Journal of Universal Computer Science, vol. 2, no. 4 (1996), 224-240
submitted: 13/10/95, accepted: 9/4/96, appeared: 28/4/96 © Springer Pub. Co.

The power of restricted splicing with rules from a regular language

Lila Kari
Department of Mathematics University of Western Ontario
London, Ontario, Canada N6A 5B7

Gheorghe PĂUN
Institute of Mathematics of the Romanian Academy
PO Box 1 – 764, 70700 Bucureşti, Romania

Arto SALOMAA
Academy of Finland and Turku University, Department of Mathematics
20500 Turku, Finland

Abstract: We continue the investigations begun in [11] on the relationships between several variants of the splicing operation and usual operations with formal languages. The splicing operations are defined with respect to arbitrarily large sets of splicing rules, codified as simple languages. The closure properties of families in Chomsky hierarchy are examined in this context. Several surprising results are obtained about the generative or computing power of the splicing operation. Many important open problems are mentioned.

Key Words: DNA recombination, splicing systems, molecular genetics, Chomsky hierarchy, regulated rewriting, abstract families of languages
Category: F4.3, F1.1

1 Introduction

The splicing operation has been introduced in [4] as a model of the recombinant behavior of DNA sequences. Specifically, given a quadruple $(u_1, u_2; u_3, u_4)$, of strings over some alphabet V, and two strings of the form $x = x_1u_1u_2x_2, y = y_1u_3u_4y_2$, we produce the string $z = x_1u_1u_4y_2$. We say that z is the result of *splicing* x and y according to the rule $(u_1, u_2; u_3, u_4)$ (applied at the sites u_1u_2, u_3u_4 specified above). In fact, this is a slight generalization of the operation in [4], where, for instance, also the string $y_1u_3u_2x_2$ is produced at the same time with $x_1u_1u_4y_2$; this is however equivalent to adding the symmetric rule, $(u_3, u_4; u_1, u_2)$

In the previous definition (and in many papers devoted to the splicing operation) the application of the rule $(u_1, u_2; u_3, u_4)$ to the strings x, y takes place without any restriction. A series of regulations of the splicing were considered in [11]. For example, one asks to produce strings z with certain properties (the evolution by splicing is supposed to have a purpose): to be a prolongation of x to the right, to be strictly larger/shorter than the involved strings x, y, to be an element of a given target language. Moreover, we can restrict the strings x, y to which a rule is applied: x, y have to be elements of the same class of a given partition of the set of all strings, for instance, they have to be equal, or of the same length, etc.

However, in [11] only the splicing with respect to a finite set of rules is considered. In the style of [7], we can also consider infinite sets of splicing rules, codified as languages in the following way: take two symbols $\#, \$$ not in V and write a rule $(u_1, u_2; u_3, u_4)$ as the string $u_1\#u_2\$u_3\#u_4$ over $V \cup \{\#, \$\}$. Sets of splicing rules of a given type in the Chomsky hierarchy can then be considered. Such splicing operations are investigated here, hence combining the restrictions in [11] about using rules with the generalization in [7] about the number of rules.

From [1], [2], [5], [9], [10], [12], etc we know already that the splicing operation is "very powerful", especially when it is iterated (roughly speaking, the splicing can simulate both concatenation and the erasing of prefixes/suffixes, and for certain of the above mentioned variants it induces the closure under doubling or under intersection). Some of the results we report below are still quite surprising (mainly because we work in the uniterated case). For instance, we find that each recursively enumerable language can be obtained by splicing a linear (or context-free) language with respect to a linear (or context-free) set of splicing rules, and this happens in most of the regulated types of splicing – including the free one, without any restriction. Such results can be interpreted also as new rather unexpected general models of computation. Similarly unexpected is the fact that non-context-free languages are obtained in a number of cases when we start from regular languages and the set of splicing rules is linear or context-free.

Most of our presentation below will concentrate on technical details. Further motivation and background material can be found in [4], [5], [7], [8], [9], [10], [12], [13].

2 Definitions

We start by recalling from [11] the definitions of the variants of the splicing operation we investigate here.

As usual, we denote: V^* = the free monoid generated by the alphabet V, λ = the empty string, $V^+ = V^* - \{\lambda\}$, $|x|_a$ = the number of occurrences in $x \in V^*$ of $a \in V$, $|x|_U = \sum_{a \in U} |x|_a$, REG, LIN, CF, CS, RE = the families in the Chomsky hierarchy (regular, linear, context-free, context-sensitive, recursively enumerable), FIN = the family of finite languages, $\partial^l_x(L) = \{w \in V^* \mid xw \in L\}$ (the left derivative of $L \subseteq V^*$ with respect to $x \in V^*$), $\partial^r_x(L) = \{w \in V^* \mid wx \in L\}$ (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 with respect to L_2), $L_2 \backslash L_1 = \{w \in V^* \mid xw \in L_1$ for some $x \in L_2\}$ (the left quotient), $x ⧢ y = \{x_1y_1 \ldots x_ny_n \mid n \geq 1, x = x_1x_2 \ldots x_n, y = y_1y_2 \ldots y_n, x_i, y_i \in V^*, 1 \leq i \leq n\}$ (the shuffle of $x, y \in V^*$), $Pref(L)$ $(Suf(L), Sub(L))$ = the set of prefixes (suffixes, substrings, respectively) of strings in L. A *coding* is a morphism which maps symbols to symbols, a *weak coding* is a morphism which maps symbols to symbols or to the empty string. A morphism $h : (V_1 \cup V_2)^* \longrightarrow V_1^*$ such that $h(a) = \lambda$ for

$a \in V_2$ and $h(a) \neq \lambda$ for $a \in V_1$, is said to be *restricted* on a language $L \subseteq (V_1 \cup V_2)^* V_1 (V_1 \cup V_2)^*$ if there is a constant k such that if $x \in Sub(L) \cap V_2^*$, then $|x| \leq k$. Two languages are considered equal if they differ by at most the empty string. For further elements of formal language theory, the reader is referred to [13], [14]. For details about abstract families of languages and their closure properties we refer also to [3].

Consider a given alphabet V and two special symbols $\#, \$$, not in V.

