The power of H systems : does representation matter?

Hendrik Jan Hoogeboom
Nikè van Vugt

Dept. of Computer Science, Leiden University
P.O. Box 9512, 2300 RA Leiden, The Netherlands
e-mail : { hoogeboom, nvvugt } @ wi.leidenuniv.nl

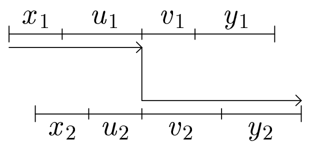
Abstract

Splicing rules of the form (u, v, w, x) are usually represented as strings of the form $u\#v\$w\#x$. The effect of splicing with rules from several families of languages has been determined in the literature.

We investigate whether the results about splicing systems obtained in this way are indeed properties of the *splicing system* and not of the specific *string representation* of the rules. We study in detail single and iterated splicing systems, by considering the alternative string representation $u\#w\$v\#x$, and indeed obtain the same classifications as for the standard representation. We briefly discuss some related representations.

1 Introduction

Analogous to the splicing of two molecules with the help of restriction enzymes to produce one or two other molecules, 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 give another string $z = x_1u_1v_2y_2$:



In the literature on splicing sytems, r is usually represented by the string $u_1\#v_1\$u_2\#v_2$, and so a set of splicing rules is a language. A classification of the generating power of splicing systems with rules defined by the six families of the Chomsky hierarchy is given in [HPP97].

Although this string representation of splicing rules is very natural, other representations are possible. The question arises whether these results are properties of the *splicing systems* or of the specific *representation* of splicing rules that is chosen. For example, would we get different results if we first write the left contexts of the rule and then the right contexts, choosing $u_1\#u_2\$v_1\#v_2$ instead of $u_1\#v_1\$u_2\#v_2$ as the string representation of (u_1, v_1, u_2, v_2) ? We will answer this question in detail for single and iterated splicing systems, and show that the families of splicing languages we consider are not influenced by this change in representation. We briefly discuss some other, related, string representations.

A preliminary version of these results was presented at the workshop on Molecular Computing, August 1997 in Mangalia, Romania. We have since included new results, most notably those from Section 5.2.

2 Preliminaries

We use classical notions from formal language theory, see, e.g, [HU79]. For notions related to splicing theory, the reader may consult the survey [HPP97]. We give some definitions here, mainly to fix our notation.

2.1 Formal language theory

For two sets A and B, $A \subset B$ denotes the proper inclusion of A in B, and $A \subseteq B$ denotes the inclusion of A in B (where A and B may be equal).

We denote the empty word by λ.

The quotient of two languages L_1 and L_2, denoted L_1/L_2, is defined as the set $\{x \mid \text{there exists } y \text{ in } L_2 \text{ such that } xy \text{ is in } L_1\}$.

A generalized sequential machine (gsm) is a 6-tuple $\mathcal{A} = (Q, V_1, V_2, \delta, q_0, F)$, where Q is the finite set of states, V_1 is the input alphabet, V_2 is the output alphabet, $q_0 \in Q$ is the initial state, $F \subseteq Q$ is the set of final states, and δ is a mapping from $Q \times V_1$ into finite subsets of $Q \times V_2^*$, the transition relation. By omitting all reference to the output alphabet V_2, we obtain a finite automaton.

We use FIN, REG, LIN, CF, CS and RE to denote the families of finite, regular, linear (context-free), context-free, context-sensitive and recursively enumerable languages, respectively. These families form the Chomsky hierarchy.

2.2 Splicing theory

We give, in an adapted form, some definitions concerning splicing rules and H systems from [HPP97].

Definition 1 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

$$\begin{aligned}(x, y) \vdash_r z \quad \text{iff} \quad & x = x_1u_1v_1y_1,\ y = x_2u_2v_2y_2, \text{ and} \\ & z = x_1u_1v_2y_2, \text{ for some } x_1, y_1, x_2, y_2 \in V^*.\end{aligned}$$

The string z is said to be obtained by *splicing* the strings x and y *using* the rule r. □

Definition 2 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*.
For a given H system $h = (V, L, R)$ we define

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

to be the (*single splicing*) *language* generated by h. □

In [HPP97], splicing rules are represented as strings rather than 4-tuples : a splicing rule $r = (u_1, v_1, u_2, v_2)$ is given as the string $Z(r) = u_1\#v_1\$u_2\#v_2$ ($\#$ and $\$$ are special symbols not in V), i.e., Z is a mapping from $(V^*)^4$ to $V^*\#V^*\$V^*\#V^*$, that gives a *string representation* of each splicing rule. We extend Z in the natural way to a mapping from splicing relations to languages. The name of this mapping is suggested by a more graphical notation for the splicing rule r, that is used in [PRS96a] :

$$\begin{array}{c|c} u_1 & v_1 \\ \hline u_2 & v_2 \end{array}$$

and by the way the diagram is then read to get the $u_1\#v_1\$u_2\#v_2$ notation.

Since a splicing relation R is now represented by the language $Z(R)$, 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 linear splicing rules?

