1-splicing vs. 2-splicing: separating results

Sergey Verlan[1] Rosalba Zizza[2]

[1] Laboratoire d'Informatique Théorique et Appliquée
Université de Metz, France
E-mail: verlan@sciences.univ-metz.fr

[2] Dipartimento di Informatica ed Applicazioni
Università di Salerno, 84081 Baronissi (SA), Italy
E-mail: zizza@unisa.it

Abstract

Splicing is a binary operation on strings defined by T. Head in 1987 which is inspired by the cut and paste phenomenon on DNA molecules [2]. Based on this operation, *splicing systems*, or *H systems*, were proposed as a generative device of languages: starting from a set of words, by repeated application of the splicing operation, new words may be generated. When we consider Păun's definition, two types of H systems may be defined, according to the fact that one word or two words are produced as a result of the splicing operation, called *1-splicing* and *2-splicing* operation, respectively. While the power of splicing languages was investegated in comparison with classes of Chomsky hierarchy it wasn't known if 1-splicing and 2-splicing languages represent the same class. In this paper we present non trivial classes of regular languages which are generated by using the 1-splicing operation (over a finite H system) but which cannot be generated by using the 2-splicing operation. Thus, we show that 1-splicing operation is more powerful than the 2-splicing operation.

1 Introduction

Splicing systems were proposed by T. Head in 1987 as a generative device of languages, formal counterpart of the recombinant process of DNA molecules (*splicing*) [2]. Two strands of DNA are cut at specified substrings (sites) recognized by restriction enzymes and then the fragments are pasted by ligase enzymes, so new molecules may be produced. Head was concerned with the structure of the languages of those DNA molecules (strings) which could be produced through the splicing operation performed by a *splicing system* consisting of a finite set of initial DNA molecules (axioms) and a finite set of enzymes (splicing rules). He showed that, under some conditions, the language generated by the repeated applications of splicing rules starting with a (finite) set of axioms is strictly locally testable [3]. From his pioneering work, several variants of the splicing operation were proposed and different variants of splicing systems were introduced as well as a more general model of a splicing system having an infinite number of axioms and an infinite set of rules (see [4, 9] for a complete survey).

In this paper we deal with H systems, *i.e.*, triples $H = (V, A, R)$, where V is a finite alphabet, A is a set of axioms (words) and R is a set of rules. The (splicing) language generated by an H system is the smallest language containing A which is closed under the application of rules from R (see Section 2 for definitions). The computational power of H systems depends on which level in the Chomsky hierarchy A and R are situated. It can reach the power of Turing machines

[4, 9]. At the lowest level, the languages generated by a regular set of axioms and a finite set of rules are proved to be regular [7]. This result was extended to a set of axioms which belong to a full AFL [8]. On the other hand, it is known that the class of languages generated by *finite* H systems, *i.e.*, H systems where both A and R are finite sets, is a proper subclass of the family of regular languages, but its structure is still unknown [4, 9].

In the literature, two types of the splicing operation have been considered: the *1-splicing* operation, when only *one* word is generated when a rule is applied to two words and the *2-splicing* operation, when *two* words are generated as a result. It is worthy of note that the membership of the splicing language to the family of Chomsky hierarchy does not depend on this choice, *i.e.*, if we fix two families of languages for axioms and splicing rules then H_1-systems (H systems generating languages through the 1-splicing operation) are at the same level as H_2-systems (H systems generating languages through the 2-splicing operation) with respect to Chomsky hierarchy. In particular, let us denote by Fin the class of finite languages and by $H_1(Fin, Fin)$ (resp. $H_2(Fin, Fin)$) the class of languages generated by *finite* H_1-systems (resp. H_2-systems). Both classes $H_1(Fin, Fin)$ and $H_2(Fin, Fin)$ are proper subclasses of the class of regular languages [9].

As a consequence, it is interesting to investigate the relation between $H_1(Fin, Fin)$ and $H_2(Fin, Fin)$. It is already known that $H_2(Fin, Fin)$ is a subset of $H_1(Fin, Fin)$ which is in turn a proper subset of the class of regular languages. The main aim of this paper is to show that the first inclusion is also strict. Precisely, by using combinatorics on words, we construct non trivial classes of regular languages which can be generated by using the 1-splicing operation (over a finite H system) but which cannot be generated using the 2-splicing operation (over a finite H system). As a result we obtain that in the case of finite H systems 1-splicing operation is more powerful than the 2-splicing operation.

The paper is organized as follows. Section 2 contains some basics on words and all the necessary definitions and notations for the splicing operation. Known results are presented in Section 3 and last sections are devoted to our contributions. More precisely, in Section 4 we present classes of 2-splicing languages and conjectures for more general classes of languages. Regular languages which are 1-splicing languages but not 2-splicing languages are given in Section 5. Finally, in Section 6, we prove that some regular languages are not 1-splicing languages, *i.e.*, they cannot be generated by any finite H system.