A *splicing rule* (over V) is a string $u_1\#u_2\$u_3\#u_4$, where u_1, u_2, u_3, u_4 are strings over V. For such a rule r and for $x, y \in V^*$, we write

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

The letter f stands for *free*: no restriction is imposed on the use of r on the strings x, y. We say that x is the *first term* and y is the *second term* of the splicing. When the rule r is understood or it is not important, we shall omit it and we write $\vdash^f$ only.

A *splicing scheme* is a pair $\sigma = (V, R)$, where V is an alphabet and R is a set of splicing rules (hence a language over $V \cup \{\#, \$\}$ consisting of strings of the form $u_1\#u_2\$u_3\#u_4$). If the language R belongs to a specified family FA, then we say that σ is a splicing scheme *of type FA*. Having defined the relation $\vdash_r^f$ as above, for a language $L \subseteq V^*$ we define

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

Therefore, a splicing scheme defines an operation with languages. It can be investigated in the general frame of abstract families of languages, relating the splicing to other known operations on languages. This has been done in [7] for the free splicing with respect to finite and regular sets of rules and in [11] for regulated variants of it, but only for splicing schemes with finite sets of rules. We consider here the general case.

Let us recall the definitions in [11].

Definition 1. Given a splicing scheme $\sigma = (V, R)$, for $x, y, z \in V^*$ and $r = u_1\#u_2\$u_3\#u_4 \in R$, we define

$$\begin{aligned}(x, y) \vdash_r^{pr} \; z & \text{ iff } (x, y) \vdash_r^f \; z \text{ and } x \in Pref(z), x \neq z, \\ (x, y) \vdash_r^{in} \; z & \text{ iff } (x, y) \vdash_r^f \; z \text{ and } |z| > \max\{|x|, |y|\}, \\ (x, y) \vdash_r^{de} \; z & \text{ iff } (x, y) \vdash_r^f \; z \text{ and } |z| < \min\{|x|, |y|\}, \\ (x, y) \vdash_r^{mi} \; z & \text{ iff } (x, y) \vdash_r^f \; z \text{ and } |z| \geq |z'| \text{ for all} \\ & \qquad z' \text{ such that } (x, y) \vdash_r^f \; z', \\ (x, y) \vdash_r^{md} \; z & \text{ iff } (x, y) \vdash_r^f \; z \text{ and } |z| \leq |z'| \text{ for all} \\ & \qquad z' \text{ such that } (x, y) \vdash_r^f \; z'.\end{aligned}$$

The indications pr, in, de, mi, md stand here for *prefix, length-increasing, length-decreasing, most-increasing, and most-decreasing* use of the rule, respectively. (The md mode is not considered in [11], although it is a natural counterpart of the in mode.)

Definition 2. A splicing scheme with a *target* is a triple $\sigma = (V, R, T)$, where $T \subseteq V^*$ is a regular language. For $r \in R$ and $x, y, z \in V^*$ we write

$$(x, y) \vdash_r^{tr} z \text{ iff } (x, y) \vdash_r^{f} z \text{ and } z \in T.$$

Definition 3. A splicing scheme with *clusters* is a triple $\sigma = (V, R, C)$, where C is a partition of V^*. For $r \in R$ and $x, y, z \in V^*$, we write

$$(x, y) \vdash_r^{cl} z \text{ iff } (x, y) \vdash_r^{f} z \text{ and } x, y \text{ belong to the same class of } C.$$

When C is a finite set of regular languages we write $\vdash_r^{fr}$ instead of $\vdash_r^{cl}$. When C consists of singleton classes, $\{x\}, x \in V^*$, then we write $\vdash_r^{sf}$ (from *self-splicing*), and when C consists of classes $C_i = \{x \in V^* \mid |x| = i\}, i \geq 0$, then we write $\vdash_r^{sl}$ (from *same-length* splicing). In fact, the general relation $\vdash_r^{cl}$ will not be considered in this paper.

Definition 4. An *ordered* splicing scheme is a triple $\sigma = (V, R, \geq)$, where $\geq$ is a partial order over R. When $r \geq r'$ and $r \neq r'$, we write $r > r'$. For $x, y, z \in V^*$ and $r \in R$, we write

$$(x, y) \vdash_r^{or} z \text{ iff } (x, y) \vdash_r^{f} z \text{ and there is no } r' \in R \text{ such that } r' > r \text{ and } (x, y) \vdash_{r'}^{f} z' \text{ for some } z' \in V^*.$$

(We use a rule which is maximal, in the sense of the given relation over R.)

We denote by D the set $\{f, pr, in, de, mi, md, tr, fr, sf, sl, or\}$, identifying the variants of the splicing operation defined above. For a splicing scheme σ with the alphabet V and the set of rules R, and for $L \subseteq V^*$ and $g \in D$, we define

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

Then, for two families of languages, FA_1, FA_2, we define

$$S_g(FA_1, FA_2) = \{\sigma_g(L) \mid L \in FA_1, \sigma \text{ of type } FA_2\},$$

for all $g \in D$.

The aim of this paper is to compare the families $S_g(FA_1, FA_2)$ with FA_1, for various values of $g \in D$ and FA_2 (mainly for FA_1, FA_2 in the Chomsky hierarchy). For $FA_2 = FIN$ this has been done in [11]. We consider here the cases $FA_2 \in \{REG, LIN, CF\}$ for FA_1 one of REG, LIN, CF, CS (of course, $S_g(RE, FA_2) = RE$, hence the case of $FA_1 = RE$ is not of interest.)

3 The power of the splicing operations

In [7] it is proved that $S_f(FA, REG) \subseteq FA$, for $FA \in \{REG, CF\}$, but $S_f(LIN, FIN) - LIN \neq \emptyset$.

Given a regular target splicing scheme $\sigma = (V, R, T)$, if we denote $\sigma' = (V, R)$, then we have

$$\sigma_{tr}(L) = L \cup (\sigma'_f(L) \cap T),$$

hence for all families closed under intersection with regular sets (and union), the closure under the f splicing implies the closure under the tr splicing, for all types of splicing schemes.

