DNA and circular splicing *

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

[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
rosziz@bioinformatics.bio.disco.unimib.it
[2] Dipartimento di Informatica ed Applicazioni,
Università di Salerno, 84081 Baronissi (SA), Italy
defelice@unisa.it

Abstract. Circular splicing has been very recently introduced to model a specific recombinant behaviour of circular DNA, carrying on the investigation initiated with linear splicing. In this paper we restrict ourselves to the relationship between circular regular languages and circular splicing languages. We provide partial results towards a characterization of the class of circular regular languages generated by finite circular splicing systems. We consider a class of languages X^* closed under conjugacy relation and with X a regular languages, called here *star languages*. Using automata theory and combinatorial techniques on words, we show that for a subclass of star languages the corresponding circular languages are circular (Paun) splicing languages. In particular, star languages with X being a finite set or X^* being a free monoid belong to this subclass.

1 Introduction

The notion of *splicing systems* was firstly introduced in [11], where T. Head modelled a recombinant behaviour of DNA molecules (under the action of restriction and ligase enzymes) as a particular operation between words in a formal language and so suggesting the possibility of using molecules to perform computations. Since then, the way from DNA Recombination to DNA Computing via formal languages has been largely explored for both theoretical and technological implications. In this context, the *splicing* operation has been used to model specific recombinant behaviours. In short, a restriction enzyme is able to recognize a pattern in a DNA molecule and to cut this molecule inside the pattern in a specific position, so providing two segments of DNA. Then a ligase enzyme binds together pairs of sequences coming from different molecules, so generating a new molecule. In order to treat this cut and paste phenomenon in

* Partially supported by MURST Project *"Unconventional Computational Models: Syntactic and Combinatorial Methods"* - *"Modelli di calcolo innovativi: Metodi sintattici e combinatori"*.

** E-mail address for correspondence: rosziz@bioinformatics.bio.disco.unimib.it

formal language theory, an initial set of strings (the initial set of DNA molecules) and a set of splicing rules or special words (simulators of enzymes behaviour) are given, such that the set of all possible molecules generated by the biochemical process of cut and paste is nothing but the language of all possible words that may be generated by applying the rules to the strings. We do not take into account the bidimensional nature of DNA even if very recently a link has been given between splicing systems and picture languages (see [10] for a survey on picture languages). We also deliberately ignore other molecular considerations and basic details (for a complete monograph see [17]).

DNA occurs in both linear and circular form and, correspondingly, linear and circular splicing systems have been defined. The introduction of circular splicing rules may be justified as follows: an example of this type of splicing occurs in a recombinant mechanism (transposition) between bacteria and plasmides. Moreover, the replication is possible only if DNA has a circular form in the host cell. Furthermore, it could be interesting to use circular DNA in Adleman's experiment, since plasmides autoreplicate themselves without errors and exponential weight increasing, which are weak points in Adleman's approach [1]. In this paper we mainly deal with circular splicing systems, providing interesting links with the linear case, which has been deeply investigated. Furthermore, in our theoretical model there is no hypothesis on the number of copies of the initial set of molecules.

Classical results in formal language theory and combinatorial tools have been of great use for reaching the results given in this paper. It is well known that three definitions of (linear) splicing systems are given by Head, Paun and Pixton. The computational power of these systems, i.e. the class of languages which are generated by them, has been completely described. This computational power depends on which level in the Chomsky hierarchy the initial set I and the set of the rules R belong to. In particular, if the last two sets are finite, the splicing-generated languages are a subclass of the regular languages, completely characterized in [4]. We now state some results towards the proof of a counterpart of this characterization for circular splicing. Precisely, three corresponding definitions of circular splicing operations have been given (by Head, Paun and Pixton). We show that under an additional hypothesis, for a star language (i.e. a language X^* closed under conjugacy relation and with X a regular language), the corresponding circular language is generated by a circular Paun splicing system with a finite initial set I and a finite set R of rules (Theorem 1). Furthermore, this result finds a deeper insight in the framework of the theory of variable-length codes [2]. Indeed, we prove that for all star languages X^* with X being a finite set or X being a rational code, the corresponding circular languages are circular splicing languages (Proposition 5 and Corollary 1). Observe that in contrast with the linear case, a finite initial set and a finite set of rules do not guarantee that the splicing-generated language is regular.

Let us briefly sketch the contents of this paper. In Section 2 we gathered an overview on linear splicing and easy new and known results on circular splicing. In the same section, we also explicitly observe that the computational power of the three circular splicing systems is already known to be different. As far as we know, very few results on circular splicing systems are known, and these are surveyed in Sections 2, 4. For the sake of completeness, in the same Section 4 we reported several additional hypotheses which could or could not be added to circular splicing systems. Some questions are also asked. In our results, presented in Section 3, we restrict ourselves to Paun's definition with no additional hypotheses. All the proofs which are not contained in this paper may be found in an extended version of the paper [5].

2 Languages and splicing: definitions

In the next part of this paper, we denote by A^* the free monoid over a finite alphabet A. We also set $A^+ = A^* \setminus 1$, where 1 is the empty word.

