Regular Languages Generated by Reflexive Finite Splicing Systems *

Paola Bonizzoni[1], Clelia De Felice[2], Giancarlo Mauri[1], and Rosalba Zizza[2]

[1] Dipartimento di Informatica Sistemistica e Comunicazione
Università degli Studi di Milano - Bicocca
Via Bicocca degli Arcimboldi 8, 20126 Milano - Italy
{bonizzoni, mauri}@disco.unimib.it
[2] Dipartimento di Informatica ed Applicazioni,
Università di Salerno, 84081 Baronissi (SA), Italy
{defelice,zizza}@unisa.it

Abstract. *Splicing systems* are a generative device inspired by a cut and paste phenomenon on DNA molecules, introduced by Head in 1987 and subsequently defined with slight variations also by Paun and Pixton respectively [8, 13, 17]. We will face the problem of characterizing the class of regular languages generated by finite splicing systems. We will solve this problem for the special class of the *reflexive finite splicing systems* introduced in [9, 10]. As a byproduct, we give a characterization of the regular languages generated by *finite Head splicing systems*. As in already known results, the notion of *constant*, given by Schützenberger in [19], intervenes.

1 Introduction

In this paper we will face the problem of characterizing the class of regular languages generated by finite linear splicing systems. *Splicing systems* are a generative device introduced by Head in 1987 as a formal model of certain cut and paste biochemical transformation processes of an initial collection of DNA strands under the simultaneous influence of enzymes [8]. This topic can be considered a part of the more general area of Molecular Computing [15].

A splicing system (or H-system) is a triple $H = (A, I, R)$, where A is a finite alphabet, $I \subseteq A^*$ is the initial language and R is the set of rules, $R \subseteq (A')^*$, $A \subseteq A'$ (see Section 2.2 for the definitions). The formal language generated by the splicing system is the smallest language containing I and closed under the splicing operation, which makes the rule intervene. Different variants of the original definition of the splicing operation given by Head have been proposed briefly; in particular two of them have been introduced by Paun and Pixton respectively [13–15, 18].

* Partially supported by MIUR Project *"Linguaggi Formali e Automi: teoria ed applicazioni"*, by the contribution of EU Commission under The Fifth Framework Programme (*project MolCoNet* IST-2001-32008) and by 60% Project *"Linguaggi Formali e Modelli di Calcolo"* (University of Salerno).

The computational power of (iterated linear) splicing systems has been investigated thoroughly and it mainly depends on which level of the Chomsky hierarchy I, R belong to. Precisely, let F_1, F_2 be two families of languages in the Chomsky hierarchy. Following [10, 15], we set $H(F_1, F_2) = \{L(H) \mid H = (A, I, R)$ with $I \in F_1, R \in F_2\}$. In [10] it is proved that either $H(F_1, F_2)$ is a specific class of languages in the Chomsky hierarchy or it is strictly intermediate between two of them. More precisely, according to some hypotheses on F_1, F_2, splicing systems can reach the same power of the Turing machines [10, 13, 14]. On the other hand, let FIN (resp. REG) be the class of finite (resp. regular) languages. In [18] the author proved that $H(REG, FIN) = REG$, while the class $H(FIN, FIN)$, as shown in [10, 15, 18], turns out to be strictly intermediate between FIN and REG. As an example, $(aa)^*b$ is a regular language which belongs to $H(FIN, FIN)$ [4, 9], whereas $(aa)^* \notin H(FIN, FIN)$ [6, 10]. Now two problems arise and Problem 2 is the special question we will focus on in this paper.

Problem 1. Given $L \in REG$, can we decide whether $L \in H(FIN, FIN)$?

Problem 2. Characterize regular languages which belong to $H(FIN, FIN)$.

The paper [3] is a complete survey on the state of the art of the results concerning the two above-mentioned problems. In particular, the search for a characterization of the regular languages in $H(FIN, FIN)$ is a largely investigated but difficult open problem. The difficulty is also due to the fact that, as observed in [5], the computational power of finite splicing systems (i.e., splicing systems $H = (A, I, R)$ with $I, R \in FIN$) increases when we substitute Head's systems with Paun's systems and Paun's systems with Pixton's systems.

The main result of this paper is the solution of Problem 2 for the special class of the *reflexive finite splicing systems* introduced in [9, 10]. As a byproduct, we give a characterization of the regular languages generated by *finite Head splicing systems*. To be more precise, let us first consider the more largely used Paun's definition of splicing. When the binary relation induced by the set of rules is reflexive, we have a reflexive Paun splicing system. One of the results in this paper characterizes finite Head splicing systems as reflexive Paun splicing systems which, in addition, satisfy a transitive hypothesis, i.e., we will state the following result:

Main result 1. A regular language L is generated by a finite Head splicing system if and only if L is generated by a finite Paun reflexive splicing system $H = (A, I, R_B \cup R_C)$ which is R_B-transitive and R_C-transitive (Theorem 2).