Example 3 Let $L = (L_1 \cdot d) \cup (d \cdot L_2)$, where $L_1 = \{a^n b^n \mid n \geq 1\} \in \mathsf{LIN}$ and $L_2 = \{b^n c^n \mid n \geq 1\} \in \mathsf{LIN}$. Let $h = (\{a, b, c, d\}, L, R)$ be a splicing system with splicing relation $R = \{(a, b^i d, d, b^i c) \mid i \geq 1\}$. Then $Z(R) = \{a\#b^i d\$d\#b^i c \mid i \geq 1\} \in \mathsf{LIN}$. The language generated by h is

$$\sigma(h) = \{a^n b^n c^n \mid n \geq 1\} \notin \mathsf{LIN}.$$

□

Given two families of languages, $\mathcal{F}_1$ and $\mathcal{F}_2$, a family $S(\mathcal{F}_1, \mathcal{F}_2)$ of single splicing languages (obtained by splicing $\mathcal{F}_1$ languages with $\mathcal{F}_2$ rules) is defined in the obvious way :

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

The families $S(\mathcal{F}_1, \mathcal{F}_2)$ are investigated in [Pău96a] and [PRS96b], for $\mathcal{F}_1$ and $\mathcal{F}_2$ in the Chomsky hierarchy : FIN, REG, LIN, CF, CS, RE. An overview of these results is presented in [HPP97], upon which we base our discussions. When $S(\mathcal{F}_1, \mathcal{F}_2)$ was not found to be equal to one of these six families, the greatest lower bound $\mathcal{F}_3$ and the smallest upper bound $\mathcal{F}_4$ among them are given : $\mathcal{F}_3 \subset S(\mathcal{F}_1, \mathcal{F}_2) \subset \mathcal{F}_4$.

These results are collected in Table 1 from [HPP97], which we repeat here. $\mathcal{F}_1$ is listed from top to bottom, $\mathcal{F}_2$ from left to right. As an example, the optimal classification of splicing LIN languages with REG rules is $\mathsf{LIN} \subset S(\mathsf{LIN}, \mathsf{REG}) \subset \mathsf{CF}$.

	FIN	REG	LIN	CF	CS	RE
FIN	FIN	FIN	FIN	FIN	FIN	FIN
REG	REG	REG	REG, LIN	REG, CF	REG, RE	REG, RE
LIN	LIN, CF	LIN, CF				
CF	CF	CF				
CS				RE		
RE						

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

3 The problem

Splicing with 'matching' splicing sites may be modelled by the set of rules $R = \{(a^n, b^m, a^n, b^m) \mid m, n \geq 1\}$. This set is not 'context-free', as a consequence of the specific representation Z that we use.

Let us now consider an alternative representation : first writing the left contexts, and then the right contexts of the splicing sites. Formally we use the mapping $\text{И} : (V^*)^4 \to V^*\#V^*\$V^*\#V^*$, with $\text{И}(u_1, v_1, u_2, v_2) = u_1\#u_2\$v_1\#v_2$. Then, for the 'matching' rules above, $\text{И}(R) = \{a^n\#a^n\$b^m\#b^m \mid m, n \geq 1\}$ is a context-free language!

Consequently the question arises whether the results stated in Table 1 are properties of the splicing systems or of the specific representation of splicing rules that was chosen. In particular, would Table 1 look different if we chose И instead of Z as our string representation?

To find the answer to this question, we slightly extend the notation for the families of single splicing languages. Let $\rho : (V^*)^4 \to W^*$ be a given string representation of splicing rules over the alphabet V, for some alphabet W. Then define

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

Hence, by definition, $S(\mathcal{F}_1, \mathcal{F}_2) = S_{\mathsf{Z}}(\mathcal{F}_1, \mathcal{F}_2)$ for the standard string representation Z.

We will also directly consider the family of *splicing relations* defined by the family of languages $\mathcal{F}$ under the representation ρ,

$$\mathcal{R}_\rho(\mathcal{F}) = \{R \subseteq (V^*)^4 \mid \rho(R) \in \mathcal{F}\}.$$

In the next section we investigate whether the splicing relations defined by the language families from the Chomsky hierarchy are changed when we move from the Z-representation to the И-representation. In other words, we determine whether or not $\mathcal{R}_{\mathsf{Z}}(\mathcal{F}_2) = \mathcal{R}_{\text{И}}(\mathcal{F}_2)$. It is clear from the example above that for $\mathcal{F}_2 = \mathsf{CF}$ this is *not* the case. Obviously $\mathcal{R}_{\mathsf{Z}}(\mathcal{F}_2) = \mathcal{R}_{\text{И}}(\mathcal{F}_2)$ implies $S_{\mathsf{Z}}(\mathcal{F}_1, \mathcal{F}_2) = S_{\text{И}}(\mathcal{F}_1, \mathcal{F}_2)$ for all $\mathcal{F}_1$.

In Section 5 we consider the remaining cases (for which $\mathcal{R}_{\mathsf{Z}}(\mathcal{F}_2) \neq \mathcal{R}_{\text{И}}(\mathcal{F}_2)$) and we prove that even for those families we have $S_{\mathsf{Z}}(\mathcal{F}_1, \mathcal{F}_2) = S_{\text{И}}(\mathcal{F}_1, \mathcal{F}_2)$.