2.1 Linear splicing

In this paper we only deal with circular splicing, but for completeness, let us briefly sketch some concepts and results on linear splicing. Literature presents three definitions of the linear splicing operation [11, 16, 18]. One of them has been introduced by Paun, where the biological process presented in Section 1 has been formalized as follows [16]. The enzymes are encoded by a *splicing rule* $r = u_1|u_2\$u_3|u_4$, for $u_i \in A^*$ and $|, \$ \notin A$; the molecules are sequences $w', w'' \in A^*$. If $w' = x_1u_1u_2x_2$ and $w'' = y_1u_3u_4y_2$, the rule r can be applied to w' and w'', producing two new sequences $w = x_1u_1u_4y_2$ and $z = y_1u_3u_2x_2$. A splicing system S consists of a set R of rules (enzymes) and an initial language $I \subseteq A^*$ (molecules). The global recombinant behaviour of DNA molecules is described by the splicing language $L(S)$ generated by S. $L(S)$ is the smallest set of strings s.t. $I \subseteq L(S)$ and if w', w'' are in $L(S)$ and r is a rule that can be applied to w' and w'', producing w, z, then w and z must also be in $L(S)$. Let us denote $H(F_1, F_2)$ the class of languages generated by splicing systems $S = (I, R)$, where $I \in F_1$, $R \in F_2$ and F_1, F_2 are classes of languages. It is known that splicing systems can reach the same power of Turing Machines [16, 12]. As we have already said, when we restrict ourselves to splicing systems with a finite set R of rules and a finite set I of strings, we get a class $H(Fin, Fin)$ which is a proper subclass of regular languages, as shown in [8, 9, 14, 18]. Despite the interesting results obtained in this framework, no algorithm was yet known for deciding whether a regular language is generated by splicing. In [4] the first decision procedure to test whether a regular language can be generated by the splicing operation is given, deriving a complementary procedure to the ones proposed in [8, 9, 14, 18]. A very interesting consequence of this result is that this procedure allows us to compare the generative power of different splicing systems. Indeed, in [7] the authors point out that there is a regular language generated by Paun's (resp. Pixton's) splicing operation that cannot be generated by Head's (resp. Paun's) splicing operation.

2.2 Circular languages

Circular words have already been examined in formal language theory. Indeed, for a given word $w \in A^*$, a circular word $^\sim w$ is the equivalence class of w w.r.t. the *conjugacy* relation $\sim$ defined by $xy \sim yx$, for $x, y \in A^*$ (see [15]). $A^\sim$ is the set of all such circular words over A, i.e. the quotient of A^* w.r.t. $\sim$. Given $L \subseteq A^*$, $Cir(L) = \{^\sim w | w \in L\}$ is the *circularization* of L, i.e. the set of all circular words corresponding to elements of L, while any language L such that $Cir(L) = C$, for a given circular language $C \subseteq A^\sim$, is called a *linearization* of C. The set of all strings in A^* corresponding to elements of C is the *full linearization* of C, written $Lin(C)$. Circular splicing deals with circular strings and circular languages and as a result with formal languages which are full linearizations of circular languages. This latter representation will be used in the next part of this paper, where we will consider languages closed under conjugacy and the action of circular splicing over their circularization. We denote Fin (resp. Reg, CF) the class of finite (resp. regular, context free) languages. Given a family of languages FA in Chomsky hierarchy, $FA^\sim$ consists of all those circular languages C which have some linearization in FA. In particular, $C \in Reg^\sim$ if and only if its full linearization is regular [13, 18, 20]. In this paper we mainly deal with $Reg^\sim$.

2.3 Circular splicing

The splicing operation in the circular case deals with biological phenomena which are different from the ones modelled in the linear case, but the three definitions given for linear splicing systems can be adapted to the circular case.

`Head's definition [20].` A *circular (Head) splicing system* is a 4-tuple $SC_H = (A, I, T, P)$, where $I \subseteq A^\sim$ is the initial circular language, $T \subseteq A^* \times A^* \times A^*$ and P is a binary relation on T, such that, if $p, x, q, u, y, v \in A^*$, $(p, x, q), (u, y, v) \in T$ and $(p, x, q)P(u, y, v)$ then $x = y$. Thus, given $^\sim hpxq$, $^\sim kuxv \in A^\sim$ with $(p, x, q)P(u, x, v)$, the splicing produces $^\sim hpxvkuxq$.

`Paun's definition [16].` A *circular (Paun) splicing system* is a 3-tuple $SC_{PA} = (A, I, R)$, where $I \subseteq A^\sim$ is the initial circular language, $R \subseteq A^*|A^*\$A^*|A^*$, with $|, \$ \notin A$, is the set of rules. Then, given a rule $r = u_1|u_2\$u_3|u_4$ and two circular words $^\sim hu_1u_2, ^\sim ku_3u_4$, the rule cuts and linearizes the two strings obtaining u_2hu_1 and u_4ku_3, and pastes and circularizes them obtaining $^\sim u_2hu_1u_4ku_3$.

`Pixton's definition [18].` A *circular (Pixton) splicing system* is a 3-tuple $SC_{PI} = (A, I, R)$ where A is a finite alphabet, $I \subseteq A^\sim$ is the initial circular language, $R \subseteq A^* \times A^* \times A^*$ is the set of rules. R is s.t. if $r = (\alpha, \alpha'; \beta) \in R$ then $\overline{r} = (\alpha', \alpha; \beta') \in R$. Thus, given two circular words $^\sim \alpha h, ^\sim \alpha' k$, the two rules $r, \overline{r}$ cut and linearize the two strings, obtaining $h\alpha, k\alpha'$, and then paste, substitute and circularize them, producing $^\sim h\beta k\beta'$.