2 Basics on words and splicing operation

2.1 Words

For definitions on languages we will refer to [5]. We denote by V^* be the free monoid over a finite alphabet V and $V^+ = V^* \setminus \varepsilon$, where ε is the empty word. If it is not differently supposed L will always be a regular language over V. Let $a \in V$ and $w \in V^*$. The number of occurrences of a in w is denoted by $|w|_a$. Given V' a finite alphabet, we set $|w|_{V'} = \sum_{a \in V'} |w|_a$ the sum of occurrences of letters from V' in w.

For a word $w \in V^*$, we denote by $Fact(w) = \{x \in V^+ | w = w_1 x w_2,\ w_1, w_2 \in V^*\}$ the set of all factors of w. We recall the important notion of a *constant* for a regular language L given by Schützenberger in [10]. We report an equivalent, but simpler definition given in [2, 3].

Definition 2.1. [2, 3] A word $w \in V^*$ is a *constant* for a regular language L if whenever the words $xwy, uwv \in L$, then the words $xwv, uwy \in L$.

The following observation will be frequently used in the sequel.

Remark 2.1. Let $L \subseteq V^*$ be a regular language. It is clear that if $c \notin V$ then c is a constant for the languages Lc, cL and LcL. It is also not too difficult to prove that if $c, d \notin V$ then c, d are constants for the language $cL + Ld$. Finally $\{ca \mid a \in V\}$ is a finite set of constants for the language cLc.

2.2 Splicing operation

An *(abstract) molecule* is simply a word over a finite alphabet. A *splicing rule* (over alphabet V) is a quadruple (u_1, u_2, u_3, u_4) of words $u_1, u_2, u_3, u_4 \in V^*$ which is often written as follows: $u_1\#u_2\$u_3\#u_4$ where $\#$ and $\$$ are special symbols which are not in V.

A splicing rule $r = u_1\#u_2\$u_3\#u_4$ is applicable to two molecules x, y if there are words $x_1, x_2, y_1, y_2 \in V^*$ with $x = x_1u_1u_2x_2$ and $y = y_1u_3u_4y_2$, and produces two new molecules $w_1 = x_1u_1u_4y_2$ and $w_2 = y_1u_3u_2x_2$. In this case we write $(x, y) \vdash_r (w_1, w_2)$. This operation is also called *2-splicing*. We can take only w_1 as a result instead of w_1 and w_2. In this case the corresponding operation is called *1-splicing* and we denote it by $(x, y) \vdash_r w_1$. We also say that r is applied in *1-splicing* mode in this case and in *2-splicing* mode in the previous case.

A pair $\sigma = (V, R)$, where V is an alphabet and R is a set of splicing rules, is called a *splicing scheme* or an H *scheme*.

For an H scheme $\sigma = (V, R)$ and a language $L \subseteq V^*$ we define:

$$\sigma_1(L) \stackrel{\text{def}}{=} \{w_1 \in V^* | \exists x, y \in L, r \in R : (x, y) \vdash_r w_1\}$$

$$\sigma_2(L) \stackrel{\text{def}}{=} \{w_1, w_2 \in V^* | \exists x, y \in L, r \in R : (x, y) \vdash_r (w_1, w_2)\}$$

where x, y, w_1, w_2 and r are specified above. We also define:

$\sigma_1^0(L) = L$,
$\sigma_1^{i+1}(L) = \sigma_1^i(L) \cup \sigma_1(\sigma_1^i(L))$,
$\sigma_1^*(L) = \cup_{i \geq 0} \sigma_1^i(L)$.

We define $\sigma_2^*(L)$ similarly.

Definition 2.2. [2, 4] A *Head splicing system*, or H *system*, is a construct $H = (\sigma, A) = ((V, R), A)$, simply denoted $H = (V, A, R)$, where V is a finite alphabet, $A \subseteq V^*$ is a set of initial words over V, called *axioms*, and $R \subseteq V^*\#V^*\$V^*\#V^*$ is a set of splicing rules.

The *language generated* by $H = (V, A, R)$ is $L(H) \stackrel{\text{def}}{=} \sigma_1^*(A)$ (resp. $L(H) \stackrel{\text{def}}{=} \sigma_2^*(A)$).

Thus the language generated by the H system H is the set of all molecules that can be generated starting with A, as initial molecules, by iteratively applying splicing rules from R in one of the splicing modes to copies of the molecules already generated. If we use rules from R in 1-splicing mode (resp. 2-splicing mode) then H is also called an H_1-system (resp. H_2-system).

Remark 2.2. Consider two H_1-systems (resp. H_2-systems) $S = (V, A, R)$ and $S' = (V, A, R')$ having $R' \subseteq R$. It is easy to observe that $L(S') \subseteq L(S)$.

We denote by $H_1(Fin, Fin)$ (or simply H_1) the family of languages generated by finite H_1-systems, *i.e.*, H systems with a finite set of axioms, a finite set of rules and which uses the 1-splicing operation for producing words. Similarly, we denote by $H_2(Fin, Fin)$ (or H_2) the family of languages generated by finite H_2-systems *i.e.*, H systems with a finite set of axioms, a finite set of rules and which use the 2-splicing operation for producing words. The language L generated by a finite H_1-system (resp. H_2-system) S, *i.e.*, $L = L(S)$, is called a 1-splicing (resp. 2-splicing) language.

