1

A Proof of Regularity for Finite Splicing

Vincenzo Manca

Università di Verona
Dipartimento di Informatica
Ca' Vignal 2 - Strada Le Grazie 15, 37134 - Verona - Italy
vincenzo.manca@univr.it

Keywords and Phrases: String Rewriting, Formal Systems, Regular Languages, H Systems, Splicing Systems, DNA Computing, Molecular Computing.

Summary. A new proof is given that languages generated by H (nonextended) systems with finite axioms and rules are regular.

1.1 Introduction

Splicing is the basic combinatorial operation which DNA Computing [8] is based on. It was introduced in [4] as a formal representation of DNA recombinant behavior and opened new perspectives in the combinatorial analysis of strings, languages, grammars, and automata. Indeed, biochemical interpretations were found for concepts and results in formal language theory [11, 8] and Molecular Computing emerged as a new field covering these subjects, where synergy between Mathematics, Computer Science and Biology yields an exceptional stimulus for developing new theories and applications based on discrete formalizations of biological processes.

In this paper we express the combinatorial form of splicing by four cooperating combinatorial rules: two cut rules (suffix and prefix deletion), one paste rule, and one (internal) deletion rule. This natural translation of splicing allows us to prove in a new manner the regularity of (non extended) splicing with finite axioms and rules.

1.2 Preliminaries

Consider an alphabet V and two symbols $\#, \$$ not in V. A *splicing rule* over V is a string $r = u_1\#u_2\$u_3\#u_4$, where $u_1, u_2, u_3, u_4 \in V^*$. For a rule r and for any $x_1, x_2, y_1, y_2 \in V^*$ we define the (ternary) splicing relation $\Longrightarrow_r$ such that:

$$x_1u_1u_2x_2\ ,\ \ y_1u_3u_4y_2 \Longrightarrow_r x_1u_1u_4y_2.$$

In this case we say that $x_1u_1u_4y_2$ is obtained by a *splicing step*, according to the rule r, from the *left argument* $x_1u_1u_2x_2$ and the *right argument* $y_1u_3u_4y_2$. The strings u_1u_2 and u_3u_4 are respectively the left and right *splicing points* or *splicing sites* of the rule r.

An *H system*, according to [8], can be defined by a structure $\Gamma = (V, A, R)$ where V is an *alphabet*, that is, a finite set of elements called *symbols*, A is a set of strings over this alphabet, called *axioms* of the system, and R is a set of splicing rules over this alphabet.

$L_0(\Gamma)$ consists of the axioms of Γ. For $n \geq 0$, $L_{n+1}(\Gamma)$ consists of the strings generated by one splicing step from strings of $L_n(\Gamma)$, by applying all the rules in R that can be applied. The language $L(\Gamma)$ *generated* by Γ is the union of all languages $L_n(\Gamma)$ for $n \geq 0$.

If a terminal alphabet $T \subset V$ is considered, and $L(\Gamma)$ consists of the strings over T^* generated by Γ, then we obtain an *extended* H system $\Gamma = (V, T, A, R)$.

H systems are usually classified by means of two classes of languages FL_1, FL_2: a H system is of type $H(FL_1, FL_2)$ when its axioms are a language in the class FL_1 and its rules, which are strings of $(V \cup \{\#, \$\})^*$, are a language in the class FL_2; $EH(FL_1, FL_2)$ is the subtype of *extended* H systems of type $H(FL_1, FL_2)$. We identify a type $C = H(FL_1, FL_2)$ of H systems with the class of languages generated by C. Let FIN, REG, RE indicate the classes of finite, regular, and recursively enumerable languages respectively. It is known that: $H(FIN, FIN) \subset REG$, $H(REG, FIN) = REG$, $EH(FIN, FIN) = REG$, $EH(FIN, REG) = RE$. Comprehensive details can be found in [8]. We refer to [11] and [8] for definitions and notations in formal language theory.

