if R_1, R_2, R_3, R_4 are regular. Of course, if R is a regular language, then it is also of type $REG/2$, and if it is of type $REG/2$, then it is of type $REG/4$, too.

Consider now the mode of using the splicing rules. For $x, y, z \in V^*$ and $r : u_1\#u_2\$u_3\#u_4$ in R, we write

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

The substring u_1u_2 is identified in x, the substring u_3u_4 is identified in y, without any further restriction in any of these cases. This is the *free* mode of using the rule r. However, we can consider many other natural modes. We specify them only for x, the case of y being similar.

We say that u_1u_2 appears in x in the mode:

$$\begin{array}{rl}
\textit{free} & \text{iff } x = x_1u_1u_2x_2, \text{ for some } x_1, x_2 \in V^*, \\
\textit{prefix} & \text{iff } x = u_1u_2x_2, \text{ for some } x_2 \in V^*, \\
\textit{suffix} & \text{iff } x = x_1u_1u_2, \text{ for some } x_1 \in V^*, \\
\textit{total} & \text{iff } x = u_1u_2, \\
\textit{leftmost} & \text{iff } x = x_1u_1u_2x_2, \text{ for some } x_1, x_2 \in V^* \\
 & \quad \text{and there is no rule } r' \colon u_1'\#u_2'\$u_3'\#u_4' \text{ in } R \\
 & \quad \text{such that } x = x_1'u_1'u_2'x_2', \text{ for } x_1', x_2' \in V^* \text{ with } |x_1'| < |x_1|, \\
\textit{rightmost} & \text{iff } x = x_1u_1u_2x_2, \text{ for some } x_1, x_2 \in V^* \\
 & \quad \text{and there is no rule } r' \colon u_1'\#u_2'\$u_3'\#u_4' \text{ in } R \\
 & \quad \text{such that } x = x_1'u_1'u_2'x_2', \text{ for } x_1', x_2' \in V^* \text{ with } |x_2'| < |x_2|, \\
\textit{maximal} & \text{iff } x = x_1u_1u_2x_2, \text{ for some } x_1, x_2 \in V^* \\
 & \quad \text{and there is no rule } r' \colon u_1'\#u_2'\$u_3'\#u_4' \text{ in } R \\
 & \quad \text{such that } x = x_1'u_1'u_2'x_2', \text{ for } x_1', x_2' \in V^* \\
 & \quad \text{with } |x_1'| \leqslant |x_1|, \ |x_2'| \leqslant |x_2| \text{ and } |u_1'u_2'| > |u_1u_2|.
\end{array}$$

We denote these modes by f, p, s, t, l, r, m, respectively, and their set by D. For $g_1, g_2 \in D$ and $r \colon u_1\#u_2\$u_3\#u_4 \in R$, we write

$$\begin{array}{ll}
(x, y) \vdash_r^{g_1, g_2} z & \text{iff } u_1u_2 \text{ appears in } x \text{ in the mode } g_1, \\
 & \quad u_3u_4 \text{ appears in } y \text{ in the mode } g_2, \\
 & \quad \text{and for these occurrences of } u_1u_2, u_3u_4 \\
 & \quad \text{we obtain } z \text{ by splicing.}
\end{array}$$

With respect to a splicing system $H = (V, T, A, R)$ as above, a language $L \subseteq V^*$, and $g_1, g_2 \in D$, we define

$$\sigma_{g_1,g_2}(L) = L \cup \{z \in V^* \mid (x, y) \vdash_r^{g_1,g_2} z, \text{ for some } x, y \in L, \ r \in R\}.$$

Then we define

$$\sigma_{g_1,g_2}^*(L) = \bigcup_{i \geqslant 0} \sigma_{g_1,g_2}^{(i)}(L),$$

where

$$\sigma^{(0)}_{g_1,g_2}(L) = L,$$
$$\sigma^{(i+1)}_{g_1,g_2}(L) = \sigma_{g_1,g_2}(\sigma^{(i)}_{g_1,g_2}(L)), \quad i \geqslant 0.$$

The language generated by the splicing system H in the mode (g_1, g_2) is defined by

$$L_{g_1,g_2}(H) = \sigma^{*}_{g_1,g_2}(A) \cap T^{*}.$$

We denote by $EH_{g_1,g_2}(F_1,F_2)$ the family of languages generated by extended H systems $H = (V,T,A,R)$, in the mode (g_1, g_2), with the axiom language A of the type F_1, and the language of rules, R, of the type F_2. Here we consider F_1 to be one of FIN, REG, CF and F_2 one of $FIN, REG, CF, REG/2, REG/4, RE$. In total we obtain in this way

$$3 \times 6 \times 7^2 = 882$$

families of languages. Fortunately, many of them are identical (namely with known families, all of the latter in the Chomsky hierarchy).

The family of languages generated by H systems of the form $H = (T,T,A,R)$, hence without extended symbols, in the mode (g_1, g_2), with A, R of types F_1, F_2, respectively, is denoted by $H_{g_1,g_2}(F_1,F_2)$. In this case we write the system in the form $H = (T,A,R)$. If we take $H = (V,T,A,R)$ extended and $H' = (V,A,R)$ non-extended associated with H, then $L_{g_1,g_2}(H) = L_{g_1,g_2}(H') \cap T^{*}$.

