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 $x = x_1 u_1 u_2 x_2$ and $y = y_1 u_3 u_4 y_2$, and produces two new molecules $w_1 = x_1 u_1 u_4 y_2$ and $w_2 = y_1 u_3 u_2 x_2$. 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]* $H_2(Fin, Fin) \subseteq H_1(Fin, Fin)$.

The above proposition has been stated by taking into account an hypothesis on the set of rules in an H system. We say that an H system is *symmetric* if for each rule $r = u_1\#u_2\$u_3\#u_4 \in R$, the rule $r' = u_3\#u_4\$u_1\#u_2 \in R$. Now it is easy to observe that each H_2-system is implicitly supposed to be symmetric. Thus, each language generated by a finite H_2-system may be generated by a finite H_1-system which is symmetric. This argument has been used in [4].

The next result is a byproduct of a more general result stated by Head in [3]. It regards regular languages $L \subseteq V^*$ having the form Lc, cL, LcL, cLc with $c \notin V$, *i.e.*, c is a constant for such languages. These languages will be named here *constant languages*.

Theorem 3.1. *[3] Let* $L \subseteq V^*$ *be a regular language and* $c, d \notin V$*. Then* Lc, cL, LcL, cLc *and* $cL + Ld \in H_2(Fin, Fin)$*. Moreover, each rule in an H system generating* Lc *(resp.* cL, LcL, cLc *and* $cL + Ld$*) has the form* $m\#\varepsilon\$m'\#\varepsilon$ *or* $\varepsilon\#m\$\varepsilon\#m'$*, where* m, m' *are constants for* Lc *(resp.* cL, LcL, cLc *and* $cL + Ld$*) and both* m *and* m' *contain* c *in the case of* Lc, cL, LcL *and* cLc*.*

Remark 3.1. We explicitly note that in [3] the author used the 1-splicing operation. So the result reported above should be referred to $H_1(Fin, Fin)$. However, it is possible to prove that this result holds also for $H_2(Fin, Fin)$. Indeed, in [1] it is provided a generalization of Head's result by using 2-splicing operation.

4 2-splicing languages

In this section we construct classes of 2-splicing languages starting from constant languages. Moreover, we present some classes of languages for which we are able to prove the membership to $H_1(Fin, Fin)$, whereas we conjecture that they are members of $H_2(Fin, Fin)$. Non trivial examples which confirm our conjecture will end this section.

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

Proof. First, we observe that c is a constant for Lc, see Remark 2.1. From Theorem 3.1 we obtain that there is an H_2 system $S_2 = (V \cup \{c\}, A, R)$ which generates Lc. Let us consider $\bar{S}_2 = (V \cup \{c\}, A, R \cup \{r\})$ where $r = \varepsilon\#c\$c\#\varepsilon$. ¿From Remark 2.2 we have that $L(S_2) \subseteq L(\bar{S}_2)$.

It easy to observe that $L(\bar{S}_2) = L + Lc^*$. Indeed, if we apply r to $w = lc$ and lc, $l \in L$, we obtain l and lcc. If we further apply r on lc and lcc then we obtain $lccc$, and so on. More precisely, let us first show that $L + Lc^* \subseteq L(\bar{S}_2)$. Let $w \in L + Lc^*$. If $w \in Lc$, then $w \in L(S_2) \subseteq L(\bar{S}_2)$. Suppose that $w \in L$. Then $wc \in Lc \subseteq L(S_2) \subseteq L(\bar{S}_2)$. So we can apply r on wc: $(wc, wc) \vdash_r (w, wcc)$. Now suppose that $w \in Lc^i$, $i \geq 1$, *i.e.*, $w = w'c^i$, $w' \in L$. We prove that $w \in L(\bar{S}_2)$ by induction on i. For $i = 1$ it is already proved that $w \in L(\bar{S}_2)$. Suppose that $wc^j \in L(\bar{S}_2)$, for each $j < i$. Then we can use r on wc and wc^{i-1}: $(wc, wc^{i-1}) \vdash_r (w, wc^i)$.

