A Characterization of Non-Iterated Splicing with Regular Rules*

Ray Dassen Hendrik Jan Hoogeboom
Nikè van Vugt

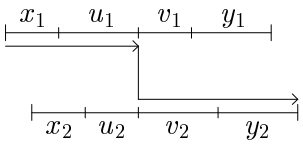
LIACS, Universiteit Leiden, The Netherlands
http://www.liacs.nl/CS/

Abstract

The family $S(\mathsf{LIN}, \mathsf{REG})$ of languages obtained by (non-iterated) splicing linear languages using regular rules does not coincide with one of the Chomsky families. We give a characterization of this family, and show that we can replace the regular rule set by a finite one.

1 Introduction

The molecular operation of cutting two DNA molecules with the help of restriction enzymes, and recombining (ligating) the halves into new molecules can be modelled within formal language theory as the *splicing* operation, as follows : two strings $x = x_1u_1v_1y_1$ and $y = x_2u_2v_2y_2$ can be spliced according to a splicing rule $r = (u_1, v_1, u_2, v_2)$ to produce another string $z = x_1u_1v_2y_2$:



Here, the strings u_1v_1 and u_2v_2 represent the specific sites where restriction enzymes cut the DNA molecules. This operation was proposed by Head [Hea87], long before the study of such operations became fashionable following Adleman's DNA implementation of an algorithm to solve the Hamiltonian Path Problem [Adl94, DNA].

The power of the splicing operation, with sets of rules classified within the Chomsky hierarchy, is investigated in [HPP97, PRS96]. We reconsider

*Published in: *Where Mathematics, Computer Science and Biology Meet* (C. Martin-Vide, V. Mitrana, eds.), Kluwer Academic Publishers, November 2000, pp. 319-327.

the non-iterated case of this operation, i.e., we consider the splicing operation as an operation on languages rather than as a language generating mechanism. In particular, we study the family of languages $S(\mathcal{F}, \mathsf{REG})$ that is obtained by splicing languages from a given family $\mathcal{F}$ using regular sets of rules. For families within the Chomsky hierarchy a precize characterization of $S(\mathcal{F}, \mathsf{REG})$ is known, except for the linear languages : $S(\mathsf{LIN}, \mathsf{REG})$ lies strictly in between the linear and context-free languages.

Here we obtain an elementary characterization of $S(\mathcal{F}, \mathsf{REG})$ in terms of $\mathcal{F}$, and moreover, we show that the regular set of rules may be replaced by a finite set. For the linear languages this then yields a new characterization of $S(\mathsf{LIN}, \mathsf{REG})$, Corollary 7.

In our final Section 5 we again try to reduce regular rule sets to finite ones, for a restricted form of splicing, where the rules may only be applied in a certain context [PRS96, KPS96]. We do not fully succeed. For the case of 'increasing mode' which we consider, the reduction is implemented at the cost of losing words of length one from the generated language.

2 Preliminaries

The empty word is denoted by λ.

A *generalized sequential machine* (gsm) is a finite state machine with additional output. It has a finite set of transitions of the form $(p, a, q, w) \in Q \times \Sigma \times Q \times \Delta^*$, where Q is the finite set of states, Σ and Δ are the input alphabet and output alphabet. Using such a transition, the machine may change from state p into state q, while reading the letter a on its input and writing the string w to its output. The gsm defines a relation in $\Sigma^* \times \Delta^*$, called a gsm mapping.

The *Chomsky hierarchy* is formed by the families $\mathsf{FIN}, \mathsf{REG}, \mathsf{LIN}, \mathsf{CF}, \mathsf{CS}$, and RE, of finite, regular, linear, context-free, context-sensitive, and recursively enumerable languages. For a language family $\mathcal{F}$ we use $\mathcal{F} \oplus \mathcal{F}$ to denote finite unions of elements of $\mathcal{F}^2$, i.e., languages of the form $K_1 \cdot L_1 \cup \ldots \cup K_n \cdot L_n$, $n \geq 0$, with $K_i, L_i \in \mathcal{F}$. Assuming $\{\lambda\}$ and $\varnothing$ are elements of $\mathcal{F}$, then $\mathcal{F} \oplus \mathcal{F}$ equals $\mathcal{F}$ iff $\mathcal{F}$ is closed under union and concatenation. Hence $\mathcal{F} \oplus \mathcal{F} = \mathcal{F}$ for each Chomsky family, except LIN.