3. Preliminary results

The following relations follow from definitions:

Lemma 1. (i) $H_{g_1,g_2}(F_1,F_2) \subseteq H_{g_1,g_2}(F_1',F_2')$, $EH_{g_1,g_2}(F_1,F_2) \subseteq EH_{g_1,g_2}(F_1',F_2')$, *for all* $F_1 \subseteq F_2', F_2 \subseteq F_2', g_1, g_2 \in D$.
(ii) $H_{g_1,g_2}(F_1,F_2) \subseteq EH_{g_1,g_2}(F_1,F_2)$, *for all* F_1, F_2, g_1, g_2.

In what concerns the type of the language R, of splicing rules, it is easy to see that we have

$$FIN \subset REG \subset \frac{REG}{2} \subset \frac{REG}{4},$$

and that CF is incomparable with $REG/2$ and $REG/4$.

Moreover, languages in $REG/2, REG/4$ are not necessarily "simple". Specifically, *there are languages in REG/2 which are not recursively enumerable*. Indeed, take a mapping $f : 2 \cdot \mathbf{N} \to 2 \cdot \mathbf{N}$ which is not computable. The set $\mathbf{N} - f(2 \cdot \mathbf{N})$ is countable (and infinite). Enumerate it: $n_1, n_2, \ldots$ and consider the mapping $g : \mathbf{N} \to \mathbf{N}$ defined by

$$g(i) = \begin{cases} f(i), & i \text{ even}, \\ n_{\frac{i+1}{2}}, & i \text{ odd}. \end{cases}$$

Consider the language

$$R_f = \{a^i\#\$a^{g(i)}\# \mid i \geqslant 1\}.$$

Because $R_{12} = R_{34} = a^*\#$, we have $R_f \in REG/2$, but, clearly, R_f is not in RE.

For this reason, from now on when we say that R is of type $REG/2$ or $REG/4$ it is assumed that $R \in RE$, too.

Because $L_{g_1,g_2}(H) = A$ for any $H = (T, A, \emptyset)$, we have

Lemma 2. *$F_1 \subseteq H_{g_1,g_2}(F_1, F_2)$, for all F_1, F_2 and all g_1, g_2.*

Moreover, from the Turing–Church thesis we obtain

Lemma 3. *$EH_{g_1,g_2}(F_1, F_2) \subseteq RE$, for all F_1, F_2, g_1, g_2.*

The splicing modes $g \in D$ are not very important from the generative point of view, and this is quite surprising and different from the situation in other areas of formal language theory (such as regulated rewriting area [3], or contextual grammars [13]).

Lemma 4. *$H_{f,g_2}(F_1, F_2) \subseteq H_{g_1,g_2}(F_1, F_2)$, $H_{g_1,f}(F_1, F_2) \subseteq H_{g_1,g_2}(F_1, F_2)$, $EH_{f,g_2}(F_1, F_2) \subseteq EH_{g_1,g_2}(F_1, F_2)$, $EH_{g_1,f}(F_1, F_2) \subseteq EH_{g_1,g_2}(F_1, F_2)$, for all $g_1, g_2 \in D$, for all F_1, and for all F_2 different from FIN.*

Proof. Take an extended H system $H = (V, T, A, R)$ and construct

$$H' = (V, T, A, \{V^*u_1\#u_2V^*\$u_3\#u_4 \mid u_1\#u_2\$u_3\#u_4 \in R\}),$$

$$H'' = (V, T, A, \{u_1\#u_2\$V^*u_3\#u_4V^* \mid u_1\#u_2\$u_3\#u_4 \in R\}).$$

We obtain

$$L_{f,g_2}(H) = L_{g_1,g_2}(H'), \qquad L_{g_1,f}(H) = L_{g_1,g_2}(H''),$$

for all $g_1, g_2 \in D$. Clearly, H', H'' are of the same type as H: if R', R'' are the languages of rules of H', H'', respectively, then we have

$$R'_{12} = V^*R_{12}V^*, \qquad R'_{34} = R_{34},$$

$$R''_{12} = R_{12}, \qquad R''_{34} = V^*R_{34}V^*,$$

$$R'_1 = V^*R_1, \qquad R'_2 = R_2V^*, \qquad R'_3 = R_3, \qquad R'_4 = R_4,$$

$$R'_1 = R_1, \qquad R''_2 = R_2, \qquad R''_3 = V^*R_3, \qquad R''_4 = R_4V^*.$$

In the case of non-extended systems we have $V = T$, hence the same construction can be used.

Therefore, we have the inclusions in the lemma. □

Lemma 5. *If $L \in EH_{g_1,g_2}(F_1,F_2)$, $L \subseteq V^*$, then for any F_1,F_2 and $c \notin V$, we have*
1. $\{c\}L \in EH_{f,g_2}(F_1,F_2)$, *for* $g_1 = p$, g_2 *arbitrary,*
2. $\{c\}L \in EH_{g_1,f}(F_1,F_2)$, *for* $g_2 = p$, g_1 *arbitrary,*
3. $L\{c\} \in EH_{f,g_2}(F_1,F_2)$, *for* $g_1 = s$, g_2 *arbitrary,*
4. $L\{c\} \in EH_{g_1,f}(F_1,F_2)$, *for* $g_2 = s$, g_1 *arbitrary,*
5. $\{c\}L\{c\} \in EH_{f,g_2}(F_1,F_2)$, *for* $g_1 = t$, g_2 *arbitrary,*
6. $\{c\}L\{c\} \in EH_{g_1,f}(F_1,F_2)$, *for* $g_2 = t$, g_1 *arbitrary.*

All the corresponding assertions are true also for non-extended families.