Conversely, let us show that $L(\bar{S}_2) \subseteq L + Lc^*$. We use the induction on the number k of iterations used to generate $w \in L(\bar{S}_2)$. If $k = 0$, then $w \in A \subseteq L(S_2) = Lc \subseteq L + Lc^*$. Let $k > 0$, *i.e.*, there are $x, y \in L(\bar{S}_2)$ and $r' \in R \cup \{r\}$ such that $(x, y) \vdash_{r'} (w_1, w_2)$ and $w \in \{w_1, w_2\}$. By induction hypothesis, $x, y \in L + Lc^*$. If $r' \in R$, then $x, y \in Lc^+$ so $w_1, w_2 \in Lc^+$ (see Theorem 3.1). If $r' = r$, then $x = x_1u_1u_2x_2, y = y_1u_3u_4y_2$ and we have $x_1u_1 \in Lc^*$, $u_2x_2 \in c^+$, $y_1u_3 \in Lc^+$ and $u_4y_2 \in c^*$. Then $w_1 = x_1u_1u_4y_2 \in Lc^*$ and $w_2 = y_1u_3u_2x_2 \in Lc^+$.

Similarly we can show that $\mathcal{L}' = L + c^*L$ is a 2-splicing language. □

Proposition 4.2. *Let $L \subseteq V^*$ be a regular language and $c \notin V$. Then $\mathcal{L}_1 = L + cL$ and $\mathcal{L}_2 = L + Lc$ are 1-splicing languages.*

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

As $c \notin V$, then c is a constant for Lc. So there is an H_1-system (Theorem 3.1) $S = (V \cup \{c\}, A, R)$ such that $Lc = L(S)$. Similarly we consider $r = \varepsilon\#c\$c\#\varepsilon$ and $\bar{S} = (V \cup \{c\}, A, R \cup \{r\})$. It is clear that $L + Lc = L(\bar{S})$, since the rules in R allow the generation of Lc, while r allows to generate L. Observe that r cannot be used in the 2-splicing mode. □

Conjecture 4.1. *Let $L \subseteq V^*$ be a regular language and $c \notin V$. Then $\mathcal{L}_1 = L + cL$ and $\mathcal{L}_2 = L + Lc$ are 2-splicing languages.*

We will present one possible direction for solving this conjecture. Suppose we can find $z_i \in V^*$ and $w_i \in L$ such that $\forall x w_i y \in L$ the following two conditions hold:

a) $xw_i \in L$

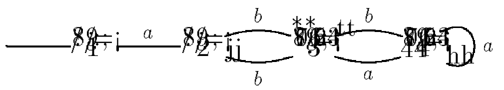
b) $\forall w \in L : w = w'z_i \Rightarrow wy = w'z_iy \in L$

If we can find such z_i and w_i, then the rules $z_i\#c\$w_i\#\varepsilon$ permit to generate L from Lc. The word w_i shall be added to axioms as well. We can formulate similar conditions for producing L from cL.

We shall present the motivation of the previous statement. First of all we observe that we can produce the language Lc (see Theorem 3.1). After that the main idea is to produce a word x from L by removing the trailing c from the word xc from Lc. In order to do this we use rules $z_i\#c\$w_i\#\varepsilon$. The conditions a) and b) are needed in order to assure the consistency of the obtained words (both resulting words shall be from L or Lc).

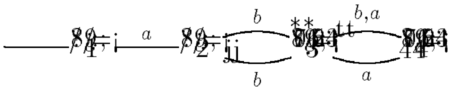
Example 4.1. The languages $a^* + a^*b$, $a^* + ba^*$ are easy examples of regular languages which confirm the above conjecture. For the first language it suffices to take $i = 1$, $w_1 = a$ and $z_1 = \varepsilon$ to satisfy both conditions from above (so the rule $\varepsilon\#b\$a\#\varepsilon$ permits to generate a^* from a^*b). We recall that a^*b can be produced by an H_2-system (see Theorem 3.1).

Example 4.2. Consider the regular language L recognized by the following finite state automaton (the initial state is indicated by an arrow without a label going into it and final states are indicated by a double circle).