In what follows we shall consider only non-finite regular languages, as well as only finite H systems.

3 1-splicing and 2-splicing: known results

The aim of this paper is to compare two classes of regular languages: $H_2(Fin, Fin)$ and $H_1(Fin, Fin)$. Below we report an already known result.

Proposition 3.1. *[4, 9]* $H_2(Fin, Fin) \subseteq H_1(Fin, Fin)$.

This proposition is based on the following reasoning. An H system is called symmetric if for any rule $r = u_1\#u_2\$u_3\#u_4 \in R$, R contain the rule $r' = u_3\#u_4\$u_1\#u_2$. It is easy to observe that an H system based on 2-splicing is implicitly supposed to be symmetric. Consequently, every language produced by a system from the family $H_2(FIN, FIN)$ can be produced by a system in $H_1(FIN, FIN)$ which is symmetric. This argument was used in [4].

This property is one of the reasons why 2-splicing is used mainly in the literature: all results obtained for 2-splicing can be easily reformulated in terms of 1-splicing.

We shall concentrate below on languages obtained by union of a regular language $L \subseteq V^*$ with the languages Lc, cL, LcL and cLc, where $c \notin V$, *i.e.* c is a constant for L. We shall call these languages *constant languages*. The next theorem is a corollary of a more general result proved by T. Head in [3]. It shows that some of the languages above are splicing languages.

Theorem 3.1. *[3] Let $L \subseteq V^*$ be a regular language and $c, d \notin V$. Then the languages Lc, cL, LcL, cLc and $cL + Ld$ are in $H_2(FIN, FIN)$. Moreover, each rule of the H system which generates Lc, respectively cL, LcL, cLc and $cL + Ld$, is of form $m\#\varepsilon\$m'\#\varepsilon$ or $\varepsilon\#m\$\varepsilon\#m'$, where m and m' are constants for Lc, respectively cL, LcL, cLc and $cL + Ld$. The words m and m' contain also c in the case of languages Lc, cL, LcL and cLc.*

Remark 3.1. We note that the author used 1-splicing in [3]. But, as it was shown in [1], this result holds in case of 2-splicing.

4 2-splicing languages

Now we present some classes of 2-splicing languages that can be obtained starting from the languages described above.

Proposition 4.1. *Let $L \subseteq V^*$ be a regular language and $c \notin V$. Then the regular languages $\mathcal{L} = L + Lc^*$ and $\mathcal{L}' = L + c^*L$ are 2-splicing languages.*

Proof. Let us consider the language $\mathcal{L}$. From Theorem 3.1 we obtain that there is an H system $S = (V \cup \{c\}, A, R)$ based on 2-splicing which generates Lc. Let $\bar{S} = (V \cup \{c\}, A, R \cup \{r\})$, where $r = \varepsilon\#c\$c\#\varepsilon$. It is clear that $L(S) \subseteq L(\bar{S})$, see Remark 2.2.

It is easy to see that $L(\bar{S}) = \mathcal{L}$. If we apply r to lc and to lc, $l \in L$, we obtain l and lcc. If we apply once more r to lc and to lcc, we obtain $lccc$ and so on.

More formally, at first we shall show by induction that $\mathcal{L} \subseteq L(\bar{S})$. Let w be in $\mathcal{L}$. If w is in Lc, we obtain that w is in $L(S)$ which is a subset of $L(\bar{S})$. If w is in L, then there is a word wc in Lc and we can generate it in $\bar{S}$. After that we can apply r to wc and itself: $(wc, wc) \vdash_r (w, wcc)$ obtaining w.

Now, let us suppose that w is in Lc^i, $i > 0$, *i.e.* $w = w'c^i$, where $w' \in L$. Let us show that $w'c^{i+1} \in L(\bar{S})$. As wc and wc^i are in $L(\bar{S})$ by the induction hypothesis, we obtain wc^{i+1} by applying r to these two words: $(wc, wc^{i-1}) \vdash_r (w, wc^i)$.

Conversely, let w be in $L(\bar{S})$. We use the induction on the number k of iterations of σ used to produce w. If $k = 0$, then the word w is in A which is a part of $L(S)$ which is equal to Lc. Let us consider now a word w produced in $k+1$ iterations of σ. Let the last application of σ be the following: $(x, y) \vdash_{r'} (w_1, w_2)$ and $w \in \{w_1, w_2\}$. By the induction hypothesis x and y are in $\mathcal{L}$. If r' is a rule from R, we get that x and y are in Lc^+, because the sites of r' contain c, see Theorem 3.1. Consequently, w is also in Lc^+. If $r' = r$, we obtain that $x = x_1x_2$, with $x_1 \in Lc^*$ and $x_2 \in c^+$ and $y = y_1y_2$, with $y_1 \in Lc^+$ and $y_2 \in c^*$. In this case $w_1 = x_1y_2$ belongs to Lc^* and $w_2 = y_1x_2$ belongs to Lc^+, *i.e.*, w is in $\mathcal{L}$.