Proof. For $H = (V,T,A,R)$ and $c \notin V$, consider

$$H'_p = (V \cup \{c\}, T \cup \{c\}, \{c\}A, \{c\}R),$$
$$H''_p = (V \cup \{c\}, T \cup \{c\}, \{c\}A, \{u_1\#u_2\$cu_3\#u_4 \mid u_1\#u_2\$u_3\#u_4 \in R\}.$$

Clearly, $L_{f,g_2}(H'_p) = \{c\}L_{p,g_2}(H)$, $L_{g_1,f}(H''_p) = \{c\}L_{g_1,p}(H)$, for all g_1,g_2. Similar constructions prove the other assertions. For instance, for (5) $g_1 = t$, we use the rules

$$\{cu_1\#u_2c\$u_3\#u_4 \mid u_1\#u_2\$u_3\#u_4 \in R\},$$

whereas for (6) $g_2 = t$ we use

$$\{u_1\#u_2\$cu_3\#u_4c \mid u_1\#u_2\$u_3\#u_4 \in R\}. \qquad \square$$

Theorem 1. *$REG = EH_{g_1,g_2}(REG,FIN)$, $CF = EH_{g_1,g_2}(CF,FIN)$, for all $g_1,g_2 \in \{f,p,s,t\}$.*

Proof. In [2, 14] it is proved that $H_{f,f}(REG,FIN) \subseteq REG$, whereas in [15] it is proved that $H_{f,f}(CF,FIN) \subseteq CF$. From Lemma 5 ($REG$ and CF have the closure properties involved in the previous proof) we get $EH_{g_1,g_2}(REG,FIN) \subseteq REG$, $EH_{g_1,g_2}(CF,FIN) \subseteq CF$, for $g_1,g_2 \in \{f,p,s,t\}$. With Lemmas 1 and 2 above, we have also the converse inclusions. $\square$

Corollary. *$EH_{g_1,g_2}(FIN,FIN) \subseteq REG$, for all $g_1,g_2 \in \{f,p,s,t\}$.*

For the modes l,r,m we have only the following partial result:

Lemma 6. *If $L \in EH_{g_1,g_2}(F_1,F_2)$, $L \subseteq V^*$, for any F_1, for $F_2 \in \{REG, REG/2\}$ and $c \notin V$, we have*
1. $\{c\}L \in EH_{f,g_2}(F_1,F_2)$, *for* $g_1 = l$, g_2 *arbitrary,*
2. $\{c\}L \in EH_{g_1,f}(F_1,F_2)$, *for* $g_2 = l$, g_1 *arbitrary,*
3. $L\{c\} \in EH_{f,g_2}(F_1,F_2)$, *for* $g_1 = r$, g_2 *arbitrary,*
4. $L\{c\} \in EH_{g_1,f}(F_1,F_2)$, *for* $g_2 = r$, g_1 *arbitrary,*
5. $\{c\}L\{c\} \in EH_{f,g_2}(F_1,F_2)$, *for* $g_1 = m$, g_2 *arbitrary,*
6. $\{c\}L\{c\} \in EH_{g_1,f}(F_1,F_2)$, *for* $g_2 = m$, g_1 *arbitrary.*

All the corresponding assertions are true also for non-extended families.

Proof. Take $H = (V,T,A,R)$ working in the (l,g_2) mode. Denote by $\gamma_d(R_{12}V^*)$ the language obtained from $R_{12}V^*$ by inserting the symbol d at the right of a symbol in V, non-deterministically chosen (hence γ can be affected by a gsm). If $R_{12} \in REG$, then $\gamma_d(R_{12}V^*) \in REG$. Construct the system

$$H' = (V \cup \{c\}, T \cup \{c\}, \{c\}A, R'_l),$$

where

$$R'_l = h((\{c\}V^*\{d\}V^*\{\#\}V^* - \{c\}V^*\gamma_d(R_{12}V^*))\{\$\}V^*\{\#\}V^* \cap \{c\}V^*\{d\}R),$$

h being the morphism defined by $h(a) = a$, $a \in V \cup \{\#,\$,c\}$, $h(d) = \lambda$. We obtain $\{c\}L_{l,g_2}(H) = L_{f,g_2}(H')$: the difference $\{c\}V^*\{d\}V^*\{\#\}V^* - \{c\}V^*\gamma_d(R_{12}V^*)$ selects the strings which contain occurrences of strings $u_1\#u_2 \in R_{12}$ only in the right hand of the occurrence of the symbol d. Hence to the left of d we add a prefix cw which ensures that the use of u_1u_2 in a splicing $(x,y) \vdash z$ is leftmost in x.

Because REG is closed under difference and intersection, we have R'_l of the same type as R. This proves point (1).