1.3 Cut-and-Paste Splicing

A most important mathematical property of splicing is that the class of languages generated by a *finite splicing*, that is, by finite number of splicing rules, from a finite initial set of strings, is a subclass of regular languages: $H(FIN, FIN) \subset REG$ and, more generally, $H(REG, FIN) = REG$. The proof of this result has a long history. It originates in [1, 2] and was developed in [9], in terms of a complex inductive construction of a finite automaton. In [8] Pixton's proof is presented (referred to as *Regularity preserving Lemma*). More general proofs, in terms of closure properties of abstract families of languages, are given in [5, 10]. In [6] a direct proof was obtained by using ω-splicing.

In this section we give another and more direct proof of this lemma, as a natural consequence of a representation of splicing rules.

Let Γ be a H system of alphabet V, with a finite number of splicing rules. The language $L(\Gamma)$ generated by a H system Γ can be obtained in the following way. Replace every splicing rule

$$r_i = u_1\#u_2\$u_3\#u_4$$

of Γ by the following four rules (two cut rules, a paste rule, a deletion rule) where $\bullet_i$ and $\odot_i$ are symbols that do not belong to V, and variables $x_1, x_2, y_1, y_2, x, y, w, z$ range over the strings on V.

1. $x_1u_1u_2x_2 \Longrightarrow_{r_i} x_1u_1\bullet_i$ right cut rule
2. $y_1u_3u_4y_2 \Longrightarrow_{r_i} \odot_i u_4y_2$ left cut rule
3. $xu_1\bullet_i, \quad \odot_i u_4y \Longrightarrow_{r_i} xu_1 \bullet_i \odot_i u_4y$ paste rule
4. $w \bullet_i \odot_i z \Longrightarrow_{r_i} wz$ deletion rule

If we apply these rules in all possible ways, starting from the axioms of Γ, the set of strings so generated that also belong to V^* coincides with the language $L(\Gamma)$.

This representation of splicing has a very natural biochemical reading [3]. Actually, the first two rules are an abstract formulation of the action of restriction enzymes, where the symbols $\bullet_i$ and $\odot_i$ correspond to the sticky ends (here complementarity is between $\bullet_i$ and $\odot_i$). The third rule is essentially the annealing process that joins strands with matching sticky ends but leaves a hole (represented by the string $\bullet_i\odot_i$) in the fosphodiesteric bond between the 5' and 3' loci. The final rules express the hole repair performed by ligase enzyme.

The proof that will follow is inspired by this analysis of the splicing mechanism and develops an informal idea already considered in [7].

Theorem
Let Γ be a H system with a finite number of axioms and rules, then $L(\Gamma)$ is regular.

Proof. Let $r_1, r_2, \ldots r_n$ be the rules of Γ. For each rule $r_i = u_1\#u_2\$u_3\#u_4$, introduce two new symbols $\bullet_i$ for $i = 1, \ldots n$, and $\odot_i$ we call *bullet* and *antibullet* of the rule r_i.

If u_1u_2 and u_3u_4 are the left and the right splicing sites of r_i, then symbols $\bullet_i$ and $\odot_i$ can be used in order to highlight the left and right splicing sites that occur in a string.

More formally, let h be the morphism

$$h : (V \cup \{\bullet_i | i = 1, \ldots n\} \cup \{\odot_i | i = 1, \ldots n\})^* \to V^*$$

that coincides with the identity on V while erases all the symbols that do not belong to V, that is, associates the empty string λ to them. In the following V' will abbreviate $(V \cup \{\bullet_i | i = 1, \ldots n\} \cup \{\odot_i | i = 1, \ldots n\})^*$. For any rule $r_i = u_1\#u_2\$u_3\#u_4$, with $i = 1, \ldots, n$, we say that a string of V' is $\bullet_i$-factorizable if it includes a substring $u_1'u_2' \in V'$ such that $h(u_1') = u_1$, $h(u_2') = u_2$. In this case the relative $\bullet_i$-factorization of the string is obtained by replacing $u_1'u_2'$ with $u_1' \bullet_i u_2'$. Analogously, a string of V' is $\odot_i$-factorizable if it includes a substring $u_3'u_4' \in V'$ such that $h(u_3') = u_3$, $h(u_4') = u_4$. In this case the relative $\odot_i$-factorization of the string is obtained by replacing $u_3'u_4'$ with $u_3' \odot_i u_4'$.