Remark 1. We must note that in the original definition of circular splicing language given by Paun in [12], rules in R can be used in two different ways: one way has been described above, the other, called *self-splicing*, will be defined in

Section 4. In this paper in order to make the three definitions uniform, we have preferred to delete self-splicing from Paun's definition.

We now give the definition of circular splicing languages. For a given splicing system SC_X, with $X \in \{H, PA, PI\}$, we denote $(w', w'') \vdash_r w$ the fact that w is produced from (or spliced by) w', w'' by using a rule r. Given a language $C \subseteq A^\sim$, we denote $\sigma(C) = \{z \in A^\sim \mid (w', w'') \vdash_r z$, for $w', w'' \in C$ and $r \in R\}$. Thus, we define $\sigma^0(C) = C$, $\sigma^{i+1}(C) = \sigma^i(C) \cup \sigma(\sigma^i(C))$, $i \geq 0$, and then $\sigma^*(C) = \bigcup_{i \geq 0} \sigma^i(C)$.

Definition 1. *Given a splicing system SC_X, with $X \in \{H, PA, PI\}$, the circular language $C(SC_X) = \sigma^*(I)$ is the language generated by the system SC_X, I being the initial language in SC_X.*
A circular language C is CX generated (or C is a circular splicing language) if a splicing system SC_X exists such that $C = C(SC_X)$.

2.4 On the generating power of circular splicing systems

In this paper we deal with Paun's systems $SC_{PA} = (A, I, R)$, with both I, R finite sets, and with the corresponding class of generated languages, denoted $C(Fin, Fin)$. As a matter of fact, the investigation about circular splicing systems is usually limited to ones with a finite or regular initial language. Furthermore, it goes without saying that the set of rules is finite in each definition.

Example 1. In [20] it is shown that $(aa)^*$ is not a linear splicing language, while the corresponding circular language $\{^\sim(aa)^n | n \geq 0\}$ is CH generated by $SC_H = (A, I, T, P)$ with $A = \{a\}$, $I = \{^\sim 1, ^\sim aa\}$, $T = \{(1, a, 1)\}$, and $(1, a, 1)P(1, a, 1)$. We can observe that $\{^\sim(aa)^n | n \geq 0\}$ is CPA generated, by choosing $A = \{a\}, I = \{^\sim aa\}, R = \{aa|1\$1|aa\}$. On the contrary, let us consider the regular language $(aa)^*b$. Using results in [4], we get $(aa)^*b \in H(Fin, Fin)$. Take its circularization $C =^\sim (aa)^*b$. Notice that there is a linearization of C, $\{a^k b a^k : k \in \mathbb{N}\}$ which is not regular. However, the full linearization is $(aa)^*\{b, aba\}(aa)^*$, which is regular and given in [18]. Thus $C \in Reg^\sim$, in virtue of the definition [18, 20]. We immediately see that C cannot be CH or CPA generated (Proposition 2). Nevertheless, it is not too difficult to prove that C is CPI generated, when we choose, for example, $A = \{a, b\}, I = \{^\sim b, ^\sim a^2b, ^\sim a^4b\}, R = \{(a^2, a^2b; a^2), (a^2b, a^2; 1)\}$.

The previous example shows the difference between linear and circular splicings. The same example also suggests the problem, formalized below, of comparing the computational power of the three definitions of the circular splicing operation given in Section 2.3.

Problem 1. Given $C = C(SC_X)$, with $X \in \{H, PA, PI\}$, does SC_Y exist, with $Y \neq X$, $Y \in \{H, PA, PI\}$, such that $C = C(SC_Y)$?

It is not too difficult to prove Proposition 1. This proposition and Example 1 show that the computational power of Pixton's systems is greater than the other two. This gives a partial answer to Problem 1.

Proposition 1. *If $C \subseteq A^\sim$ is CH generated, then C is CPA generated. Moreover, if $C \subseteq A^\sim$ is CPA generated, then C is CPI generated.*

Proposition 2, quoted in Example 1, allows us to state that some circular languages are not generated by Head's or Paun's systems.

Proposition 2. *Let $L = Lin(C)$ be an infinite language which is the full linearization of a circular language C. Suppose that L satisfies the following condition*

$$\forall l, m \in L : \quad lm \notin L.$$

Then $Cir(L)$ is not CPA generated.

Example 2. By using Proposition 2, we can easily state that some circular regular languages cannot be generated by circular Paun's splicing systems. For instance, let $L_1 = \{w \in A^* \mid \exists h, k \in \mathbb{N}\ |w|_a = 2k, |w|_b = 2h\}$ and $L_2 = \{w \in A^* \mid \exists h, k \in \mathbb{N}\ |w|_a = 2k+1, |w|_b = 2h+1\}$. L_1 and L_2 are languages closed under conjugacy. Moreover, L_1 and L_2 are regular languages (L_1 is the shuffle of $(aa)^*$ and $(bb)^*$, L_2 is the shuffle of $(aa)^*a$ and $(bb)^*b$). Thus, $Cir(L_1)$ and $Cir(L_2)$ are circular regular languages. By using Proposition 2, we can see that $Cir(L_2)$ is not CH or CPA generated. On the contrary, we will see in Section 3 that $Cir(L_1)$ is CPA generated.

