Some Interesting Properties Of Splicing Languages

Pethuru Raj, Naohiro Ishii
Dept. of Intelligence and Computer Science
Nagoya Institute of Technology
Nagoya 466 8555, Japan
peter@ics.nitech.ac.jp

Abstract

Tom Head[4] introduced the exciting concept of splicing system as an effective mathematical model for representing the reactionary behaviors of DNA molecules under the influence of certain restriction enzymes and ligases in a test tube environment. A corresponding splicing language can be generated by splicing system. In this paper, we have given some new characterizations for splicing languages, analyzed various types of splicing systems and brought out a number of new algebraic properties.

1 Introduction

The genome of complex organisms are organized into chromosomes which contain genes arranged in linear order. It is true that DNA is a language for specifying the structures and processes of life. Recently, it has been proved that very simple genes could be described by means of regular grammars[2]. In the course of its evolution, the genome of an organism mutates by different processes. At the level of individual genes the evolution proceeds by local mutations which substitute, insert and delete nucleotides of DNA sequence. These operations reveal the evolutionary and functional relationships between genes. Also, it has been found that there are a number of large-scale rearrangements in one evolutionary event. They are inversion, transposition, duplication and splicing. In particular, we are to discuss about the effects of splicing operation on DNA sequences.

DNA(deoxyribonucleic acid [8]) is found in all living organisms as the storage medium for genetic information. It consists of polymer chains, customarily referred to as DNA strands. A chain is composed of nucleotides, also referred to as bases. The four DNA nucleotides or bases are denoted by A(adenine), C(cytosine), G(guanine),and T(thymine). The DNA alphabet $\Sigma_{DNA} = \{A, C, G, T\}$. Thus, DNA strands may be viewed as words over the DNA alphabet. A restriction enzyme has the capability of recognizing short sequences of double-stranded DNA molecules and cuts it in a specific way. The following matter appears to be of interest from the computational biology point of view.

Suppose we are given a finite set M of DNA molecules and a finite set N of restriction enzymes. What is the nature of the language consisting of all DNA molecules that can arise through the action of the N number of restriction enzymes on the set M of DNA molecules?.

Tom Head proposed a mathematical model for this biological problem by introducing a novel concept called splicing operation as a language theoretic operation. Here we are to discuss and analyze some of the interesting properties of splicing languages and in that process we have introduced a number of useful characterizations and results.

2 Some terminologies on Splicing system

We now recall some definitions and notations from formal language theory and formalize the operations mentioned above.

Let Σ be an alphabet; we denote by Σ^*, the set of all strings over Σ, by ϵ, the empty word, $\Sigma^+ = \Sigma^* - \{\epsilon\}$,and by $|w|$ the length of $w \in \Sigma^*$. The families of finite, regular, linear, context-free, context-sensitive and recursively enumerable languages are denoted by FIN, REG, LIN, CF, CS, RE, respectively and Pref(x) denotes the set of all prefixes of x.

Further, we define the mirror image *mi(x)* of a word $x = a_1a_2a_3...a_n, a_i \in \Sigma$ for $1 \leq i \leq n$, by $mi(a_1a_2...a_n) = a_na_{n-1}...a_1$ and for a word $x \in \Sigma^*$, let $Perm(x) = \{y/|y|_a = |x|_a \forall a \in \Sigma\}$, be the set of all words over V which are permutations of x.

For a string $x \in \Sigma^*$, the inversion is defined as $Inv(x) = \{x_1mi(x_2)x_3/x = x_1x_2x_3, x_2 \in \Sigma^*\}$.

For a string $x \in \Sigma^*$, the transposition is defined as $Tr(x) = \{x_1x_3x_4x_2x_5x_6$ for $x = x_1x_2x_3x_4x_5x_6, (x_2, x_4, x_5) \in (\Sigma^*)^3\}$.

For a string x in Σ^*, the duplication is defined as $Dupl(x) = \{x_1x_2x_3x_4x_2x_5x_6$ for $x = x_1x_2x_3x_4x_5x_6, (x_2, x_4, x_5) \in (\Sigma^*)^3\}$.

