Linear 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
Viale Sarca 202, 20126 Milano - Italy
{bonizzon,mauri,rosalba}@dsi.unimi.it
[2] Dipartimento di Informatica ed Applicazioni, Università di Salerno,
84081 Baronissi (SA), Italy
defelice@dia.unisa.it

1 Introduction

The notion of *splicing systems* was first introduced in [7], where T. Head translated 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 to use molecules to perform computations. This idea was subsequently developed in [1]. 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 characterized to model specific recombinant behaviours. In short, a restriction enzyme is able to recognize itself in a DNA molecule, to cut this molecule in a specific position, providing two segments of DNA, while a ligase enzyme binds together pairs of suitable molecules. In order to formulate this cut and paste phenomenon in formal language theory, an initial set of strings (the initial set of DNA molecules) and 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 purposely ignore other molecular considerations and basic details (see [9]). Furthermore, there is no hypothesis on the number of copies of the initial set of molecules and, as DNA occurs in both linear and circular form, we separate these cases, but provide interesting links between them. Classical results in formal language theory and combinatorial tools have been of great use for reaching the results given in this paper. It is worth mentioning that, due to the bidimensional nature of DNA, very recently a link has been given between splicing systems and picture languages, which were completely surveyed in [6] ([2]).

Let us briefly sketch the contents of this paper. Section 2 is devoted to the linear case. Three definitions of splicing operation are known in the literature.

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

They are collected in Section 2.1 with the problem, still open, of comparing the computational power of the three corresponding splicing systems. In Section 2.2 we will present the main result contained in this paper. Namely, we will limit ourself to the definition of splicing given by Paun. Corresponding to this operation, the computational power of splicing systems S_{PA} has been investigated in [9]. This computational power depends on the initial set of words and on the initial set of splicing rules. In particular, if this initial set and the set of splicing rules are both finite, each language which is generated by those splicing systems S_{PA} is regular. Theorem 1 states that an algorithm exists for deciding whether a regular language is generated by such a splicing system S_{PA}.

In Section 3 we gathered all the results on the circular splicing. Precisely, three corresponding definitions of circular splicing operations are given in Section 3.1. In the same section, we also explicitly observe that the computational power of the corresponding splicing systems is already known to be not the same. The circular framework has not been as thoroughly investigated as the linear case. Even if we restrict ourself to one definition, as observed in Section 3.2, several properties could or could not be added. In Section 3.2, we give a conjecture on the structure of the circular regular languages which are generated by a splicing system SC_{PA} (finite scheme, without self splicing). We also give a conjecture on the structure of the circular context-free languages which are generated by a splicing system SC_{PA} (finite scheme, without self splicing).

2 Linear splicing

2.1 Three definitions

In this section, we discuss the linear form of DNA molecules as finite words over a finite alphabet. The splicing operation, informally described above, has been formalized into three definitions.

`Head's definition [7].` A *splicing system* $S_H = (V, I, B, C)$ consists of a finite alphabet V, a finite set $I \subseteq V^*$ of initial strings, finite sets B, C of triples (c, x, d) with $c, x, d \in V^*$ (*patterns*). We also say that cxd is a *site* and x is a *crossing*. Given two words $ucxdv, pexfq \in V^*$ and two patterns $(c, x, d), (e, x, f)$ both in B or in C, the splicing operation produces $ucxfq, pexdv$. The *splicing language* generated by S_H is $L = L(S_H)$, which is the smallest language obtained by using the splicing operation starting with I. In the following we also refer to $L = L(S_H)$ as an $H-$generated language.

Example 1. [7] Let $V = \{a, b\}$, $I = \{ababa\}, B = \{(a, b, a)\}$ and $C = \emptyset$. $S_H = (V, I, B, C)$ generates $L = (ab)^*aba$.

Example 2. Head [7] explicitly linked the splicing operation with the concept of *constant* introduced in [13]. In the same paper, he proved that a significant subclass of splicing languages (persistent splicing languages) coincide with strictly locally testable languages and may be generated by $S = (V, I, V^P, V^P)$, for $P \geq 1$ (uniform splicing systems). To do this, he used a result in [5]. It is worth observing that $H(Fin, Fin) \not\subseteq LT$ and $LT \not\subseteq H(Fin, Fin)$ where LT denotes the class of locally testable languages, i.e. $H(Fin, Fin)$ is not comparable (with respect to the inclusion) with LT [2].

`Paun's definition [9].` Given a finite alphabet V and two special symbols $|, \$$ not in V, a *splicing rule* (over V) is $r = u_1|u_2\$u_3|u_4$, ($u_i \in V^*, i = 1, 2, 3, 4$). For such a rule r and $x, y, z \in V^*$, we write $(x, y)\vdash_r z$ if and only if $x = x_1u_1u_2x_2, y = y_1u_3u_4y_2$, $z = x_1u_1u_4y_2$, $(x_1, x_2, y_1, y_2 \in V^*)$. A *splicing system* $S_{PA} = (V, I, R)$ consists of a set R of rules and an initial language I. The *splicing language* $L = L(S_{PA})$ is obtained by iteration of the rules in R applied to I. We will say that L is PA-generated.