So, finite Head systems are all reflexive and this explains why reflexivity is a quite natural property of splicing systems.

The characterization of reflexive splicing languages we give here, extends the description of languages generated by a special class of reflexive Paun splicing systems (with rules with one-sided contexts, see Section 3) obtained in [9]. It is worthy of note that all these results are obtained by using only the classical notion of *constant* introduced by Schützenberger [19]. Indeed, we prove that, when

we take into account Paun's and Pixton's definitions, L is a reflexive splicing language if and only if there exist a finite set $\mathcal{M}$ of constants for L such that L is of the following form: $L = Y \cup \bigcup_{m \in \mathcal{M}} L(m) \cup \bigcup_{(\alpha,\beta) \in J} L_{(\alpha,\beta)}$, where $L(m)$ is a set of words having m as a factor (*constant languages*), Y is a finite set of words such that no m is a factor of a word of Y. The (finite) set J and the structure of $L_{(\alpha,\beta)}$ depend on the splicing operation we choose. $L_{(\alpha,\beta)}$ is a language obtained by extending the splicing operation to two constant languages $L(m)$ and $L(m')$. As a consequence, we have three definitions for $L_{(\alpha,\beta)}$. Depending on which definition of splicing we take into account, $L_{(\alpha,\beta)}$ will be termed a X-split language, whereas L will be a X-con-split language, with $X \in \{H, PA, PI\}$ (see Definitions 7 and 13). In conclusion, we will state the following result:

Main result 2. A regular language is generated by a finite Paun (resp. Pixton) reflexive splicing system if and only if L is a PA-con-split (resp. PI-con-split) language (Theorems 1, 3).

and, as a consequence, we give a structural property of the regular languages generated by finite Head's systems (Corollary 1). In the proofs of these results we exhibit the splicing systems that generate reflexive languages. However, Problem 1 remains open even for reflexive languages. As a final observation, we strongly guess the existence of finite Paun (or Pixton) splicing languages which are not reflexive.

This paper is organized as follows. Basics on words and linear splicing are gathered in Section 2, together with a decidability property for a regular language used in the proofs of our results. Definitions and preliminary properties of reflexive Paun splicing languages are collected in Section 3. The description of our class of splicing languages and the characterization of reflexive Paun splicing languages is presented in Section 4. Regular languages generated by finite Head splicing systems are presented in Section 5 and the characterization of reflexive Pixton splicing languages is given in Section 6. An extended version of this paper contains all the missing proofs of the results presented in this abstract [2].

2 Preliminaries

2.1 Words

Let A^* be the free monoid over a finite alphabet A and let $A^+ = A^* \setminus 1$, where 1 is the empty word. In the following $\mathcal{A} = (Q, A, \delta, q_0, F)$ will be a finite state automaton, where Q is a finite set of states, $q_0 \in Q$ is the initial state, $F \subseteq Q$ is the set of final states, δ is the transition function and $L(\mathcal{A})$ is the language recognized by $\mathcal{A}$ [1, 11, 16]. A finite state automaton $\mathcal{A}$ is *deterministic* if, for each $q \in Q$, $a \in A$, there is at most one state $q' \in Q$ so that $\delta(q, a) = q'$. Furthermore, $\mathcal{A}$ is *trim* if each state is accessible and coaccessible, i.e., for each state $q \in Q$ there are $x, y \in A^*$ such that $\delta(q_0, x) = q$ and $\delta(q, y) \in F$. Given a regular language $L \subseteq A^*$ it is well known that there is a *minimal* finite state automaton $\mathcal{A}$ recognizing it. This automaton is unique up to a possible renaming

of the states, is deterministic, trim and has the minimal number of states. As usual, in the transition diagram of a finite state automaton $\mathcal{A}$, each final state will be indicated by a double circle and the initial state will be indicated by an arrow without a label going into it. If it is not differently supposed, $L \subseteq A^*$ will always be a regular language.
Let us recall the definition of a constant, already introduced by Schützenberger in [19]. Let $\mathcal{A}$ be the minimal finite state automaton recognizing a regular language L and let $w \in A^*$. We will set $Q_w(\mathcal{A}) = \{q \in Q \mid \delta(q, w) \ \textit{is defined}\}$, simply indicated Q_w when the context makes the meaning evident. The *left* and *right* contexts of a word $w \in A^*$ are therefore defined as follows: $C_{\mathcal{L}}(w, L) = \{z \in A^* \mid \exists q \in Q_w : \delta(q_0, z) = q\}$, $C_{\mathcal{R},q}(w, L) = \{y \in A^* \mid \delta(q, wy) \in F\}$, $C_{\mathcal{R}}(w, L) = \bigcup_{q \in Q_w} C_{\mathcal{R},q}(w, L)$.