For all $S \in \{mi, Perm, Inv, Transp, Dupl\}$, the operations can naturally be extended to languages by $S(L) = \cup_{x \in L} S(x)$ [1].

The iterated versions of the above operations are naturally defined as follows:

For $S \in \{Inv, Transp, Dupl\}$, we get $S^0(L) = L$, $S^{i+1}(L) = S(S^i(L))$, $S^* = \cup_{i \geq 0} S^i(L)$.

For example, if we take the language L with a single word *abab* and $(ab, a, b) \in (\Sigma^*)^3$ and it is easy to see that the iterated duplication $Dupl^*(L) = \{a^nb^na^nb^n/n \geq 1, m \geq 1\}$.

The full quotient of $N \subseteq \Sigma^*$ by $M \subseteq \Sigma^*$, written $M\backslash\backslash N$ is defined as follows:

$M\backslash\backslash N = \{u \in \Sigma^*/\exists w \in M,\ wu \in N \wedge \forall v \in pref_+(u), wv \ni M\}$

A self cross-over system is a triple: $SCO = (V, A, R)$, where V is an alphabet, A is a finite subset of V^*, and R is a finite commutative relation, $R \subset (V^* \times V^*)^2$.

With respect to a self cross-over system as above, for $x \in V^+$, $x \bowtie y$ iff i) $x = x_1\alpha\beta x_2 = x_3\gamma\delta x_4$ ii) $y = x_1\alpha\delta x_4$ iii) $(\alpha, \beta)R(\gamma, \delta)$. The language generated by a self cross-over system as above is

$L(SCO) = \{x \in V^* | w \bowtie^* x,\ w \in A\}$

This section formally defines splicing system, which was first introduced in [4].

2.1 Definition

A splicing system is a quadruple $S = (\Sigma, I, B, C)$, where

Σ : a finite alphabet,

I : a set of initial strings in Σ^* and

B, C : finite sets of triples $(a, x, b), a, x, b \in \Sigma^*$.

The sets B and C are called left-hand patterns and right-hand patterns respectively. String x is called the crossing of the site axb. (In a biological interpretation, one may take I as the initial set of DNA molecule sequences, B and C as the sets of splicing rules specified by restriction enzymes and a ligase).

2.2 Definition

For a splicing system $S = (\Sigma, I, B, C)$, $L(S)$ is the set of strings generated by S which is formally defined as follows. If $w_1 = uaxbv$ and $w_2 = pcxdq$ are in I and axb and cxd are sites of the same hand, then $w_3 = uaxdq$ and $w_4 = pcxbv$ are in I_1.

If w_1 and w_2 are two strings in $I \cup I_1$, then I_2 contains w_3 and w_4.

For a set of initial strings $I \subseteq \Sigma^*$, the non-iterated splicing operation is defined as follows:

$L(S) = I \cup I_1$ and the iterated splicing operation is defined as $L^*(S) = I \cup I_1 \cup I_2 ...$, i.e., $L^*(S)$ is the minimal subset of Σ^* which contains I and is closed under the operation of splicing.

2.3 Definition

A splicing system $S = (\Sigma, I, B, C)$ is persistent if, for each pair of strings $uaxbv$ and $pcxdq$ in Σ^* with sites axb and cxd of the same hand, it has the following property: if y is a substring of uax(respectively, xdq) that is the crossing of a site in $uaxbv$ (respectively, $pcxdq$), then this same substring y of $uaxdq$ 'contains an occurrence of' the crossing of a site in $uaxdq$.(Intuitively, in a persistent splicing system, it is possible to apply consecutive splicing operations infinitely many times).

A splicing system is permanent [3] if the words 'contains an occurrence of' in the above definition, is replaced by 'is'.

A null context splicing system is a splicing system $S = (\Sigma, I, B, C)$ for which each cleavage pattern in B and each in C has the form $(1, x, 1)$ and a uniform splicing system is a null context splicing system $S = (\Sigma, I, X, X)$,where $X = B \cup C$, for which there is a positive integer P such that $X = \Sigma^P$.