3 Regular CPA generated languages and star languages

It is already known that in contrast with the linear case, $C(Fin, Fin)$ is not intermediate between two classes of languages (in the Chomsky hierarchy). For example, $\sim a^n b^n$ is a circular context-free language which is not a circular regular language (since it has no regular linearization), but $\sim a^n b^n$ is CH generated with finite initial circular language [20]. On the other hand, we have already seen examples of regular circular languages that cannot be generated by using CPA splicing systems with finite initial circular language (see Example 1). So far, we have not yet discovered whether $C(Fin, Fin)$ contains any context-sensitive or recursive enumerable language which is not context-free. We believe that $C(Fin, Fin) \subseteq NP$. Nevertheless, in this paper we restrict ourselves to the following problem.

Problem 2. Characterize $Reg^\sim \cap C(Fin, Fin)$.

We approach Problem 2 by dealing with the class of languages defined below.

Definition 2. *A* star language *is a language $L \subseteq A^*$ that satisfies the following conditions:*
(1) $L = X^*$, with X a regular language,
(2) L is closed under conjugacy relation.

Under an additional hypothesis, a star language has its circularization in $C(Fin, Fin)$, as shown in Theorem 1, whose proof can be obtained thanks to the following proposition.

Proposition 3. *Let X^* be a star language and let $SC_{PA} = (A, I, R)$ be a splicing system. If $I \subseteq Cir(X^*)$ then $C(SC_{PA}) \subseteq Cir(X^*)$.*

Proof. Let us prove that $C(SC_{PA}) \subseteq Cir(X^*)$ holds, by using induction over the length $|y|$ of $\sim y \in C(SC_{PA})$. If $\sim y \in I$ then $\sim y \in Cir(X^*)$, since $I \subseteq Cir(X^*)$. Otherwise, $y' \in \sim y$ exists such that $y' = u_2hu_1u_4ku_3$ with $\sim u_2hu_1, \sim u_4ku_3 \in C(SC_{PA})$ and $\sim u_2hu_1 \neq \sim y$, $\sim u_4ku_3 \neq \sim y$. Then $|u_2hu_1| < |y|, |u_4ku_3| < |y|$. By induction hypothesis, $\sim u_2hu_1, \sim u_4ku_3 \in Cir(X^*)$ and, X^* being closed under conjugacy, $u_2hu_1, u_4ku_3 \in X^*$. Thus, X^* being a submonoid, $u_2hu_1u_4ku_3 = y' \in X^*$, and so $\sim y \in Cir(X^*)$. □

In Theorem 1, for a star language X^* satisfying an additional hypothesis, we state how $Cir(X^*)$ is CPA generated (with I, R finite sets). We also use the following notations and we suppose the reader is familiar with elementary finite state automaton theory (see [2, 13]). Let $\mathcal{A} = (A, Q, \delta, i, F)$ be a finite state automaton. $\mathcal{A}$ is *trim* if each state is accessible and coaccessible. A word $c \in A^+$ is a *cycle* (resp. *simple cycle*) in $\mathcal{A}$ if $q \in Q$ exists such that $\delta(q, c) = q$ and the internal states crossed by the transition are different from q (resp. each other and w.r.t. q). Given a regular language L and a finite state automaton $\mathcal{A}$ recognizing L, we inductively define the *fingerprint* $F_{n_c}(c)$ of a cycle c in $\mathcal{A}$ and the class of *fingerprint closed languages* L w.r.t. $\mathcal{A}$.

Definition 3 (Fingerprint of a cycle). *Let $\mathcal{A}$ be a finite state automaton recognizing a regular language L, i.e. $L = L(\mathcal{A})$. The set $\mathcal{C}(L) = \bigcup_{c\ cycle} F_{n_c}(c)$ of the fingerprints $F_{n_c}(c)$ for any cycle c in $\mathcal{A}$ is defined inductively as follows:*

- *if c is simple, then $F_{n_c}(c) = \{c^{n_c}\}$, for minimal $n_c > 0$ s.t. $c^{n_c} \in L$, otherwise $F_{n_c}(c) = \emptyset$;*
- *if c is not simple, i.e. $c = u_1c_1^{p_1}u_2c_2^{p_2} \cdots u_kc_k^{p_k}u_{k+1}$, where u_i are labels of transitions in which no state is crossed twice, $u_1, u_{k+1} \neq 1$, c_i are cycles, $p_i \geq 1$, $F_{n_{c_1}}(c_1), \cdots, F_{n_{c_k}}(c_k)$ are nonempty fingerprints of $c_1, \ldots, c_k$, then $F_{n_c}(c) = \{(u_1x_1^{t_1}u_2x_2^{t_2} \cdots u_kx_k^{t_k}u_{k+1})^{n_c} \mid x_i \text{ s.t. } x_i^{n_{c_i}} \in F_{n_{c_i}}(c_i),\ 0 \leq t_i \leq n_{c_i},\ i = 1, \ldots, k\}$, for minimal $n_c > 0$ such that $F_{n_c}(c) \subseteq L$, otherwise $F_{n_c}(c) = \emptyset$.*

Let c be a not simple cycle. Observe that $F_{n_c}(c) = \emptyset$ if either there exists j s.t. $F_{n_{c_j}}(c_j) = \emptyset$, or there exist $x_i^{n_{c_i}} \in F_{n_{c_i}}(c_i)$, $0 \leq t_i \leq n_{c_i}$, $i = 1, \ldots, k$ s.t. $(u_1x_1^{t_1}u_2x_2^{t_2} \cdots u_kx_k^{t_k}u_{k+1})^* \cap L = \emptyset$.