Notice that these definitions are slightly different from the ones given in [1]. Furthermore, we denote $C(w, L) = \{(x, y) \in A^* \times A^* \mid xwy \in L\}$ the set of *contexts* of w with respect to L. We recall that two words w, w' are equivalent with respect to the *syntactic congruence* of L if they have the same set of contexts with respect to L, i.e., $w \equiv_L w' \Leftrightarrow [\forall x, y \in A^*, xwy \in L \Leftrightarrow xw'y \in L] \Leftrightarrow C(w, L) = C(w', L)$ [12, 16].

A word $w \in A^*$ is a *constant* for a regular language L if $A^*wA^* \cap L \neq \emptyset$ and $C(w, L) = C_{\mathcal{L}}(w, L) \times C_{\mathcal{R}}(w, L)$ [19]. A characterization of constants, which is more or less folklore, is stated below.

Proposition 1. *Let $L \subseteq A^*$ be a regular language and let $\mathcal{A}$ be the minimal finite state automaton recognizing L. A word $w \in A^*$ is a constant for L if and only if $Q_w \neq \emptyset$ and there exists $q_w \in Q$ such that for all $q \in Q_w$ we have $\delta(q, w) = q_w$.*

If it is not differently supposed, for a regular language $L \subseteq A^*$, we will always refer to the minimal finite state automaton $\mathcal{A}$ recognizing L. Thus, the definitions of $C_{\mathcal{L}}(w, L)$ and $C_{\mathcal{R}}(w, L)$ can be given as follows: $C_{\mathcal{L}}(w, L) = \{z \in A^* \mid \exists y \in A^* : zwy \in L\}$, $C_{\mathcal{R}}(w, L) = \{z \in A^* \mid \exists y \in A^* : ywz \in L\}$. Obviously, $Q_1 = Q$ and $\delta(q, 1) = q$, for each $q \in Q$. Thus, in virtue of Proposition 1, if w is a constant for L we have $w \neq 1$ unless Q has only one element.

2.2 Linear Splicing

As we have already said, there are three definitions of linear splicing operation, given by Head, Paun and Pixton respectively [10, 15]. The difference among them depends on the biological phenomena that they want to model, an aspect which will not be discussed here. Paun's definition is given below, whereas Head's (resp. Pixton's) definition is given in Section 5 (resp. 6).

`Paun's definition [13].` A *Paun splicing system* is a triple $S_{PA} = (A, I, R)$, where $I \subset A^*$ is a set of strings, called *initial language*, R is a set of *rules* $r = u_1 \# u_2 \$ u_3 \# u_4$, with $u_i \in A^*, i = 1, 2, 3, 4$ and $\#, \$ \notin A$. Given two words $x = x_1 u_1 u_2 x_2, y = y_1 u_3 u_4 y_2$, $x_1, x_2, y_1, y_2 \in A^*$ and the rule $r = u_1 \# u_2 \$ u_3 \# u_4$, the splicing operation produces $w' = x_1 u_1 u_4 y_2$ and $w'' = y_1 u_3 u_2 x_2$, denoted

$(x, y) \vdash_r (w', w'')$. We also say that u_1u_2, u_3u_4 are *sites* of splicing and we denote $SITES(R)$ the set of sites of the rules in R.

Our study will consider having an unlimited number of copies of each word in the set, so that a pair of strings (x, y) can generate more than one pair of words with the use of different rules. Let $L \subseteq A^*$. We denote $\sigma'(L) = \{w', w'' \in A^* \mid (x, y) \vdash_r (w', w''),\ x, y \in L, r \in R\}$. The (iterated) splicing operation is defined as follows: $\sigma^0(L) = L, \sigma^{i+1}(L) = \sigma^i(L) \cup \sigma'(\sigma^i(L)),\ i \geq 0,\ \sigma^*(L) = \bigcup_{i \geq 0} \sigma^i(L)$.

Definition 1 (Paun splicing language). *Given a splicing system $S_{PA} = (A, I, R)$, the language $L(S_{PA}) = \sigma^*(I)$ is the language generated by S_{PA}. A language L is S_{PA}* generated *(or is a* Paun splicing language*) if a splicing system S_{PA} exists such that $L = L(S_{PA})$.*

A splicing system is *finite* when I, R are finite sets.

2.3 Languages Closed with respect to a Set of Rules

In the next part of this paper we suppose that each rule r, in a given splicing system S_{PA}, is *useful*, i.e., there exist $x, y, w', w'' \in L(S_{PA})$ such that $(x, y) \vdash_r (w', w'')$. A notion which will be used in the sequel is that of *languages closed with respect to a rule*. This definition was formally given in [4] and in [7] and is reported below.

Definition 2. *[4, 7] A language $L \subset A^*$ is* closed with respect to *a rule r if and only if for each $x, y \in L$, if $(x, y) \vdash_r (w', w'')$ then $w', w'' \in L$.*

Indeed, in order to prove that a language L is generated by a splicing system, in particular we must find a language $I \subseteq L$, such that, starting from I, the application of the splicing rules generates words in L. Problem 3 is a natural question which follows.