In Section 6 we investigate the effect of using the И-representation instead of the Z-representation for iterated splicing systems, while in Section 7 we discuss related string representations.

4 Families of splicing relations

In this section we compare the families of splicing relations defined by the two string representations of the splicing rules.

Observe that Z and И define a one-to-one correspondence between a splicing rule (u, v, w, x) and the string representations $u\#v\$w\#x$ and $u\#w\$v\#x$ of this splicing rule, respectively. Hence, when considering the Z-representation, one has $\mathsf{Z}(R) = L$ iff $R = \mathsf{Z}^{-1}(L)$ and $L \subseteq V^*\#V^*\$V^*\#V^*$. Consequently we may write $\mathcal{R}_{\mathsf{Z}}(\mathcal{F}) = \{\mathsf{Z}^{-1}(L) \mid L \subseteq V^*\#V^*\$V^*\#V^* \text{ and } L \in \mathcal{F}\}$.

Proving that $\mathcal{R}_{\mathsf{Z}}(\mathcal{F}) \subseteq \mathcal{R}_{\text{И}}(\mathcal{F})$ amounts to verifying that $\text{И}\mathsf{Z}^{-1}(L) \in \mathcal{F}$ for every $L \in \mathcal{F}$ with $L \subseteq V^*\#V^*\$V^*\#V^*$. Note that this is a *closure property* of the family $\mathcal{F}$; closure under the operation $\text{И}\mathsf{Z}^{-1}$ that maps a string $u\#v\$w\#x$ to the string $u\#w\$v\#x$.

The converse inclusion $\mathcal{R}_{\mathsf{Z}}(\mathcal{F}) \supseteq \mathcal{R}_{\text{И}}(\mathcal{F})$ then follows immediately, as $\text{И}\mathsf{Z}^{-1} = \mathsf{Z}\text{И}^{-1}$; the operation is its own inverse.

4.1 FIN, REG, CS and RE splicing rules

In this subsection, we prove that $\mathcal{R}_{\mathsf{Z}}(\mathcal{F}) = \mathcal{R}_{\text{И}}(\mathcal{F})$, for $\mathcal{F} \in \{\mathsf{FIN}, \mathsf{REG}, \mathsf{CS}, \mathsf{RE}\}$.

For the finite languages, it should be clear that the two representations are equivalent : if R is a finite splicing relation, then both $\mathsf{Z}(R)$ and $\text{И}(R)$ are finite languages.

For the regular, context-sensitive and recursively enumerable languages, we use the following lemma.

Lemma 4 REG, CS *and* RE *are closed under* $\text{И}\mathsf{Z}^{-1}$.

Proof. The mapping $\text{И}\mathsf{Z}^{-1}$ can be realized by a 2-way deterministic generalized sequential machine (a finite state device with a 2-way input tape and a 1-way output tape, see [AU70]) : on input $u\#v\$w\#x$ it outputs $u\#$, skips v, outputs $w\$$, returns on the input to the first $\#$, outputs $v\#$, skips w and finally outputs x. Since $\text{И}\mathsf{Z}^{-1}$ is its own inverse, it can also be realized by an inverse 2dgsm mapping.

The result now follows from the closure of REG, CS and RE under inverse 2dgsm mappings, see [AU70, Theorem 2].

Note that the closure of REG under inverse 2dgsm mappings follows quite easily from the fact that 2-way finite state automata accept only regular languages, cf. [HU79, Theorem 2.5]. □

Consequently we have equality for the two classes of splicing relations for the families under consideration, and thus for H systems with FIN, REG, CS or RE splicing rules, representation does *not* matter.

Corollary 5 $\mathcal{R}_{\mathsf{Z}}(\mathcal{F}) = \mathcal{R}_{\text{И}}(\mathcal{F})$ *for* $\mathcal{F} = \mathsf{REG}, \mathsf{CS}, \mathsf{RE}$.

Theorem 6 $S_{\mathsf{Z}}(\mathcal{F}_1, \mathcal{F}_2) = S_{\text{И}}(\mathcal{F}_1, \mathcal{F}_2)$ *for* $\mathcal{F}_2 = \mathsf{FIN}, \mathsf{REG}, \mathsf{CS}, \mathsf{RE}$.

4.2 LIN splicing rules

In Section 3 we have already seen that $\mathcal{R}_{\mathsf{Z}}(\mathsf{CF}) \neq \mathcal{R}_{\text{И}}(\mathsf{CF})$. In this subsection we additionally show the same inequality for LIN.

Consider the splicing relation $R = \{(a^p, c^q, b^r, d^s) \mid p, q, r, s \geq 1 \text{ and } p+q = r+s\}$. Then the Z-representation of R is a linear language, but the И-representation is not. To prove the latter, we use Lemma 2 from [Gre79], which we repeat here.