A similar reduction to the free splicing is true for the fr variant. Specifically, if $\sigma = (V, R, C)$ is a splicing scheme with $C = \{C_1, C_2, \ldots, C_n\}$ a partition of V^* into a finite number of regular sets, then, for $\sigma' = (V, R)$, we have

$$\sigma_{fr}(L) = L \cup \bigcup_{i=1}^{n} \sigma'_f(L \cap C_i).$$

For families closed under intersection with regular sets (and union), the closure under the f splicing implies the closure under the fr splicing.

Consequently, we do not have to consider explicitly the tr and fr variants.

As previously presented, the definition of the or splicing is too general. More specifically, the order restriction is natural for splicing schemes with finite sets of rules, but, in the general form considered above, it leads to artificial results, due to the unrestricted character of the order relation.

Consider, for instance the regular language

$$L = ca^+c \cup \{d\},$$

and the ordered regular splicing scheme $\sigma = (\{a, c, d\}, R, \leq)$, where

$$R = ca^+c\#\$\#d \cup c\#\$d\#,$$

with the order relation defined by

$$c\#\$d\# > ca^ic\#\$\#d, \text{ for } i \neq 2^j, j \geq 0.$$

We obtain

$$\sigma_{or}(L) \cap ca^+cd = \{ca^{2^j}cd \mid j \geq 0\}.$$

Indeed, the rule $c\#\$d\#$ prevents the use of $ca^ic\#\$\#d$, for all $i \neq 2^j, j \geq 0$, hence only rules $ca^{2^j}c\#\$\#d, j \geq 0$, can be used in order to produce strings in ca^+cd.

The language obtained is not context-free, but the non-context-freeness is introduced by the way the order relation is defined (it selects from the regular language R a non-context-free language of applicable rules).

A possible modification of the definition of the ordered splicing schemes is to consider the order relation defined among the classes of a finite partition of the set of rules, namely consisting of regular languages. We shall not investigate this variant here.

In fact, in most of our considerations below, the order restriction will be omitted from the set D of the variants of the splicing operation. In particular, this will happen in our main Theorem 4.1. (We have introduced the order restriction for the sake of completeness and, mainly, because the idea of a priority relation on the set of splicing rules is quite natural and it deserves further investigations.)

We give now a series of lemmas, relating the splicing variants to other operations with languages, then we shall collect the results for families in the Chomsky hierarchy in a summarizing theorem.

Lemma 1. *If FA_1 and FA_2 are two families of languages such that $FA_1 \subseteq FA_2$ and*

1. *FA_1 is closed under shuffle with regular sets, intersection with regular sets, quotient with regular sets, and restricted morphisms, whereas*
2. *FA_2 is closed under shuffle, intersection with regular sets, restricted morphisms, and λ-free gsm mappings,*

then $S_{pr}(FA_1, REG) \subseteq FA_2$.

Proof. Take a language $L \subseteq V^*, L \in FA_1$, and a splicing scheme $\sigma = (V, R)$ with $R \in REG$. Define the coding $h : V \longrightarrow V'$, where $V' = \{a' \mid a \in V\}$, by $h(a) = a', a \in V$. Consider a finite automaton $A = (K, V \cup \{\#, \$\}, s_0, F, \delta)$ for the language R.

Construct the following series of languages:

$$\begin{aligned}
R_1 &= \{s_0 u_1 (s_1, s_2) u_2 s_2 u_3 [s_2, s_3] u_4 s_f \mid u_1 \# u_2 \$ u_3 \# u_4 \in R,\\
&\quad s_1 \in \delta(s_0, u_1), s_2 \in \delta(s_1, \# u_2), s_3 \in \delta(s_2, \$ u_3), s_f \in \delta(s_3, \# u_4), s_f \in F\},\\
R_2 &= V^* \{s_0 u_1 (s_1, s_2) u_2 s_2 \mid s_0 u_1 (s_1, s_2) u_2 s_2 u_3 [s_2, s_3] u_4 s_f \in R_1\} V^*,\\
L_1 &= (L \sqcup\!\sqcup \{s_0 (s_1, s_2) s_2 \mid s_1, s_2 \in K\}) \cap R_2,\\
L_1' &= \{x_1 u_1 (s_1, s_2) u_2 x_2 \mid x_1 s_0 u_1 (s_1, s_2) u_2 s_2 x_2 \in L_1\},\\
R_3 &= V^* \{s_2 u_3 [s_2, s_3]^2 u_4 s_f \mid s_0 u_1 (s_1, s_2) u_2 s_2 u_3 [s_2, s_3] u_4 s_f \in R_1\} V^*,\\
L_2 &= (L \sqcup\!\sqcup \{s_2 [s_2, s_3]^2 s_f \mid s_2, s_3 \in K, s_f \in F\}) \cap R_3,\\
R_4 &= \{w_1 s_2 w_2 [s_2, s_3] \mid w_1, w_2 \in V^*, s_2, s_3 \in K\},\\
L_3 &= R_4 \backslash L_2 = \{[s_2, s_3] u_4 s_f y_2 \mid y_1 s_2 u_3 [s_2, s_3]^2 u_4 s_f y_2 \in L_2\},\\
L_3' &= \{[s_2, s_3] u_4 y_2 \mid [s_2, s_3] u_4 s_f y_2 \in L_3\},\\
R_5 &= V^* \{(s_1, s_2)[s_2, s_3] \mid s_1, s_2, s_3 \in K\} \{a a' \mid a \in V\}^* V'^+,\\
L_4 &= (L_1' \sqcup\!\sqcup h'(L_3')) \cap R_5,
\end{aligned}$$

where $h' : (V \cup \{[s, s'] \mid s, s' \in K\})^* \longrightarrow (V' \cup \{[s, s'] \mid s, s' \in K\})^*$ is the coding obtained by extending h, that is $h'(a) = a'$ for $a \in V$, and $h'([s, s']) = [s, s'], s, s' \in K$.

Note that we consider here as symbols both the elements of V, of K, as well as the pairs $(s, s'), [s, s']$ for all $s, s' \in K$. Observe also the fact that the pairs (s_1, s_2), $[s_2, s_3]$ identify rules in R via the regular languages R_1, R_2, R_3.