Problem 3. Let L be a regular language and let R be a set of rules in a finite Paun splicing system $S_{PA} = (A, I, R)$. Is L closed with respect to R?

In [4], the authors provided a characterization of languages closed with respect to a rule via automata. This characterization, reported in Lemma 1 below, gives a decision algorithm for Problem 3.

Lemma 1. *[4] Let $S_{PA} = (A, I, R)$ be a finite Paun splicing system, let $L \subseteq A^*$ be a regular language and let $\mathcal{A}$ be the minimal finite state automaton recognizing L. Then $L = L(\mathcal{A})$ is closed with respect to a rule $r = u_1 \# u_2 \$ u_3 \# u_4 \in R$ if and only if for each pair $(p, q) \in Q_{u_1u_2} \times Q_{u_3u_4}$, we have*
(1) $C_{\mathcal{R},p}(u_1u_2, L) \subseteq C_{\mathcal{R},q}(u_3u_2, L)$,
(2) $C_{\mathcal{R},q}(u_3u_4, L) \subseteq C_{\mathcal{R},p}(u_1u_4, L)$.

Lemma 2 shows a relationship between languages which are closed with respect to a rule and splicing languages. This relationship will be used to characterize reflexive splicing languages.

Lemma 2. *Let $S_{PA} = (A, I, R)$ be a splicing system and let $L \subseteq A^*$. If $I \subseteq L$ and L is closed with respect to each rule in R, then $L(S_{PA}) \subseteq L$.*

3 Reflexive Paun Splicing Languages

In this section we define the class of splicing languages we deal with when we restrict ourselves to Paun's definition of splicing. Let $S_{PA} = (A, I, R)$ be a finite Paun splicing system, let $R' \subseteq R$, let $SITES(R')$ be the set of sites of the rules in R'. We denote $Rel(R')$ the binary relation over $\{u_1\#u_2 \mid u_1u_2 \in SITES(R')\}$ induced by $SITES(R')$, that is $u_1\#u_2\ Rel(R')\ u_3\#u_4$ if and only if $u_1\#u_2\$u_3\#u_4 \in R'$.

Definition 3 (reflexive Paun splicing system). *[9] A finite splicing system $S_{PA} = (A, I, R)$ is a* reflexive Paun splicing system *if and only if the relation $Rel(R)$ is reflexive, i.e., if $u_1\#u_2\$u_3\#u_4 \in R$, then $u_1\#u_2\$u_1\#u_2$ and $u_3\#u_4\$ u_3\#u_4 \in R$.*

A language L is called *PA-reflexive* if there exists a finite reflexive Paun splicing system S_{PA} such that $L = L(S_{PA})$. PA-reflexive languages have been introduced for the first time in [9, 10] and Lemma 3 shows how they can be easily characterized in terms of constants.

Lemma 3. *A regular language $L \subseteq A^*$ is PA-reflexive if and only if there exists a finite splicing system $S_{PA} = (A, I, R)$ so that each site of the rules in R is a constant for L and $L = L(S_{PA})$.*

A special class of reflexive Paun splicing languages, the *constant languages* have been considered in [9]. Constant languages are the simplest regular Paun reflexive languages and we recall their definition below.

Definition 4 (constant language). *Let L be a regular language and let $m \in A^*$ be a constant for L. The* constant language *associated with m is the language $L(m) = \{y \in L \mid y = y_1' m y_2', y_1', y_2' \in A^*\}$.*

We also need Proposition 2 concerning languages which are finite union of constant languages.

Proposition 2 (union of constant languages). *[9] Let $L \subseteq A^*$ be a regular language and let $\mathcal{M} \subseteq A^*$ be a finite set of constants for L. Then $L' = \cup_{m \in \mathcal{M}} L(m)$ is a PA-reflexive language. Furthermore, L can be generated by a Paun splicing system in which each rule has either the form $u\#1\$v\#1$ or $1\#u\$1\#v$ (one-sided contexts), where u, v are constants for L.*

4 Main Result

In this section we illustrate our first main result, i.e., a complete characterization of the class of reflexive Paun splicing languages (Theorem 1). Given two constants m, m' for L and the two sets of the factorizations of these constants into two words, we begin with the definition of a language obtained by splicing the two constant languages $L(m), L(m')$ (Definition 6). By means of this operation, in Definition 7 we introduce the class $\mathcal{D}_{PA-con-split}$ which we prove to be the class of the reflexive Paun splicing languages. Roughly, a language L is in $\mathcal{D}_{PA-con-split}$ if and only if L can be obtained by the application of the above-mentioned operation to a finite number of pairs of constant languages and starting with the union of a finite set and a finite union of constant languages. Example 1 shows that this operation must be necessarily introduced.