We can take $i = 2$ and $z_1 = a$, $z_2 = b$, $w_1 = ababa$, $w_2 = abbb$ which satisfy both conditions from above. In this case the obtained rules $a\#c\$ababa\#\varepsilon$ and $b\#c\$abbb\#\varepsilon$ permit to generate L from Lc. We recall that Lc can be produced by an H_2-system (see Theorem 3.1).

Example 4.3. Consider the regular language L recognized by the following automaton



We can take $i = 4$ and the following 4 rules $ba\#c\$ababa\#\varepsilon$, $aa\#c\$abbb\#\varepsilon$, $ab\#c\$abbb\#\varepsilon$ and $bb\#c\$abbb\#\varepsilon$ which satisfy both conditions from above. So, they permit to generate L from Lc. We recall that Lc can be produced by an H_2-system (see Theorem 3.1).

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

The next proposition shows that there are regular languages which are not 2-splicing languages.

Proposition 5.1. *Let $L \subseteq V^*$ be a regular language and $c, d \notin V$. Then $\mathcal{L} = L + cL + Ld \notin H_2(Fin, Fin)$.*

Proof. Let us show that $\mathcal{L}$ is not a 2-splicing language. By contradiction, suppose that there exist $S_2 = (V, A, R)$ such that $L(S_2) = \mathcal{L}$. Take any rule $r = u_1\#u_2\$u_3\#u_4 \in R$. Let us denote $V' = \{c, d\}$. We will prove the claim according to $|u_i|_{V'}$, $i \in \{1, 2, 3, 4\}$. Clearly, we consider only the cases in which $|u_i|_{V'}$ is such that u_1u_2 and u_3u_4 are factors of words in $\mathcal{L}$. Moreover we denote $x = x_1u_1u_2x_2 \in \mathcal{L}$, $y = y_1u_3u_4y_2 \in \mathcal{L}$ such that $(x, y) \vdash_r (w_1, w_2)$, $w_1 = x_1u_1u_4y_2$ and $w_2 = y_1u_3u_2x_2$. Notice that for each $w \in \mathcal{L}$, we have $0 \leq |w|_{V'} \leq 1$. In other words, if $|u_1|_{V'} \neq 0$ (resp. $|u_2|_{V'} \neq 0$), then either $|u_1|_c \neq 0$ or $|u_1|_d \neq 0$ (resp. either $|u_2|_c \neq 0$ or $|u_2|_d \neq 0$). In addition, if $|u_1|_c \neq 0$ (resp. $|u_1|_d \neq 0$), we have $|u_1|_d = 0 = |u_2|_d = |u_2|_c$ (resp. $|u_1|_c = 0 = |u_2|_c = |u_2|_d$). The same argument holds for u_2, u_3, u_4.

We show that using the rule r, we either can obtain a word which is not in $\mathcal{L}$ (*i.e.*, it contains more than one occurrence of letters from V') or both resulting words contain one occurrence of c or one occurrence of d, *i.e.*, we cannot produce words in L using this rule. We notice the closure property of $\mathcal{L}$: $\mathcal{L} = \sigma_2(\mathcal{L})$ where σ is the underlying splicing scheme (V, R).

Let us consider the following cases.

1. $|u_1|_{V'} = 1$ and $|u_3|_{V'} = 1$ (then $|u_2|_{V'} = 0$ and $|u_4|_{V'} = 0$).
   Thus, for each $x, y \in \mathcal{L}$, we have $|w_1|_{V'} = |x_1u_1u_4y_2|_{V'} \geq 1$ and $|w_2|_{V'} = |y_1u_3u_2x_2|_{V'} \geq 1$. In both cases we have $w_1, w_2 \notin L$.

2. $|u_1|_{V'} = 1$ (then $|u_2|_{V'} = 0$) and $|u_3|_{V'} = 0$.
   If $|u_4|_{V'} = 0$ then there exists $y \in \mathcal{L}$ having $|y_2|_{V'} \geq 1$. Then $|w_1|_{V'} = |x_1u_1u_4y_2|_{V'} \geq 2$.
   If $|u_4|_{V'} = 1$ then for each $y \in \mathcal{L}$ that can enter the rule r we have $|w_1|_{V'} = |x_1u_1u_4y_2|_{V'} \geq 2$. In both cases $w_1 \notin \mathcal{L}$.