A string α is a maximal factorization of a string η if α is a factorization such that $h(\alpha) = \eta$, α contains no two consecutive occurrences of the same bullet or antibullet, while any further factorization of α contains two consecutive occurrences of the same bullet or antibullet. It easy to verify that the maximal factorization of a string is unique.

Now, given a H system, we factorize its axioms in a maximal way. Let $\alpha_1, \alpha_2, \ldots, \alpha_m$ be these factorization and let $\triangleright\alpha_1\triangleright, \triangleright\alpha_2\triangleright, \ldots, \triangleright\alpha_m\triangleright$ be their extensions with a symbol marking the start and the end of these factorizations. From the set Φ of these *factorization strings* we construct the following labeled directed graph G_0, which we call *axiom factorization graph* of the system, where:

1. A node is associated to each occurrence of a bullet or antibullet that occurs in in a strings of Φ, while a unique *entering node* $\triangleright\bullet$ is associated to all the occurrences of symbol $\triangleright$ at beginning of the strings of Φ, and a unique *exiting node* $\bullet\triangleright$ is associated to all the occurrences of symbol $\triangleright$ at end of the strings of Φ.
2. From any node n there is an arc to the node m if the occurrence to which m is associated is immediately after the occurrence to which n is associated; this arc is labeled with the string between these two occurrences.
3. Each bullet is linked by an arc with empty label to the antibullet of the same index.

As an example, apply this procedure to the following H system specified by:

Alphabet: $\{a, b, c, d\}$
Axioms: $\{dbab\ ,\ cc\}$
Rules: $\{r_1 :\ a\#b\$\lambda\#ab\ , r_2 :\ baa\#aaa\$\lambda\#c\}$

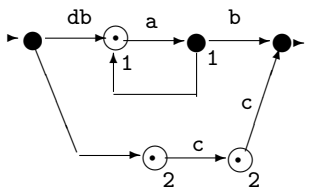

Fig. 1.1. An Axiom Factorization Graph

In Figure 1.1 the graph G_0 of our example is depicted. Hereafter, unless it is differently specified, when we say path we intend a path going from the entering node to the exiting node. If one concatenates the labels along a path, one gets a string generated by the given H system. However, there are strings generated by the system that are not generated as (concatenation of labels of) paths of this graph. For example, the strings $dbaac, dbaacc$ do not correspond to any path of the graph above. In order to overcome this inadequacy, we extend the graph that factorizes the axioms by adding new possibilities of factorizations, where the strings u_1 and u_4 of the rules are included and other paths with these labels are possible.

Before going on, let us sketch the main intuition underlying the proof, that should appear completely clear when the formal details are developed. The axiom factorization graph suggests that all the strings generated by a given H system are combinations of: i) strings that are labels of the axiom factorization graph, and ii) strings u_1 and u_4 of splicing rules of the system. Some combinations of these pieces can be iterated, but, although paths may include cycles, there is only a finite number of ways to combine these pieces in paths going from the entering node to the exiting node. An upper bound on this number is determined by: i) the number of substrings

of the axioms and of the rules, and ii) the number of factorization nodes that can be inserted in a factorization graph when extending the axiom factorization with the rules, as it will be shown in the proof.

The detailed construction of the proof is based on two procedures that expand the axiom factorization graph G_0. We call the first procedure **Rule expansion** and we call the second one **Cross bullet expansion**. In the following, for the sake of brevity, we identify paths with factorization strings.

Rule expansion
Let G_0 be the graph of the maximal factorizations of the axioms of Γ. Starting from G_0, we generate a new graph G_e we call *rule expansion* of G_0.

Consider a symbol $\otimes$ we call *cross bullet*. For this symbol h is assumed to be a deleting function, that is, $h(\otimes) = \lambda$. For every rule $r_i = u_1\#u_2\$u_3\#u_4$ of Γ, add to G_0 two rule components: the u_1 component that consists of a pair of nodes $\otimes$ and $\bullet_i$, with an arc from $\otimes$ to $\bullet_i$ labeled by the string u_1, and the u_4 component that consista of a pair of nodes $\otimes, \odot_i$ with an arc from $\odot_i$ to $\otimes$ labeled by the string u_4.