In the sequel, we need closure under gsm mappings and under union to be able to apply our constructions. Any family having these closure properties and contains all finite languages is called *friendly*. All Chomsky families, with the exception of CS, are friendly. We need a simple technicality.

Lemma 1 *A friendly family is closed under concatenation with symbols.*

Proof. Let K be an element of the friendly family $\mathcal{F}$. We show that $K\{a\} \in \mathcal{F}$. When $\lambda \notin K$, mapping K onto $K\{a\}$ can be performed by a gsm. If K contains λ, then observe that $K - \{\lambda\} \in \mathcal{F}$ as intersection with a regular

language can be computed by a gsm. Hence, $K\{a\} = (K - \{\lambda\})\{a\} \cup \{a\} \in \mathcal{F}$, by closure under union.

A symmetric argumentation holds for $\{a\}K$. □

3 The splicing operation

We give basic notions and results concerning splicing and H systems, slightly adapted from [HPP97].

Definition 2 A *splicing rule* (over an alphabet V) is an element of $(V^*)^4$. For such a rule $r = (u_1, v_1, u_2, v_2)$ and strings $x, y, z \in V^*$ we write

$$(x, y) \vdash_r z \quad \text{iff} \quad x = x_1 u_1 v_1 y_1,\ y = x_2 u_2 v_2 y_2,\ \text{and}$$
$$z = x_1 u_1 v_2 y_2,\ \text{for some } x_1, y_1, x_2, y_2 \in V^*.$$

We say that z is obtained by *splicing* strings x and y *using* rule r. □

Definition 3 An *H system* (or *splicing system*) is a triple $h = (V, L, R)$ where V is an alphabet, $L \subseteq V^*$ is the *initial language* and $R \subseteq (V^*)^4$ is a set of splicing rules, the *splicing relation*. The (*non-iterated*) *splicing language* generated by h is defined as

$$\sigma(h) = \{\ z \in V^* \mid (x, y) \vdash_r z \text{ for some } x, y \in L \text{ and } r \in R\ \}.$$

□

Usually a splicing rule $r = (u_1, v_1, u_2, v_2)$ is given as the string $\asymp(r) = u_1 \# v_1 \$ u_2 \# v_2$ (# and \$ are special symbols not in V), i.e., $\asymp$ is a mapping from $(V^*)^4$ to $V^* \# V^* \$ V^* \# V^*$, that gives a *string representation* of each splicing rule. Now that the splicing relation R is represented by the language $\asymp(R)$, we can consider the effect of splicing with rules from a certain family of languages : for instance, what is the result of splicing linear languages with finite sets of splicing rules?

Example 4 Let $L = \{a^n b^n \mid n \geq 1\} \cup \{c^n d^n \mid n \geq 1\}$, thus $L \in \mathsf{LIN}$. Let $h = (\{a, b, c, d\}, L, R)$ be a splicing system with splicing relation $R = \{\ (b, \lambda, \lambda, c)\ \}$ consisting of a single rule. The language generated by h is $\sigma(h) = \{\ a^{n_1} b^{m_1} c^{m_2} d^{n_2} \mid n_i \geq m_i \geq 1\ (i = 1, 2)\ \}$, which is not in LIN. □

For any two families of languages $\mathcal{F}_1$ and $\mathcal{F}_2$, the family $S(\mathcal{F}_1, \mathcal{F}_2)$ of non-iterated splicing languages (obtained by splicing $\mathcal{F}_1$ languages using $\mathcal{F}_2$ rules) is defined in the obvious way :

$$S(\mathcal{F}_1, \mathcal{F}_2) = \{\ \sigma(h) \mid h = (V, L, R) \text{ with } L \in \mathcal{F}_1 \text{ and } \asymp(R) \in \mathcal{F}_2\ \}.$$