3. $|u_1|_{V'} = 0$ and $|u_2|_{V'} = 0$.
   (a) If $|u_3|_{V'} = 1$ (then $|u_4|_{V'} = 0$) then there exists $x \in \mathcal{L}$ having $|x_2|_{V'} = 1$. Then $|w_2|_{V'} = |y_1u_3u_2x_2|_{V'} \geq 2$, *i.e.*, $w_2 \notin \mathcal{L}$.
   (b) Let $|u_3|_{V'} = 0$.
   If $|u_4|_{V'} = 0$ then there exist $x, y \in \mathcal{L}$ having $|x_1|_{V'} = 1 = |y_2|_{V'}$. Then $|w_1|_{V'} = |x_1u_1u_4y_2|_{V'} \geq 2$, *i.e.*, $w_1 \notin \mathcal{L}$.
   If $|u_4|_{V'} = 1$ then there exists $x \in \mathcal{L}$ having $|x_1|_{V'} = 1$. Then $|w_1|_{V'} = |x_1u_1u_4y_2|_{V'} \geq 2$, *i.e.*, $w_1 \notin \mathcal{L}$.

4. $|u_1|_{V'} = 0$ and $|u_2|_{V'} = 1$.

   (a) If $|u_3|_{V'} = 1$ (then $|u_4|_{V'} = 0$) then for each $x, y \in \mathcal{L}$ that can enter the splicing rule r we have $|w_2|_{V'} = |y_1 u_3 u_2 x_2|_{V'} \geq 2$, *i.e.*, $w_2 \notin \mathcal{L}$.

   (b) Let $|u_3|_{V'} = 0$.
   If $|u_4|_{V'} = 0$ then there exists $y \in \mathcal{L}$ having $|y_1|_{V'} = 1$. Then $|w_2|_{V'} = |y_1 u_3 u_2 x_2|_{V'} \geq 2$, *i.e.*, $w_2 \notin \mathcal{L}$.
   If $|u_4|_{V'} = 1$ then for each $x, y \in \mathcal{L}$ that enter r we have $|w_1|_{V'} = |x_1 u_1 u_4 y_2|_{V'} \geq 1$ and $|w_2|_{V'} = |y_1 u_3 u_2 x_2|_{V'} \geq 1$, *i.e.*, $w_1, w_2 \notin L$.

Thus, we have two types of rules: rules of the first type applied to two words $x, y \in \mathcal{L}$ produce w_1 and w_2 such that $|w_1|_{V'}, |w_2|_{V'} = 1$ (*i.e.*, $w_1, w_2 \notin L$). For rules of the second type we can indicate an example of words $x, y \in \mathcal{L}$ such that one of the results of the application of a such rule on these words does not belong to $\mathcal{L}$. This contradicts the fact that $\mathcal{L}$ is closed under splicing. So we obtained a contradiction with the fact that $\mathcal{L}$ is a 2-splicing language. □

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

Corollary 5.1. *Let $L \subseteq V^*$ be a regular language and $c \notin V$. Then $\mathcal{L}' = L + cL + Lc \notin H_2(Fin, Fin)$.*

The theorems below show that the above languages belong to $H_1(Fin, Fin)$ and so 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 $\mathcal{L} = L + cL + Ld \in H_1(Fin, Fin) \setminus H_2(Fin, Fin)$.*

Proof. ¿From Proposition 5.1, we have that $\mathcal{L} = L + cL + Ld \notin H_2(Fin, Fin)$. Let us show that $\mathcal{L} \in H_1(Fin, Fin)$. Since c is a constant for cL and d is a constant for Ld then we have that cL and Ld are 2-splicing languages (and also 1-splicing languages), see Theorem 3.1. We denote by $S_c = (V \cup \{c\}, A_c, R_c)$ the H_2-system such that $cL = L(S_c)$ (resp. $S_d = (V \cup \{d\}, A_d, R_d)$ the H_2-system such that $Ld = L(S_d)$). Moreover, the language $cL + Ld$ is a 2-splicing language (see Theorem 3.1). We claim that the H_1-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.*, $\mathcal{L} = L(S)$ (observe that we can use also $r = \varepsilon \# c \$ c \# \varepsilon$). Indeed, by using the rules from R_c and R_d we generate cL and Ld, respectively. Finally, by using the rule $r = \varepsilon \# d \$ d \# \varepsilon$ which can be applied (only) to pair of words from Ld, we generate words in L by 1-splicing. We stress that S is not an H_2-system, since the rule r cannot be used in the 2-splicing mode. □