For point (2), take a system $H = (V,T,A,R)$ and construct

$$H'' = (V \cup \{c\}, T \cup \{c\}, \{c\}A, R''_l),$$

with

$$R''_l = h(\{u_1\#u_2\$cwu_3\#u_4 \mid u_1\#u_2\$u_3\#u_4 \in R, cwdu_3\#u_4 \in (\{c\}V^*\{d\}V^*\{\#\}V^* - \{c\}V^*\gamma_d(R_{34}V^*)) \cap \{c\}V^*\{d\}\{u_3\#u_4\}\}.$$

The way of constructing R''_l ensures the leftmost use of u_3u_4, because the string cw added in front of $u_3\#u_4$ ensures that no rule in R can have the string $u'_3u'_4$ to the left of u_3u_4. Consequently, $L_{g_1,f}(H'') = \{c\}L_{g_1,l}(H)$, which proves point (2).

The other points can be proved in a similar way. □

4. Equalizing the power of Turing machines

For many variants of extended splicing systems, we shall obtain characterizations of recursively enumerable languages, hence such systems (even with finite sets of axioms and with rather simple sets of splicing rules) have the same generative power as Turing machines (and all other equivalent class of algorithms).

Theorem 2. *$RE = EH_{g_1,g_2}(F_1,F_2)$, for all $g_1,g_2 \in D$, $F_1 \in \{FIN,REG,CF\}$, $F_2 \in \{REG/2,REG/4,RE\}$.*

Proof. In view of Lemmas 1, 3, 4, it is enough to prove the relation $RE \subseteq EH_{f,f}(FIN,REG/2)$. Consider a type-0 grammar $G = (N,T,S,P)$ with the rules $u \to v \in P$ having $|u| \leqslant 2$, $|v| \leqslant 2$ (for instance, take G in Kuroda normal form). Construct the splicing system $H = (V, T, A, R)$, where

$$V = N \cup T \cup \{X_1, X_2, X_3, Y_1, Y_2, Z\},$$

$$A = A_0 \cup A_1 \cup A_2 \cup A_3,$$

with

$$A_0 = \{X_3X_2\} \cup \{X_3\alpha X_3 \mid \alpha \in N \cup T\},$$

$$A_1 = \{X_1Y_1Y_2SX_2\},$$

$$A_2 = \{ZY_2vX_3 \mid u \to v \in P\},$$

$$A_3 = \{Z\alpha Y_1Y_2X_3, ZY_1Y_2\alpha X_3 \mid \alpha \in N \cup T\},$$

and R contains the following groups of rules (we write the rules as in Fig. 3, for an easier readability):

(1) $\dfrac{ZY_2x \mid X_3}{X_3 \mid \alpha X_3}$, for $\alpha \in N \cup T$, $x \in (N \cup T)^*$,

(2) $\dfrac{ZY_2x \mid X_3}{X_3 \mid X_2}$, for $x \in (N \cup T)^*$,

(3) $\dfrac{Z\alpha Y_1Y_2x \mid X_3}{X_3 \mid \beta X_3}$, for $\alpha, \beta \in N \cup T$, $x \in (N \cup T)^*$,

(4) $\dfrac{Z\alpha Y_1Y_2x \mid X_3}{X_3 \mid X_2}$, for $\alpha \in N \cup T$, $x \in (N \cup T)^*$,

(5) $\dfrac{ZY_1Y_2x \mid X_3}{X_3 \mid \alpha X_3}$, for $\alpha \in N \cup T$, $x \in (N \cup T)^+$,

(6) $\dfrac{ZY_1Y_2x \mid X_3}{X_3 \mid X_2}$, for $x \in (N \cup T)^+$,

(7) $\dfrac{X_1xY_1 \mid Y_2uwX_2}{Z \mid Y_2vwX_2}$, for $u \to v \in P$, $x, w \in (N \cup T)^*$,

(8) $\dfrac{X_1x \mid Y_1Y_2\alpha wX_2}{Z \mid \alpha Y_1Y_2wX_2}$, for $\alpha \in N \cup T$, $x, w \in (N \cup T)^*$,

(9) $\dfrac{X_1x \mid \alpha Y_1Y_2wX_2}{Z \mid Y_1Y_2\alpha wX_2}$, for $\alpha \in N \cup T$, $x, w \in (N \cup T)^*$,

(10) $\dfrac{X_1w \mid Y_1Y_2X_2}{X_1wY_1Y_2 \mid X_2}$, for $w \in T^*$,

(11) $\dfrac{\lambda \mid X_1wX_2}{X_1 \mid wX_2}$, for $w \in T^*$,

(12) $\dfrac{w \mid X_2}{wX_2 \mid \lambda}$, for $w \in T^*$,

Observe that $A \in FIN$ and that $R \in REG/2$.

We have two main sets of rules, those in groups 1–6, and those in groups 7–12. The first ones are *initial*, in the following sense. Each rule in this group involves two strings containing each an occurrence of the symbol X_3, each rule in the second set involves two strings containing each an occurrence of the symbol X_2. Only the axiom in A_1 contains the symbol X_2, but no rule in groups 7–12 can use two copies of $X_1Y_1Y_2SX_2$ for a splicing. Therefore, the process starts from axioms in A_0, A_2, A_3, using rules of types 1–6.