Definition 5 (split of a constant). *Let L be a regular language and let m be a constant for L. A* split of the constant m *is a pair (x_1, x_2) of words in A^* such that $x_1 x_2 = m$.*

We denote by $F(m) = \{(x_1, x_2) \mid x_1 x_2 = m\}$ the set of the splits of the constant m. Given two constants m, m' for L and the sets $F(m)$, $F(m')$ of the splits for these constants, we can define a language obtained "by splicing" the two constant languages $L(m), L(m')$.

Definition 6 (PA-split language). *Let L be a regular language and let m and m' be two constants for L. Given $\alpha \in F(m)$ and $\beta \in F(m')$, such that $\alpha = (\alpha_1, \alpha_2)$ and $\beta = (\beta_1, \beta_2)$, the PA-*split language *generated by α and β is the language:*

$$L_{(\alpha,\beta)} = C_{\mathcal{L}}(m, L)\ \alpha_1\beta_2\ C_{\mathcal{R}}(m', L) \cup C_{\mathcal{L}}(m', L)\ \beta_1\alpha_2\ C_{\mathcal{R}}(m, L)\ .$$

The above definition is necessary to complete the characterization of Paun reflexive languages. Indeed, in Theorem 1, we will show that L is a PA-reflexive language if and only if $L \in \mathcal{D}_{PA-con-split}$, where $\mathcal{D}_{PA-con-split}$ is defined below. We recall that given two words $w, x \in A^*$, w is a *factor* of x if there exist $y, z \in A^*$ so that $x = ywz$.

Definition 7 (PA-con-split language). *Let L be a regular language and let $\mathcal{M}$ be a finite set of constants for L. Let Y be a finite subset of L such that m is not a factor of a word in Y, for each $m \in \mathcal{M}$. Let $J \subseteq \{(\alpha, \beta) \mid \alpha \in F(m), \beta \in F(m'), m, m' \in \mathcal{M}\}$. L is a PA-*con-split language *(associated with $Y, \mathcal{M}, J$) if and only if*

$$L = Y\ \cup \bigcup_{m \in \mathcal{M}} L(m)\ \cup \bigcup_{(\alpha,\beta) \in J} L_{(\alpha,\beta)}\ .$$

$\mathcal{D}_{PA-con-split}$ is the class of PA-con-split languages.

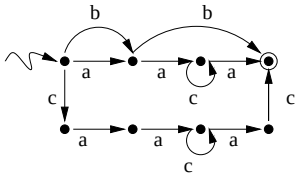

Fig. 1. A PA-reflexive splicing language which is not a finite union of constant languages

Example 1. In Figure 1 we report the transition diagram of a finite state automaton recognizing the language $L = ab \cup bb \cup aac^*a \cup bac^*a \cup caac^*ac$. Let us check that L is a PA-con-split language which is not a finite union of constant languages. Observe that ba, caa, bb and ab are constants for L and so bac^*a, $caac^*ac$, bb and ab are constant languages. On the contrary, it is not too difficult to see that aac^*a is not a finite union of constant languages. However, we have that $aac^*a \cup bb = L_{(\alpha,\beta)}$ where $m = ba$, $m' = ab$, $L(m) = bac^*a$, $L(m') = ab$, $\alpha = (b, a)$, $\beta = (a, b)$. Thus, L is a PA-con-split language. Finally, by using the characterization given in [9], we stress that L is not a finite union of constant languages since it is not too difficult to see that, for any finite set F of constants for L, we have that $aac^*a \subseteq L \setminus \cup_{f \in F} A^* f A^*$.

We end this section with one of the main results in this paper.

Theorem 1. *A regular language $L \subseteq A^*$ is a PA-reflexive language if and only if L belongs to $\mathcal{D}_{PA-con-split}$.*

Remark 1. Notice that $L_{(\alpha,\beta)}$ can be considered as the result of an extension of the splicing operation to languages $L(m) = C_{\mathcal{L}}(m, L)\ \alpha_1\alpha_2\ C_{\mathcal{R}}(m, L)$ and $L(m') = C_{\mathcal{L}}(m', L)\ \beta_1\beta_2\ C_{\mathcal{R}}(m', L)$, $m = \alpha_1\alpha_2 \in \mathcal{M}$, $m' = \beta_1\beta_2 \in \mathcal{M}$. Indeed, for proving that a PA-con-split language L is PA-reflexive, we construct a splicing system $S_{PA} = (A, I, R)$ in which, among the rules in R, we find rules with the form $\alpha_1\#\alpha_2\$\beta_1\#\beta_2$, for each $\alpha = (\alpha_1, \alpha_2) \in F(m)$, $\beta = (\beta_1, \beta_2) \in F(m')$. Conversely, for showing that a PA-reflexive language $L = L(S_{PA})$ is a PA-con-split language, we define J as above starting with the rules in S_{PA}.

Example 2. Let L be the language considered in Example 1. Following the proof of Theorem 1, we have that $L = L(S_{PA})$, where $S_{PA} = (A, I, R)$, $I = \{ab, bb, caaac, caacac, aaa, aaca, baa, baca\}$ and $R = \{caa\#1\$caac\#1, ba\#1\$\, bac\, \#1, b\#a\$a\#b\}$.