$\mathcal{F}_2$:	FIN	REG	LIN	CF	CS	RE
$\mathcal{F}_1$: FIN	FIN	FIN	FIN	FIN	FIN	FIN
REG	REG	REG	REG, LIN	REG, CF	REG, RE	REG, RE
LIN	LIN, CF	LIN, CF	RE			
CF	CF	CF				
CS	RE					
RE						

Table 1: The position of $S(\mathcal{F}_1, \mathcal{F}_2)$ in the Chomsky hierarchy

The families $S(\mathcal{F}_1, \mathcal{F}_2)$ are investigated in [Pău96] and [PRS96], for $\mathcal{F}_1$ and $\mathcal{F}_2$ in the Chomsky hierarchy. An overview of these results is presented in [HPP97], from which we copy Table 1. As an example, the optimal classification within the Chomsky families of splicing LIN languages with FIN rules is $\mathsf{LIN} \subset S(\mathsf{LIN}, \mathsf{FIN}) \subset \mathsf{CF}$. It was shown in [HvV98] that this table does not change when using the equally natural representation $u_1\#u_2\$v_1\#v_2$ instead of $u_1\#v_1\$u_2\#v_2$ for rule (u_1, v_1, u_2, v_2).

Additionally we will consider the family $S(\mathcal{F}, [1])$ of languages obtained by splicing $\mathcal{F}$ languages using rules of *radius* 1, i.e., for (u_1, u_2, u_3, u_4) we have $|u_i| \leq 1$ for $i = 1, 2, 3, 4$.

4 Unrestricted splicing

We start by considering finite sets of rules.

Lemma 5 *Let $\mathcal{F}$ be a friendly family. Then $S(\mathcal{F}, [1]) = S(\mathcal{F}, \mathsf{FIN}) = \mathcal{F} \oplus \mathcal{F}$.*

Proof. The inclusion $S(\mathcal{F}, [1]) \subseteq S(\mathcal{F}, \mathsf{FIN})$ is immediate. We prove two other inclusions to obtain the result.

First we show $S(\mathcal{F}, \mathsf{FIN}) \subseteq \mathcal{F} \oplus \mathcal{F}$. Let $h = (V, L, R)$ be an H system with a finite number of rules, and with $L \in \mathcal{F}$.

Consider the rule $r = (u_1, v_1, u_2, v_2)$. When r is applied to strings $x_1u_1v_1y_1$ and $x_2u_2v_2y_2$, then only the substrings x_1u_1 and v_2y_2 are visible in the resulting string $x_1u_1v_2y_2$. We define two languages derived from the initial language following that observation: let $L_{\langle r} = \{xu_1 \mid xu_1v_1y \in L, \text{ for some } y \in V^*\}$, and let $L_{r\rangle} = \{v_2y \mid xu_2v_2y \in L, \text{ for some } x \in V^*\}$.

Observe that both $L_{\langle r}$ and $L_{r\rangle}$ can be obtained from L by a gsm mapping, and consequently these languages are in $\mathcal{F}$. Clearly, $\sigma(h) = \bigcup_{r \in R} L_{\langle r} L_{r\rangle}$, thus, $\sigma(h) \in \mathcal{F} \oplus \mathcal{F}$.

Second, we show $\mathcal{F} \oplus \mathcal{F} \subseteq S(\mathcal{F}, [1])$. Consider $K_1 \cdot L_1 \cup \ldots \cup K_n \cdot L_n$ with $K_i, L_i \subseteq V^*$ in $\mathcal{F}$, for some alphabet V. This union is obtained by splicing the initial language $\bigcup_{i=1}^n K_i c_i \cup \bigcup_{i=1}^n c_i' L_i$ with rules $(\lambda, c_i, c_i', \lambda)$,

$i = 1, \ldots, n$, where the c_i, c_i' are new symbols. Note that the languages $K_i c_i$ and $c_i' L_i$ belong to $\mathcal{F}$ by Lemma 1. □

The equality $S(\mathcal{F}, [1]) = S(\mathcal{F}, \mathsf{FIN})$ appears as [HPP97, Lemma 3.10].