Lemma 7 *Let* $L \subseteq a^+b^+c^+d^+$ *be a language such that*

1. $a^n b^n c^k d^k \in L$ *for all* $n, k \geq 1$,

2. *if* $a^n b^n c^k d^\ell$ *is in* L, *then* $k \leq \ell$, *and*

3. there are integers $t_1, t_2 \geq 1$ such that, if $a^n b^m c^k d^\ell$ is in L and $n > m$, then $(n-m)t_1 \leq (k+\ell)t_2$.

Then L is not linear context-free.

Lemma 8 $L = \{a^p b^r c^q d^s \mid p,q,r,s \geq 1$ *and* $p+q = r+s\} \notin \mathsf{LIN}$.

Proof. Lemma 7 is applicable, because obviously $L \subseteq a^+b^+c^+d^+$, conditions (1) and (2) hold, and moreover, taking $t_1 = t_2 = 1$, we can see that (3) also holds. Consequently, L is not linear context-free. □

Since LIN is closed under homomorphisms (see [HU79]), from Lemma 8 it follows that $\{a^p\#b^r\$c^q\#d^s \mid p,q,r,s \geq 1 \text{ and } p+q = r+s\} \notin \mathsf{LIN}$ and consequently $\mathcal{R}_{\mathrm{Z}}(\mathsf{LIN}) \neq \mathcal{R}_{\text{И}}(\mathsf{LIN})$.

Theorem 9 $\mathcal{R}_{\mathrm{Z}}(\mathcal{F}) \neq \mathcal{R}_{\text{И}}(\mathcal{F})$ *for* $\mathcal{F} = \mathsf{LIN}, \mathsf{CF}$.

In a way, for these splicing relations, representation *does* matter!

5 Families of single splicing languages

From Section 4.1 we know that $\mathcal{R}_{\mathrm{Z}}(\mathcal{F}_2) = \mathcal{R}_{\text{И}}(\mathcal{F}_2)$ for $\mathcal{F}_2 \in \{\mathsf{FIN}, \mathsf{REG}, \mathsf{CS}, \mathsf{RE}\}$, and thus that $S_{\mathrm{Z}}(\mathcal{F}_1, \mathcal{F}_2) = S_{\text{И}}(\mathcal{F}_1, \mathcal{F}_2)$ for these families $\mathcal{F}_2$ and each of the six families $\mathcal{F}_1$ considered here.

For $\mathcal{F}_2 \in \{\mathsf{LIN}, \mathsf{CF}\}$, however, we have demonstrated that $\mathcal{R}_{\mathrm{Z}}(\mathcal{F}_2) \neq \mathcal{R}_{\text{И}}(\mathcal{F}_2)$, and consequently, we still have to investigate the situation for these two possibilities for $\mathcal{F}_2$.

Because of the nature of the classifications given in the LIN and CF columns of Table 1, we consider the splicing of non-REG languages apart from the splicing of REG-languages.

For the remainder of this section, let $\mathcal{F}_2 \in \{\mathsf{LIN}, \mathsf{CF}\}$.

5.1 Splicing non-REG languages with LIN or CF splicing rules

We show that $S_{\mathrm{Z}}(\mathcal{F}_1, \mathcal{F}_2) = S_{\text{И}}(\mathcal{F}_1, \mathcal{F}_2)$, for $\mathcal{F}_1 \neq \mathsf{REG}$, by demonstrating that the results used in [HPP97] to fill the corresponding part of Table 1 also hold when the И-representation is used. These results are the following :

1. $S_{\mathrm{Z}}(\mathsf{FIN}, \mathcal{F}_2) \subseteq \mathsf{FIN}$ (obvious),
2. $\mathcal{F}_1 \subseteq S_{\mathrm{Z}}(\mathcal{F}_1, \mathcal{F}_2)$ [HPP97, Lemma 3.2],
3. $L_1/L_2 \in S_{\mathrm{Z}}(\mathcal{F}_2, \mathcal{F}_2)$ for each $L_1, L_2 \in \mathcal{F}_2$ [HPP97, Lemma 3.7].

The proof of (1) is independent of the splicing rules, therefore $S_{\text{И}}(\mathsf{FIN}, \mathcal{F}_2) \subseteq \mathsf{FIN}$ holds. In the proof of (2), only one splicing rule is used, (λ, c, c, λ), for which the Z-representation is equal to the И-representation. For the splicing relation used to prove (3), which is $R = \{(\lambda, wc, c, \lambda) \mid w \in L_2\}$, where $L_2 \in \mathcal{F}_2, L_2 \subseteq V^*$ and $c \notin V$, it should be clear that both $\mathrm{Z}(R) = \#L_2c\$c\#$ and $\text{И}(R) = \#c\$L_2c\#$ belong to $\mathcal{F}_2$.

By (1) and (2) we have $\mathsf{FIN} \subseteq S_{\text{И}}(\mathsf{FIN}, \mathcal{F}_2) \subseteq \mathsf{FIN}$. As each RE-language is the quotient of two linear languages, from (3) the inclusion $\mathsf{RE} \subseteq S_{\text{И}}(\mathsf{LIN}, \mathsf{LIN})$ $\subseteq S_{\text{И}}(\mathsf{LIN}, \mathsf{CF})$ follows.

Hence this part of the table does not change, and since exact classifications are found, we have the following result.