As far as we know, it is not known whether we can decide if a language L belongs to $\mathcal{D}_{PA-con-split}$, i.e., whether L is a PA-reflexive language. Observe that this problem generalizes the analogous question, proposed in [9], concerning languages which are union of a finite set and a finite union of constant languages.

Remark 2. It seems that reflexive Head (resp. Pixton) splicing systems have not been considered in the literature. On the other hand for each Head (resp. Paun) splicing system S_H (resp. S_{PA}) there exists a canonical transformation ϕ such that $\phi(S_H)$ is a Paun (resp. $\phi(S_{PA})$ is a Pixton) splicing system S'_{PA} (resp. S'_{PI}) and $L(S_H) = L(S'_{PA})$ (resp. $L(S_{PA}) = L(S'_{PI})$) [15]. In the next two sections, we give the definition of a reflexive Head (resp. Pixton) splicing system which allows us to preserve these canonical transformations.

5 A Description of Finite Head Splicing Languages

In this section we prove the other main results of the paper (Theorem 2, Corollary 1). Corollary 1 is a consequence of Theorem 2 which characterizes the structure of finite Head splicing systems inside the class of finite Paun splicing systems. Indeed, Theorem 2 shows the equivalence between finite Head splicing systems and a subclass of reflexive Paun splicing systems satisfying a transitive hypothesis. We also state that Head splicing languages are a proper subset of PA-reflexive languages (see Example 1 and [5]). Let us first recall the original definition of splicing operation given by Head.

`Head's definition [8].` A *Head splicing system* is a 4-uple $S_H = (A, I, B, C)$, where $I \subset A^*$ is a finite set of strings, called *initial language*, B and C are finite sets of triples (α', μ, β'), called *patterns*, with $\alpha', \beta', \mu \in A^*$ and μ called the *crossing* of the triple. Given two words $u\alpha'\mu\beta'v$, $p\alpha''\mu\beta''q \in A^*$ and two patterns $p_1 = (\alpha', \mu, \beta')$ and $p_2 = (\alpha'', \mu, \beta'')$ that have the same crossing and are both in B or both in C, the splicing operation produces $w' = u\alpha'\mu\beta''q$ and $w'' = p\alpha''\mu\beta'v$, denoted $(x, y)\vdash_{p_1,p_2}(w', w'')$. We also say that $\alpha'\mu\beta'$, $\alpha''\mu\beta''$ are *sites* of splicing.

Clearly, a Head splicing system is *finite* when I, B, C are finite sets. Furthermore, we set $\sigma'(L) = \{w', w'' \in A^* \mid (x, y)\vdash_{p_1,p_2} (w', w''),\ x, y \in L, p_1, p_2$ both in B or both in $C\}$ and, as in Paun's systems, given $S_H = (A, I, B, C)$, the language $L(S_H) = \sigma^*(I)$ is the language generated by S_H. A language L is S_H *generated* (or L is a *Head splicing language*) if a splicing system S_H exists such that $L = L(S_H)$.

Remark 3. Looking for a definition of a reflexive Head splicing system which takes into account Remark 2, we find that each Head splicing system S_H can be implicitly supposed to be reflexive. This property is satisfied since for each pattern $p = (\alpha', \mu, \beta')$ we can always apply the splicing operation to two copies of a word having $\alpha'\mu\beta'$ as a factor [10]. Furthermore, we can easily prove that each site of a pattern (in B or C) is a constant for $L = L(S_H)$ (see [2]).

In order to prove the main results presented in this section, we consider again Paun's systems and we define a new relation $Rel_\mu(R')$.

Definition 8 ($Rel_\mu(R')$ relation). *Let $S_{PA} = (A, I, R)$ be a finite reflexive Paun splicing system and let $\mu \in A^*$. For $R' \subseteq R$, we set $Rel_\mu(R')$ the binary relation over $\{u_1\#u_2 \mid u_1u_2 \in SITES(R')\}$ defined as follows:*

$u_1\#u_2\ Rel_\mu(R')\ u_3\#u_4$ *if and only if* $u_1 = u_1'\mu, u_3 = u_3'\mu$.

Definition 9 (**Rel_μ-full**). *Let $S_{PA} = (A, I, R)$ be a finite reflexive Paun splicing system and let $\mu \in A^*$. The set $R' \subseteq R$ is* Rel_μ-full *if and only if for each $u_1\#u_2, u_3\#u_4 \in \{x_1\#x_2 \mid x_1x_2 \in SITES(R')\}$, we have $u_1\#u_2\ Rel_\mu(R')\ u_3\#u_4$ and $u_1\#u_2\ Rel(R')\ u_3\#u_4$.*

We denote $R'_\mu = \{u_1\#u_2\$u_3\#u_4 \in R' \mid u_1\#u_2\ Rel_\mu(R')\ u_3\#u_4\}$. Roughly, R'_μ contains rules in R' such that for all $u_1u_2, u_3u_4 \in SITES(R'_\mu)$, u_1, u_3 satisfy the condition on the existence of the common suffix μ. Furthermore, the set R'_μ is Rel_μ-full if for all $u_1u_2, u_3u_4 \in SITES(R'_\mu)$, there exists a rule $u_1\#u_2\$u_3\#u_4$ in R'_μ.