Example 3. [3] The language $L = (aa)^*b$ is $PA-$generated by the system S_{PA}, with $V = \{a, b\}$, $I = \{b, a^2b, a^4b\}$, $R = \{a^2|b\$a^2|a^2b, a^2|b\$\epsilon|a^4b\}$.

`Pixton's definition [12].` Given a finite alphabet V, a *splicing rule* is a triple $r = (\alpha, \alpha'; \beta)$ that applied to $w = \epsilon\alpha\eta$ and $w' = \epsilon'\alpha'\eta'$ produces $\epsilon\beta\eta'$. A *splicing system* is $S_{PI} = (V, I, R)$ where R is a finite collection of splicing rules and I is the initial set of strings. The *splicing language* $L = L(S_{PI})$ is the set of words obtained from I by repeated splicing using R. We will say that L is $PI-$generated.

Example 4. The language $L = (aa)^*b$ is $PI-$ generated, by S_{PI} with $V = \{a, b\}$, $I = \{aab, b\}$, $R = \{(aab, aab, aaaab)\}$. Moreover, a^*b^* is $PI-$generated, by choosing $I = \{ab\}$, $R = \{(ab, ab, \beta)|\beta \in \{a^2b^2, a^2b, ab^2, \epsilon, b, a\}\}$.

Behind each definition, there is a biological motivation. Indeed, as splicing systems should capture the enzymes' behaviour on DNA, the difference between the three definitions depends on the phenomena that they want to model. Without going into detail, we observe that the enzymes' action on DNA produces blunt ends and staggered cleavages and since in the recombinant processes of two segments the overhangs must match under some conditions, Head divides them into two classes (this explains why pair B, C appears in his definition). On the contrary, Paun and Pixton work more abstractly on single strands of DNA.

It is natural to compare the computational power of each type of splicing operation which is the same as solving the problem below.

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

We believe that the power of the three definitions is not the same.

2.2 Known and new results

Another kind of question can be considered if we restrict ourselves to the investigation of only one of the three definitions above. Indeed in [9] a description of the computational power of splicing systems S_{PA} is given. This computational power obviously depends on I, R. Precisely, let us denote $H(F_1, F_2)$ the family of languages generated by S_{PA} with $I \in F_1, R \in F_2$, F_1, F_2 being classes of languages in the Chomsky hierarchy. Thus, either $H(F_1, F_2)$ is a specific class of languages in the above hierarchy, or it is strictly intermediate between two of them (see [9] for further details). In particular, $Fin \subseteq H(Fin, Fin) \subseteq Reg$, Reg (resp. Fin) being the class of regular (resp. finite) languages.

Example 5. We can see that $(aa)^* \notin H(Fin, Fin)$ (see [9,3]), but $(aa)^*b \in H(Fin, Fin)$ (see Example 3).

The above example shows that $H(Fin, Fin)$ is strictly contained in Reg and strictly contains Fin. Thus, a natural problem is the characterization of the regular languages which belong to $H(Fin, Fin)$. In [3] this problem is solved, capturing the behaviour of $H(Fin, Fin)$ and providing the first characterization of the class of languages generated by these systems in terms of properties of finite state automata.

Theorem 1. *[3] It is decidable whether a regular language L is generated by a splicing system S_{PA} with a finite initial language and a finite set of rules. Moreover there is an algorithm to construct the splicing system associated with the minimum finite state automaton $\mathcal{A}$ for the given splicing language.*

Since we consider a splicing system with finite rules and a finite initial language, we need to consider a number of conditions (on the state graph $G_{\mathcal{A}}$, associated to $\mathcal{A}$) which is finite, but sufficient to generate all the sequences accepted by $\mathcal{A}$. In order to do so, three properties that may be tested on $G_{\mathcal{A}}$ are given. These properties are necessary and sufficient for $G_{\mathcal{A}}$ to be associated with a regular language generated by splicing. Finally, an algorithm is given which starts from a finite initial set of "canonical" crossings of successful paths and produces the set of rules in order to generate L.

3 Circular splicing

As already mentioned in Section 1, we now consider the splicing operation on circular strings. 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. Alongside biological motivations, circular strings have already been treated in formal language theory. Indeed, a circular word $\sim w$ is nothing but the equivalence class of $w \in V^*$, V being a finite alphabet, with respect to the *conjugacy* relation $\sim$ defined by $xy \sim yx$, for $x, y \in V^*$ (see for example [11]). $V^\sim$ is the set of all such circular strings over V, i.e. the quotient of V^* with respect to $\sim$.