We can show in a similar manner that $\mathcal{L}'$ is a 2-splicing language. □

Proposition 4.2. *Let $L \subseteq V^*$ be a regular language and $c \notin V$. Then, regular languages $\mathcal{L} = L + Lc$ and $\mathcal{L}' = L + cL$ are 1-splicing languages.*

Proof. We use the same construction as in the previous proposition.

As $c \notin V$, c is a constant for Lc. Therefore, there an H system based on 1-splicing $S = (V \cup \{c\}, A, R)$ such that $Lc = L(S)$ (Theorem 3.1). Similarly to Proposition 4.1, we consider the rule $r = \varepsilon\#c\$c\#\varepsilon$ and the system $\bar{S} = (V \cup \{c\}, A, R \cup \{r\})$. It is obvious that $L + Lc = L(\bar{S})$, because rules of R permit to produce Lc, while r permits to produce L. We note that it is important to use 1-splicing instead of 2-splicing in order to produce L. □

We do not know if this result remains true if 2-splicing is used, but we suppose this.

Conjecture 4.1. *Let $L \subseteq V^*$ be a regular language and $c \notin V$. Then, regular languages $\mathcal{L} = L + Lc$ and $\mathcal{L}' = L + cL$ are 2-splicing languages.*

We shall show now one possibility to solve this conjecture. Let us suppose that it is possible to find $z_i \in V^*$ and $w_i \in L$ such that for all $xw_iy \in L$ the following two conditions are satisfied:

a) $xw_i \in L$

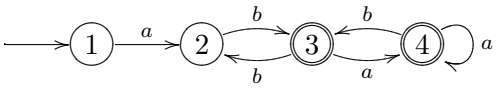
b) $\forall w \in L : w = w'z_i \Rightarrow wy = w'z_iy \in L$

If it is possible to find such z_i and w_i, then after adding words w_i to axioms the rules $z_i\#c\$w_i\#\varepsilon$ permit to generate L starting from Lc. Similar conditions may be formulated for the generation of L from cL.

The motivation of the previous statement is the following. First we observe that we can generate the language Lc, see Theorem 3.1. After that, the main idea is to produce a word x of L from the word $xc \in Lc$ by erasing c at the end. This may be done by using rules $z_i\#c\$w_i\#\varepsilon$. Conditions a) and b) guarantee the consistence of obtained words because both resulting words must belong to L or Lc.

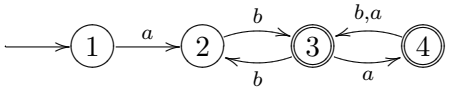
Example 4.1. Languages $a^* + a^*b$ and $a^* + ba^*$ are examples, which confirm the conjecture above. For the first language it is sufficient to take $i = 1$, $w_1 = a$ and $z_1 = \varepsilon$ in order to satisfy both conditions above. The rule $\varepsilon\#b\$a\#\varepsilon$ then permits to produce a^* from a^*b. We note that the language a^*b may be produced by an H system based on 2-splicing (Theorem 3.1).

Example 4.2. Let us consider the regular language L recognised by the following finite automata:



If we take $i = 2$ and $z_1 = a$, $z_2 = b$, $w_1 = ababa$, $w_2 = abbb$, then both previous conditions are satisfied. In this case the rules $a\#c\$ababa\#\varepsilon$ and $b\#c\$abbb\#\varepsilon$ permit to produce L from Lc. We note that Lc may be generated by an H system based on 2-splicing (Theorem 3.1).

Example 4.3. Let us consider the regular language L recognised by the following finite automata:



In this case $i = 4$ and the following 4 rules $ba\#c\$ababa\#\varepsilon$, $aa\#c\$abbb\#\varepsilon$, $ab\#c\$abbb\#\varepsilon$ and $bb\#c\$abbb\#\varepsilon$ satisfy both previous conditions. Therefore, they permit to produce L from Lc. We note that Lc may be generated by an H system based on 2-splicing (Theorem 3.1).

5 Classes of 1-splicing languages which are not 2-splicing languages

The next proposition shows a class of regular languages which cannot be 2-splicing languages.

Proposition 5.1. *Let $L \subseteq V^*$ be a regular language and $c, d \notin V$. Then, the language $\mathcal{L} = L + cL + Ld$ cannot be a 2-splicing language.*

Proof. We shall prove this result by contradiction. We suppose that there is an H system based on 2-splicing $S = (V, A, R)$ such that $L(S) = \mathcal{L}$. We consider any rule $r \in R$ and we show that using this rule we obtain either a word which does not belong to $\mathcal{L}$ as it will contain two occurrences of letters from V', where $V' = \{c, d\}$, or both resulting words will contain one occurrence of c or d, *i.e.*, we cannot produce words from L by using this rule. We also notice the closure property of $\mathcal{L}$: $\mathcal{L} = \sigma_2(\mathcal{L})$ where σ is the H scheme (V, R).