Definition 10 (**R'-transitive**). *Let $S_{PA} = (A, I, R)$ be a finite reflexive Paun splicing system and let $R' \subseteq R$. S_{PA} is* R'-transitive *if there exist $m \geq 1$, $\mu_1, \ldots, \mu_m \in A^*$ such that $\bigcup_{i=1}^{m} R'_{\mu_i} = R'$ and each R'_{μ_i} is Rel_{μ_i}-full, for $1 \leq i \leq m$.*

Theorem 2. *1) Each finite Head splicing system $S_H = (A, I, B, C)$ is equivalent to a finite Paun splicing system $S_{PA} = (A, I, R_B \cup R_C)$ which is reflexive and is R_B-transitive and R_C-transitive.*

2) Each finite Paun splicing system $S_{PA} = (A, I, R_B \cup R_C)$ which is reflexive, R_B-transitive and R_C-transitive is equivalent to a finite Head splicing system $S_H = (A, I, B, C)$.

Thanks to Theorems 1 and 2, we can give a structural property of regular languages generated by finite Head splicing systems. Indeed, the required R_B-transitivity and R_C-transitivity for the set $R = R_B \cup R_C$ of the rules in a Paun system imply an additional hypothesis on the set J introduced in Definition 7. Precisely, an *H-split language* is a PA-split language. An *H-con-split language* is a PA-con-split language such that $\mathcal{M} = \mathcal{M}' \cup \mathcal{M}''$, $\mathcal{M}', \mathcal{M}'' \subseteq \mathcal{M}$ and $J = J_{\mathcal{M}'} \cup J_{\mathcal{M}''}$, where $J_{\mathcal{M}'} = \{(\alpha, \beta) \mid \alpha = (\alpha_1, \alpha_2) \in F(m), \beta = (\beta_1, \beta_2) \in F(m'), m = x\mu y \in \mathcal{M}', \alpha_1 = x\mu, m' = x'\mu y' \in \mathcal{M}', \beta_1 = x'\mu\}$ and $J_{\mathcal{M}''} = \{(\alpha, \beta) \mid \alpha = (\alpha_1, \alpha_2) \in F(m), \beta = (\beta_1, \beta_2) \in F(m'), m = x\mu y \in \mathcal{M}'', \alpha_1 = x\mu, m' = x'\mu y' \in \mathcal{M}'', \beta_1 = x'\mu\}$. Roughly, in Head's case the pairs $((\alpha_1, \alpha_2), (\beta_1, \beta_2))$ in $J_{\mathcal{M}'}$ (or in $J_{\mathcal{M}''}$) must satisfy the additional condition on the existence of a common suffix for the first components α_1, β_1 of the factorizations.

Corollary 1. *Let $L \subseteq A^*$ be a finite Head splicing language. Then L is an H-con-split language (generated by a finite Paun splicing system $S_{PA} = (A, I, R_B \cup R_C)$ which is reflexive, R_B-transitive and R_C-transitive).*

Let $L \subseteq A^*$ be an H-con-split language generated by a finite Paun splicing system $S_{PA} = (A, I, R_B \cup R_C)$ which is reflexive, R_B-transitive and R_C-transitive. Then L is a finite Head splicing language.

6 Reflexive Pixton Splicing Languages

In order to end the characterization of reflexive splicing languages, we briefly present their structure when we take into account Pixton's definition of the splicing operation.

`Pixton's definition [18].` A *Pixton splicing system* is a triple $S_{PI} = (A, I, R)$, where $I \subset A^*$ is a set of strings, called *initial language*, R is a finite collection of rules $r = (\alpha', \alpha''; \beta')$, $\alpha', \alpha'', \beta' \in A^*$. Given two words $x = \epsilon\alpha'\eta$, $y = \epsilon'\alpha''\eta'$ and the rule $r = (\alpha', \alpha''; \beta')$, the splicing operation produces $w = \epsilon\beta'\eta'$, denoted $(x, y) \vdash_r w$. We also say that α', α'' are *sites* of splicing.

A Pixton splicing system is *finite* if I, R are finite sets and a language L is S_{PI} *generated* (or L is a *Pixton splicing language*) if a splicing system S_{PI} exists such that $L = L(S_{PI})$.