2.4 Definition

A splicing system $S = (\Sigma, I, B, C)$ is crossing disjoint if there do not exist patterns (a, x, b) in B and (c, x, d) in C with the same crossing x.

The set of patterns in S is full context if when (a, x, d) and (c, x, d) are in B or C, then (a, x, d) is in B or C.

3 Examples and Counter-Examples

3.1 Proposition

Let S be a splicing system with only one crossing. If there is only one occurrence of the crossing in all the initial strings for a splicing system S, then the splicing language and its Kleene's closure are one and the same, i.e., $L(S) = L^*(S)$.

Consequently, if there are more than one crossing or more than one occurrence of crossing in any one of the initial strings, then the resulting splicing language is infinite.

3.2 Example

Let $\Sigma = \{a, b, c, d\}, I = \{abcd, dcba\}, B = \{(1, b, 1), (1, c, 1)\}, C = \emptyset$. Then the splicing language is $L(S) = \{a(bc)^+d, d(cb)^+a, aba, dcd, abcba, dcbcd, ...\}$, an infinite language and

let $\Sigma = \{a\}, I = \{aaa\}, B = \{(1, aa, 1)\}, C = \emptyset$, then the splicing language $L(S) = \{a^3a^*\}$, an infinite language.

3.3 Lemma

For any family F of languages, $F \subseteq L^*(S)$, S is the splicing system with F as the initial set.

As the relevant processing is being initiated on the initial set of strings, it is obvious that the initial set is a subset of the resulting set what ever be the initial set.

3.4 Theorem

The families FIN and RE are closed under non-iterated splicing operation[7].

Clearly, $L^*(S)$ is finite for any set of patterns and for any finite initial language FIN. Using previous lemma, the closure of the family FIN is proved.

For the family RE, we use previous lemma and Church's Thesis.

3.5 Lemma

The families REG ,CF and RE are closed under $L^*(S)$ and the family FIN is not closed under $L^*(S)$.

Note A splicing language need not be a strictly locally testable language and hence a persistent splicing language.

Let $S = (\Sigma, I, B, C)$ be a splicing system where $\Sigma = \{a, b, c\}$,

$I = \{baa, bb, aaac, cc, ac\}$,

$B = \{(b, a, \wedge), (ba, a, \wedge), (\wedge, b, \wedge), (\wedge, aa, c), (\wedge, aa, ac), (\wedge, \wedge)\}$,

$C = \emptyset$.

Here $L^*(S) = bb^*a^* + a^*c^*c$ and L is neither strictly locally testable nor persistent.

3.6 Examples

Let $S = (\Sigma, I, B, C)$ be a splicing system with $\Sigma = \{a\}$ and $I = \{aa\}, B = \{(1, a, 1)\}$, $C = \emptyset$. The language generated by S is $L^*(S) = \{a^n : n \geq 1\}$.

If $\Sigma = \{a, b\}$ and $I = \{ab, ba\}$, $B = \{(1, a, 1), (1, b, 1)\}$, $C = \emptyset$, then $L^*(S) = \{(ab)^n a^m\} \cup \{(ba)^n b^m : n, m \geq 0\}$.

The language $L^*(S) = \{a^i b^j a^m b^n\}$ is a splicing language and can not be generated by any self cross-over system.

The self cross-over language $\{ba^{2n}b : n \geq 0\} \cup \{bb\}$ can not be generated by any splicing system.

The regular language $(aa)^*$ is not a splicing language.

3.7 Results

The family of splicing languages $L^*(S)$ is incomparable with the families of regular, self cross-over and context-free languages.

The family of splicing languages is incomparable with each of the following families: DOL languages, OL languages, EOL languages.

4 Closure and Decidability Properties

In this section, we shall discuss about the closure and decidability properties of splicing languages.

4.1 Theorem

The family of splicing languages is an anti-AFL and it is not closed under left/right derivatives and complement too.