Now let us consider a rule $r = u_1\#u_2\$u_3\#u_4 \in R$ and let $x = x_1u_1u_2x_2$, $y = y_1u_3u_4y_2$, $w_1 = x_1u_1u_4y_2$, $w_2 = y_1u_3u_2x_2$ be such that we can have the following application of r: $(x, y) \vdash_r (w_1, w_2)$.

It is clear that $|u_1u_2|_{V'} \leq 1$ and $|u_3u_4|_{V'} \leq 1$. We have the following cases with respect to the number of c and d in the components u_i of the rule r, $1 \leq i \leq 4$:

1. $|u_i|_{V'} = 0$. We obtain a contradiction if we take x in cL and y in Ld, as w_1 will contain c and d in the same time.

2. $|u_1u_2|_{V'} = 0$ and $u_3u_4 \in Fact(cL)$. We have two subcases:

   (a) $u_3 \neq \varepsilon$. If we take x in Ld and y in cL, then we obtain that $|w_2|_{V'} = 2$, which is a contradiction.

   (b) $u_3 = \varepsilon$. If we take x and y in cL, we have a contradiction, as the resulting word w_1 will contain two letters c.

3. We can reason in a similar way for other combinations for which one of sites contain a letter from V' and another one does not contain any letter of V'.

4. $u_1u_2 \in Fact(cL)$ and $u_3u_4 \in Fact(cL)$. We have 3 subcases:

   (a) $u_3 = \varepsilon$ and $u_1 \neq \varepsilon$ (or $u_1 = \varepsilon$ and $u_3 \neq \varepsilon$). If we take x and y in cL, then we obtain a resulting word containing two occurrences of c.

   (b) $u_1, u_3 = \varepsilon$. Both resulting words are identical to the initial ones.

(c) $u_1, u_3 \neq \varepsilon$. Both resulting words belong to cL.
We see that we cannot produce a word from L by using rules of last two types.

5. We can reason in a similar manner for the case when $u_1u_2 \in Fact(Ld)$ and $u_3u_4 \in Fact(Ld)$.

6. $u_1u_2 \in Fact(cL)$ and $u_3u_4 \in Fact(Ld)$. We have 4 subcases:

   (a) $u_1, u_4 \neq \varepsilon$. We obtain a contradiction if we take x in cL and y in Ld, as w_1 contains both c and d.

   (b) $u_1 \neq \varepsilon, u_4 = \varepsilon$. We obtain that $|w_1|_c = 1$ and $|w_2|_d = 1$.

   (c) $u_1 = \varepsilon, u_4 \neq \varepsilon$. We obtain that $|w_1|_d = 1$ and $|w_2|_c = 1$.
   We see that we cannot produce a word from L by using rules of last two types.

   (d) $u_1, u_4 = \varepsilon$. A contradiction is obtained if we take x in cL and y in Ld, as the resulting word w_2 contains both c and d.

Shortly speaking, we have rules of two types. Rules of the first type produce words having one occurrence of c or d. Consequently, these words cannot belong to L. If we suppose that we can generate L by using rules of another type, then we can show for each rule of this type an example of words such that by applying this rule to them we will produce a word having two occurrences of letters from V'. This means that the corresponding word does not belong to $\mathcal{L}$. But this contradicts to the fact that $\mathcal{L}$ is closed with respect to the splicing operation. Consequently, we obtain a contradiction with the fact that $\mathcal{L} = L + cL + Ld$ is a 2-splicing language. □

We can easily derive the same result in the case when $|V'| = 1$, *i.e.*, $c = d$.

Corollary 5.1. *Let $L \subseteq V^*$ be a regular language and $c \notin V$. Then, the language $\mathcal{L}' = L + cL + Lc$ cannot be a 2-splicing language.*

The following theorems show that the languages above belong to the family $H_1(FIN, FIN)$. This means that they are in $H_1(FIN, FIN) \setminus H_2(FIN, FIN)$.

Theorem 5.1. *Let $L \subseteq V^*$ be a regular language and $c, d \notin V$. Then, the language $\mathcal{L} = L + cL + Ld$ is in $H_1(FIN, FIN) \setminus H_2(FIN, FIN)$.*

Proof. Proposition 5.1 gives us that the language $\mathcal{L} = L+cL+Ld$ cannot be in $H_2(FIN, FIN)$. Now it is enough to show that $\mathcal{L}$ is in $H_1(FIN, FIN)$. Since c is a constant for cL and since d is a constant for Ld, cL and Ld are 2-splicing languages, see Theorem 3.1, and, consequently, 1-splicing languages, see Proposition 3.1. Let $S_c = (V \cup \{c\}, A_c, R_c)$ be the H system which generates cL, *i.e.*, $L(S_c) = cL$, and let $S_d = (V \cup \{d\}, A_d, R_d)$ be the H system which generates Ld, *i.e.* $L(S_d) = Ld$. The language $cL+Ld$ is a 2-splicing language as well, see Theorem 3.1. We affirm that the system $S = (V \cup \{c, d\}, A_c \cup A_d, R_c \cup R_d \cup \{r\})$, where $r = \varepsilon\#d\$d\#\varepsilon$, generates $\mathcal{L}$, *i.e.* $\mathcal{L} = L(S)$. We note that we can also use the rule $r = \varepsilon\#c\$c\#\varepsilon$. Indeed, we can generate cL and Ld by using rules from R_c and R_d. Finally, the rule $r = \varepsilon\#d\$d\#\varepsilon$ which may be applied to a pair of words from Ld only, permits to generate words from L by 1-splicing. We note that the obtained system S is based on 1-splicing, since r cannot be used to perform a 2-splicing. □