Definition 11 (reflexive Pixton splicing system). *A finite splicing system $S_{PI} = (A, I, R)$ is a PI-reflexive splicing system if and only if for each rule $(\alpha', \alpha''; \beta') \in R$ we also have $(\alpha', \alpha'; \alpha')$, $(\alpha'', \alpha''; \alpha'') \in R$.*

A language L is called *PI-reflexive* if there exists a finite Pixton reflexive splicing system S_{PI} such that $L = L(S_{PI})$. As in Paun's case we can prove that a Pixton splicing language L is PI-reflexive if and only if there exists a finite splicing system $S_{PI} = (A, I, R)$ so that each site of the rules in R is a constant for L and $L = L(S_{PI})$ [2].

The definitions and results which follow are the extension to Pixton's systems of the analogous definitions and results given in Section 4.

Definition 12 (PI-split language). *Let $L \subseteq A^*$ be a regular language, let $\mathcal{M}$ be a finite set of constants for L, let $J \subseteq \{(m, m') \mid m, m' \in \mathcal{M}\}$. Let $\gamma : J \to A^*$ be a mapping. The PI-*split language *generated by m and m' is the language*

$$L_{(m,m')} = C_{\mathcal{L}}(m, L)\ \gamma(m, m')\ C_{\mathcal{R}}(m', L)\ .$$

Definition 13 (PI-con-split language). *Let $L \subseteq A^*$ be a regular language, let $\mathcal{M} \subseteq A^*$ be a finite subset of constants for L. Let $J \subseteq \{(m, m') \mid m, m' \in \mathcal{M}\}$ and let $\gamma : J \to A^*$ be a mapping. Let Y be a finite subset of L such that no $m \in \mathcal{M}$ is a factor of a word in Y.*
*L is a PI-*con-split language *(associated with $Y, \mathcal{M}, \gamma$) if and only if*

$$L = Y\ \cup \bigcup_{m \in \mathcal{M}} L(m)\ \cup \bigcup_{(m,m') \in J} L_{(m,m')}\ .$$

$\mathcal{D}_{PI-con-split}$ is the class of the PI-con-split languages.

Theorem 3. *A regular language $L \subseteq A^*$ is a PI-reflexive language if and only if L belongs to $\mathcal{D}_{PI-con-split}$.*

References

1. Berstel, J., Perrin, D.: Theory of codes. Academic Press, New York (1985)
2. Bonizzoni, P., De Felice, C., Mauri, G., Zizza, R.: The structure of reflexive regular splicing languages via Schützenberger constants. *manuscript* (2003)
3. Bonizzoni, P., De Felice, C., Mauri, G., Zizza, R.: Decision Problems on Linear and Circular Splicing. In: Ito, M., Toyama, M. (eds.): DLT 2002. Lecture Notes in Computer Science, Springer-Verlag, New York (2003)
4. Bonizzoni, P., De Felice, C., Mauri, G., Zizza, R.: On the power of linear and circular splicing. *submitted* (2002)
5. Bonizzoni, P., Ferretti, C., Mauri, G., Zizza, R.: Separating some splicing models. Information Processing Letters **79:6** (2001) 255 – 259
6. Gatterdam, R.W.: Algorithms for splicing systems. SIAM Journal of Computing **21:3** (1992) 507 – 520
7. Goode, E., Head, T., Pixton, D. *private communication* (2002)
8. Head, T.: Formal Language Theory and DNA. An analysis of the generative capacity of specific recombinant behaviours. Bull. Math. Biol. **49** (1987) 737– 759
9. Head, T.: Splicing languages generated with one sided context. In: Paun, Gh. (ed.): Computing with Bio-molecules. Theory and Experiments, Springer-Verlag Singapore (1998)
10. Head, T., Paun, Gh., Pixton, D.: 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
11. Hopcroft, J.E., Motwani, R., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation. 2nd edn. Addison-Wesley, Reading, Mass. (2001)
12. McNaughton, R., Papert, S.: Counter-Free Automata. MIT Press, Cambridge, Mass. (1971)
13. Paun, Gh.: On the splicing operation. Discrete Applied Mathematics **70** (1996) 57 – 79
14. Paun, Gh., Rozenberg, G., Salomaa, A.: Computing by splicing. Theoretical Computer Science **168:2** (1996) 321 – 336
15. Paun, Gh., Rozenberg, G., Salomaa, A.: DNA computing, New Computing Paradigms. Springer-Verlag (1998)
16. Perrin, D.: Finite Automata. In: van Leeuwen, J. (ed.): Handbook of Theoretical Computer Science, Vol. B. Elsevier (1990) 1 – 57
17. Pixton, D.: Linear and Circular Splicing Systems. In: Proc. of 1st Int. Symp. on Int. in Neural and Biological Systems (1996) 181 – 188
18. Pixton, D.: Regularity of splicing languages. Discrete Applied Mathematics **69** (1996) 101 – 124
19. Schützenberger, M.-P.: Sur certaines opérations de fermeture dans le langages rationnels. Symposia Mathematica **15** (1975) 245 – 253