The characteristics of simple splicing languages over permutation groups

Cite as: AIP Conference Proceedings **2266**, 060004 (2020); https://doi.org/10.1063/5.0018731
Published Online: 06 October 2020

Mathuri Selvarajoo, Fong Wan Heng, Nor Haniza Sarmin, and Sherzod Turaev

The Characteristics of Simple Splicing Languages Over Permutation Groups

Mathuri Selvarajoo[1,a),*], Fong Wan Heng[2,b)], Nor Haniza Sarmin[2,c)] and Sherzod Turaev[3,d)]

[1]*Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA, 40450 Shah Alam, Selangor, Malaysia.*
[2]*Department of Mathematical Science, Faculty of Science, Universiti Teknologi Malaysia, Johor Bharu, Johor, Malaysia.*
[3]*3Faculty of Engineering and Natural Sciences, International University of Sarajevo, Hrasnicka cesta 15, 71210, llidza, Sarajevo, Bosnia i Herzegovina.*

[a)]Corresponding author: mathuri@fskm.uitm.edu.my
[b)]fwh@utm.my
[c)]nhs@utm.my
[d)]sturaev@ius.edu.ba

Abstract. Head in 1987 was the first person to introduce the concept of splicing system as a theoretical model for DNA based computation using splicing operation. Splicing operation is a method of cutting and recombining DNA molecules under the influence of restriction enzymes such as ligase. Previous researches have proven that splicing systems with finite sets of axioms and rules generate only regular languages. Hence, in order to increase the computational power of the languages generated by splicing systems, several restrictions in the use of rules have been considered. In this paper, simple splicing systems controlled by permutation groups are defined and the computational power of the languages generated by this variable is explored.

INTRODUCTION

Deoxyribonucleic acid (DNA) are known as the genetic material of organisms in a chain of nucleotides. The nucleotides are differ by their chemical bases that are known as adenine (A), guanine (G), cytosine (C), and thymine (T). The DNA bases are pair up with each other, A with T and C with G to form units called base pairs. Hence, nucleotides are arranged in two long strands which look like a ladder to form a spiral called double helix. DNA are represented as strings over four alphabets, i.e. D = [A/T], [C/G], [G/C], [T/A]. The recombination behaviour of restriction enzymes and ligases was introduced by Head in 1987, in which the behaviour was modelled in the form of splicing systems and languages [1]. The restriction enzyme, found naturally in bacteria will cut DNA fragments at specific sequences, known as restriction sites. While, the enzyme, ligase, will re-join DNA fragments which have complementary ends. This model has been modelled to investigate the recombinant behaviour of DNA molecules in the presence of restriction enzymes and ligases.

The computational power of the languages generated by the unrestricted splicing systems belongs to the lower level of languages according to the Chomsky Hierarchy [2]. In order to increase the computational power of the languages generated by splicing systems, several restrictions on splicing operation have been considered. A splicing systems with restrictions can be considered as theoretical models of universal programmable DNA based computers which is important from the point of view in DNA computing as stated by Adleman in [3]. In order to denote the languages, there are few operations such as concatenation and multiplication that act on the strings to form an expression. The pattern of the languages generated from splicing system will be classified in the family of languages according

*Corresponding author

Proceedings of the 27th National Symposium on Mathematical Sciences (SKSM27)
AIP Conf. Proc. 2266, 060004-1–060004-6; https://doi.org/10.1063/5.0018731
Published by AIP Publishing. 978-0-7354-2029-8/$30.00

to the Chomsky hierarchy using grammars. Some of the classes for family of languages are defined by Pixton [4], whereas relations on several variants of splicing systems are mentioned by Paun [5].

Probabilistic and weighted concepts in formal language and automata theories can also be adapted in DNA computing theory. Turaev *et al*. [6], have proven that languages with higher computational power can be generated by the probabilistic splicing system compared to the languages generated by splicing systems without any restriction. Selvarajoo *et al*. [7], also obtained the same result when probability is used as a restriction in sticker operation. Therefore, new and significant results can be obtained by introducing the probability concept on splicing systems and sticker systems, as well as using them as molecular models for stochastic methods.