The previous result remain true if we take $\mathcal{L}' = L + cL + Lc$. More exactly, we observe that the proof that $\mathcal{L} = L + cL + Ld$ is a 1-splicing language uses the hypothesis that c and d are two constants for $\mathcal{L}$. From the other side, it is possible to show that $\mathcal{L}' = L + cL + Lc$ is a 1-splicing language because the sets $\{ca \mid a \in V\}$ and $\{ac \mid a \in V\}$ are two finite sets of constants for $\mathcal{L}'$. In this case, in order to generate L we can use the set of rules $r_a = a\#c\$ac\#\varepsilon$, respectively $r_a = \varepsilon\#ca\$c\#a$, for all $a \in V$ in order to generate L. Such a rule r_a is applied to two words from Lc, respectively cL, and produce a word from L.

Theorem 5.2. *Let $L \subseteq V^*$ be a regular language and $c \notin V$. Then the language $\mathcal{L} = L + LcL$ cannot be a 2-splicing language.*

Proof. We shall prove this result by contradiction. We suppose that there is an H system based on 2-splicing $S = (V, A, R)$ such that $L(S) = \mathcal{L}$. We consider any rule $r \in R$ and we show that using this rule we obtain either a word which does not belong to $\mathcal{L}$ as it will contain two occurrences of c, or both resulting words will contain one occurrence of c. This means that we cannot produce words from L by using this rule. We also notice the closure property of $\mathcal{L}$: $\mathcal{L} = \sigma_2(\mathcal{L})$ where σ is the H scheme (V, R).

Now let us consider a rule $r = u_1\#u_2\$u_3\#u_4 \in R$ and let $x = x_1u_1u_2x_2$, $y = y_1u_3u_4y_2$, $w_1 = x_1u_1u_4y_2$, $w_2 = y_1u_3u_2x_2$ such that we can have the following application of r: $(x, y) \vdash_r (w_1, w_2)$.

It is clear that $|u_1u_2|_c \leq 1$ and $|u_3u_4|_c \leq 1$. Therefore, we have the following cases with respect to the number of c in components u_i of the rule r, $1 \leq i \leq 4$:

1. $|u_i|_c = 0, i = 1, \ldots, 4$.
   We obtain a contradiction if we take x, y such that $|x_1|_c = |y_2|_c = 1$, because $|w_1|_c = 2$.

   We affirm that it is possible to find such x and y because of the form of $\mathcal{L}$. More precisely, if we can apply r to x and to y and if $|x_1|_c = 0$, then necessarily $x = x_1u_1u_2w'cw''$. Since $x_1u_1u_2w' \in L$, there is a word $x' = w_3cx_1u_1u_2w' \in LcL$ having $|x'_1|_c = |w_3cx_1|_c = 1$. Then the rule r may be applied to x' and to y.

2. $|u_1|_c = |u_2|_c = 0$ and $|u_3u_4|_c = 1$

   (a) $|u_3|_c = 1$ ($|u_4|_c = 0$): if we take x such that $|x_2|_c = 1$, we obtain a contradiction, as $|w_2|_c = 2$.

   (b) $|u_3|_c = 0$ ($|u_4|_c = 1$): if we take x such that $|x_1|_c = 1$, we obtain a contradiction again, as this time $|w_1|_c = 2$.

2′. Similarly, we obtain a contradiction in the case when $|u_3|_c = |u_4|_c = 0$ and then $|u_1u_2|_c = 1$.

3. $|u_1|_c = 1$ ($|u_2|_c = 0$ and $|u_3u_4|_c = 1$). We have two subcases:

   (a) $|u_3|_c = 1$ ($|u_4|_c = 0$): then $|w_1|_c = |w_2|_c = 1$, and we cannot produce a word in L using this rule.

   (b) $|u_4|_c = 1$ ($|u_3|_c = 0$): we have a contradiction, as $|w_1|_c = 2$.

3′. Similarly if $|u_2|_c = 1$ ($|u_1|_c = 0$ and $|u_3u_4|_c = 1$).