Theorem 10 *$S_{\mathsf{Z}}(\mathcal{F}_1, \mathcal{F}_2) = S_{\text{И}}(\mathcal{F}_1, \mathcal{F}_2)$ for $\mathcal{F}_1 \neq \mathsf{REG}$, $\mathcal{F}_2 = \mathsf{LIN}, \mathsf{CF}$.*

5.2 Splicing REG languages with LIN or CF splicing rules

We now show that $S_{\mathsf{Z}}(\mathsf{REG}, \mathcal{F}_2) = S_{\text{И}}(\mathsf{REG}, \mathcal{F}_2)$ also holds, by giving a direct proof.

We start by providing a normal form for splicing systems with regular initial language (and rules from a family that is closed under gsm mappings). According to this normal form, every splicing rule is of the form (u, p, q, x), where u and x are strings, while p and q are *symbols.*

This normal form is suggested by the fact that, when a splicing rule (u_1, v_1, u_2, v_2) is applied, the strings v_1 and u_2 do not appear in the result. We only need the fact that the initial strings have these substrings next to the cutting points, which appears to be a finite state property. However, the interchange of these two strings causes the fact that $\mathcal{R}_{\mathsf{Z}}(\mathsf{LIN}) \neq \mathcal{R}_{\text{И}}(\mathsf{LIN})$ and $\mathcal{R}_{\mathsf{Z}}(\mathsf{CF}) \neq \mathcal{R}_{\text{И}}(\mathsf{CF})$, as explained in Section 4.2. If we are able to restrict v_1 and u_2 to symbols rather than strings, we do not have this problem.

Let L be a regular language, that is accepted by a finite automaton $\mathcal{A} = (Q, V_1, \delta, q_0, F)$ with $Q \cap V_1 = \varnothing$, which is 'reduced', i.e., every state in Q occurs on a path from the initial state to a final state.

Let $p \in Q$ and $u \in V_1^*$. We use $p\xrightarrow{u}$ to denote the fact that p has an outgoing path with label u in the state transition diagram of $\mathcal{A}$, i.e., $\delta(p, u) \neq \varnothing$. Similarly, we write $\xrightarrow{u}p$ if p has an incoming path with label u, i.e., $p \in \delta(q, u)$ for some $q \in Q$. We introduce two copies, Q' and Q'', of the set Q.

Consider the splicing rule (u_1, v_1, u_2, v_2). We replace v_1 and u_2 with symbols that convey essentially the same information, by changing (u_1, v_1, u_2, v_2) into (u_1, p_1', p_2'', v_2) where $p_1' \in Q'$ and $p_2'' \in Q''$ such that $p_1\xrightarrow{v_1}$ and $\xrightarrow{u_2}p_2$. That is, v_1 is replaced by a state in $\mathcal{A}$ from which we can read v_1, and u_2 is replaced by a state in $\mathcal{A}$ where we can arrive after reading u_2.

Having adapted the splicing rules, we change the initial language to fit the new rules. Let $L_{\rightarrow p}$ be the language accepted by $\mathcal{A}_{\rightarrow p} = (Q, V_1, \delta, q_0, \{p\})$, and similarly let $L_{p\rightarrow}$ be the language accepted by $\mathcal{A}_{p\rightarrow} = (Q, V_1, \delta, p, F)$. Now consider the language

$$L' = \bigcup_{p \in Q} ((L_{\rightarrow p} \cdot p') \cup (p'' \cdot L_{p\rightarrow}))$$

over the extended alphabet $Q' \cup Q'' \cup V_1$. Since both $L_{\rightarrow p}$ and $L_{p\rightarrow}$ are regular, L' is a regular language.

Lemma 11 *Splicing $L \in \mathsf{REG}$ with splicing relation R yields the same language as splicing $L' \in \mathsf{REG}$ with the adapted set of splicing rules $R' = \{(u_1, p_1', p_2'', v_2) \mid (u_1, v_1, u_2, v_2) \in R$ and $p_1' \in Q', p_2'' \in Q''$ such that $p_1\xrightarrow{v_1}$ and $\xrightarrow{u_2}p_2\}$.*

Proof. From the construction above, it is clear that $x = x_1u_1v_1y_1$ and $y = x_2u_2v_2y_2$ are in L if and only if $x' = x_1u_1{\cdot}p'_1 \in L_{\to p_1}{\cdot}p'_1$ and $y' = p''_2{\cdot}v_2y_2 \in p''_2{\cdot}L_{p_2\to}$, for some p_1 such that $p_1\xrightarrow{v_1}$ and p_2 such that $\xrightarrow{u_2}p_2$. Moreover, $r = (u_1, v_1, u_2, v_2) \in R$ if and only if $r' = (u_1, p'_1, p''_2, v_2) \in R'$.