For a given finite state automaton $\mathcal{A}$, we say that $L = L(\mathcal{A})$ is a *fingerprint closed language* w.r.t. $\mathcal{A}$, whenever $F_{n_c}(c) \neq \emptyset$, for any cycle c in $\mathcal{A}$. Loosely speaking, if L is a fingerprint closed language we are sure that for any label of closed path we have a finite crossing of this label in L.

Thus we can prove our main result.

Theorem 1. *Let X^* be a fingerprint closed star language w.r.t. a trim deterministic automaton $\mathcal{A}$ recognizing X^*. Then $Cir(X^*) \in C(Fin, Fin)$.*

Sketch of the proof. Let us consider $SC_{PA} = (A, I, R)$ where I and R are finite sets defined by means of $\mathcal{C}(L)$. Precisely, we define I_1 as the set containing $\mathcal{C}(L)$, along with the labels of the successful paths in $\mathcal{A}$ in which each cycle c is crossed at most n_c times: $I_1 = \{w \in A^* \mid \delta(i, w) \in F$ and $\forall x, z \in A^*, c \in A^+$, $q \in Q$ if $w = x\alpha^k z$ with $\delta(i, x) = q$, $\delta(q, c) = q$, α s.t. $\alpha^{n_c} \in F_{n_c}(c)$, then $k \le n_c\} \cup \mathcal{C}(L)$. Let $I = Cir(I_1)$. Note that I_1 is finite and $I = Cir(I_1) \subseteq Cir(X^*)$. Furthermore, we set $R = \{1|1\$1|\alpha; \alpha \in F_{n_c}(c), F_{n_c}(c) \subseteq \mathcal{C}(L), c\ cycle\}$.

Let us denote $C = C(SC_{PA})$ and let us prove that $C = Cir(X^*)$.

We will firstly show that $Cir(X^*) \subseteq C$. Let $\sim y \in Cir(X^*)$. By induction on $|y|$, we prove that $\sim y \in C$, i.e. $\sim y$ is generated by the splicing system given above. By definition, y is the label of a successful path: if $\sim y$ does not contain cyles or each cycle c is crossed at most n_c times (for n_c the one given in the definition of I_1), then, by construction of I, $\sim y \in I$, and so $\sim y \in C$ (by definition). Otherwise, each element $y \in \sim y$ is a successful path in which a cycle c is crossed more than n_c times (for n_c the one given in the definition of I_1). Precisely, $y = xc^r z$ with $x, z \in A^*$, $\delta(i, x) = q$, $\delta(q, c) = q$, $\delta(q, z) \in F$ and $r > n_c$, i.e., $r - n_c > 0$. Choose x with $|x|$ maximal w.r.t. this condition. Since y is a successful path, then $y' = xc^{r-n_c}z$ is a successful path and, by induction hypothesis, $\sim y' \in C$.

We claim that $c^{n_c} \in \mathcal{C}(L)$, where we refer to the occurence of the word c as the cycle c in the successful path y. This allows us to prove that $\sim y \in Cir(X^*)$. Indeed, if $c^{n_c} \in \mathcal{C}(L)$, then $\sim c^{n_c} \in I$. By splicing definition, $\sim c^{n_c} \in C$. Consider the linearization zxc^r of $\sim y$. Set $zxc^r = u_2hu_1u_4ku_3$, with $h = zxc^{r-n_c}$, $u_1 = u_2 = k = u_3 = 1$, $u_4 = c^{n_c}$. Clearly, $u_1|u_2\$u_3|u_4 = 1|1\$1|c^{n_c} \in R$. Moreover, $\sim u_2hu_1 = \sim zxc^{r-n_c} \in C$ and $\sim u_4ku_3 = \sim c^{n_c} \in C$: by splicing definition $\sim zxc^r = \sim y \in C$.

We now prove the above stated claim: $c^{n_c} \in \mathcal{C}(L)$. Otherwise, by Definition 3, c is not a simple cycle in y and c^{n_c} is not a fingerprint of the cycle c in y. Then $c = u(c')^t v$, where c' is a cycle in y, $t > n_{c'}$ and $u, v \neq 1$. Thus $y = x(u(c')^t v)^r z$ with $|xu| > |x|$, against the maximality of $|x|$.