More specifically, the strings in R_1 are exactly the strings $u_1 \# u_2 \$ u_3 \# u_4$ of R with the occurrences of the marker $\#$ replaced by pairs $(s_1, s_2), [s_2, s_3]$ of states, with \$ replaced by s_2, and bordered by s_0 and some $s_f \in F$, where s_0, s_1, s_2, s_3, s_f are states used in a recognition of $u_1 \# u_2 \$ u_3 \# u_4$ by the automaton A. The "first half" of the strings in R are arbitrarily prolonged to the two ends with strings in V^*; the "second half" of strings in R_1 are arbitrarily prolonged to the two ends with strings in V^* and the symbol $[s_2, s_3]$ is doubled. The obtained languages are denoted by R_2, R_3. Intersecting these languages R_2, R_3 with the language L shuffled with strings of the form $s_0(s_1, s_2)s_2$, $s_2[s_2, s_3]^2 s_f$, respectively, we identify occurrences of $u_1 u_2$, $u_3 u_4$, respectively, in the strings of L. The resulting languages are L_1 and L_2. The language L_1' is a slight variant of L_1: the symbols s_0 and s_2 are erased. The obtained strings are meant to be the first term of the splicing operations, hence we need them as they are (by the splicing, they should be prolonged to the right with at least one symbol).

From the strings in L_2 we erase the prefix bounded by the first copy of $[s_2, s_3]$; this is done by the left quotient of L_2 with respect to R_4. We obtain the language L_3. Then, as when passing from L_1 to L_1', we remove the state s_f from the strings of L_3 (it is now useless), leading to L_3'.

The passing from L_1', L_3' to L_4 simulates the splicing: the symbols of L_3' are primed and then L_3' is shuffled with L_1' in such a way that the prefix of the strings in L_1' bounded by (s_1, s_2) is preserved. This is ensured by the intersection with R_5, which also selects from the result of the shuffling those strings where the remaining symbols of the string in L_1' are interleaved with symbols in the strings in L_3'.

Therefore, each $w \in L_4$ is of the form

$$\begin{aligned} w = \{ & x_1u_1(s_1, s_2)[s_2, s_3]a_1a_1' \dots a_ka_k'a_{k+1}a_{k+1}' \dots a_{k+r}a_{k+r}' \\ & a_{k+r+1}' \dots a_{k+r+p}' \mid x_1u_1(s_1, s_2)a_1 \dots a_ka_{k+1} \dots a_{k+r} \in L_1', \\ & \text{with } a_1 \dots a_k = u_2, k \geq 0, r \geq 0, \text{ and} \\ & [s_2s_3]a_1 \dots a_ka_{k+1} \dots a_{k+r}a_{k+r+1} \dots a_{k+r+p} \in L_3', p \geq 1\}. \end{aligned}$$

Note that in the above writing we have $p \geq 1$ indeed: each string in R_5 ends with at least a primed symbol which is not paired with its non-primed variant.

We have now only to remove from the strings of L_4 those symbols which do not appear in the output of the splicing. This is an easy operation which can be performed by as follows.

Consider a gsm γ which scans strings of L_4 of the above form and:

- leaves unchanged the prefix x_1u_1,
- replaces $(s_1, s_2)[s_2, s_3]$ by cc, for a new symbol c,
- from that point on, replaces each occurrence of symbols $a \in V$ with c and each occurrence of symbols $a' \in V'$ with the corresponding $a \in V$.

Consider also the morphism $h'' : (V \cup \{c\})^* \longrightarrow V^*$ defined by $h''(a) = a, a \in V$ and $h''(c) = \lambda$.

We obtain

$$\sigma_{pr}(L) = h''(\gamma(L_4)).$$

The equality follows from the way of constructing the languages $L_1, L_1', L_2, L_3, L_3', L_4$ (see the explanations above) and of defining h, h', h'', γ.

The languages R_1, R_2, R_3, R_4, R_5 are regular, all languages $L_1, L_1', L_2, L_3, L_3'$ are in FA_1. Then L_4 is in FA_2, hence also $\sigma_{pr}(L)$ is in FA_2 (clearly, h'' is a restricted morphism). □

Corollary. *$S_{pr}(FA, REG) \subseteq CS$, for $FA \in \{LIN, CF\}$, and $S_{pr}(REG, REG) \subseteq REG$.*

Observe in the previous proof, when applied for $FA_1 = LIN$ (or $FA_1 = CF$) and $FA_2 = CS$, how the quotient by R_4, in the definition of L_3, is done in FA_1, thus avoiding the erasing of an unbounded prefix of the second term of the splicing operation after passing to a language in the family $FA_2 = CS$, which is not closed under arbitrary erasings (by a quotient or in any other way).

The proof above can be modified in order to cover also other splicing variants.

Lemma 2. *If FA_1, FA_2 are families as in Lemma 1, then $S_{de}(FA_1, REG) \subseteq FA_2$.*

Proof. As in the previous proof, take $L \subseteq V^*, L \in FA_1$, and $\sigma = (V, R), R \in FA_2$, then define the coding h. Take the finite automaton $A = (K, V \cup \{\#, \$\}, s_0, F, \delta)$ recognizing the language R.

As in the proof of Lemma 1, construct the languages $R_1, R_2, R_3, L_1, L_1', L_2$. Then construct

$$L_2' = \{y_1 u_3 [s_2, s_3] u_4 y_2 \mid y_1 s_2 u_3 [s_2, s_3] u_4 s_f y_2 \in L_2\}$$

as well as

$$L_3 = (L_1'/V^+)\{d\},$$
$$L_4 = \{d\}(V^+ \backslash L_2'),$$

where d is a new symbol. (Observe that the quotient by V^+ removes in each case at least one symbol; instead of the removed symbols one adds one occurrence of d.)