It is easy to see that starting from a string in A_2, using a rule in group 1 to splice it with strings of the form $X_3\alpha X_3$ in A_0 we can obtain all strings of the form ZY_2vwX_3, for $u \to v \in P$, $w \in (N \cup T)^*$. To such a string, only rules in group 1 can be used, splicing again with some $X_3\alpha X_3$, or a rule in group 2, splicing with X_3X_2. We obtain a string ZY_2vwX_2, for $u \to v \in P$ and $w \in (N \cup T)^*$. Let us denote by A_2' the set of all strings of this form.

Similarly, one can see that starting from a string $Z\alpha Y_1Y_2X_3$ in A_3 and using a rule in group 3 for splicing it with some $X_3\beta X_3$, then using a rule in group 4 for splicing the current string with X_3X_2, we can obtain all strings of the form $Z\alpha Y_1Y_2xX_2$, for $\alpha \in N \cup T$, $x \in (N \cup T)^*$.

If we start from a string $ZY_1Y_2\alpha X_3$ in the same A_3 and we use rules in group 5 for splicing it with some $X_3\beta X_3$, then we use a rule in group 6, for splicing the current string with X_3X_2, we can produce all strings of the form $ZY_1Y_2\alpha xX_2$, for $\alpha \in N \cup T$, $x \in (N \cup T)^*$.

We denote by A_3' the set of all such strings (ended by X_2) obtained from the strings in A_3.

Due to the presence of markers Z, X_3, X_2 in the rules of types 1–6, all these rules are applied in a unique mode – the total one – which hence is at the same time free, prefix, suffix, etc., that is, all the modes coincide for these rules.

The rules in groups 1–6 cannot be used for splicings involving a string in $A_1 \cup A_2' \cup A_3'$. From now on, only rules in groups 7–12 are applied and they are meant to simulate derivations in G. The string in A_1 will be the starting point of each such simulation.

Each splicing which uses rules of types 7–9 will use a string produced by splicing, at an earlier step of the simulation, and a string in A_2' or in A_3'. Rules in group 7 simulate the rewriting rules of P. This is done in the presence of the pair Y_1Y_2. This subword Y_1Y_2 can be moved to the left using the rules in group 8 and to the right using the rules in group 9. Rules in groups 10, 11 cannot use strings in $A \cup A_2' \cup A_3'$, hence only strings produced during the simulation can be used by these rules.

Using the rules in group 7 we get

$$(X_1xY_1Y_2uwX_2, ZY_2vwX_2) \vdash_7^{f,f} X_1xY_1Y_2vwX_2,$$

which corresponds to the derivation step $xuw \Rightarrow xvw$ in G by the rule $u \to v$.

(Note that the assertion above holds for all modes of applying these splicing rules, because all strings obtained by splicing, using rules in groups 7–9, contain the markers X_1, X_2 at the ends, and all strings in A_2', A_3' start with the marker Z and end with X_2. Therefore all modes coincide, the rules in groups 7–9 (and 10) are forced to be used in the total mode, which is at the same time prefix, suffix, maximal, etc.)

Using the rules in groups 8, 9 we get

$$(X_1xY_1Y_2\alpha wX_2, Z\alpha Y_1Y_2wX_2) \vdash_8^{f,f} X_1x\alpha Y_1Y_2wX_2,$$

$$(X_1x\alpha Y_1Y_2wX_2, ZY_1Y_2\alpha wX_2) \vdash_9^{f,f} X_1xY_1Y_2\alpha wX_2,$$

hence we interchange the places of Y_1Y_2 and α.

Because of the matching substrings w in rules of types 7–9, by splicing we get a string identical to the first string we start with, modulo the specified modification: replacing u with v, for $u \to v \in P$, and interchanging Y_1Y_2 with α, $\alpha \in N \cup T$.

Obviously, in this way we can simulate any derivation in G. More exactly, we get strings of the form $X_1xY_1Y_2X_2$ for $S \Rightarrow^* x$ in G. Now, using rules of type 10 we can remove Y_1Y_2, then we can remove X_1 by a rule of type 11, and X_2 by a rule of type 12 – these operations being possible if x above is a terminal string.

Consequently, $L(G) \subseteq L_{g_1,g_2}(H)$.

Conversely, all strings in A contain either the symbol X_3 or the symbol X_2.

The symbol X_3 can be removed only by rules in groups 2, 4, 6. What we obtain are strings in the above-mentioned sets A'_2 and A'_3, all of them containing the symbol X_2.

Now, the symbol X_2 can be removed only by using a rule of type 12. All the other rules in groups 7–11 need the presence of X_2 in both strings participating to splicing. No string in $A \cup A'_2 \cup A'_3$ is of the form xX_2 with $x \in T^*$, such that applying a rule of type 12 to it we obtain a terminal string. Consequently, we must use at least once one of the rules in groups 7–11. This implies that X_1 is also present, hence we must start the elimination of X_2 by using the string $X_1Y_1Y_2SX_2$ in A_1. As we have pointed out, all splicings using rules in groups 7–9 must be performed for strings x, y with x obtained by a previous splicing and y in $A'_2 \cup A'_3$. Moreover, all X_1,X_2 and the pair Y_1Y_2 must be present. This means that we can do nothing else than to simulate rules $u \to v \in P$ and to move the pairs Y_1Y_2 to the left and to the right. The rules in group 11 cannot be used before the rules in group 10, and no one can be used after the rules in group 12. Consequently, the splicing process will end by using rules in groups 10–12, in this order. The obtained string will be terminal, and it corresponds to a derivation in G. All the rules must be used in the t mode, the only possible, except for the rules of type 12, which are forced to be used in the suffix mode. But, because $w\#X_2\$wX_2\#$ appears as a rule for all $w \in T^*$, we can use this rule in each mode we need. Consequently, $L_{g_1,g_2}(H) \subseteq L(G)$, which completes the proof. $\square$