Consequently, $(x, y) \vdash_r x_1u_1v_2y_2$ if and only if $(x', y') \vdash_{r'} x_1u_1v_2y_2$. $\square$

Let $\mathcal{F}$ be a family of languages that is closed under gsm mappings. Consider the Z-representation of a splicing rule, $u_1\#v_1\$u_2\#v_2$. The translation of $u_1\#v_1\$u_2\#v_2$ into $u_1\#p'_1\$p''_2\#v_2$ with $p'_1 \in Q'$ and $p''_2 \in Q''$ such that $p_1\xrightarrow{v_1}$ and $\xrightarrow{u_2}p_2$ can be realized by a gsm mapping, that simulates the transition diagram of $\mathcal{A}$. Hence there exists an effective construction that transforms an H system of $(\mathsf{REG}, \mathcal{F})$-type into an equivalent H system of $(\mathsf{REG}, \mathcal{F})$-type that is in normal form.

Furthermore, the translation of the Z-representation of a rule *in normal form* into the $\reflectbox{$\mathsf{N}$}$-representation (i.e., $u_1\#p'_1\$p''_2\#v_2 \mapsto u_1\#p''_2\$p'_1\#v_2$) can also be realized by a gsm mapping. Consequently, for a splicing relation R in normal form, $\mathsf{Z}(R) \in \mathcal{F}$ if and only if $\reflectbox{$\mathsf{N}$}(R) \in \mathcal{F}$.

Hence, we have the following result, which is applicable for $\mathcal{F} = \mathsf{LIN}, \mathsf{CF}$.

Theorem 12 $S_{\mathsf{Z}}(\mathsf{REG}, \mathcal{F}) = S_{\reflectbox{$\scriptstyle\mathsf{N}$}}(\mathsf{REG}, \mathcal{F})$ *whenever $\mathcal{F}$ is closed under gsm mappings.*

Summarizing the results from Sections 4 and 5, we have the equality $S_{\mathsf{Z}}(\mathcal{F}_1, \mathcal{F}_2) = S_{\reflectbox{$\scriptstyle\mathsf{N}$}}(\mathcal{F}_1, \mathcal{F}_2)$ for all $\mathcal{F}_1, \mathcal{F}_2$ in the Chomsky hierarchy.

6 Iterated splicing systems

We discuss one other type of splicing system that is treated in [HPP97] : iterated splicing. We start by repeating some definitions, again in a slightly adapted form.

Definition 13 The *iterated splicing language* $\sigma^*(h)$ generated by an H system $h = (V, L, R)$ is defined by

$$\begin{aligned}
\sigma^0(h) &= L, \\
\sigma^{i+1}(h) &= \sigma^i(h) \cup \sigma(\sigma^i(h)), i \geq 0, \text{ and} \\
\sigma^*(h) &= \bigcup_{i\geq 0} \sigma^i(h).
\end{aligned}$$

$\square$

Let ρ be a given string representation of splicing rules over the alphabet V. Similar to the uniterated case, families of iterated splicing languages are defined :

$$H_\rho(\mathcal{F}_1, \mathcal{F}_2) = \{\sigma^*(h) \mid h = (V, L, R) \text{ with } L \in \mathcal{F}_1 \text{ and } \rho(R) \in \mathcal{F}_2\}$$

for $\mathcal{F}_1, \mathcal{F}_2 \in \{\mathsf{FIN}, \mathsf{REG}, \mathsf{LIN}, \mathsf{CF}, \mathsf{CS}, \mathsf{RE}\}$. A first notable result was obtained in [Hea87], namely that iterated splicing of REG languages by FIN rules does not lead outside REG : $H_{\mathsf{Z}}(\mathsf{REG}, \mathsf{FIN}) = \mathsf{REG}$. These families were further investigated for $\rho = \mathsf{Z}$ in [CH91], [Pix96], [Pău96a], [Pix95] and [Pău96b], and the results are

	FIN	REG	LIN	CF	CS	RE
FIN	FIN, REG	FIN, RE	FIN, RE	FIN, RE	FIN, RE	FIN, RE
REG	REG	REG, RE	REG, RE	REG, RE	REG, RE	REG, RE
LIN	LIN, CF	LIN, RE	LIN, RE	LIN, RE	LIN, RE	LIN, RE
CF	CF	CF, RE	CF, RE	CF, RE	CF, RE	CF, RE
CS	CS, RE	CS, RE	CS, RE	CS, RE	CS, RE	CS, RE
RE	RE	RE	RE	RE	RE	RE

Table 2: The position of $H_{Z}(\mathcal{F}_1, \mathcal{F}_2)$ in the Chomsky hierarchy

listed in Table 2 from [HPP97], which we repeat here. Again, $\mathcal{F}_1$ is listed from top to bottom and $\mathcal{F}_2$ from left to right.