Proof:

1. Union: The languages $L_1 = \{a(xb)^n xc : n \geq 0\}$ and $L_2 = \{d(xb)^n xf : n \geq 0\}$ are two splicing languages but not their union.
2. Concatenation: The catenation of above two splicing languages can not be generated by any splicing system and thus under kleene's operation.

3. Intersection with regular language: Consider the intersection between the splicing language $\{a^n : n \geq 1\}$ and the regular language $\{a^{2n} : n \geq 1\}$, which is not a splicing language.

4. Morphisms: Take the morphism $h : \{a, b\}^* \to \{a, \lambda\}^*$ defined by $h(b) = \lambda, h(a) = a$ and the splicing language is $L = b(aa)^*$. However, $h(L) = (aa)^*$ is not a splicing language.

5. Inverse homomorphisms: The image of a splicing language is not a splicing language under inverse homomorphism.

6. Left derivatives: Take $L^*(S) = b(aa)^*$, a splicing language. But $\partial(L) = (aa)^*$ is not a splicing language.

4.2 Proposition

It is algorithmically decidable whether a splicing system S is persistent as well as permanent.

4.3 Proposition

For $S = (\Sigma, I, B, C)$, a crossing disjoint, reduced, permanent splicing system, it is algorithmic to determine if $L^*(S)$ is a finite language and to determine the equivalence of two languages $L^*(S)$ and $L(S')$.

Note Every persistent splicing system need not be permanent.

For instance, let $\Sigma = \{a, b, c, d, x, y\}$, $B = \{(a, xy, b), (c, xy, d), (ax, y, d), (c, x, yb)\}$, $C = \emptyset$.

On splicing any two arbitrary initial strings *axyb* and *cxyd* using their respective patterns (a, xy, b) and (c, xy, d) with the crossing 'xy'', we get *axyd* and *cxyb*. But there is no pattern with crossing 'xy' with a site in *axyd*. Thus, this splicing system is not permanent.

However, this is persistent, since the same crossing 'xy' containing y, which is a crossing occurs in a site in *axyd*.

5 Some observations on Persistence and Permanence

- Let $S = (\Sigma, I, B, C)$ be a full context splicing system. If the length of crossing in all the patterns is equal, then $X = B \cup C$ is both persistent and permanent.

- If there are no patterns with crossings which are substrings of other crossings of the patterns in X, then X is both persistent and permanent.

- If $(a, p, b), (c, pq, d)$ and (ap, q, b) and/or (a, p, qb) are in X, then (ap, q, d) and/or (a, p, qd) should be in X for S to be both persistent and permanent.

6 Characterizations of Splicing languages by syntactic congruence

To each language L, we associate canonically a congruence for all $x, y \in \Sigma^*$. The congruence $\sim_L$ for $u, v \in \Sigma^*$ by $u \sim_L v$ iff xuy and xvy are both in L or both in $\Sigma^* \setminus L$ for all $x, y \in \Sigma^*$. The congruence $\sim_L$ is called the syntactic congruence of L and the monoid $M(L) = \Sigma^* \setminus \sim_L$ is called the syntactic monoid of L. Kleene's theorem shows that a language L over Σ is regular iff its syntactic monoid is finite. That is, there exists a smallest deterministic finite automaton recognizing L. Every persistent splicing language is regular.

6.1 Proposition

Let $S = (\Sigma, I, B, C)$ be a splicing system with only one crossing. If the crossing occurs only once in all the initial strings and the symbols are single letters, then the resulting splicing language is finite and $L^*(S) = I_0 \cup I_1$ and $I_2, I_3, \ldots$ are $\emptyset$ and every element is square-free.

If the symbols are strings, i.e., suppose $w \in I$, the set of initial strings, is such that $w = uaxbv = u'axbv'v$, where $v' \in \Sigma^+$ and $u, a, x, b, v \in \Sigma^*$. On splicing and recombination, we get $uaxbv'v$ and $u'axbv$.