We can directly apply Lemma 5 to obtain the (known) characterizations of $S(\mathcal{F}, \mathsf{FIN})$ for the friendly families $\mathcal{F} = \mathsf{FIN}, \mathsf{REG}, \mathsf{CF}$ and RE. The equality $S(\mathsf{LIN}, \mathsf{FIN}) = \mathsf{LIN} \oplus \mathsf{LIN}$ appears to be new, although the family $\mathsf{LIN} \oplus \mathsf{LIN}$ is hinted at in the proof of Theorem 3 of [Pău96], when it is demonstrated that $S(\mathsf{LIN}, \mathsf{REG})$ is strictly included in CF.

Refining the above proof, we can extend it to regular rule sets.

Theorem 6 *Let $\mathcal{F}$ be a friendly family. Then $S(\mathcal{F}, \mathsf{FIN}) = S(\mathcal{F}, \mathsf{REG})$.*

Proof. By Lemma 5, it suffices to prove the inclusion $S(\mathcal{F}, \mathsf{REG}) \subseteq \mathcal{F} \oplus \mathcal{F}$.

Let $h = (V, L, R)$ be an H system with regular rule set, and initial language in $\mathcal{F}$. Assume $\asymp(R) \subseteq V^*\#V^*\$V^*\#V^*$ is accepted by the finite state automaton $\mathcal{A} = (Q, \Sigma, \delta, q_{in}, F)$, with $\Sigma = V \cup \{\#, \$\}$.

Now, for $p \in Q$, let

$$\begin{aligned} L_{\langle p} = \{\ xu \quad &| \quad xuvy \in L, \text{ for some } x, u, v, y \in V^*, \\ &\quad \text{such that } p \in \delta(q_{in}, u\#v)\ \}, \\ L_{p\rangle} = \{\ vy \quad &| \quad xuvy \in L, \text{ for some } x, u, v, y \in V^*, \\ &\quad \text{such that } \delta(p, u\#v) \cap F \neq \varnothing\ \} \end{aligned}$$

Observe that both $L_{\langle p}$ and $L_{p\rangle}$ can be obtained from L by a gsm mapping. For example, the gsm computing $L_{\langle p}$ guesses the start of the segment u on its input, and simulates $\mathcal{A}$ on this segment (all the time copying its input to the output). At the end of u (nondeterministically guessed), it simulates the step of $\mathcal{A}$ on # and continues to simulate $\mathcal{A}$ on the input, without writing output, while checking whether state p is reached.

Some care has to be taken here. By definition, a gsm cannot use a λ-transition to simulate $\mathcal{A}$ on the additional symbol # that is not part of the input. As a solution, the gsm may keep in its finite state the values of both $\delta(q_{in}, u')$ and $\delta(q_{in}, u'\#)$ for the prefix u' of u that has been read.

Hence the languages $L_{\langle p}$ and $L_{p\rangle}$ are in $\mathcal{F}$.

We claim that $\sigma(h) = \bigcup_{(p,\$,q)\in\delta} L_{\langle p} L_{q\rangle}$, and consequently, $\sigma(h) \in \mathcal{F} \oplus \mathcal{F}$.

We prove the claim here in one direction : Assume $z \in L_{\langle p} L_{q\rangle}$ for some $(p, \$, q) \in \delta$. Then there exist $x_1, y_1, x_2, y_2, u_1, v_1, u_2, v_2 \in V^*$ such that $z = x_1u_1 \cdot v_2y_2$, $x_1u_1v_1y_1 \in L$, $p \in \delta(q_{in}, u_1\#v_1)$, $x_2u_2v_2y_2 \in L$, and $\delta(q, u_2\#v_2) \cap F \neq \varnothing$.

As $q \in \delta(p, \$)$, we conclude that $\delta(q_{in}, u_1\#v_1\$u_2\#v_2) \cap F \neq \varnothing$, and so $r = (u_1, v_1, u_2, v_2) \in R$. Hence, $z \in \sigma(h)$, as it is obtained by splicing $x = x_1u_1v_1y_1$ and $y = x_2u_2v_2y_2$ in L using r from R. □

Again, for most of the Chomsky families (including CS which is not friendly) the last result is implicit in Table 1. Here it is obtained through direct construction. We summarize the new results obtained for LIN.