Now consider the language

$$L_5 = (L_3 \sqcup\!\sqcup\, h'(L_4)) \cap R_5,$$

where h' is obtained by extending h by $h'(d) = d$, and $h'([s_2, s_3]) = [s_2, s_3], s_2, s_3 \in K$, and

$$R_5 = \{d\}\{ab' \mid a, b \in V\}^* \{(s_1, s_2)[s_2, s_3] \mid s_1, s_2, s_3 \in K\}\{ab' \mid a, b \in V\}^* \{d\}.$$

Consider now the gsm γ which replaces by a new symbol c all primed symbols to the left of (s_1, s_2) and the non-primed symbols to the right of $[s_2, s_3]$, as well as the symbols $(s_1, s_2,), [s_2, s_3]$ and d, then consider the morphism h'' erasing the symbol c. We obtain the equality

$$\sigma_{de}(L) = h''(\gamma(L_5)).$$

(The correct use of a splicing rule is ensured, as in the proof of Lemma 1, by the construction of the involved languages; the fact that we obtain a string which is strictly shorter than the terms of the splicing is entailed by the presence of the symbol d.) Consequently, $\sigma_{de}(L) \in FA_2$. □

Corollary. *$S_{de}(FA, REG) \subseteq CS$, for each $FA \in \{LIN, CF\}$, and $S_{de}(REG, REG) \subseteq REG$.*

A much stronger result is true for the length-increasing variant.

Lemma 3. *$S_{in}(CS, CS) \subseteq CS$.*

Proof. For $L \subseteq V^*, L \in CS$, and $\sigma = (V, R), R \in CS$, take the coding $h : V \longrightarrow V'$ as above. Consider two new symbols, c, d, and construct

$L_1 = (L \sqcup\!\sqcup\, \{d\})c^*,$
$L_2 = c^*(h(L) \sqcup\!\sqcup\, \{d\}),$
$R_1 = \{ac \mid a \in V\}^+ \{ab' \mid a, b \in V\}^* \{dd\}\{ab' \mid a, b \in V\}^* \{ca' \mid a \in V\}^+,$
$L_3 = (L_1 \sqcup\!\sqcup\, L_2) \cap R_1.$

Clearly, $L_3 \in CS$. Take a length-increasing grammar for L_3 and modify it to obtain a grammar G (of type-0) working as follows:

– generate a string $w \in L_3$; it is of the form $w = a_1c \ldots a_r c a_{r+1} b'_1 \ldots a_{r+k} b'_k dd a_{r+k+1} b'_{k+1} \ldots a_{r+k+l} b'_{k+l} c a'_{r+k+l+1} \ldots c a'_{r+k+l+p}$, with $r \geq 1, k \geq 0, l \geq 0, p \geq 1$;
– check whether or not around the substring dd the string u_1 appears to the left and u_2 appears to the right of dd, on symbols in V (that is, u_1 is a suffix of $a_1 \ldots a_r a_{r+1} \ldots a_{r+k}$ and u_2 is a prefix of $a_{r+k+1} \ldots a_{r+k+l}$), as well as u_3 to the left and u_4 to the right, on positions where primed symbols appear (that is, u_3 is a suffix of $b'_1 \ldots b'_k$ and u_4 is a prefix of $b'_{k+1} \ldots b'_{k+l} a'_{r+k+l+1} \ldots a'_{r+k+l+p}$), for some rule $u_1\#u_2\$u_3\#u_4$ in R (this can be done by guessing the rule, hence nondeterministically generating an element of R somewhere in the current string, then checking the presence of u_1, u_2, u_3, u_4 for the guessed rule, on the indicated positions; in this way, the length of the string is multiplied at most by 2),
– erase all occurrences of the symbols c, d, as well as all occurrences of primed symbols from the left of dd and the occurrences of non-primed symbols from the right of dd; moreover, all primed symbols from the right of dd are replaced by the corresponding non-primed symbols.

From the above construction, we have $L(G) = \sigma_{in}(L)$.

Because the obtained string, w, is strictly longer than the strings $x, y \in L$ whose splicing is simulated by G, the workspace of G is linearly bounded with respect to $|w|$ (it is at most $4|w| + k$, where k is the number of possible scanners, markers and other auxiliary symbols – whose number of occurrences is precisely defined in the construction of G).

Consequently, $\sigma_{in}(L) = L(G) \in CS$. □

Corollary. *$S_{in}(FA_1, FA_2) \subseteq CS$ for all $FA_1, FA_2 \subseteq CS$.*

A modification of the construction in Lemma 1 similar to that in Lemma 2 is possible for the length-increasing splicing for the remaining case (REG, REG).

Lemma 4. $S_{in}(REG, REG) \subseteq REG$.

Proof. We proceed as in the proofs of Lemma 1 and 2, namely we construct L_1, L'_1 and L_2 as in the proof of Lemma 1, then L'_2 as in the proof of Lemma 2.

Then, we consider the language

$$L_3 = (L'_1 ⧢ h'(L'_2)) \cap R_4,$$

for

$$R_4 = V^+\{ab' \mid a, b \in V\}^*\{(s_1, s_2)[s_2, s_3] \mid s_1, s_2, s_3 \in K\}\{ab' \mid a, b \in V\}^* V'^+.$$

The presence of V^+, V'^+ ensures the fact that by splicing (by simulating the splicing using the mentioned constructions) we get a string which is strictly longer than the strings we have started with. By a gsm γ we can remove the primed symbols to the left of (s_1, s_2), the non-primed symbols to the right of $[s_2, s_3]$, as well as the mentioned symbols $(s_1, s_2), [s_2, s_3]$, and we can replace the primed symbols to the right of $[s_2, s_3]$ by their non-primed counterparts. Therefore, $\gamma(L_3) = \sigma_{in}(L)$ is a regular language. □

We have started this section by mentioning the result in [7] that $S_f(FA, REG) \subseteq FA$, for families FA with certain (rather weak) closure properties. A counterpart of this result is also true, interchanging the places of FA and REG in $S_f(FA, REG)$ above.

Lemma 5. *If FA is a family of languages closed under concatenation with regular sets, substitution with regular sets, intersection with regular sets, and under arbitrary gsm mappings, then* $S_f(REG, FA) \subseteq FA$.