The previous result may be also obtained for $\mathcal{L}' = L + cL + Lc$. Loosely speaking, observe that the proof that $\mathcal{L} = L + cL + Ld$ is a 1-splicing language uses the hypothesis that c, d are two constants for $\mathcal{L}$. On the other hand, we can prove that $\mathcal{L}' = L + cL + Lc$ is a 1-splicing language too, since $\{ca \mid a \in V\}$ and $\{ac \mid a \in V\}$ are two (finite) sets of constants for $\mathcal{L}'$. Thus, in order to generate L, we can use the set of rules $r_a = a \# c \$ ac \# \varepsilon$ (resp. $r_a = \varepsilon \# ca \$ c \# a$), $\forall a \in V$. Such a rule r_a can be applied to two words from Lc (resp. cL) generating words from L only.

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

Proof. We prove the statement by contradiction. We suppose that there exist an H_2-system $S = (V, A, R)$ such that $L(S) = \mathcal{L}$. We consider any rule $r \in R$ and we show that using this rule we either can obtain a word which is not in $\mathcal{L}$ (which contains two occurrences of c) or both resulting words contain c, *i.e.*, we can not produce words in L using this rule. We notice the closure property of $\mathcal{L}$: $\mathcal{L} = \sigma_2(\mathcal{L})$ where σ is the underlying splicing scheme (V, R).

So, 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: $(x, y) \vdash_r (w_1, w_2)$.

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

1. $|u_i|_c = 0, i = 1, \ldots, 4$.
   We get the contradiction if we take x, y such that $|x_1|_c = |y_2|_c = 1$ (as $|w_1|_c = 2$).

   We state that we can find such x and y because of the form of $\mathcal{L}$, *i.e.*, if we can apply r to x and y and $|x_1|_c = 0$, then $x = x_1u_1u_2w'cw''$. As $x_1u_1u_2w' \in L$, then there is a word $x' = w_3cx_1u_1u_2w' \in LcL$ having $|x_1'|_c = |w_3cx_1|_c = 1$ and the rule r can be applied to x' and 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 $|x_2|_c = 1$, then we get a contradiction as $|w_2|_c = 2$.

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

2′. Similarly we obtain a contradiction in the case if $|u_3|_c = |u_4|_c = 0$ (and so $|u_1u_2|_c = 1$).

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

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

   (b) $|u_4|_c = 1$ ($|u_3|_c = 0$) then we get a contradiction as $|w_1|_c = 2$.

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

So, we have rules of two types. The rules of the first type produce words having one occurrence of c, thus these words cannot be from L. If we suppose that we can generate L using other rules then we can find for each such rule an example of words such that if we apply that rule to these words, then one of resulting words will contain two occurrences of c, *i.e.*, a word which is not in $L + LcL$. This contradicts the fact that $\mathcal{L}$ is closed under splicing. So we obtained a contradiction with the fact that $\mathcal{L} = L + LcL$ is a 2-splicing language. □

Conjecture 5.1. *Let $L \subseteq V^*$ be a regular language and $c \notin V$. Then $L + LcL \in H_1(Fin, Fin) \setminus H_2(Fin, Fin)$.*

One of the possibilities to prove $L + LcL \in H_1(Fin, Fin)$ is closely related to the proposed solution of Conjecture 4.1 (whether $L + Lc \in H_2(Fin, Fin)$). We recall that the Conjecture 4.1 is true if we can find $z_i \in V^*$ and $w_i \in L$ such that $\forall xw_iy \in L$ the following two condition hold:

a) $xw_i \in L$,

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

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

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

Indeed, from Theorem 5.2 we obtain $\mathcal{L} \notin H_2(Fin, Fin)$. Now we use the remark given upper and we pose $w_1 = a$ and $z_1 = \varepsilon$. It is easy to observe that the condition b) is satisfied. So the rule $\varepsilon\#b\$ba\#\varepsilon$ permits to generate a^* from a^*ba^*. To conclude the proof we add that a^*ba^* is a 2-splicing language (and thus 1-splicing language), see Theorem 3.1.