Using a suitable prefix code, the elements in $V^\sim$ can be mapped in a particular monoid (see [14] for further details).

Given $L \in V^*$, $Cir(L)$ 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 $C \subseteq V^{\sim}$, is called a linearization of C. The set of all strings in V^* corresponding to elements of C is the *full linearization* of C, written $Lin(C)$. Moreover, given a family of languages FA, $FA^{\sim}$ consists of all those circular languages C which have some linearization in FA. In particular, $L \in Reg^{\sim}$ if and only if its full linearization is regular [10].

3.1 Three definitions

The splicing operation in the circular case deals with biological phenomena which are different to the ones modelled in the linear case, but analogously there are three definitions.

`Head's definition [14].` Let $SC_H = (V, I, T, P)$ be a *splicing system*, where V is a finite alphabet, $I \subseteq V^{\sim}$ is the initial set of circular strings, $T \subseteq V^* \times V^* \times V^*$ and P is a binary relation on T, such that if $p, x, q, u, y, v \in V^*$, (p, x, q),$(u, y, v) \in T$ and $(u, x, v)P(u, y, v) \Rightarrow x = y$. Then, given SC_H and $^{\sim}hpxq,^{\sim} wuxv \in V^{\sim}$ with $(p, x, q)P(u, x, v)$, the splicing produces $^{\sim}hpxvwuxq$. The *splicing language* $C = C(SC_H)$ generated by SC_H is obtained by iteration of the splicing applied to I. We will say that C is $CH-$generated.

Example 5 (continued). [14] We have observed that $(aa)^*$ is not a linear splicing language. The corresponding circular language $\{^{\sim}(aa)^n | n \geq 0\}$ can be generated by $SC_H = (V, I, T, P)$ with $V = \{a\}$, $I = \{^{\sim}\epsilon,^{\sim} aa\}$, $T = \{(\epsilon, a, \epsilon)\}$, and $(\epsilon, a, \epsilon)P(\epsilon, a, \epsilon)$.

`Paun's definition [9].` Given a splicing rule $r = u_1|u_2\$u_3|u_4$ applied to two circular strings $^{\sim}xu_1u_2,^{\sim} yu_3u_4$, the rule cuts and linearizes the two strings obtaining u_2xu_1 and u_4yu_3, and pastes and circularizes them obtaining $^{\sim}u_2xu_1u_4yu_3$. A *splicing system* is $SC_{PA} = (V, I, R)$ where V is a finite alphabet, $I \subseteq V^{\sim}$ is the initial circular language, R is the set of rules. The *splicing language* $C = C(SC_{PA})$ generated by SC_{PA} is obtained by iteration of the splicing applied to the initial language using R. We will say that C is $CPA-$generated.

Example 5 (continued). [4] We can observe that $\{^{\sim}(aa)^n | n \geq 0\}$ can be $CPA-$generated, by choosing $V = \{a\}$, $I = \{^{\sim}aa\}$, $R = \{aa|\epsilon\$\epsilon|aa\}$.

`Pixton's definition [12].` Given $r = (\alpha, \alpha'; \beta)$ and $\overline{r} = (\alpha', \alpha; \beta')$ applied to two circular strings $^{\sim}\alpha\epsilon,^{\sim} \alpha'\epsilon'$, the splicing produces $^{\sim}\beta\epsilon'\beta'\epsilon$. A *splicing system* is $SC_{PI} = (V, I, R)$ where V is a finite alphabet, $I \subseteq V^{\sim}$ is the initial circular language, R is the set of rules. The *splicing language* $C = C(SC_{PI})$ generated by SC_{PI} is obtained by iteration of the splicing applied to the initial language using R. We will say that C is $CPI-$generated.

Remark 1. It goes without saying that the set of rules is finite in each definition. Moreover the investigation about circular splicing systems is usually limited to ones with a finite or regular initial language.

Remark 2. [4] It is easy to prove the following statement. Let $L = Lin(C)$ be the full linearization of a language C and suppose that L satisfies the following condition

$$\forall x, y \in L, \forall x_1 \in \sim x, \forall y_1 \in \sim y : x_1 y_1 \notin L$$

If L is not a finite set then $Cir(L)$ is neither CH nor CPA generated.

Example 5 (continued). We have observed that $(aa)^* \notin H(Fin, Fin)$, but its circularization is $CH-$generated. On the contrary, the regular language $(aa)^*b$ is in $H(Fin, Fin)$ and its circularization C is regular (by definition). 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 [12]. This full linearization is not CH or CPA generated (see Remark 2). On the contrary, it is $CPI-$generated when we choose, for example, $V = \{a, b\}, I = \{\sim b, \sim a^2b, \sim a^4b\}, R = \{(a^2, a^2b; a^2), (a^2b, a^2; \epsilon)\}$ [4].