As mentioned by Turaev *et al*. [6] and Selvarajoo *et al*. [7], in probabilistic splicing systems, probabilities are associated with the axioms (not with the rules), and the probability of the generated string from two strings is calculated by multiplication of their probabilities. The application of probability as a restriction to the variants of splicing system has been done by Selvarajoo *et al*. [8, 9]. A significant result on the computation power of the languages generated has been obtained. In this paper, permutation group is used as a restriction in simple splicing system . Here, permutations are associated with the axioms (not with the rules), and the permutation of the generated string from two strings is calculated by multiplication of their permutation elements.

This paper define some necessary definitions and results from formal language theory, splicing systems, and introduction on splicing system over some permutation groups in sequel. Then, some definitions of simple splicing systems over some permutation groups are given. Furthermore, an example and important results concerning the computational power of the languages generated by simple splicing system over some permutation groups are established. The conclusion of this research is then discussed at the end of the paper.

PRELIMINARIES

Some fundamental on the basic notions of the theories of formal languages and splicing systems which are used in this paper are explained in this section. The detailed information on the preliminaries involving formal language theory and algebra can be discovered in Turaev *et al*. [6], Linz [10] and Mateescu *et al*. [11].

The families of recursively enumerable, context-sensitive, context-free, linear, regular and finite languages are denoted by RE, CS, CF, LIN, REG, and FIN respectively. For these language families, the next strict inclusions, named Chomsky hierarchy, hold

$$\text{FIN} \subset \text{REG} \subset \text{LIN} \subset \text{CF} \subset \text{CS} \subset \text{RE}.$$

Definition 1 [10]:

A deterministic finite accepter (dfa) is defined by the quintuple

$$M = (Q, \sum, \delta, q_0, F),$$

where
Q is a finite set of internal states,
$\sum$is a finite set of symbols called the input alphabet,
$\delta : Q \times \sum$ is a transition function,
$q_0 \in Q$ is the initial state, and
$F \subseteq Q$ is a set of final states.

Definition 2 [10]:

A nondeterministic finite accepter (nfa) is defined by the quintuple

$$M = (Q, \sum, \delta, q_0, F),$$

where
Q is a finite set of internal states,
$\sum$is a finite set of symbols called the input alphabet,
$\delta : Q \times (\sum \cup \{\lambda\})$ is a transition function,
$q_0 \in Q$ is the initial state, and
$F \subseteq Q$ is a set of final states.

Definition 3 [10]:

A language is called regular (REG) if and only if there exists some deterministic finite accepter M such that

$$L = L(M).$$

Definition 4 [10]:

Let V be an alphabet and #, \$ be two special symbols where #, \$ $\notin V$. A splicing rule over V is a string of the form $r = u_1\#u_2\$u_3\#u_4$, where $u_i \in V^*$ for $1 \leq i \leq 4$.

For such a rule r and strings $x, y, z \in V^*$, we write $(x, y) \underset{r}{\rightarrow} z$ if and only if $x = x_1u_1u_2x_2$, $y = y_1u_3u_4y_2$ and $z = x_1u_1u_4y_2$, for some $x_1, x_2, y_1, y_2 \in V^*$.

By splicing x, y, as indicated by the rule r; u_1u_2 and u_3u_4 which are called the sites of the splicing, z is obtained . x and y are called as the first and the second term of the splicing operation respectively. In this paper, the specification of r is omitted and $\rightarrow$ is written instead of $\underset{r}{\rightarrow}$.

Definition 5 [1]:

A splicing system (EH) is a 4-tuple $\gamma = (V, T, A, R)$ where

V is an alphabet,
$T \subseteq V$ is terminal alphabet,
A is a finite subset of V and,
R is the splicing rules.