This theorem shows that a huge number of the considered families, exactly speaking 441 of them, are equal among themselves and with RE. Remark that the set of splicing rules considered in the previous proof is not context-face, but it is of a rather simple type: it is a right-linear simple matrix language [3] (roughly speaking, it is obtained from the language $\{ww \mid w \in V^*\}$ by finitely many operations of concatenation with regular languages, union, and insertion of symbols), hence it is semi-linear, too. Further characterizations of recursively enumerable languages can be obtained from the following result (using again finitely many axioms and a language of splicing rules somewhat simpler than the previous one: it is a linear language; please note, however, that the family of linear languages is incomparable with that of right-linear simple matrix languages, [3], hence the two results do not imply one another).

Theorem 3. *$RE = EH_{g_1,g_2}(F_1, CF)$ for all $g_1, g_2 \in D$ and for all $F_1 \in \{FIN, REG, CF\}$.*

Proof. It is enough to prove the inclusions $RE \subseteq EH_{g_1,g_2}(FIN, CF)$.

According to [9], for every language $L \in RE$ there are two context-free (in fact, linear) languages L_1, L_2 such that $L = L_1/L_2$. Therefore, it is enough to prove that for every $L_1, L_2 \in CF$, $L_1, L_2 \subseteq T^*$, we have $L_1/L_2 \in EH_{g_1,g_2}(FIN, CF)$.

To this aim, we construct the system

$$H = (T \cup \{X_1, X_2, X_3, Z\}, T, A, R),$$

with

$$A = \{X_1X_2, X_3X_3, X_2ZX_2\} \cup \{X_2aX_2 \mid a \in T\},$$

and the following groups of rules:

(1) $\dfrac{X_1x \mid X_2}{X_2 \mid aX_2}$, for $x \in (T \cup \{Z\})^*$, $a \in T \cup \{Z\}$,

(2) $\dfrac{X_1xZy \mid X_2}{X_3 \mid X_3}$, for $xy \in L_1$,

(3) $\dfrac{X_1x \mid ZyX_3}{X_3X_3 \mid \lambda}$, for $x \in T^*$, $y \in L_2$,

(4) $\dfrac{\lambda \mid X_3X_3}{X_1 \mid x}$, for $x \in T^*$.

Every string in A is non-terminal. All rules of types 1–4 must involve one string in A; excepting the case of using the string X_1X_2 and $X_2\alpha X_2$, $\alpha \in T \cup \{Z\}$, in a rule of type 1, all rules also involve one string which is not in A, hence it must be produced by a previous splicing. No rule of type 3 can be used before a rule of type 2 (the symbol X_3 is not present), whereas a rule of type 2 can be used only after introducing the symbol Z by a rule of type 1. If more occurrences of Z are introduced, then a rule of type 2 is not applicable, such a string will never be used for a terminal splicing. After using a rule of type 2 or rule 3, the rules of type 1 are no longer applicable. No rule can be used after using a rule of type 4, because we need an occurrence of X_1 in all other rules. Consequently, we have to use, in this order, rules of type 1, an arbitrary number of times (but we can continue only when only one occurrence of Z is introduced), then a rule of type 2, a rule of type 3, and the one of type 4. The use of rules of type 1 leads to strings of the form X_1xZyX_2, with $xy \in T^*$. Using a rule of type 2 means to check whether or not xy in such strings belongs to L_1. We obtain X_1xZyX_3. Using a rule of type 3 means to eliminate ZyX_3, providing that $y \in L_2$. We obtain X_1x, for $x \in L_1/L_2$. Finally, a rule of type 4 removes the initial nonterminal. The rules in groups 1–3 can be used in exactly one way and this is the t mode, hence it is of all other types. A rule of type 4 can be used in each mode (t, g_2), $g_2 \in \{l, f, p\}$, but for every string x there is a rule $\#X_3X_3\$X_1\#x$, hence we can find such a rule to apply it in any mode we need, for every given string x.

In conclusion, $L_{g_1,g_2}(H) = L_1/L_2$, for all g_1, g_2. $\square$

This theorem covers further 147 cases ($X \in \{FIN, REG, CF\}, Y = CF$).

5. The other families

Let us now consider families of the form $EH_{g_1,g_2}(FIN, F_2)$, for $g_1, g_2 \in D$, and $F_2 \in \{FIN, REG\}$.