Looking at Example 5, some observations can be given. Firstly, we note the difference between linear and circular splicing. Secondly, we can reformulate Problem 1 for the circular case. We already know that the three definitions of splicing have a different computational power.

3.2 Known results and new conjectures

As we have already said, the investigation of circular splicing systems is not so developed as in the linear case. In particular, the analysis of their generative capacity remains an open problem even in this more restricted situation (finite set of rules).

In the papers which have already been published on this topic, additional hypotheses are often considered for Paun's or Pixton's definitions. Precisely:

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).

Remark 4. We must note that the original definition of circular splicing language given by Paun in [9] includes self-splicing to define a splicing language generated by a SC_{PA} system. Here, to make the three definitions uniform, we have preferred to delete this hypothesis from Paun's definition.

Let us denote $C(F_1, F_2)$ the family of languages generated by circular splicing using one of the three definitions given in Section 3.1, with $I \in F_1$ and $R \in F_2$, for F_1, F_2 classes of languages in the Chomsky hierarchy. The following problem can be posed.

Problem 2. Can we characterize $F \cap C(Fin, Fin)$ (resp. $F \cap C(Reg, Fin)$) for each class F of languages in the Chomsky hierarchy by choosing whether or not to add any possible combination of the three hypotheses above?

Below you can find a known result regarding circular splicing. This known result concerns Pixton's systems with hypotheses 1, 2.

Theorem 2. *[12] Let $SC_{PI} = (V, I, R)$ be a circular splicing system with I a regular circular language and R reflexive and symmetric. Then $C = C(SC_{PI})$ is regular.*

Furthermore, 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). Indeed, $\sim a^n b^n$ 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 [14]. On the other hand, there are regular circular languages that cannot be generated by using CH or CPA splicing systems (with finite initial circular language), e.g. $\{\sim a^n b | n \geq 1\}$, $\{\sim (aa)^n b | n \geq 0\}$ (see [14], Remark 2). Thus, we suggest the following conjectures.

Conjecture 1. [4] Let C be a circular regular language and $L = Lin(C)$ be its full linearization. Then C is CH or CPA generated (finite scheme, without self-splicing) if and only if $L = X^* \cup F$ with X, F finite sets and L closed under the conjugacy relation.

Conjecture 2. [4] Let C be a circular context-free language and $L = Lin(C)$ be its full linearization. Then C is CH or CPA generated (finite scheme, without self-splicing) if and only if $L = F \cup \bigcup_{i=1}^{k} \{u_i^n v_i^n \mid n \geq 0\}$, with F a finite set and L closed under the conjugacy relation.

So far, we have not yet discovered whether $C(Fin, Fin)$ contains any context-sensitive or recursive enumerable language which is not context-free.

References

1. L. Adleman, *Molecular Computation of Solutions of Combinatorial Problems*, Science, Vol. 266, 11 November 1994, 1021 – 1024.
2. P. Bonizzoni, *private communication.*
3. P. Bonizzoni, R. Zizza, *Deciding whether a regular language is a splicing language*, (1999), *in preparation*, 15 pages.
4. C. De Felice, G. Mauri, R. Zizza, *On the circular splicing*, *manuscript*, (1999).

5. A. De Luca, A. Restivo, *A characterization of strictly locally testable languages and its application to subsemigroups*, Information and Control, 44 (1980), 300 − 319.
6. D. Giammarresi, A. Restivo, *Two-dimensional Languages*, in: Handbook of Formal Languages, G. Rozenberg & A. Salomaa, Eds., Springer Verlag, Vol. 3, (1996), 215 − 267.
7. T. Head, *Formal Language Theory and DNA: an analysis of the generative capacity of specific recombinant behaviors*, Bull. Math. Biol., Vol. 49, No. 5, (1987), 737 − 759.
8. T. Head, *Splicing representations of strictly locally testable languages*, Discrete Applied Math., to appear, (1998).
9. 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.
10. J.E. Hopcroft, J.D. Ullman, *Introduction to Automata Theory, Languages, and Computing*, Addison-Wesley, Reading, Mass. (1979).
11. M. Lothaire, *Combinatorics on Words*, Encyclopedia of Math. and its Appl., Addison Wesley Publishing Company (1983).
12. D. Pixton, *Regularity of splicing languages*, Discr. Applied Math. 69, (1996), 101 − 124.
13. M.-P. Schützenberger, *Sur certains operations de fermeture dans les langages rationnels*, Symposia Math., 15 (1975), 245 − 253.
14. R. Siromoney, K.G. Subramanian, V. Dare, *Circular DNA and Splicing Systems*, Proc. of ICPIA, LNCS 654, Springer-Verlag, (1992), 260 − 273.