Definition 6 [11]:

The splicing language generated by splicing system γ is defined as

$$L(\gamma) = \sigma^*(L),$$

where
$\sigma(L) = \{z \in V^* | (x, y) \underset{r}{\rightarrow} z$, for some $x, y \in L,\ r \in R\}$,
$\sigma^0(L) = L$,
$\sigma^{i+1}(L) = \sigma^i(L) \cup \sigma(\sigma^i(L)), i \geq 0$,
$\sigma^*(L) = \bigcup_{i \geq 0} \sigma^i(L)$.

The family of languages generated by multiplicative splicing system of type (F_1, F_2) are denoted by $EH(F_1, F_2)$ where

$$F_1, F_2 \in \{\text{ FIN,REG,CF,LIN,CS,RE}\}\ .$$

The simplified notation $EH(F)$ of the language family generated a finite set systems with finite set of axioms were used instead of $EH(F_1, F_2)$, where $F \in \{$ FIN,REG,CF,LIN,CS,RE$\}$ shows the family of languages for splicing rules.

Theorem 1 [3]: The relations in Table 1 hold, where at the intersection of the row marked with F_1 and the column marked with F_2, there appear either the family $EH(F_1, F_2)$ or two families F_3, F_4 such that $F_3 \subset EH(F_1, F_2) \subseteq F_4$.

Definition 7 [11]:

A simple splicing system (SEH) is a triple

$$\gamma = (V, M, A),$$

where V is an alphabet, $M \subseteq V$, and A is a finite language over V. The elements of M are called markers in the form $(a, 1; a, 1)$ and those of A are called axioms.

Definition 8 [12]:
A permutation of a set A is a function ϕ: $A \rightarrow A$ that is both one to one and onto.

Definition 9 [12]:
Let A be a finite set {1, 2, 3, 4, .. , n}. The group of all permutations of A is the symmetric group on n letters and is denoted by S_n.

TABLE 1. Table for the families of languages.

	FIN	**REG**	**CF**	**LIN**	**CS**	**RE**
FIN	REG	RE	RE	RE	RE	RE
REG	REG	RE	RE	RE	RE	RE
CF	LIN,CF	RE	RE	RE	RE	RE
LIN	CF	RE	RE	RE	RE	RE
CS	RE	RE	RE	RE	RE	RE
RE	RE	RE	RE	RE	RE	RE

RESULTS

Simple splicing system over some permutation groups are explored in this section. The system is specified with permutation elements were assigned to each string. The languages generated by the splicing systems will produce new permutation element through the multiplication operation over the permutation.

Definition 10: A simple splicing system over permutation group is a 4-tuple

$$\psi = (V, M, A, S_\rho),$$

where V is an alphabet, M is the rule in the form $(a, 1; a, 1)$ for $a \in A$, S_ρ is a permutation groups and A is a finite subset of V such that

$$\sum_{(x,S_\rho)\in A} S_\rho = (1),$$

Definition 11: A simple splicing operation over permutation is defined as below:
For strings (x, S_{ρ_1}), (y, S_{ρ_2}), $(z, S_\rho) \in V^* \times S_\rho$, and $r \in M$, we say that

$$\left[(x, S_{\rho_1}), (y, S_{\rho_2})\right] \underset{r}{\rightarrow} (z, S_\rho),$$

if and only if $(x, y) \underset{r}{\rightarrow} z, S_\rho = S_{\rho_1} \cdot S_{\rho_2}$ and $r = (a, 1; a, 1) \in M$.

Definition 12: The language generated by simple splicing system over permutation ψ is defined as

$$L(\psi) = \left\{z \in V^* | (z, S_\rho) \in \sigma^*(A)\right\}.$$

From the definition of simple splicing system over permutation, the next lemma follows immediately.

Lemma 1: $SEH(FIN, F) \subseteq S_\rho SEH(F)$ for all families $F \in$ { FIN,REG,CF,LIN,CS,RE}.