From the corollary of Theorem 1 we know that $EH_{g_1,g_2}(FIN, FIN) \subseteq REG$ for $g_1, g_2 \in \{f, p, s, t\}$. On the other hand, we have

Lemma 7. *$REG \subseteq EH_{g_1,g_2}(FIN, FIN)$, for all $g_1 \in D - \{t\}, g_2 \in D$.*

Proof. Take a language $L \in REG$, $L \subseteq T^*$. We can write

$$L = \{x \in L \mid |x| \leqslant 2\} \cup \bigcup_{a,b \in T} \{ab\}(\partial^l_{ab}(L) - \{\lambda\}).$$

Every language $L_{ab} = \partial^l_{ab}(L) - \{\lambda\}$ is regular. Take a regular grammar $G_{ab} = (N_{ab}, T, S_{ab}, P_{ab})$, for L_{ab}. Because the languages L_{ab} do not contain the empty string, we may assume that no λ-rule appears in sets P_{ab}. Assume also that all sets N_{ab} are mutually disjoint.

We construct the H system

$$H = (V, T, A_1 \cup A_2 \cup A_3 \cup A_4, R_1 \cup R_2),$$

with

$$V = T \cup \{Z\} \cup \bigcup_{a,b \in T} N_{ab},$$

$$A_1 = \{x \in L \mid |x| \leqslant 2\},$$

$$A_2 = \{abS_{ab} \mid a, b \in T\},$$

$$A_3 = \{ZcY \mid X \to cY \in P_{ab},\ a, b, c \in T\},$$

$$A_4 = \{ZZc \mid X \to c \in P_{ab},\ a, b, c \in T\},$$

$$R_1 = \{de\#X\$Z\#cY \mid X \to cY \in P_{ab},\ a, b, c, d, e \in T\},$$

$$R_2 = \{de\#X\$ZZ\#c \mid X \to c \in P_{ab},\ a, b, c, d, e \in T\}.$$

No splicing can use strings in A_1, they are already terminal. The rules in R_1 must use as the second term a string from A_3, the rules in R_2 must use as the second term a string from A_4. Conversely, this is the only way to use strings in A_3 and A_4, because both the rules in R_1 and in R_2 need two terminals in the first string used in splicing. The only axioms of this type are those in A_2. They start a derivation in the corresponding regular grammar G_{ab}, also introducing the associated symbols a, b. Rules in R_1 simulate the use of non-terminal rules in sets P_{ab}, those in R_2 simulate the use of terminal rules. Because the non-terminals appear in only one position in all strings in A_2 or in strings obtained by splicing, whereas the strings in A_3, A_4 are of exactly the form of the corresponding parts of rules in R_1, R_2, the splicing can be done in exactly one way, which is simultaneously of any type (g_1, g_2) different of (t, g_2), $g_2 \in D$. Clearly, the generated language is L. □

Lemma 8. *$REG \subseteq EH_{t,g_2}(FIN, FIN)$, for all $g_2 \in D - \{t\}$.*

Proof. We use a sort of mirror image of the idea in the previous proof.

Take $L \subseteq T^*$, $L \in REG$, and write

$$L = \{x \in L \mid |x| \leqslant 2\} \cup \bigcup_{a,b \in T} (\partial^{\mathrm{r}}_{ab}(L) - \{\lambda\})\{ab\}.$$

Take a λ-free left-regular grammar $G_{ab} = (N_{ab}, T, S_{ab}, P_{ab})$, for the language $L_{ab} = \partial^{\mathrm{r}}_{ab}(L) - \{\lambda\}$, $a, b \in T$, hence with the rules in each P_{ab} of the forms $X \to Yc$, $X \to c$, $X, Y \in N_{ab}$, $c \in T$. Assume all sets N_{ab} mutually disjoint and construct the H system

$$H = (V, T, A_1 \cup A_2 \cup A_3 \cup A_4, R_1 \cup R_2),$$

where

$$V = T \cup \{Z\} \cup \bigcup_{a,b \in T} N_{ab},$$

$$A_1 = \{x \in L \mid |x| \leqslant 2\},$$

$$A_2 = \{S_{ab}ab \mid a, b \in T\},$$

$$A_3 = \{YcZ \mid X \to Yc \in P_{ab},\ a, b, c \in T\},$$

$$A_4 = \{cZZ \mid X \to c \in P_{ab},\ a, b, c \in T\},$$

$$R_1 = \{Yc\#Z\$X\#de \mid X \to Yc \in P_{ab},\ a, b, c, d, e \in T\},$$

$$R_2 = \{c\#ZZ\$X\#de \mid X \to c \in L_{ab},\ a, b, c, d, e \in T\}.$$

As in the previous proof, it is easy to see that $L_{g_1,g_2}(H) = L$ for all $g_1, g_2 \in D$ with $g_2 \neq t$. □

Theorem 4. *$REG = EH_{g_1,g_2}(FIN, FIN)$, $g_1, g_2 \in \{f, p, s, t\} - \{(t, t)\}$.*

There remains the case of the mode (t, t).

Theorem 5. $EH_{t,t}(FIN, FIN) = FIN$, $REG \subseteq EH_{t,t}(FIN, REG)$.

Proof. The first relation is obvious.

For the second one we repeat the proof of Lemma 7, but in the construction of the set R of rules we take

$$R_1' = T^*\{de\#X\$Z\#cY \mid X \to cY \in P_{ab},\ a,b,c,d,e \in T\},$$

$$R_2' = T^*\{de\#X\$ZZ\#c \mid X \to c \in P_{ab},\ a,b,c,d,e \in T\}.$$

Now all rules can be used in the t mode. More exactly, for each currently produced string xX we find a rule in R_1 or in R_2 of the form $x\#X\$Z\#cY$ or $x\#X\$ZZ\#c$, respectively. □

The characterization of families $EH_{g_1,g_2}(F_1, REG)$, $F_1 \in \{FIN, REG, CF\}$, remains *open*.