Vice versa, $C \subseteq Cir(X^*)$ follows by using Proposition 3, since $I \subseteq Cir(X^*)$, i.e. $C = Cir(X^*)$. $\square$

Example 3. Star languages exist which do not satisfy the hypothesis contained in Theorem 1. For instance, consider $L = A^* \setminus a^* = (a^*ba^*)^*$ over a two-letter alphabet $A = \{a, b\}$. It is clear that L is a star language. Furthermore, a trim deterministic automaton $\mathcal{A}$ exists such that a is a cycle in $\mathcal{A}$ and $a^* \cap L = \emptyset$. However, L is not CPA generated. Indeed, by contradiction suppose that $C = Cir(L) = C(SC_{PA})$ with $SC_{PA} = (A, I, R)$. Since $\sim(a^*b) \subseteq C$, $n \in \mathbb{N}$ exists such that $w = \sim (a^n b) \in C \setminus I$. Thus, $\sim hu_1u_2 \in C$, $\sim ku_3u_4 \in C$ also exist such that $w = \sim u_2hu_1u_4ku_3$. Since we have only one occurrence of b's in w, we get $\sim hu_1u_2 \in \sim (a^*)$ or $\sim ku_3u_4 \in \sim (a^*)$, a contradiction.

As far as we know, the structure of the regular languages which are closed under the conjugacy relation is unknown, even if we restrict ourselves to the simple case of languages which are the Kleene closure of a regular language. Nevertheless, this structure has been completely described in [3, 19] when X^* is a free monoid. The necessary definitions for recalling this result can be found in [2]. We briefly report them below.

An algebraic description of some subclasses of the class of the regular languages $L \subseteq A^*$ has been given by means of the *syntactic monoid* $\mathcal{M}(L)$ of L. This is the quotient of A^* w.r.t. the syntactic congruence $\equiv_L$, defined as follows: $w \equiv_L w'$, with $w, w' \in A^*$, if and only if $xwy \in L \Leftrightarrow xw'y \in L$, for all $x, y \in A^*$. If L is regular, a well known result states that $\mathcal{M}(L)$ is a finite monoid.

Historically, this notion arose in the context of variable-length codes. We recall that X^* is a free monoid if and only if X is a *code*, i.e. for all $x_1, \ldots, x_n$, $x'_1, \ldots, x'_m \in X$, we have

$$x_1 \cdots x_n = x'_1 \cdots x'_m \Leftrightarrow n = m \text{ and } \forall i \in \{1, \ldots, n\}\ x_i = x'_i$$

A remarkable class of codes is the class of *biprefix codes* C. $C \subseteq A^*$ is biprefix if no word in C is a proper prefix or a proper suffix of another element in C, i.e. $C \cap CA^+ = C \cap A^+C = \emptyset$. For instance, uniform codes $A^d, d \geq 1$, are biprefix codes.

Here we report two known results on codes. Theorem 2 has been used to prove Proposition 4. Theorem 3 completely describes finite group codes.

Theorem 2. *[3, 19] Let $X \subseteq A^+$ be a code. Then X^* is closed under conjugacy relation if and only if $\mathcal{M}(X^*)$ is a group.*

We explicitly note that, under the hypothesis that X is a regular language, $\mathcal{M}(X^*)$ is a group if and only if X is a *group code*, i.e. a group G and a surjective morphism $\phi: A^* \to G$ exist such that $X^* = \phi^{-1}(H)$, H being a subgroup of G. Group codes are biprefix codes.

Theorem 3. *[3, 19] Let $X \subseteq A^+$ be a group code. X is finite if and only if $X = A^d$, $d \geq 1$.*

Example 4. [2] Group codes exist which are not finite. Let $A = \{a, b\}$ and $M_a = (b^*(ab^*a)^*)^*$ be the set of words with an even number of a. M_a is generated by $X_a = b \cup ab^*a$, which is a (biprefix) code. Then $M_a = X_a^*$ is a free monoid. Moreover, X_a is a regular group code, being $X_a^* = \phi^{-1}(0)$, where $\phi: A^* \to \mathbb{Z}/2\mathbb{Z}$ is the morphism given by $\phi(a) = 1, \phi(b) = 0$ and with $\{0\}$ a subgroup of $\mathbb{Z}/2\mathbb{Z}$ (w.r.t. $+$). Observe that for the submonoid $X^* = \{w \in A^* \mid |w|_a = |w|_b\}$ composed of the words in A^* having as many a's as b's, we have $X^* = \phi^{-1}(0)$ where $\phi: A^* \to \mathbb{Z}$ is such that $\phi(a) = 1, \phi(b) = -1$, and with $\{0\}$ a subgroup of $\mathbb{Z}$ (w.r.t. $+$). X is called a *Dyck code* over A and it is a (non regular) group code.

Example 2 (continued). Consider $L_1 = \{w \in A^* \mid \exists h, k \in \mathbb{N}\ |w|_a = 2k, |w|_b = 2h\}$. Obviously, $L_1 = M_a \cap M_b$. Thus L_1 is a free monoid which is closed under conjugacy, since it is the intersection of two free monoids both closed under conjugacy. We have already observed that L_1 is a regular language. Furthermore, since L_1 is a free monoid, it follows that $L_1 = X^*$, where X is a code [2]. Moreover, it is well known that when X is a code, X^* is regular if and only if

X is regular [2]. Since X is a code which is regular and $X^* = L_1$ is closed under conjugacy relation, by Theorem 2 we have that X is a group code. Consequently, $L_1 = X^*$ is a star language and $Cir(L_1) = Cir(X^*)$ is CPA generated, thanks to Corollary 1. A splicing system generating $Cir(L_1)$ is $SC_{PA} = (A, I, R)$ with $I = \{{}^\sim aa, {}^\sim bb, {}^\sim aabb, {}^\sim abab\} \cup 1$ and $R = \{a|1\$1|a, b|1\$1|b, a|1\$1|b\}$.