Corollary 7 $S(\mathsf{LIN}, \mathsf{FIN}) = S(\mathsf{LIN}, \mathsf{REG}) = \mathsf{LIN} \oplus \mathsf{LIN}$.

5 Restricted splicing

In this section we try to extend the result that a regular set of rules can be reduced to a finite set of rules (Theorem 6). We consider the setting where the general splicing operation $(x, y) \vdash_r z$ may only be applied in a certain context, as inspired by [PRS96, KPS96].

We splice in *increasing mode* if the result z is as least as long as both inputs x and y. Formally,

$$(x, y) \vdash_r^{in} z \quad \text{iff} \quad (x, y) \vdash_r z \text{ and } |z| \geq |x|, |z| \geq |y|.$$

Example 8 [KPS96] Let $h = (\{a, b\}, L, R)$, where $L = ca^*b^*c \cup cb^*a^*c$ and $\asymp(R) = \{ca^n\#b^nc\$c\#b^ma^mc \mid n, m \geq 1\} \in \mathsf{CF}$. Then $\sigma_{in}(h) = \{ca^nb^ma^mc \mid n, m \geq 1 \text{ and } n \leq 2m\}$, which is not context-free. □

With this restricted operation we define in the obvious way the language $\sigma_{in}(h)$ for a splicing system h. Thus, we consider the families $S_{in}(\mathcal{F}_1, \mathcal{F}_2)$, and we study the relation between $S_{in}(\mathcal{F}, \mathsf{FIN})$ and $S_{in}(\mathcal{F}, \mathsf{REG})$. We summarize the results concerning $S_{in}(\mathcal{F}_1, \mathcal{F}_2)$ obtained in [KPS96].

Proposition 9 $S_{in}(\mathsf{REG}, \mathsf{CF}) - \mathsf{CF} \neq \varnothing$, $S_{in}(\mathsf{REG}, \mathsf{REG}) \subseteq \mathsf{REG}$, *and* $S_{in}(\mathsf{CS}, \mathsf{CF}) \subseteq \mathsf{CS}$.

Observe that $S_{in}(\mathcal{F}, \mathsf{FIN}) \subseteq S_{in}(\mathcal{F}, \mathsf{REG})$ by definition. We could not show the converse inclusion $S_{in}(\mathcal{F}, \mathsf{REG}) \subseteq S_{in}(\mathcal{F}, \mathsf{FIN})$. However, the families are *almost* equal, in the sense that for every language K_r in $S_{in}(\mathcal{F}, \mathsf{REG})$ there is a language K_f in $S_{in}(\mathcal{F}, \mathsf{FIN})$ such that K_r and K_f differ only by words of length at most one.

Theorem 10 *Let $\mathcal{F}$ be a friendly family. Then $S_{in}(\mathcal{F}, \mathsf{FIN}) = S_{in}(\mathcal{F}, \mathsf{REG})$, almost (in the sense explained above).*

Proof. We develop some ideas from the proof of Theorem 6.

Let $h = (V, L, R)$ be an H system with regular rule set, and initial language in $\mathcal{F}$. We construct an H system with finite rule set that defines a language 'almost' equal to $\sigma(h)$. Assume $\asymp(R) \subseteq V^*\#V^*\$V^*\#V^*$ is accepted by the finite state automaton $\mathcal{A} = (Q, \Sigma, \delta, q_{in}, F)$, with $\Sigma = V \cup \{\#, \$\}$, and $Q \cap \Sigma = \varnothing$. Assuming that the automaton is reduced (each state lies on a path from the initial to a final state) we can split the set of

states into two disjoint subsets, $Q = Q_1 \cup Q_2$, such that Q_1 (Q_2) contains the states on a path before (after) the symbol \$ is read. Let ι be a new symbol.