Proof. Take $L \subseteq V^*, L \in REG$, and $\sigma = (V, R)$ with $R \in FA$. Consider the regular substitution $s : (V \cup \{\#, \$\})^* \longrightarrow 2^{(V \cup \{c,d\})^*}$ defined by

$$\begin{aligned} &s(a) = \{a\},\ a \in V,\\ &s(\#) = \{c\},\\ &s(\$) = V^*\{d\}V^*, \end{aligned}$$

and construct the language

$$L_1 = V^* s(R) V^*.$$

Consider also the regular language

$$L_2 = (L \sqcup\!\sqcup \{c\})\{d\}(L \sqcup\!\sqcup \{c\}).$$

Then we have $L_1 \cap L_2 \in FA$ and the strings $w \in L_1 \cap L_2$ are of the form

$$w = x_1 u_1 c u_2 x_2 d y_1 u_3 c u_4 y_2,$$

for $x_1 u_1 u_2 x_2 \in L, y_1 u_3 u_4 y_2 \in L$, and $u_1 \# u_2 \$ u_3 \# u_4 \in R$.

If γ is a gsm which erases the substring $c u_2 x_2 d y_1 u_3 c$ from strings w as above, then we get

$$\sigma_f(L) = \gamma(L_1 \cap L_2),$$

hence $\sigma_f(L) \in FA$. □

Corollary. $S_f(REG, FA) \subseteq FA$ *for* $FA \in \{LIN, CF\}$.

As we have pointed out at the beginning of this section, the result in Lemma 5 also implies $S_g(REG, FA) \subseteq FA$ for $g \in \{tr, fr\}$ and all families FA with the mentioned closure properties (hence for $FA \in \{LIN, CF\}$, too).

We give now a surprisingly simple and surprisingly powerful result:

Lemma 6. *If FA is a family of languages which is closed under concatenation with symbols, then for all* $L_1, L_2 \in FA$, $L_1, L_2 \subseteq V^*$, *and for all* $g \in \{f, tr, or, fr, de, sl, sf, mi, md\}$ *we can write* $L_1/L_2 = L \cap V^*$, *where* $L \in S_g(FA, FA)$.

Proof. Take two languages $L_1, L_2 \in FA, L_1, L_2 \subseteq V^*$, and a symbol $c \notin V$. For the splicing scheme

$$\sigma = (V \cup \{c, d\}, \# L_2 c \$ c \#).$$

We obtain

$$L_1/L_2 = \sigma_f(L_1\{c\}) \cap V^*.$$

Indeed, the only possible splicing of strings in $L_1\{c\}$ is of the form

$$(x_1 x_2 c, yc) \vdash^f\ x_1,\ \text{for } x_1 x_2 \in L_1, x_2 \in L_2, y \in L_1.$$

Therefore, also the cases $g \in \{mi, md\}$ are covered.

Taking V^* as a target language, we have directly $L_1/L_2 = \sigma_{tr}(L_1\{c\})$.

For $g = or$ it is enough to consider the trivial order on the rules of σ. The case $g = fr$ is covered taking the one-class partition of $(V \cup \{c\})^*$.

For $g = sl$ and $g = sf$ we take $(x_1x_2c, x_1x_2c) \vdash^g x_1$ for all $x_1x_2 \in L_1$, $x_2 \in L_2$.

As $|x_1| < |x_1x_2c|$, we also have the equality for the de mode. □

Corollary 1. *Each language $L \in RE, L \subseteq V^*$, can be written as $L = L' \cap V^*$, for $L' \in S_g(FA_1, FA_2)$, for $g \in \{f, tr, or, fr, de, sl, sf, mi, md\}$ and $FA_1 \in \{LIN, CF\}$, $FA_2 \in \{LIN, CF\}$.*

Proof. According to [6], each language $L \in RE$ can be written in the form $L = L_1/L_2$ for two linear languages L_1, L_2. Because LIN has the closure properties in the previous lemma, $L_1/L_2 = L' \cap V^*$ for $L' \in S_g(LIN, LIN)$, therefore $L = L' \cap V^*$, $L' \in S_g(FA_1, FA_2)$ for all FA_1, FA_2 containing $LIN$and g as above. □

Corollary 2. *$S_g(FA_1, FA_2) - CS \neq \emptyset$ for all g, FA_1, FA_2 as in Corollary 1.*

Proof. The family CS is closed under intersection with regular sets and strictly included in RE. □

The lemmas which follow refer to precise families $S_g(FA_1, FA_2)$, with FA_1, FA_2 in the Chomsky hierarchy.

Lemma 7. $S_{sl}(REG, REG) \subseteq REG$.

Proof. We slightly modify the proof given for the corresponding relation $S_{sl}(REG, FIN) \subseteq REG$ in the proof of Theorem 1 in [7].

Take $L \in REG, L \subseteq V^*$, and $\sigma = (V, R), R \in REG$. Consider the symbols $a, c \notin V$ and construct the language

$$L_1 = (L\{c\}L \sqcup\!\sqcup \{a^nca^n \mid n \geq 1\}) \cap (V\{a\})^+\{cc\}(V\{a\})^+.$$

Because the language $\{a^nca^n \mid n \geq 1\}$ is linear, $L\{c\}L$ is regular and LIN is closed under shuffle with regular sets and intersection with regular sets, we have $L_1 \in LIN$. The strings of L_1 are of the form

$$w = b_1ab_2a \dots b_naccd_1ad_2a \dots d_na,$$

for $x = b_1b_2 \dots b_n \in L, y = d_1d_2 \dots d_n \in L$, for some $n \geq 1$. Clearly, the splicing according to the rules in (the regular language) R can be performed by a gsm associated with a finite automaton recognizing R and transforming w as above to z such that $(x, y) \vdash^{sl} z$. Therefore $\sigma_{sl}(L) \in LIN$. □.

In the same way as above, the proof of the relation $S_{sf}(REG, FIN) \subseteq CS$ in [11] can be modified to obtain the inclusion $S_{sf}(REG, REG) \subseteq CS$.

The results which follow give examples of families $S_g(FA_1, FA_2)$ containing non-context-free languages. They are rather surprising, because in each case we splice regular languages (using, however, splicing schemes with linear or context-free sets of rules).

Lemma 8. $S_{sl}(REG, LIN) - CF \neq \emptyset$.