Now, $uaxbv'v = u'axbv'v'v = u'axb(v')^2v$

On continuing, we get $L^*(S) = \{uaxb(v')^n v : n \geq 0\}$

Thus, we get an infinite language even with a single occurrence of the crossing in the initial string.

6.2 Proposition

If at least one initial string contains more than one occurrence of the same crossing, then the resulting splicing language is infinite and $L^*(S) = I_0 \cup I_1 \cup \cdots$. and the elements are non square-free.

6.3 Example

Let $S = (\Sigma, I, B, C)$ be a splicing system, where $I = \{axbxc, dxexf\}$ and $B = \{(1, x, 1)\}, C = \emptyset$. The language generated by S is $L^*(S) = \{a(xb)^m(xe)^n xc\} \cup \{a(xb)^m(xe)^n xf\} \cup \{d(xb)^m(xe)^n xc\} \cup \{d(xb)^m(xe)^n xf\}, m, n \geq 0$. The automaton recognizing this language is as follows:

Note There will not be any loop in the automaton recognizing a finite language and there will be only one loop in the state graph of the automaton recognizing an infinite language with the same crossing being repeated in the initial strings.

Otherwise, there will be as many loops as the number of different crossings. This gives an idea for the following two propositions.

6.4 Proposition

The syntactic monoid generated by a finite splicing language does not have any idempotent element other than the zero element.

6.5 Proposition

The syntactic monoid for an infinite splicing language should have at least one idempotent element other than the zero element.

Notes The idempotent elements are of the form f_{xb} and f_{bx} where 'x' is the crossing and 'b' is the symbol found in between the occurrences of the crossing 'x' in the words of the infinite splicing language. The number of idempotent elements in the syntactic monoid for an infinite splicing language is two times the number of different crossings.

7 Properties of Persistent Splicing system

7.1 Definition

Let $S_1 = (\Sigma_1, I_1, B_1, C_1)$ and $S_2 = (\Sigma_2, I_2, B_2, C_2)$ be two splicing systems.

Then the union of S_1 and S_2 is defined as $S = (\Sigma_1 \cup \Sigma_2, I_1 \cup I_2, B_1 \cup B_2, C_1 \cup C_2)$ and the intersection is defined as $S = (\Sigma_1 \cap \Sigma_2, I_1 \cap I_2, B_1 \cap B_2, C_1 \cap C_2)$.

7.2 Definition

Let S_1 and S_2 be two splicing systems.The concatenation is defined as follows:

Let $\alpha, \eta, \beta \notin \Sigma_1 \cup \Sigma_2$. Let $\overline{\Sigma_1} = \Sigma_1 \cup \{\eta, \beta\}, \overline{\Sigma_2} = \Sigma_2 \cup \{\eta, \alpha\}, \overline{I_1} = \{u\eta\beta | u \in I_1\}, B = B_1 \cup B_2 \cup \{(\wedge, \eta, \wedge)\}, C = C_1 \cup C_2$ and $S = (\Sigma, I, B, C)$.

7.3 Proposition

The union of two persistent splicing systems need not be persistent.

Let $S_1 = (\Sigma_1, I_1, B_1, C_1)$ and $S_2 = (\Sigma_2, I_2, B_2, C_2)$ be two persistent splicing systems, where $\Sigma_1 = \{a, b, c, x\}$, $\Sigma_2 = \{e, b, f, x\}$, $I_1 = \{axbxc\}$, $I_2 = \{exbxf\}$, $B_1 = \{(a, x, b), (b, x, c), (a, x, c), (b, x, b)\}$, $B_2 = \{(e, x, b), (e, x, f), (b, x, f), (b, x, b)\}$, $C_1 = C_2 = \emptyset$.

Consider the strings $axbxc$ and $exbxf$ from $I_1 \cup I_2$. On splicing these two strings and recombining the pieces obtained, we get $axbxf, axf, exbxc, exc$. But there is no pattern with site axf, exc in B. Thus, S is not persistent.