Then, add arcs with empty label from the new node $\bullet_i$ to the corresponding antibullet nodes $\odot_i$ that were already in the graph. Analogously, add arcs with empty label from the nodes $\bullet_i$ that were already in the graph to the new antibullet node $\odot_i$.

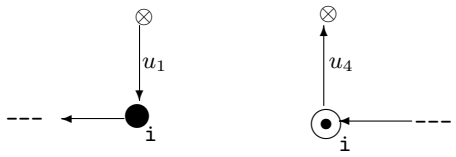

Fig. 1.2. Rule i Expansion Components

In the case of the graph in Figure 1.1, if we add the rule expansions of the two rules of the system, we get the graph of Figure 1.3.

Cross bullet Expansion
Now we define the cross bullet expansion procedure. Consider a symbol $\circ$, which we call *empty bullet*. For this symbol, h is assumed to be a deleting function, that is, $h(\circ) = \lambda$.

Suppose that in G_e there is a cycle $\odot_j - \bullet_j$. A cycle intoduces new factorization possibilities that are not explicitly present in the graph, but that appear when we go around the cycle a certain number of times. Let us assume that, by iterating this cycle, a path θ is obtained that generates a new splicing site for some rule.

We distinguish two cases. In the first case, which we call *1-cross bullet expansion*, the path θ, considered as a string, includes the splicing site u_1u_2 of a rule:

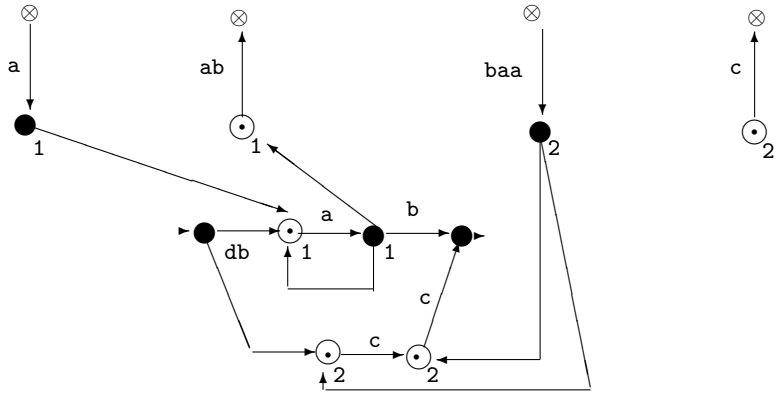

Fig. 1.3. The Rule Expansion of the Graph of Figure 1.1

$$\theta = \eta\xi\beta$$

for some ξ beginning with a symbol of V such that $h(\xi) = u_1u_2$.

In the second case, which we call *4-cross bullet expansion*, the path θ, considered as a string, includes the splicing site u_3u_4 of a rule:

$$\theta = \eta\xi\beta$$

for some ξ ending with a symbol of V such that $h(\xi) = u_3u_4$.

As it is illustrated in the following pictures, the positions where the beginning of ξ or the end of ξ are respectively located could be either external to the cycle or internal to it.

In both cases we insert an empty bullet in the path θ.

- In the case of a 1-cross bullet expansion, we insert an empty bullet $\circ$, exactly before ξ, with an arrow going from this empty bullet to the cross bullet of the u_1 expansion component of r_i. Let $1/i$ be the type of this empty bullet. This expansion is performed unless η does not already include an empty bullet of type $1/i$ after its last symbol of V.
- In the cases of a 4-cross bullet expansion, we insert an empty bullet $\circ$, exactly after ξ, with an arrow, entering this empty bullet and coming from the cross bullet of the u_4 expansion component of r_i. Let $4/i$ be the type of this empty bullet. This expansion is performed unless β does not already include an empty bullet of type $4/i$ before its first symbol of V.

The two cases are illustrated in the following pictures (in 1-cross bullet expansion the beginning of the splicing site is external to the cycle, in the 4-cross bullet expansion the end of the splicing point is internal to the cycle).

The general situation of cross bullet expansion is illustrated in figure 1.3.