Shortly, we have rules of two types. Rules of the first type produce words having one occurrence of c. Therefore, these words cannot belong to L. If we suppose that we can generate L by using rules of another type, then we can present for each rule of this type an example of

words such that by applying this rule to them we will produce a word having two occurrences of c. This means that the corresponding word does not belong to $\mathcal{L}$. But this contradicts the fact that $\mathcal{L}$ is closed with respect to the splicing operation. Consequently, we obtain a contradiction with the fact that $\mathcal{L} = L + LcL$ is a 2-splicing language. □

We do not know if the above language is a 1-splicing language, but we suppose that it is the case.

Conjecture 5.1. *Let $L \subseteq V^*$ be a regular language and $c \notin V$. Then the language $L + LcL$ is in $H_1(FIN, FIN)$, but not in $H_2(FIN, FIN)$.*

One of the possibilities to solve this conjecture, *i.e.*, whether the language $L + LcL$ is in $H_1(FIN, FIN)$, is directly related to the solution proposed for Conjecture 4.1 whether $L + Lc \in H_2(FIN, FIN)$. We recall that Conjecture 4.1 is true if we can find $z_i \in V^*$ and $w_i \in L$ such that for all xw_iy in L the following two conditions are satisfied:

a) $xw_i \in L$,

b) $\forall w \in L : w = w'z_i \Rightarrow wy = w'z_iy \in L$.

In our case, rules $z_i\#c\$cw_i\#\varepsilon$ permit to produce L from LcL. The motivation of this assertion is the same as for Conjecture 4.1, but in this case the problem is even simpler as condition a) is always true.

Example 5.1. $\mathcal{L} = a^* + a^*ba^* \in H_1(FIN, FIN) \setminus H_2(FIN, FIN)$.

From Theorem 5.2 we obtain that $\mathcal{L} \notin H_2(FIN, FIN)$. Now we use the previous remark and put $w_1 = a$ and $z_1 = \varepsilon$. It is easy to see that condition b) is satisfied. The rule $\varepsilon\#b\$ba\#\varepsilon$ permits to generate a^* from a^*ba^*. To finish the proof we note that a^*ba^* is a 2-splicing language and, consequently, a 1-splicing language, see Theorem 3.1.

Example 5.2. If we consider the language L from example 4.2, we obtain immediately that $L + LcL$ is a 1-splicing language. The same result holds for the language L from example 4.3.

6 Examples of regular languages which are not 1-splicing languages

We present in this section a class of regular languages which cannot be 1-splicing languages. This class is constructed from the language cLc, $L \subseteq V^*$, $c \notin V$.

Theorem 6.1. *Let $L \subseteq V^*$ be a regular language and $c \notin V$. Then, the language $\mathcal{L} = L + cLc$ cannot be a 1-splicing language.*

Proof. We shall prove this result by contradiction. We suppose that there is an H system based on 1-splicing $S = (V, A, R)$ such that $L(S) = \mathcal{L}$. We consider any rule r of R and we show that using this rule we obtain either a word which does not belong to $\mathcal{L}$ as it will contain one occurrence of c, or both resulting words will contain two occurrences of c. This means that we cannot produce words from L by using this rule. We also notice the closure property of $\mathcal{L}$: $\mathcal{L} = \sigma_1(\mathcal{L})$ where σ is the H scheme (V, R).

Now let us consider a rule $r = u_1\#u_2\$u_3\#u_4 \in R$ and let $x = x_1u_1u_2x_2 \in \mathcal{L}$ and $y = y_1u_3u_4y_2 \in \mathcal{L}$. We denote by w the result of the application of r to these two words.

We note that $|u_1u_2|_c \leq 2$ et $|u_3u_4|_c \leq 2$. Therefore, we have to consider only the cases when every $|u_i|_c$, $i = 1 \ldots 4$, varies between 0 and 2. We can throw away cases where $|u_1|_c > 0$ or $|u_4|_c > 0$, because in these cases we cannot produce a word from L.

Then we fix $|u_1|_c = |u_4|_c = 0$ and we consider $0 \leq |u_2|_c, |u_3|_c \leq 2$. This gives us 9 cases. We number cases by two digits; the first digit indicates the number of c in u_2 and the second digit indicates the number of c in u_3.

Case 00: (*i.e.*, $|u_2|_c = 0 = |u_3|_c$). Then there are x', y' in L such that $(cx'c, y') \vdash_r w$ and $|w|_c = 1$, therefore $w \notin \mathcal{L}$.

Case 01: (*i.e.*, $|u_2|_c = 0$ et $|u_3|_c = 1$). We take an y which contains the site u_3u_4 of the rule r.
If $u_4y_2 \neq \varepsilon$, then there is x' in L such that $(x', y) \vdash_r w$ and $|w|_c = 1$ as $|u_4y_2|_c = 1$.
If $u_4y_2 = \varepsilon$, then there is x' in L such that $(cx'c, y) \vdash_r w$ as $w = x_1u_1$ and $|w|_c = 1$.

In both cases $w \notin \mathcal{L}$.

Case 02: (*i.e.*, $u_4 = \varepsilon$). Then there is x' in L such that $(cx'c, y) \vdash_r w$ and $|w|_c = 1$ as $w = x_1u_1$. Hence, $w \notin \mathcal{L}$.