Proof. Take the regular language

$$L = b_1a^+b_2a^+b_3$$

and the splicing scheme $\sigma = (\{a, b_1, b_2, b_3\}, R)$ with

$$R = \{b_1 a^n b_2 a^m b_3 \# \$ \# b_1 a^n b_2 a^p b_3 \mid n, m, p \geq 1\}.$$

Clearly, R is a linear language. Consider also the regular language

$$L' = b_1 a^+ b_2 a^+ b_3 b_1 a^+ b_2 a^+ b_3.$$

We obtain

$$\sigma_{sl}(L) \cap L' = \{b_1 a^n b_2 a^m b_3 b_1 a^n b_2 a^m b_3 \mid n, m \geq 1\}.$$

Indeed, if

$$(b_1 a^n b_2 a^m b_3, b_1 a^p b_2 a^q b_3) \vdash^{sl} \ w,$$

then we must have $n+m = p+q$ (the two strings must have the same length) and $n = p$ (from the form of rules in R). Consequently, $m = q$, hence the obtained string is

$$w = b_1 a^n b_2 a^m b_3 b_1 a^n b_2 a^m b_3,$$

which implies that $\sigma_{sl} \cap L'$ is not a context-free language, hence $\sigma_{sl}(L) \notin CF$.□

Lemma 9. $S_{pr}(REG, LIN) - CF \neq \emptyset$.

Proof. Take the regular language

$$L = ca^+b^+a^+c \cup ca^+b^+a^+cd$$

and the splicing scheme $\sigma = (\{a, b, c, d\}, R)$ with

$$R = \{c\#a^n b^m a^m c\$\#ca^p b^q a^n cd \mid n, m, p, q \geq 1\}.$$

Clearly, $R \in LIN$. However,

$$\sigma_{pr}(L) \cap cca^+b^+a^+cd = \{cca^n b^n a^n cd \mid n \geq 1\}.$$

Indeed, if $(x, y) \vdash^{pr} \ z$, then we must have

$$x = ca^n b^m a^p c, \ n, m, p \geq 1,$$
$$y = ca^{n'} b^{m'} a^{p'} cd, \ n', m', p' \geq 1,$$

because the splicing according to the rules in R removes the suffix $a^n b^m a^m c\alpha$ from $ca^n b^m a^m c\alpha$ in L, for $\alpha \in \{\lambda, d\}$, and adds $ca^p b^q a^n cd$, which is a string in L. This implies that $\alpha = \lambda$, otherwise the obtained string is not a strict prolongation of x.

Moreover, from the form of rules in R, in x we must have $m = p$ and in y we must have $p' = n$. As $x \in Pref(z)$, we also have $n = n', m = m', p = p'$. Together with $m = p, p' = n$, these relations imply $n = n' = m = m' = p = p'$, hence

$$z = cca^n b^n a^n cd.$$

Consequently, $\sigma_{pr}(L) \notin CF$. □

Lemma 10. $S_{in}(REG, CF) - CF \neq \emptyset, \ S_{de}(REG, CF) - CF \neq \emptyset$.

Proof. Take the regular language

$$L = ca^+b^+c \cup cb^+cb^+c$$

and the splicing scheme $\sigma = (\{a, b, c\}, R)$, with

$$R = \{ca^n\#b^nc\$c\#b^mcb^mc \mid n, m \geq 1\}.$$

Clearly, this is a context-free language. However,

$$\sigma_{in}(L) \cap ca^+b^+cb^+c = \{ca^nb^mcb^mc \mid n, m \geq 1, n < 2m + 1\}.$$

Indeed, if $(x, y) \vdash^{in} z$, then we must have

$$x = ca^nb^nc, \ n \geq 1,$$
$$y = cb^mcb^mc, \ m \geq 1$$

(from the form of rules in R), hence we obtain

$$z = ca^nb^mcb^mc.$$

In order to have $|z| > |x|$ we must have $|b^nc| < |b^mcb^mc|$, hence $n + 1 < 2m + 2$, that is $n < 2m + 1$. (On the other hand, the relation $|z| > |y|$ is obvious.)

The obtained language is not context-free, hence $\sigma_{in}(L) \notin CF$.

For the length-decreasing mode of splicing we consider the same splicing scheme, σ. For the regular language

$$L' = ca^+b^+c \cup c^+b^+cb^+c$$

we obtain

$$\sigma_{de}(L') \cap ca^+b^+cb^+c = \{ca^nb^mcb^mc \mid n, m \geq 1, n > 2m + 1\}.$$

Now we have $(x, y) \vdash^{de} z$ for

$$x = ca^nb^nc, \ n \geq 1,$$
$$y = c^pb^mcb^mc, \ m, p \geq 1,$$

and we obtain

$$z = ca^nb^mcb^mc,$$

for $p > n + 1$ (in order to have $|z| < |y|$) and $n + 1 > 2m + 2$, that is $n > 2m + 1$ (in order to have $|z| < |x|$).

Again we obtain a language which is not context-free. □

4 Main results

Theorem 1. *The closure properties in Tables 1, 2, 3, 4 hold, where for each triple* $(g, FA_1, FA_2), g \in D$, *at the intersection of the row of* g *and the column marked with* FA_2 *in the table associated with* FA_1 *we have written the smallest family* FA *(among the families considered here) such that* $S_g(FA_1, FA_2) \subseteq FA$ *or the overlined* $\overline{CF}$ *for the cases when the corresponding family* $S_g(FA_1, FA_2)$ *contains non-context-free languages.* (*When* $FA = FA_1$, *this means that the family* FA_1 *is closed under the type* g *of splicing with respect to splicing schemes of type* FA_2; *conversely,* $FA_1 \neq FA$ *indicates the nonclosure.*)

	FIN	REG	LIN	CF
f	REG	REG	LIN	CF
pr	REG	REG	$\overline{CF}$	
in	REG	REG		$\overline{CF}$
de	REG	REG		$\overline{CF}$
mi	REG			
md	REG			
tr	REG	REG	LIN	CF
fr	REG	REG	LIN	CF
sf	CS	CS		
sl	LIN	LIN	$\overline{CF}$	

Table 1 ($FA_1 = REG$)