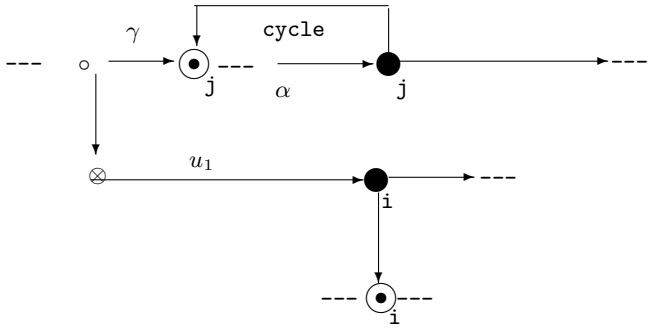

Fig. 1.4. 1-Cross bullet Expansion: $\gamma\alpha^n$ includes a string of $h^{-1}(u_1u_2)$ as its prefix

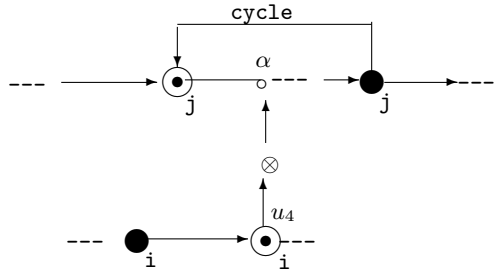

Fig. 1.5. 4-Cross bullet Expansion: α^n includes a string of $h^{-1}(u_3u_4)$ as its suffix

If we apply cross bullet expansion to the graph of Figure 1.3 we get the graph of Figure 1.7 where only an 1-cross bullet expansion was applied. The following lemma establishes the termination of the cross bullet expansion procedure.

Lemma 1. *If, starting from G_e, we apply again and again the cross bullet expansion procedure, then the resulting process eventually terminates, that is, after a finite number of steps we get a final graph where no new cycles can be introduced.*

Proof. This holds because in any cross bullet expansion we insert an empty bullet ∘ and an arc with empty label connecting it to a cross node, but empty bullets, at most one for each type, are always inserted between two symbols of V starting from the graph G_e, which is fixed at beginning of the cross bullet expansion process. Therefore, only a finite number of empty bullets can be inserted. This ensures that the expansion process starting from G_e will eventually stop.

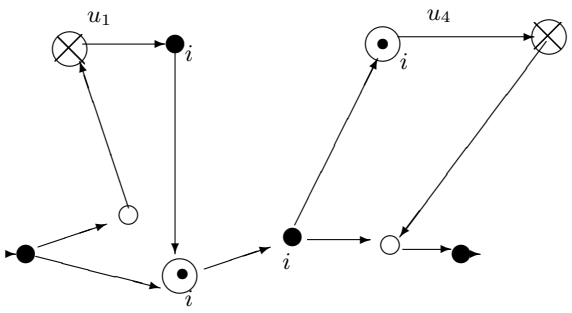

Fig. 1.6. Cross bullet Expansions of rule i

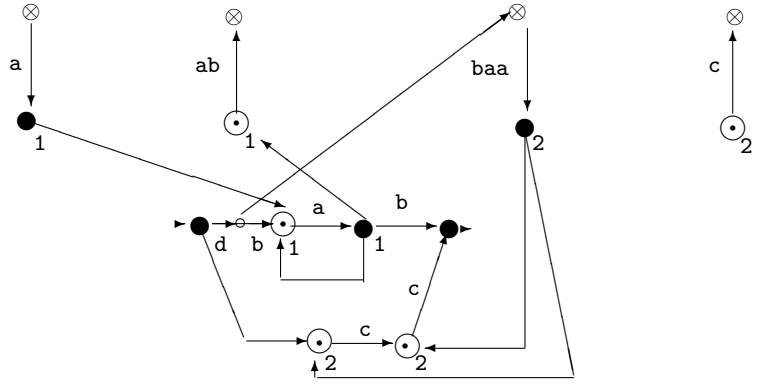

Fig. 1.7. Cross bullet Expansion of the Graph of Figure 1.3

Let G be the *completely expanded graph*, the cross bullet expansion procedure is performed by a (finite) sequence of steps. At each step an empty bullet is inserted and an arc is added that connects the empty bullet with a cross bullet.