Proof.

Let $\gamma = (V, M, A)$, be a simple splicing system generating the language $L(\gamma) \in SEH(FIN, F)$ where $F \in$ { FIN,REG,CF,LIN,CS,RE}.
Let $A = \{x_1, x_2, \ldots, x_n\}, n \geq 1$. Define a simple splicing system over permutation $\psi = (V, M, A', S_\rho)$ where the set of axioms is defined by

$$A' = \{(x_i, S_{\rho_i}) | x_i \in A, 1 \leq i \leq n\}$$

for all $1 \leq i \leq n$, then $\sum_{i=1}^{n} S_{\rho_i} = (1)$.
Hence, the language generated by simple splicing system over permutation is defined as $L(\psi, > (1))$, then it is not difficult to see that

$$L(\gamma) = L(\psi, > (1)) = S_\rho SEH(F).$$

Next, an example is given to illustrate the application of permutation groups to the simple splicing system.

Example 1

Let $(S_2, \bullet, (1))$be a permutation group of $S_2 = \{(1), (12)\}$ with the multiplicative operation, $\bullet$ and the identity, (1). A simple splicing system over S_2 is a construction of $\psi = (V, M, A, S_2)$ where:

$V = \{a, c, d\}$,
$M \subseteq \{a\#1\$a\#1\}$ is the set of splicing rules,
A is a finite subset of $V \times S_2 = \{daac, (12)\}$.

Step 1: Apply rule:$(a\#1\$a\#1)$ to initial strings $(daac, (12))$, the new strings produced are:$[\{daaac, dac\}, (12)(12)] = [\{daaac, dac\}, (1)]$.

The splicing scheme for this stage is $\sigma_1(A) = \sigma_0(A) \cup \{daaac(1), dac(1)\}$.
Note that $\sigma_0(A) = A$. Therefore, $\sigma_1(A) = \{daac(12), daaac(1), dac(1)\}$.

Step 2: Apply rule: $(a\#1\$a\#1)$ to the new string produced $(daaac, (1))$ and the initial string $(daac, (12))$. The new strings produced are: $[\{daaaac, dac\}, (1)(12)] = \left[\left\{da^4c, dac\right\}, (12)\right]$.

Step 3: Apply rule:$(a\#1\$a\#1)$ to the new string produced $\left(da^4c, (12)\right)$ and the initial string $(daac, (12))$. The new strings produced are: $\left[\left\{da^5c, dac\right\}, (12)(12)\right] = \left[\left\{da^5c, dac\right\}, (1)\right]$.

Step 4: Continue the splicing process until the end, the resulting language is only accepted if the value of permutation equals to identity (1). Therefore, the language of simple splicing system over permutation S_2 is $L(\psi) = \left\{da^{2n+1}c, (1), n \geq 1\right\}$.

From the Chomsky grammar, the grammars that generate this language are context-sensitive grammar but not regular. These grammars are shown in the following.

The generating grammar (context-sensitive grammar) that produced context-sensitive language
$G = \{\{S, A\}, \{a, c, d\}, S, P\}$ with production rules:
$S \rightarrow dAc, A \rightarrow aaA | aaa$.

The derivations of a language $L(\psi) = \left\{da^{2n+1}c, (1), n \geq 1\right\}$ by using context-sensitive grammar are shown in the following.
Case 1, for $n = 1$: $S \rightarrow dAc \rightarrow daaac$.
Case 2, for $n = 2$: $S \rightarrow dAc \rightarrow daaAc \rightarrow da^5c$.
Case 3, for $n = 3$: $S \rightarrow dAc \rightarrow daaAc \rightarrow da^4Ac \rightarrow da^7c$.

Continuing this derivations, for n is a finite number, it is clear that context-sensitive grammars generate where $n = 1$.