The question is whether this table will change when the $\bowtie$-representation rather than the Z-representation is used. Since we know from Section 4 that $\mathcal{R}_{Z}(\mathcal{F}) = \mathcal{R}_{\bowtie}(\mathcal{F})$ for $\mathcal{F} \in \{\mathsf{FIN}, \mathsf{REG}, \mathsf{CS}, \mathsf{RE}\}$, while $\mathcal{R}_{Z}(\mathsf{LIN}) \neq \mathcal{R}_{\bowtie}(\mathsf{LIN})$ and $\mathcal{R}_{Z}(\mathsf{CF}) \neq \mathcal{R}_{\bowtie}(\mathsf{CF})$, we only have to check whether the results used in [HPP97] to fill the LIN and CF columns of Table 2 also hold when the $\bowtie$-representation is used. Those results are the following :

1. $\mathcal{F}_1 \subseteq H_{Z}(\mathcal{F}_1, \mathcal{F}_2)$ [HPP97, Lemma 3.12],
2. $\mathcal{F}_1 \subset H_{Z}(\mathcal{F}_1, \mathcal{F}_2)$ for $\mathcal{F}_1 \in \{\mathsf{REG}, \mathsf{LIN}, \mathsf{CF}, \mathsf{CS}\}$ [HPP97, Lemma 3.13],
3. $H_{Z}(\mathsf{FIN}, \mathsf{FIN})$ contains infinite languages [HPP97, discussion in proof of Theorem 3.3],
4. For all $L \subseteq V^*, L \in \mathcal{F}_1$ and $c, d \notin V$ we have $L' = (dc)^* L (dc)^* \cup c(dc)^* L (dc)^* d \notin H_{Z}(\mathcal{F}_1, \mathcal{F}_2)$ [HPP97, Lemma 3.16],
5. $H_{Z}(\mathcal{F}_1, \mathcal{F}_2) \nsubseteq \mathsf{CS}$ for $\mathcal{F}_2 \neq \mathsf{FIN}$ [HPP97, Lemma 3.15].

In the proofs of (1) and (3) the Z-representation of the splicing relation is in FIN, for (4) it is arbitrary, and for (5) it is in REG. These families are closed under $\bowtie Z^{-1}$, hence the same results are obtained when we use the $\bowtie$-representation.

In the proof of (2) from a splicing relation R a new splicing relation $R' = \{(u_1, cv_1, u_2c, v_2) \mid (u_1, v_1, u_2, v_2) \in R\}$ is constructed. Then $Z(R')$ can be obtained from $Z(R)$ by changing the two symbols # into $\#c$ and $c\#$, respectively; $\bowtie(R')$ can be obtained from $\bowtie(R)$ by changing the \$ into $c\$c$. The families in the Chomsky hierarchy are closed under these operations.

For $\mathcal{F}_1 \neq \mathsf{RE}$, by (1), (2), (3) we have $\mathcal{F}_1 \subset H_{\bowtie}(\mathcal{F}_1, \mathcal{F}_2)$, while from (4) and (5) it follows that the smallest upper bound for $H_{\bowtie}(\mathcal{F}_1, \mathcal{F}_2)$ is RE. By (1) immediately $\mathsf{RE} \subseteq H_{\bowtie}(\mathsf{RE}, \mathcal{F}_2)$.

Consequently, Table 2 does not change when we use the $\bowtie$-representation instead of the Z-representation. Note, however, that we have *not* proved that $H_{Z}(\mathcal{F}_1, \mathcal{F}_2) = H_{\bowtie}(\mathcal{F}_1, \mathcal{F}_2)$, for $\mathcal{F}_2 = \mathsf{LIN}, \mathsf{CF}$, except for the obvious case in which $\mathcal{F}_1 = \mathsf{RE}$ where upper and lower bound coincide with RE.

7 Other representations

We have considered one alternative string representation for splicing relations. Our representation separates left and right contexts rather than the two initial strings. However, there are 24 possible representations in the '#\$#'-style, corresponding to the permutations of the four components of the splicing rules. We do not claim that all these permutations have a natural interpretation. In the sequel we discuss the remaining possibilities in a rather informal way.

7.1 Splicing with $\mathsf{FIN}, \mathsf{REG}, \mathsf{CS}$ or RE rules

For (single or iterated) splicing with $\mathsf{FIN}, \mathsf{REG}, \mathsf{CS}$ or RE splicing rules, it does not matter which of the '#\$#'-representations is used : results as those of Section 4.1 hold for each of these representations. In other words, if ρ is one of these representations, then $\mathcal{R}_{\mathsf{Z}}(\mathcal{F}_2) = \mathcal{R}_\rho(\mathcal{F}_2)$ and consequently $S_{\mathsf{Z}}(\mathcal{F}_1, \mathcal{F}_2) = S_\rho(\mathcal{F}_1, \mathcal{F}_2)$ and $H_{\mathsf{Z}}(\mathcal{F}_1, \mathcal{F}_2) = H_\rho(\mathcal{F}_1, \mathcal{F}_2)$, for $\mathcal{F}_2 = \mathsf{LIN}, \mathsf{REG}, \mathsf{CS}, \mathsf{RE}$.

7.2 Splicing with LIN or CF rules

Single splicing non-REG languages with LIN or CF splicing rules yields the same classification in each one of the '#\$#'-representations, because the results used in [HPP97] to fill the corresponding part of Table 1 are easily seen to hold for all string representations ρ in this style, cf. Section 5.1. Since in these classifications upper and lower bound coincide, this implies that $S_{\mathsf{Z}}(\mathcal{F}_1, \mathcal{F}_2) = S_\rho(\mathcal{F}_1, \mathcal{F}_2)$, for $\mathcal{F}_1 \neq \mathsf{REG}$ and $\mathcal{F}_2 = \mathsf{LIN}, \mathsf{CF}$.