Remark 2. We can observe that each regular language closed under conjugacy is contained in a star language generated by a regular group code which is minimal w.r.t. inclusion. Indeed, let L be a regular language which is closed under conjugacy. A^* is a free monoid containing L, which is regular and closed under conjugacy. Thus, the intersection of all regular free monoids containing L which are closed under conjugacy is a free monoid containing L, closed under conjugacy (thus generated by a regular group code) and minimal w.r.t. inclusion.

Finally, Proposition 4 shows that the hypothesis contained in Theorem 1 is satisfied by regular group codes.

Proposition 4. *Let X be a regular group code and let $\mathcal{A}$ be a trim deterministic finite state automaton recognizing X^*. For any cycle c in $\mathcal{A}$, $c \in X^*$.*

Proof. Let X be a regular group code and let $\mathcal{A}$ be a trim deterministic finite state automaton recognizing X^*. Let c be a cycle in $\mathcal{A}$. Then, $x, y \in A^*$ exist such that $xcy \in X^*$ and $xy \in X^*$. Moreover, $yxc, yx \in X^*$, being X^* closed under conjugacy, and $c \in X^*$, since X^* is biunitary. We recall that X is a biprefix code if and only if X^* is biunitary, i.e. X^* is both left and right unitary, a submonoid X^* being right (resp. left) unitary if, for all $u, v \in A^*$, $u, uv \in X^*$ (resp. $u, vu \in X^*$) implies $v \in X^*$ [2]. □

Corollary 1. *For each regular group code X, $Cir(X^*)$ is a circular splicing language.*

As we have already said, we prove that all star languages X^* with X a finite set have their circularization in $C(Fin, Fin)$. In the proof of this result, we also give a splicing system generating $Cir(X^*)$ whose construction is simpler than the one given in the proof of Theorem 1.

Proposition 5. *For each finite set X, $Cir(X^*)$ is a circular splicing language.*

Proof. It is clear that the splicing system $SC_{PA} = (A, I, R)$ generates $Cir(X^*)$, when we choose $I = \{{}^\sim x \mid x \in X\} \cup 1$ and $R = \{x_i|1\$1|x_j \mid x_i, x_j \in X\}$.

4 Conclusions and future work

As we have already said in Section 1, additional hypotheses can be added to the definitions of circular splicing given by Paun and Pixton. In this section we report the result already obtained which uses the hypotheses recalled below.

Hypothesis 1. R is a symmetric scheme, i.e. for any rule $r = u_1|u_2\$u_3|u_4$ (resp. $r = (\alpha, \alpha'; \beta)$) in a splicing system SC_{PA} (resp. SC_{PI}) there is the rule $\overline{r} = u_3|u_4\$u_1|u_2$ (resp. $\overline{r} = (\alpha', \alpha; \beta')$).

Remark 3. Any set R of rules in a splicing system SC_{PI} is implicitly supposed to be symmetric.

Hypothesis 2. R is a reflexive scheme, i.e. for any rule $u_1|u_2\$u_3|u_4$ (resp. $(\alpha, \alpha'; \beta)$) in a splicing system SC_{PA} (resp. SC_{PI}) there is the rule $u_1|u_2\$u_1|u_2$ (resp. $(\alpha, \alpha; \alpha)$).

Hypothesis 3. Self-splicing. Self-splicing is defined in a splicing system SC_{PA} (resp. SC_{PI}) producing $\sim u_4 x u_1$ and $\sim u_2 y u_3$ from $\sim x u_1 u_2 y u_3 u_4$ and the rule $u_1|u_2\$u_3|u_4$ (resp. $\sim\beta\epsilon', \sim\beta'\epsilon$ starting from $\sim\alpha\epsilon\alpha'\epsilon'$ and $r, \overline{r}$ as above).

Problem 3 generalizes Problem 2 when we take into account the additional hypotheses given above.

Problem 3. Can we characterize $F \cap C(Fin, Fin)$ (resp. $F \cap C(Reg, Fin)$) for each class F of languages in the Chomsky hierarchy, for every definition of $C(Fin, Fin)$ (resp. $C(Reg, Fin)$) and for every possible combination of the three hypotheses above?

Below we report a known result which uses Pixton's systems with hypotheses $1, 2$ and which generalizes a similar theorem proved for linear splicing [18].

Theorem 4. *[18] Let $SC_{PI} = (A, I, R)$ be a circular splicing system with I a circular regular language and R reflexive and symmetric. Then $C = C(SC_{PI})$ is regular.*

We conclude this section with some remarks related to Problem 2, i.e. the characterization of the circular regular languages generated by finite circular splicing, and with some problems which are still open.

First of all, we point out that rational languages exist which are not star languages and which are the full linearization of regular circular splicing languages. Indeed, $\sim(ab)^*$ has $(ab)^* \cup (ba)^*$ as full linearization, and this is not a star language, but it is generated by $SC_{PA} = (A, I, R)$, where $A = \{a, b\}, I = \{\sim ab\}, R = \{ab|1\$1|ab\}$.