Since regular grammar is not characterized in context-sensitive grammar, then the language $L(\psi)$ will not produce a regular language. From the Chomsky hierarchy,

$$FIN \subset REG \subset LIN \subset CF \subset CS \subset RE,$$

thus $L(\psi) = \text{CS} - \text{REG}$.

The examples above illustrate that the use of permutation groups over simple splicing systems increase the computational power of splicing systems with finite components. Should also be mentioned here, there are three interesting facts of simple splicing systems over permutation groups, as stated in the following:

Theorem 1: SEH(FIN) $\subseteq S_\rho$ SEH(FIN).

Proof:

It is obvious that the language generated by a simple splicing system is also can be generated by a simple splicing system over a permutation group if we choose the trivial permutation group, i.e.$(\{e\}, \circ, e)$.

Theorem 2: REG $\subset S_\rho$ SEH(FIN) $\subseteq S_\rho$ SEH(F) = RE where S_ρ is permutation groups and F $\in$ {REG,CF,LIN,CS,RE}.

Proof:

From the Lemma 1, when the threshold value is restricted to identity, the simple splicing language generated has higher computational power than regular (REG) languages. Example 1 shows that the language generated from the simple splicing operation over permutation groups between finite languages is categorised as context sensitive (CS) languages. Hence, the language generated has higher power than REG. Moreover, the languages generated from the simple splicing operation over permutation groups between other languages is categorised as recursively enumerable (RE) languages as stated in Table 1.

Theorem 3: $S_\rho SEH(FIN) - CF \neq \emptyset$.

Proof:

The language generated from the simple splicing operation over permutation groups between finite languages is categorised as context sensitive (CS) languages, which can be referred to Lemma 1 when the threshold value is restricted to identity as in Example 1. Therefore, this language has higher power than context free (CF) languages according to the Chomsky Hierarchy. Hence, Theorem 3 is proven true.

CONCLUSION

In this paper, the characteristic of simple splicing languages over some permutation groups as restrictions have been studied and some important facts related to simple splicing over some permutation groups were established. It is shown that a permutation extension increases the computational power of the languages generated by simple splicing systems over permutation groups with finite components. In particular cases, simple splicing systems over some permutation groups can even generate non-context-free languages.

ACKNOWLEDGMENTS

The first author would like to thank the Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA for their financial funding.

REFERENCES

[1] T. Head, Bull. of Math. Bio. **49(6)**,737 - 759 (1987).

[2] N. Chomsky, Information and Control. **2**, 137 - 167 (1959).

[3] L. M. Adleman, Science. **266**, 1021 - 1024 (1994).

[4] D.Pixton, Disc. Appl. Math. **69(1)**, 101 - 124 (1996).

[5] G.Paun, G.Rozenberg, and A.Salomaa, *DNA computing: New computing paradigms* (Springer - Verlag, Berlin, 1998).

[6] S.Turaev, M.Selvarajoo, W.H Fong, and N.H.Sarmin, *Probabilistic Splicing Systems* (Springer Berlin, New York, 2013), pp 259 - 268.

[7] M.Selvarajoo, W.H.Fong, N.H.Sarmin, and S.Turaev, Mal. J. of Fund. Appl. Sci. **9(3)**, 150 - 155 (2013).

[8] M.Selvarajoo, W.H.Fong, N.H.Sarmin, and S.Turaev, J. of Math. Comp. Sci. **2(2)**, 1 - 9 (2016).

[9] M.Selvarajoo, W.H.Fong, N.H.Sarmin, and S.Turaev, American Institute of Physics, Conference Proceedings: Probabilistic simple sticker system, April 2017.

[10] P.Linz, *An Introduction to Formal Languages and Automata* (Jones and Barlett, Sudbury MA, 2006).

[11] A.Mateescu, G.Paun, G.Rozenberg, and A.Salomaa, Disc. Appl. Math. **84**, 145 - 163 (1998).

[12] B.Fraleigh, *A First Course in Abstract Algebra* (Pearson, New York, 2002).