Let $L(G)$ be the language of the strings generated by paths of G. The inclusion $L(G) \subseteq L(\Gamma)$ can be easily shown by induction on the number of cross bullet expansion steps: obviously $L(G_0) \subseteq L(\Gamma)$, thus, assume that all the paths of the graph at step i generate strings of $L(\Gamma)$, then the paths in the expanded graph at step $i+1$ generate strings spliced from paths which are present at step i. Therefore, the inclusion holds for the completely expanded graph.

For the inverse inclusion we need the following lemma which follows directly from the method of the cross bullet expansion procedure.

Lemma 2. *In the completely expanded graph G, when there is a path $\eta\theta\sigma\rho$ where $h(\theta) = u_1$, $h(\sigma) = u_2$ and u_1u_2 is the splicing site of the rule r_i, then also a path $\eta\theta'\bullet_i$ occurs in G with $h(\theta) = h(\theta')$. Analogously, if in G there is a path $\eta\theta\sigma\rho$ where $h(\theta) = u_3$, $h(\sigma) = u_4$ and u_3u_4 is the splicing site of the rule r_i, then also a path $\odot_i\sigma'\rho$ occurs in G with $h(\sigma) = h(\sigma')$.*

The inclusion $L(\Gamma) \subseteq L(G)$ can be shown by induction on the number of splicing steps. If η is an axiom, the condition trivially holds. Assume that η derives, by means of a rule r_i, from two strings. This means that these strings can be factorized as:

$$\alpha \bullet_i \beta$$

$$\gamma \odot_i \delta$$

and, by induction hypothesis, in G there are two paths θ, ρ generating $\alpha\beta$ and $\gamma\delta$ respectively. These paths include as sub-paths $h^{-1}(u_1u_2)$, $h^{-1}(u_3u_4)$ respectively, therefore, according to the previous lemma, a path $\sigma\bullet_i$ is in G where $h(\sigma) = \alpha$ and a path $\odot_i\pi$ is in G where $h(\pi) = \delta$. This means that the path $\sigma \bullet_i \odot_i\pi$ is in G, but $h(\sigma \bullet_i \odot_i\pi) = \eta$, therefore η is generated by a path of the completely expanded graph.

In conclusion, $L(\Gamma) = L(G)$.

The language $L(G)$ is regular because it is easy to define it by means of a regular expression deduced from G.

1.4 Acknowledgments

I want to express my gratitude to: Giuditta Franco, Tom Head, Victor Mitrana, Gheorghe Păun, and Giuseppe Scollo for their suggestions that were essential in improving some previous versions of this paper.

References

1. Culik II K., Harju T., The regularity of splicing systems and DNA, in Proc. ICALP 1989, LNCS 372, pp. 222-233, 1989.
2. Culik II K., Harju T., Splicing semigroups of dominoes and DNA, Discrete Applied Mathematics, 31, pp. 261-277, 1991.
3. Garret R. H., Grisham C. M., Biochemistry, Saunders College Publishing, Harcourt, 1997.
4. Head T., Formal language theory and DNA: an analysis of the generative capacity of recombinant behaviors, Bulletin of Mathematical Biology, 49, pp. 737-759, 1987.
5. Head T., Păun G., Pixton D., Language theory and molecular genetics, Chapter 7 in: [11], pp. 295-360, 1997.

6. Manca V., Splicing Normalization and Regularity, in: C. Calude, Gh. Paun, eds., Finite versus Infinite. Contributions to an Eternal Dilemma, Springer-Verlag, London, 2000.
7. Manca V., On Some Forms of Splicing, in: C. Martin-Vide, V. Mitrana eds., Words, sequences, grammars, languages: where biology, computer science, linguistics and mathematics meet, Kluwer, Dordrecht, 2000.
8. Păun G., Rozenberg G., Salomaa A., DNA Computing. New Computing Paradigms, Springer-Verlag, Berlin, 1998.
9. Pixton D., Regularity of splicing languages, Discrete Applied Mathematics (69)1-2, pp. 101-124, 1996.
10. Pixton D., Splicing in abstract families of languages, Theoretical Computer Science, 234:135–166, 2000.
11. Rozenberg G., Salomaa A.(eds.), Handbook of Language Theory, 3 Voll., Springer Verlag, Berlin, Heidelberg, 1997.