The families $H_{f,f}(F_1, F_2)$, $F_1 \in \{FIN, REG\}$, $F_2 \in \{FIN, REG\}$, are investigated also in [12, 14]. For instance, in [14] it is proved that $H_{f,f}(FIN, REG) - LIN \neq \emptyset$, but from [12] we find that $REG - H_{f,f}(REG, RE) \neq \emptyset$. A language proving this relation is

$$L = (ab)^+ \cup (ba)^+.$$

Because $EH_{t,t}(FIN, FIN) = FIN$, we have $L \notin EH_{t,t}(FIN, FIN)$, but from Lemmas 7, 8 and Theorem 5 we know that this language, being regular, can be generated in all other modes, even starting from finite sets of axioms. Moreover, we can produce this language in all modes different from the free one even without using extended symbols. This is a clear indication of the usefulness of both extended symbols and of the modes of using splicing rules different from f.

Consider, for instance, the non-extended splicing system

$$H = (\{a,b\}, \{ab, ba\}, \{ab\#\$\#ab, ba\#\$\#ba\}).$$

We obtain $L = L_{g_1,g_2}(H)$ for all $g_1, g_2 \in \{p,s,l,r,m,t\}$, such that $(g_1, g_2) \neq (t,t)$.

The case of $g_1, g_2 \in \{p,s\}$ is obvious: ab can appear as a prefix or as a suffix only in strings of $(ab)^+$ and ba can appear as a prefix or a suffix only in strings of $(ba)^+$, hence we cannot mix strings in $(ab)^+$ with those in $(ba)^+$. In the l or r modes, we observe that if, for instance, the first rule is used for a splicing of the form $(x,y) \vdash_1^{l,g_2} z$, if $x = (ba)^n$, then this is not a correct splicing, because we can use the second rule one step to the left of the place where the first rule is used. The same assertion holds for using the second rule. Again we cannot mix the strings in $(ab)^+$ with those in $(ba)^+$.

If one of the modes is t, for the corresponding term we have to use the associated string ab or ba. Because all strings in rules of H are of length 2, each use is trivially applied in the maximal mode.

For the mode (t,t) we have $L \in H_{t,t}(FIN, REG)$ (and $L \in H_{t,t}(REG, FIN)$, because $REG \subseteq H_{t,t}(REG, FIN)$ – Lemma 2). The easy proof of this assertion is left to the reader.

Acknowledgements

Thanks are due to an anonymous referee and to Lucian Ilie for the useful comments on an earlier version of this paper.

Note added in proof. In Gh. Păun, Regular extended H systems are computationally universal, *J. Automata Languages Combin.* **1** (1996) 27–36 it is proved that $RE = EH_{f,f}(FIN, REG)$. In view of Lemmas 2 and 4, this implies that $RE = EH_{g_1,g_2}(FIN, REG)$, for all $g_1, g_2 \in D$, thus solving the above-mentioned open problem and providing new proofs for Theorems 2 and 3.

References

[1] J. Collado-Vides, The search for a grammatical theory of gene regulation is formally justified by showing the inadequacy of context-free grammars, *CABIOS* **7** (1991) 321–326.

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

[3] J. Dassow and Gh. Păun, *Regulated Rewriting in Formal Language Theory* (Springer, Berlin, 1989).

[4] K.L. Denninghoff and R.W. Gatterdam, On the undecidability of splicing systems, *Int. J. Comput. Math.* **27** (1989) 133–145.

[5] R.W. Gatterdam, Splicing systems and regularity, *Int. J. Comput. Math.* **31** (1989) 63–67.

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

[7] T. Head, Splicing schemes and DNA, in: G. Rozenberg and A. Salomaa, eds., *Lindenmayer Systems; Impacts on Theoretical Computer Science and Developmental Biology* (Springer, Berlin, 1992) 371–383.

[8] L. Ilie and V. Mitrana, Crossing-over on languages. A formal representation of the recombination of genes in a chromosome, submitted, 1995.

[9] M. Latteux, B. Leguy and B. Ratoandromanana, The family of one-counter languages is closed under quotient, *Acta Inform.* **22** (1985) 579–588.

[10] A. Mateescu, Gh. Păun, G. Rozenberg and A. Salomaa, Simple splicing systems, *Discrete Appl. Math.*, to appear.

[11] Gh. Păun, On the power of the splicing operation, *Int. J. Comput. Math.* **59** (1995) 27–35.

[12] Gh. Păun, On the splicing operation, *Discrete Appl. Math.* **70** (1996) 57–79.

[13] Gh. Păun and G. Rozenberg, Contextual grammars, in: G. Rozenberg and A. Salomaa, eds., *The Handbook of Formal Languages* (Springer, Berlin, 1996).

[14] D. Pixton, Linear and circular splicing systems, in: *1st IEEE Symp. Intelligence in Neural and Biological Systems*, Washington (1995) 181–188.

[15] D. Pixton, Context-free splicing systems, manuscript, 1995.

[16] A. Salomaa, *Formal Languages* (Academic Press, New York, 1973).