Example 5.2. If we consider the language L from the Example 4.2, then we immediately obtain that $L + LcL$ is a 2-splicing language. The same result holds for the language L from the Example 4.3.

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

This section presents a class of regular languages which are not splicing languages. This class is constructed starting from the constant language cLc, $L \subseteq V^*$, $c \notin V$. We note that $\{ca \mid a \in V\}$ is a finite set of constants for the language cLc.

Theorem 6.1. $\mathcal{L} = L + cLc \notin H_1(Fin, Fin)$.

Proof. We prove the statement by contradiction. We suppose that there exist an H_1 system $S = (V, A, R)$ such that $L(S) = \mathcal{L}$. We consider any rule $r \in R$ and we show that using this rule we either can obtain a word which is not in $\mathcal{L}$ (which contains one occurrence of c) or both resulting words contain two c, *i.e.*, we cannot produce words in L using this rule. We notice the closure property of $\mathcal{L}$: $\mathcal{L} = \sigma_1(\mathcal{L})$ where σ is the underlying splicing scheme (V, R).

So, 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 also denote by w the result of application of the rule r on these words.

We shall note that $|u_1u_2|_c \leq 2$ and $|u_3u_4|_c \leq 2$. So we shall consider cases where each $|u_i|_c$, $i = 1 \ldots 4$, varies between 0 and 2. We can throw away the cases when $|u_1|_c > 0$ or $|u_4|_c > 0$ because in this case we cannot produce any word from L using such rule.

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

Case 01: ($|u_2|_c = 0$ and $|u_3|_c = 1$). We take an y that match the site u_3u_4 of the rule r.
If $u_4y_2 \neq \varepsilon$ then $\exists x' \in L$ such that $(x', y) \vdash_r w$ and $|w|_c = 1$ (as $|u_4y_2|_c = 1$).
If $u_4y_2 = \varepsilon$ then $\exists x' \in L$ such that $(cx'c, y) \vdash_r w$ (so $w = x_1u_1$ and $|w|_c = 1$).
In both cases $w \notin \mathcal{L}$.

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

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

Case 11: Here we have 4 sub-cases ($x, y \in \mathcal{L}$):

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 $|w|_c = 2$, so w cannot be in L.

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

Case 12: (*i.e.*, $u_4 = \varepsilon$). Then $\exists x' \in L$ 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 the case 02 we obtain that $\exists y' \in L$ such that $(x, cy'c) \vdash_r w$ and $|w|_c = 1$.

Case 21: (*i.e.*, $u_1 = \varepsilon$). Similarly to the case 12 we obtain that $\exists y' \in L$ 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.

So 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 $|w|_c = 2$. So, using these two types of rules we cannot obtain any word from L. Rules of the third type permit to obtain different words but if we suppose that we can produce words from L then for each rule we can indicate examples of words from L and cLc such that if we apply this rule to these words then we obtain a word containing one occurrence of c, *i.e.*, a word which is not in $L + cLc$. This contradicts the fact that $\mathcal{L}$ is closed under splicing. So, we obtained a contradiction with the fact that $\mathcal{L} = L + cLc$ is 1-splicing language. $\square$

7 Conclusions

In this paper we proved that the set $H_1(Fin, Fin) \setminus H_2(Fin, Fin)$ is not empty. The languages used to prove this result are constructed starting from constant languages where the constant is defined by considering a letter $c \notin V$. We explicitly note that we can obtain other examples of languages by considering words (instead of letters) which are constants for the language. Even if these classes of languages could be more interesting their underlying structure is identical to the structure of languages presented in this paper. We also think that the solution of conjectures which follows the scheme suggested here, in particular the one 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.