First step. We construct a new initial language $L' \subseteq (V \cup Q \cup \{\iota\})^*$ from L as follows. For each $x, u, v, y \in V^*$, with $xuvy \in L$, and each $p \in Q_1$, $q \in Q_2$ we include in L' the following words, under the given constraints :

$xup\iota^k$	where	$p \in \delta(q_{in}, u\#v)$,	$\lvert vy\rvert \geq 1,\ k = \lvert vy\rvert - 1.$
xup		$p \in \delta(q_{in}, u\#v)$,	$\lvert vy\rvert = 0.$
$\iota^\ell qvy$		$\delta(q, u\#v) \cap F \neq \varnothing$,	$\lvert xu\rvert \geq 1,\ \ell = \lvert xu\rvert - 1.$
qvy		$\delta(q, u\#v) \cap F \neq \varnothing$,	$\lvert xu\rvert = 0.$

Observe that L' can be obtained from L by a gsm mapping, and consequently L' belongs to $\mathcal{F}$.

Let R' be the (finite) set of rules $\{\ (\lambda, p, q, \lambda) \mid (p, \$, q) \in \delta\ \}$. Note that every rule (λ, p, q, λ) in R' corresponds to a (regular) set of rules $\{\ (u_1, v_1, u_2, v_2) \mid p \in \delta(q_{in}, u_1\#v_1), \delta(q, u_2\#v_2) \cap F \neq \varnothing\ \}$ in R.

Let $h' = (V \cup Q \cup \{\iota\}, L', R')$.

It is easy to understand that $\sigma_{in}(h') \subseteq \sigma_{in}(h)$, following the construction of L' and R'. If $x' = x_1u_1p\iota^k$ and $y' = \iota^\ell qv_2y_2$ in L' splice in the increasing mode to give $z = x_1u_1v_2y_2$ using rule $r' = (\lambda, p, q, \lambda)$ in R', then there are strings $x = x_1u_1v_1y_1$ and $y = x_2u_2v_2y_2$ in L that splice to give again z using rule $r = (u_1, v_1, u_2, v_2)$ in R. By construction, $|x| = |x'|$ (or $|x| = |x'| - 1$ when $|vy| = 0$) we know that $|x| \leq |x'|$, thus $|x'| \leq |z|$ implies $|x| \leq |z|$. Mutatis mutandis, this argument is also valid for y, y', so x and y splice in increasing mode too.

The reverse inclusion $\sigma_{in}(h) \subseteq \sigma_{in}(h')$ in general is not true. Assume however $z \in \sigma_{in}(h)$, obtained through $(x = x_1u_1v_1y_1, y = x_2u_2v_2y_2) \vdash_r^{in} z = x_1u_1v_2y_2$, where $r = (u_1, v_1, u_2, v_2)$ in R.

By construction one finds $x' = x_1u_1p\iota^k$ and $y' = \iota^\ell qv_2y_2$ in L' for suitable $k, \ell \in \mathbb{N}$, $p, q \in Q$. Consider $r' = (\lambda, p, q, \lambda)$ corresponding to r (as discussed above). Then $(x', y') \vdash_{r'} z$. This is increasing mode under the condition that

$$|v_1y_1| + |v_2y_2| \geq 1 \text{ and } |x_1u_1| + |x_2u_2| \geq 1.$$

This is seen as follows. If $|v_1y_1| \geq 1$, then $|x'| = |x|$ and hence $|x'| \leq |z|$. Otherwise, if $|v_1y_1| = 0$, then $x' = x_1u_1p$ is one symbol longer than x in the original splicing. However, $|x'| \leq |z|$ follows from the fact that $|v_2y_2| \geq 1 = |p|$. An analogous argument holds for $|y'| \leq |z|$.

We conclude that $z \in \sigma_{in}(h')$, except in case it can only be obtained using $|v_1y_1| = |v_2y_2| = 0$, i.e., $x = z = x_1u_1$, $y = x_2u_2$, and $r = (u_1, \lambda, u_2, \lambda)$, or $|x_1u_1| = |x_2u_2| = 0$, i.e., $x = v_1y_1$, $y = z = v_2y_2$, and $r = (\lambda, v_1, \lambda, v_2)$.

Second step. In order to accommodate almost all these cases we add additional strings to the initial language L'. Extend the alphabet by two copies

of $V \times Q$, symbols which we will denote as $\langle a-p\rangle$, $\langle a+p\rangle$, $\langle q-a\rangle$, $\langle q+a\rangle$, where $a \in V, p \in Q_1$, and $q \in Q_2$.