In the case of single splicing REG languages with CF splicing rules, we use the normal form of Section 5.2 for the rules, that makes it possible to change the Z-representation $u\#p\$q\#x$ of a splicing rule into the И-representation $u\#q\$p\#x$ by applying a gsm mapping (recall that this is possible only because p and q are symbols). Obviously, there exist gsm mappings that map $u\#p\$q\#x$ into each of the 12 representations in which u precedes x. To see that the other 12 possibilities can also be obtained by operations preserving context-freeness, note that CF is closed under the operation CYCLE, that can move x in front of u [HU79, Exercise 6.4 c]. Again, this shows that $S_{\mathsf{Z}}(\mathsf{REG}, \mathsf{CF}) = S_\rho(\mathsf{REG}, \mathsf{CF})$.

For single splicing REG languages with LIN rules, however, we give an example that shows that the '#\$#'-representations of a rule (u, v, w, x) in which x precedes u are *not* equivalent to the Z-representation.

Example 14 Consider a splicing sytem $h = (\{a, b, c, d\}, L, R)$ with $L = c \cdot \{a, b\}^* \cdot d$ and $R = \{(cb^j, d, c, a^n b^i d) \mid i, j, n \geq 1 \text{ and } i + j = n\}$. Then the 'reverse' representation of R (i.e., the representation that maps (u, v, w, x) into $x\#w\$v\#u$) is $R_r = \{a^n b^i d\#c\$d\#cb^j \mid i, j, n \geq 1 \text{ and } i + j = n\}$, which is a linear language. The single splicing language generated by h, $\sigma(h) = \{cb^j a^n b^i d \mid i, j, n \geq 1 \text{ and } i + j = n\}$, is *not* in LIN (by the pumping lemma for linear languages [HU79, Exercise 6.11]). Since $S_{\mathsf{Z}}(\mathsf{REG}, \mathsf{LIN}) \subset \mathsf{LIN}$, clearly this 'reverse' representation is not equivalent to the Z-representation. □

Of course, all representations ρ in which u precedes x *are* equivalent to the Z-representation, by using the same arguments as in the CF case. So for those representations we have $S_{\mathsf{Z}}(\mathsf{REG}, \mathsf{LIN}) = S_\rho(\mathsf{REG}, \mathsf{LIN})$.

Again, for iterated splicing, it is easy to see that the results used in [HPP97] to fill the LIN and CF columns of Table 2 hold for every '#$#'-representation, by observations as in Section 6. Hence this part of the table does not change either.

Summarizing the results of this final section, we see that for iterated splicing the classifications in Table 2 do not change when we use one of the representations in the '#$#'-style, but we do not yet know whether or not all *families* of iterated splicing languages stay the same.

For single splicing systems, however, we have seen that all '#$#'-representations are equivalent, except for the twelve LIN cases mentioned above.

Acknowledgements

We wish to thank Gh. Păun for inviting us to the workshop in Mangalia, G. Rozenberg for his support, and Jurriaan Hage for his comments on a previous version of this paper.

References

[AU70] Alfred V. Aho and Jeffrey D. Ullman. A characterization of two-way deterministic classes of languages. *Journal of Computer and System Sciences*, 4(6):523–538, 1970.

[CH91] Karel Culik II and Tero Harju. Splicing semigroups of dominoes and DNA. *Discrete Applied Mathematics*, 31:261–277, 1991.

[Gre79] Sheila A. Greibach. Linearity is polynomially decidable for realtime pushdown store automata. *Information and Control*, 42(1):27–37, 1979.

[Hea87] Thomas Head. Formal language theory and DNA : an analysis of the generative capacity of specific recombinant behaviors. *Bulletin of Mathematical Biology*, 49(6):737–759, 1987.

[HPP97] Thomas Head, Gheorghe Păun, and Dennis Pixton. Language theory and molecular genetics : Generative mechanisms suggested by DNA recombination. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages*, volume 2. Springer-Verlag, 1997.

[HU79] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to automata theory, languages, and computation.* Addison-Wesley, 1979.

[Pău96a] Gheorghe Păun. On the splicing operation. *Discrete Applied Mathematics*, 70:57–79, 1996.

[Pău96b] Gheorghe Păun. Regular extended H systems are computationally universal. *Journal of Automata, Languages and Combinatorics*, 1(1):27–36, 1996.

[Pix95] Dennis Pixton. Linear and circular splicing systems. In *Proceedings of the 1st International Symposium on Intelligence in Neural and Biological Systems*, pages 38–45. IEEE, 1995.

[Pix96] Dennis Pixton. Regularity of splicing languages. *Discrete Applied Mathematics*, 69:101–124, 1996.

[PRS96a] Gheorghe Păun, Grzegorz Rozenberg, and Arto Salomaa. Computing by splicing. *Theoretical Computer Science*, 168:321–336, 1996.

[PRS96b] Gheorghe Păun, Grzegorz Rozenberg, and Arto Salomaa. Restricted use of the splicing operation. *International Journal of Computer Mathematics*, 60:17–32, 1996.