	FIN	REG	LIN	CF
f	CF	CF	RE	RE
pr	CS	CS		
in	CS	CS	CS	CS
de	CS	CS	RE	RE
mi	CF		RE	RE
md	CF		RE	RE
tr	CF	CF	RE	RE
fr	CF	CF	RE	RE
sf	$\overline{CF}$		RE	RE
sl	$\overline{CF}$		RE	RE

Table 2 ($FA_1 = LIN$)

	FIN	REG	LIN	CF
f	CF	CF	RE	RE
pr	CS	CS		
in	CS	CS	CS	CS
de	CS	CS	RE	RE
mi	CF		RE	RE
md	CF		RE	RE
tr	CF	CF	RE	RE
fr	CF	CF	RE	RE
sf	$\overline{CF}$		RE	RE
sl	$\overline{CF}$		RE	RE

Table 3 ($FA_1 = CF$)

	FIN	REG	LIN	CF
f	RE	RE	RE	RE
pr	RE	RE	RE	RE
in	CS	CS	CS	CS
de	RE	RE	RE	RE
mi	RE	RE	RE	RE
md	RE	RE	RE	RE
tr	RE	RE	RE	RE
fr	RE	RE	RE	RE
sf	RE	RE	RE	RE
sl	RE	RE	RE	RE

Table 4 ($FA_1 = CS$)

The results corresponding to families $S_g(FA_1, FIN)$ are from [11] (with the exception of $g = md$); however, the relations $S_g(FA_1, FIN) \subseteq CS, g \in \{pr, de\}$, for $FA_1 \in \{LIN, CF\}$, are improvements in the relations $S_g(FA_1, FIN) \subseteq RE$ given in [11]. The other relations are proved in the preceding section (or are direct consequences of the lemmas in the preceding section).

The empty boxes in Tables 1 and 2 indicate *open problems*. Also for the boxes where $\overline{CF}$ appears we have the problem of finding families containing the corresponding families $S_g(FA_1, FA_2)$ (we only know that they contain non-context-free languages). Do the families $S_g(REG, LIN)$, $g \in \{in, de\}$, contain non-context-free languages ?

Several remarks about the results in Table 1 are worth mentioning:

- Many families $S_g(FA_1, FA_2)$ are "almost equal" with the family RE of recursively enumerable languages (see again Lemma 6), hence new characterizations of RE are obtained in this way. We stress the fact that these characterizations are obtained by using *uniterated* splicing operations. Compare this with the series of characterizations of RE obtained in [10], where iterated splicing is involved.

– Splicing schemes with linear sets of rules are "universal" as far as power is concerned, leading in most cases (when starting from linear languages) to simple characterizations of RE; this suggests using splicing schemes (and systems: a construct $H = (V, A, R)$, where (V, R) is a splicing scheme and A is a given language of *axioms* is called a *splicing system*; a language $L(H)$ is associated with H by iterating the splicing scheme starting from A) with sublinear sets of rules (and/or a sublinear set of axioms, in the case of splicing systems), otherwise the (generative) power is too large.
– The regulated variants of the splicing operation are more powerful than the free splicing especially when starting from linear and context-free languages and using regular splicing schemes, but not for splicing schemes with linear sets of rules (already the family $S_f(FA, LIN)$ goes beyond CS); however, the pr, in, de, sf and sl variants have a significant influence on the power of the splicing.

In [10] one considers several different restricted modes of using the splicing rules: looking for leftmost/rightmost occurrences of the strings u_1u_2, u_3u_4 in the strings which are spliced, for maximal/minimal occurrences of the sites u_1u_2, u_3u_4, and so on. They are considered in [10] in the iterated mode. It remains as a research topic to investigate also their uniterated version (as operations with languages). Symmetrically, it remains to examine the generative power of splicing systems based on the splicing variants considered in [11] and in the present paper.

Notes: Research supported by the Academy of Finland, project 11281, and Grant OGP0007877 of the Natural Sciences and Engineering Research Council of Canada.

Useful remarks by two anonymous referees are gratefully acknowledged.

References

[1] K. L. Denninghoff, R. W. Gatterdam, On the undecidability of splicing systems, *Intern. J. Computer Math.*, 27 (1989), 133 – 145.
[2] R. W. Gatterdam, Splicing systems and regularity, *Intern. J. Computer Math.*, 31 (1989), 63 – 67.
[3] S. Ginsburg, *Algebraic and Automata-Theoretic Properties of Formal Languages*, North-Holland, Amsterdam, 1975.
[4] T. Head, Formal language theory and DNA: an analysis of the generative capacity of specific recombinant behaviors, *Bull. Math. Biology*, 49 (1987), 737 – 759.
[5] T. Head, Splicing schemes and DNA, in *Lindenmayer Systems; Impacts on Theoretical Computer Science and Developmental Biology* (G. Rozenberg, A. Salomaa, eds.), Springer-Verlag, Berlin, 1992, 371 – 383.
[6] M. Latteux, B. Leguy, B. Ratoandromanana, The family of one-counter languages is closed under quotient, *Acta Informatica*, 22 (1985), 579 – 588.
[7] Gh. Păun, On the splicing operation, *Discrete Appl. Math.*, to appear.
[8] Gh. Păun, The splicing as an operation on formal languages, *First IEEE Symp. on Intelligence in Neural and Biological Systems*, Washington, 1995, 176 – 180.
[9] Gh. Păun, On the power of the splicing operation, *Intern. J. Computer Math.*, 59 (1995), 27 – 35.

[10] Gh. Păun, G. Rozenberg, A. Salomaa, Computing by splicing, *Theor. Computer Sci.*, to appear.

[11] Gh. Păun, G. Rozenberg, A. Salomaa, Restricted use of the splicing operation, *Intern. J. Computer Math.*, to appear.

[12] D. Pixton, Regularity of splicing languages, *First IEEE Symp. on Intelligence in Neural and Biological Systems*, Washington, 1995.

[13] G. Rozenberg, A. Salomaa (eds.), *The Handbook of Formal Languages*, Springer-Verlag, Heidelberg, 1996.

[14] A. Salomaa, *Formal Languages*, Academic Press, New York, 1973.