Note If the alphabet sets for the two persistent splicing systems are distinct, then union is persistent. Similarly we can prove the following one.

7.4 Proposition

The class of persistent splicing systems is not closed under concatenation and under Kleen's closure.

Note If $B_1 \cup B_2$ is not full context, the union and concatenation of two persistent splicing systems are not persistent.

7.5 Proposition

The left full quotient of a persistent splicing language by a deterministic context free language is not necessarily persistent.

Consider a persistent splicing language $L^*(S) = \{a^+b^+a^+b^+\}$ and $CF = \{a^nb^na^mb^m/n \in N \setminus 0, m \in N\}$. Then $L\backslash\backslash R$=$\{\epsilon \cup b^+ \cup b^+a^+b^+ \cup \{a^nb^m/n > m > 0\}\}$, which is not persistent. We now discuss about existence and effective construction of test set

7.6 Definition

We say that a finite subset F of a language L is a test set for L, if for any pair of homomorphism $(g, h), g(x) = h(x)$ for all x in L iff $g(x) = h(x)$ for all x in F. That is, g and h are equivalent on L iff g and h are equivalent on F [5].

7.7 Example

Take $L^*(S) = \{a(xb)^nxc : n \geq 0\}$, a persistent splicing language. Let $F = \{axc, axbxc\}$ be the test set for $L^*(S)$.

We know from the literature if $h(uv) = g(uv)$ for a pair of homomorphisms $h, g \in \Sigma^*$, where $u, v, q, q' \in \Sigma^*$, $h(uqv) = g(uqv)$, $h(uq'v) = g(uq'v)$, then $h(uqq'v) = g(uqq'v)$. Thus, if $h(axc) = g(axc)$, $h(axbxc) = g(axbxc)$ for every $axc, axbxc \in F$, then $h(axbxbxc) = g(axbxbxc)$.

Continuing in this way, we get $h(a(xb)^nxc) = g(a(xb)^nxc)$ for every $n \geq 1$. Thus, F is the test set for $L^*(S)$.

7.8 Theorem

Every persistent splicing language has the test set.

We know that every persistent splicing language ,as a subclass of regular languages, recognized by a finite automaton should have the test set.

The effective and accurate procedure for obtaining the test set of a persistent splicing language is taking the initial strings in the splicing system in addition to the words accepted by the automaton without entering into the loop.

Conclusions

One of the most recent suggestions in developing new types of computers consists of considering computers based on molecular interactions, which under some circumstances, might be an viable alternative to the classical Turing/Von Neumann notion of computing. There are several theoretical as well as practical proposals for achieving universal programmable molecular computer in the near future. But still many hurdles have to be crossed to embracing the molecular age. The concept of splicing system serves as one of the strong proposals for attaining DNA based computations as it has been proved theoretically that splicing operation can generate recursively enumerable language under certain restrictions and thus has the full computing power of Turing Machines [6].

References

[1] Dassow, J., and Mitrana, V., 1997, Self Cross-Over Systems, Personal communication.

[2] Dassow, J., and Mitrana, V., 1997, On Some Operations suggested by Genome Evolution, Pacific Symposium on BioComputing '97, Hawaii.

[3] Gatterdam, R.W., 1992, Algorithms for splicing systems, SIAM Journ. of Computing, 21, 507 - 520.

[4] Head, T., 1987, Formal Language Theory and DNA: An analysis of the generative capacity of specific recombinant behaviors. Bulletin of Mathematical Biology, 49, 737 - 759.

[5] Hopcroft, J.E., and Ullman, J.D., 1979, Introduction to Automata Theory, Languages and Computation, Addison-Wesley, Reading, MA.

[6] Paun, Gh., 1997, On the power of the splicing operation, to appear in International Journal of Computer Mathematics.

[7] Pethuru raj, C., 1997, Contributions to the study of Persistent Splicing languages, Ph.D thesis, Anna University, India.

[8] Watson, J.D., Tooze, J., and Kurtz, D.T., 1983, Recombinant DNA: A short Course, New York, Freeman.