For each $x, u, v, y \in V^*$, with $xu, vy \in L$, each $a \in V$, and each $p \in Q_1$, $q \in Q_2$ we add to L' the following words, under the given constraints :

$w\langle a-p\rangle$	where	$p \in \delta(q_{in}, u\#)$,	$wa = xu$.
$\iota^k\langle q+a\rangle a$		$\delta(q, u\#) \cap F \neq \varnothing$,	$\lvert xu\rvert \geq 2,\ k = \lvert xu\rvert - 2$.
$a\langle a+p\rangle\iota^\ell$		$p \in \delta(q_{in}, \#v)$,	$\lvert vy\rvert \geq 2,\ \ell = \lvert vy\rvert - 2$.
$\langle q-a\rangle w$		$\delta(q, \#v) \cap F \neq \varnothing$,	$aw = vy$.

To R' add the set of rules $\{\ (\lambda, \langle a-p\rangle, \langle q+a\rangle, \lambda), (\lambda, \langle a+p\rangle, \langle q-a\rangle, \lambda)\ \mid (p, \$, q) \in \delta\ \}$.

These new strings and new rules can only splice among themselves, and simulate most of the remaining splicings of the original system (with regular rule set). For instance, assuming $|x_1u_1| \geq 2$, and writing $x_1u_1 = wa$, the original system splices $(x_1u_1, x_2u_2) \vdash_r^{in} x_1u_1 = z$, with $r = (u_1, \lambda, u_2, \lambda)$, iff the new system splices $(w\langle a-p\rangle, \iota^k\langle q+a\rangle a) \vdash_{r'}^{in} wa = z$ where $r' = (\lambda, \langle a-p\rangle, \langle q+a\rangle, \lambda)$ and $|\iota^k\langle q+a\rangle a| = |x_2u_2|$ to ensure increasing mode.

Conclusion. Splicings we can *not* simulate have $v_1y_1 = v_2y_2 = \lambda$ to obtain $z = x_1u_1$, or $x_1u_1 = x_2u_2 = \lambda$ to obtain $z = v_2y_2$, in both cases with $|z| \leq 1$. Which proves the result. □

The papers [KPS96, PRS96] contain many other modes of restricted splicing. Many of these modes still lack a precise characterization. More specifically in connection with our investigations, it would be interesting to relate $S_\mu(\mathcal{F}, \mathsf{FIN})$ and $S_\mu(\mathcal{F}, \mathsf{REG})$ for each mode μ.

References

[Adl94] L.M. Adleman. Molecular computation of solutions to combinatorial problems, *Science*, 226:1021–1024, November 1994.

[DNA] A bibliography of molecular computation and splicing systems (J.H.M. Dassen, P. Frisco, eds.), at url: `http://www.liacs.nl/~pier/dna.html`.

[Hea87] T. Head. Formal language theory and DNA: an analysis of the generative capacity of specific recombinant behaviours. *Bulletin of Mathematical Biology*, 49:737–759, 1987.

[HPP97] T. Head, G. Păun, and D. Pixton. Language theory and molecular genetics : Generative mechanisms suggested by DNA recombination. In: *Handbook of Formal Languages* (G. Rozenberg and A. Salomaa, eds.), volume 2. Springer-Verlag, 1997.

[HvV98] H.J. Hoogeboom, and N. van Vugt. The power of H systems: does representation matter? In: *Computing with Bio-Molecules* (G. Păun, ed.), Springer-Verlag, Singapore, 255–268, 1998.

[KPS96] L. Kari, G. Păun, and A. Salomaa. The power of restricted splicing with rules from a regular language. *Journal of Universal Computer Science*, 2(4):224-240, 1996.

[Pău96] G. Păun. On the splicing operation. *Discrete Applied Mathematics*, 70:57–79, 1996.

[PRS96] G. Păun, G. Rozenberg, and A. Salomaa. Restricted use of the splicing operation. *International Journal of Computer Mathematics*, 60:17–32, 1996.