Case 10: This case is similar to case 01. We take y' in L and we choose y' or $cy'c$ depending on x and we obtain that $|w|_c = 1$, *i.e.*, $w \notin \mathcal{L}$.

Case 11: we have 4 subcases ($x, y \in \mathcal{L}$):

a) $x_1u_1 = u_4y_2 = \varepsilon$: we obtain $w = \varepsilon$.

b) $x_1u_1, u_4y_2 \neq \varepsilon$: as $|u_2|_c = |u_3|_c = 1$ we obtain that $|x_1u_1|_c = |x_1|_c = 1$ and $|u_4y_2|_c = |y_2|_c = 1$. Then, $|w|_c = 2$ and w cannot be in L.

c,d) either $x_1u_1 = \varepsilon$, or $u_4y_2 = \varepsilon$. If $x_1u_1 = \varepsilon$, we have $w = u_4y_2$ and $|w|_c = 1$. If $u_4y_2 = \varepsilon$, we have $w = x_1u_1$ et $|w|_c = 1$. In both cases $w \notin \mathcal{L}$.

Case 12: (*i.e.*, $u_4 = \varepsilon$). Then there is an x' in L such that $(cx'c, y) \vdash_r w$ and either $w = \varepsilon$ (if $x_1u_1 = \varepsilon$), or $|w|_c = 1$.

Case 20: (*i.e.*, $u_1 = \varepsilon$). Similarly to case 02 we can find an y' in L such that $(x, cy'c) \vdash_r w$ and $|w|_c = 1$.

Case 21: (*i.e.*, $u_1 = \varepsilon$). Similarly to case 12, we obtain that there is an y' in L such that $(x, cy'c) \vdash_r w$ and either $w = \varepsilon$ (if $u_4y_2 = \varepsilon$), or $|w|_c = 1$.

Case 22: We produce ε.

Shortly speaking, rules are of three types. Rules of the first type permit to produce only ε; rules of the second type permit to obtain words w such that $|w|_c = 2$. Consequently, these words cannot belong to L. Rules of the third type permit to obtain different words, but if we suppose that we can generate L by using rules of this type, then we can find for each rule of this type an example of words from L and cLc such that by applying this rule to them we will produce a word having one occurrence of c. This means that the corresponding word does not belong to $\mathcal{L}$. This contradicts the fact that $\mathcal{L}$ is closed under splicing. Consequently, we obtain a contradiction with the fact that $\mathcal{L} = L + cLc$ is 1-splicing language. □

7 Conclusions

In this paper we compared the generating power of H systems based on 1-splicing and of H systems based on 2-splicing and we have shown that the set $H_1(FIN, FIN) \setminus H_2(FIN, FIN)$ is not empty. The languages which we used to obtain this result are constructed from constant

languages where the constant is defined by a letter $c \notin V$. We note that we can replace this letter by words which are constants for the considered language. Even if these languages seem to be more interesting, their underlying structure is identical to the structure of languages presented before. We also think that the proposal for solution of conjectures suggested for Conjecture 4.1 may be useful for more general cases.

Acknowledgments The first author wants to thank M. Margenstern and Yu. Rogozhin for their comments. The same author acknowledges the Ministry of Education of France for the financial support of his PhD. The second author wants to thank C. De Felice for her encouragement and supervision. Both authors acknowledge also the "MolCoNet" IST-2001-32008 project which made possible their collaboration.

References

[1] P.Bonizzoni, C. De Felice, G. Mauri, R. Zizza, *The structure of reflexive finite splicing languages*, submitted.

[2] T. Head, Formal Language Theory and DNA: an analysis of the generative capacity of specific recombinant behaviours, *Bull. Math. Biol.* **49** (1987), 737 – 759.

[3] T. Head, Splicing languages generated with one sided context, *in* "Computing with Biomolecules: Theory and Experiment (1998).

[4] T. Head, Gh. Paun, D. Pixton, Language theory and molecular genetics: generative mechanisms suggested by DNA recombination, *in* Rozenberg, G., Salomaa, A. (eds.): Handbook of Formal Languages, Vol. 2. Springer-Verlag (1996), 295 – 360.

[5] J.E. Hopcroft, R. Motwani, J.D. Ullman, Introduction to Automata Theory, Languages, and Computation, 2nd ed., Addison-Wesley, Reading, Mass. (2001).

[6] Gh. Paun, On the splicing operation, *Discrete Applied Math.* **70** (1996), 57 – 79.

[7] D. Pixton, Regularity of splicing languages, *Discrete Applied Math.* **69** (1996), 101 – 124.

[8] D. Pixton, Splicing in abstract families of languages, *Theoret. Comp. Science* **234** (2000), 135 – 166.

[9] Gh. Paun, G. Rozenberg, A. Salomaa, DNA computing, New Computing Paradigms. Springer-Verlag (1998).

[10] M. P. Schützenberger, Sur certaines opérations de fermeture dans les langages rationnels, *Symposia Mathematica* **15** (1975), 245 – 253.