Secondly, note that the class of regular languages closed under conjugacy relation is closed under $\cup$. However, while $Cir((A^2)^*)$, $Cir((A^3)^*) \in C(Fin, Fin)$, as shown in Section 3, we have $Cir((A^2)^*) \cup Cir((A^3)^*) \notin C(Fin, Fin)$. This means that $Reg^\sim \cap C(Fin, Fin)$ is not closed w.r.t. $\cup$. One could ask for additional hypotheses to be added so that the union of circular regular splicing languages will still be a circular splicing language. This research line could be a promising direction towards a complete characterization of the class of circular regular languages that are generated by finite circular splicing. In this direction, further investigations are contained in a work in progress [5].

Another interesting problem could be the investigation of the *splicing subsystem* of a given circular splicing system. For example, $Cir((A^2)^*)$, for $A = \{a, b\}$, is CPA generated by $I = \sim A^2$ and $R' = \{w_1|1\$1|w_2 \mid w_1, w_2 \in A^2\}$ (Proposition 5). Thus, by extracting $R_1 = \{aa|1\$1|aa\}$, we obtain $\sim(aa)^*$, with $R_2 = \{bb|1\$1|bb\}$ we generate $\sim(bb)^*$ and with $R_3 = \{ab|1\$1|ab\}$ we obtain $\sim(ab)^*$. It could be interesting to prove or to disprove that circular languages can be obtained by extraction as described before. Let us take $L_1 = \{w \in A^* \mid \exists h, k \in \mathbb{N} \ |w|_a = 2k, |w|_b = 2h\}$ as in Example 2. The circular language $Cir(L_1)$ cannot be generated by choosing a subset of $R = \{a|1\$1|a, b|1\$1|b, a|1\$1|b\}$. This investigation is related to the notion of a *minimal splicing system* introduced in [17], where it was called *descriptional complexity*. It goes without saying that this notion is the counterpart of the minimal automaton for regular languages. Here, the minimality of the system could be referred to the cardinality of R or to the length of the rules in R.

Acknowledgments

Many thanks to J. Berstel and A. Restivo for suggesting us references [3] and [19] during the conference Words99 [6]. The authors also wish thank A. Bertoni for useful discussions.

References

1. L. M. Adleman, *Molecular computation of solutions to combinatorial problems*, Science, 226, (1994), 1021 – 1024.
2. J. Berstel, D. Perrin, *Theory of codes*, Academic Press, New York, (1985).
3. J. Berstel, A. Restivo, *Codes et sousmonoides fermes par conjugaison*, Sem. LITP, 81 – 45, (1981), 10 pages.
4. P. Bonizzoni, R. Zizza, *Deciding whether a regular language is a splicing language*, *submitted*, (1999).
5. P. Bonizzoni, C. De Felice, G. Mauri, R. Zizza, *Circular splicing and regular languages*, *manuscript*, (2000).
6. P. Bonizzoni, C. De Felice, G. Mauri, R. Zizza, *Linear and circular splicing*, WORDS99, (1999).
7. P. Bonizzoni, C. Ferretti, G. Mauri, R. Zizza, *Separating some splicing models*, *Grammar Systems 2000*, (2000).
8. K. Culik, T. Harju, *Splicing semigroups of dominoes and DNA*, Discrete Appl. Math., 31, (1991), 261 – 277.
9. R. W. Gatterdam, *Algorithms for splicing systems*, SIAM Journal of Computing, 21 : 3, (1992), 507 – 520.
10. D. Giammarresi, A. Restivo, *Two-dimensional Languages*, in: Handbook of Formal Languages, G. Rozenberg & A. Salomaa, Eds., Springer Verlag, Vol. 3, (1996), 215 – 267.
11. T. Head, *Formal Language Theory and DNA: an analysis of the generative capacity of specific recombinant behaviors*, Bull. Math. Biol., 49, No. 5, (1987), 737 – 759.
12. T. Head, Gh. Paun, D. Pixton, *Language theory and molecular genetics: generative mechanisms suggested by DNA recombination*, in: Handbook of Formal Languages, G. Rozenberg & A. Salomaa, Eds., Springer Verlag, Vol. 2, (1996), 295 – 360.
13. J.E. Hopcroft, J.D. Ullman, *Introduction to Automata Theory, Languages, and Computing*, Addison-Wesley, Reading, Mass. (1979).
14. S.M. Kim, *Computational modeling for genetic splicing systems*, SIAM Journal of Computing, 26, (1997), 1284 – 1309.

15. M. Lothaire, *Combinatorics on Words*, Encyclopedia of Math. and its Appl., Addison Wesley Publishing Company (1983).
16. G. Paun, *On the splicing operation*, Discrete Applied Math., 70, (1996), 57 – 79.
17. G. Paun, G. Rozenberg, A. Salomaa, *DNA computing, New Computing Paradigms*, Springer-Verlag, (1998).
18. D. Pixton, *Regularity of splicing languages*, Discrete Applied Math. 69, (1996), 101 – 124.
19. C. Reis, G. Thierren, *Reflective star languages and codes*, Information and Control, 42, (1979), 1 – 9.
20. R. Siromoney, K.G. Subramanian, A. Dare, *Circular DNA and Splicing Systems*, Proc. of ICPIA, LNCS 654, Springer-Verlag, (1992), 260 – 273.