Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.12785/amis/090311

The Generative Capacity of Probabilistic Splicing Systems

Mathuri Selvarajoo[1], *Sherzod Turaev*[2], *Wan Heng Fong*[3,*] *and Nor Haniza Sarmin*[1]

[1] Department of Mathematical Sciences, Faculty of Science, Universiti Teknologi Malaysia, 81310 UTM Johor Bahru, Johor, Malaysia
[2] Kulliyyah of Information and Communication Technology, International Islamic University Malaysia, 53100 Kuala Lumpur, Malaysia
[3] Ibnu Sina Institute for Fundamental Science Studies, Universiti Teknologi Malaysia, 81310 UTM Johor Bahru, Johor, Malaysia

Received: 12 Jul. 2014, Revised: 13 Oct. 2014, Accepted: 14 Oct. 2014
Published online: 1 May 2015

Abstract: The concept of *probabilistic splicing system* was introduced as a model for stochastic processes using DNA computing techniques. In this paper we introduce splicing systems endowed with different continuous and discrete probabilistic distributions and call them as probabilistic splicing systems. We show that any continuous distribution does not increase the generative capacity of the probabilistic splicing systems with finite components, meanwhile, some discrete distributions increase their generative capacity up to context-sensitive languages. Finally, we associate certain thresholds with probabilistic splicing systems and this increases the computational power of splicing systems with finite components.

Keywords: DNA computing, splicing systems, probabilistic splicing systems, generative capacity

1 Introduction

DNA (Deoxyribonucleic acid) is the genetic material of organisms in a chain of nucleotides. The nucleotides differ by their chemical bases that are *adenine* (**A**), *guanine* (**G**), *cytosine* (**C**), and *thymine* (**T**). This nucleotides are paired **A-T**, **C-G** according to the so-called *Watson-Crick complementary*. *Massive parallelism*, another fundamental feature of DNA molecules, allows performing millions of cut and paste operations simultaneously on DNA strands until a complete set of new DNA strands are generated.

The famous biological experiment performed by Adleman [1] using these two features (Watson-Crick complementary and massive parallelism) of DNA molecules for solving Hamiltonian Path Problem for some instances indeed gave a high hope for the future of DNA computing. Since then, several studies have been done to show the power of DNA computing. For instance, Lipton [2] proved that DNA computing techniques can be successfully used to solve the problem of finding the satisfying assignments for arbitrary contact networks. Boneh et al. [3] showed that DNA based computers can be used to solve the satisfiability problem for Boolean circuits.

One of the earliest theoretical proposals for DNA based computation, called a *splicing system*, was introduced by Head [4] in 1987. A splicing system uses a *splicing operation* that is a formal model of the cutting and recombination of DNA molecules under the influence of restriction enzymes and ligation reactions. This process works as follows: two DNA molecules are cut at specific subsequences and the first part of one molecule is connected to the second part of the other molecule, and vice versa. This process can be formalized as an operation on *strings*, described by a so-called *splicing rule*. A system starts from a given set of strings (*axioms*) and produces a *language* by iterated splicing according to a given set of splicing rules. Because of practical reasons, the case when the components of splicing systems are finite is of special interest. But splicing systems with finite sets of axioms and splicing rules generate only regular languages (see [5]), hence, several restrictions in the use of rules have been considered (for instance, see [6]), which increase the computational power of the languages generated up to the recursively enumerable languages. This is important from the point of view of DNA computing, as splicing systems with restrictions can be considered as theoretical models of *universal programmable DNA based computers*.

* Corresponding author e-mail: fwh@ibnusina.utm.my

Different problems appearing in computer science and related areas motivate the need to consider suitable computation models for the solution. For instance, the probabilistic models have been widely used in order to develop accurate tools for natural and programming language processing [7]. In fact, adding probabilities to Chomsky grammars allows eliminating derivation ambiguity and leads to more efficient parsing and tagging algorithms for the language processing. The study of probabilistic grammars (defined by assigning a probability distribution to the productions) and probabilistic automata (defined by associating probabilities with the transitions) was started in the 1960s (for instance, see [8,9,10,11,12]). Recent results on probabilistic grammars and automata can be found, for instance, in [13,14,15].

Probabilistic concepts in formal language and automata theories can also be adapted in DNA computing theory which can produce interesting results. Authors in [16,17] studied probabilistic variants of splicing and sticker systems as well as Watson-Crick finite automata. In a probabilistic splicing system (see [18]), probabilities are initially 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 probability of a generated string in a Chomsky grammar is computed by the multiplication of the probabilities of the rules used in the derivation of the string. This probability computation is *natural*: the application of a rule $u \to v$ modifies a sentential form xuy by rewriting the substring u with the substring v resulting in the sentential form xvy i.e., the rule has "direct" contribution into the new sentential form. Thus, the probability of xvy should be computed from the probabilities of xuy and the rule $u \to v$ i.e., $p(xvy) = p(xuy)p(u \to v)$.

This idea of the computation of the probability of strings cannot be directly used in splicing systems:

First, in a splicing, two strings are used. If we follow the idea above, in order to compute the probability of the string z obtained from strings x and y using splicing rule r, we need to use probabilities $p(x)$, $p(y)$ and $p(r)$ i.e., $p(z) = p(x)p(r)p(y)$. But a splicing rule does not contribute a substring into the newly formed string: it only cuts two strings in some specific sites and combines the prefixes with the suffixes. Thus, it is not quite natural to use the probability of a splicing rule in this case.

On the other hand, if in a Chomsky grammar, sentential form xvy in a derivation is resulted from sentential form xuy by rule $u \to v$, then we can also consider that the probability of xvy is computed from the probabilities of the *two* strings xuy and v i.e., $p(xvy) = p(xuy)p(v)$, where the probability of the rule $u \to v$ can be considered as the probability of v. Thus, we can adapt this "modified" definition of the probability computation for splicing systems.

Hence, the computation of the probability of the generated string from two strings by multiplication of their probabilities is natural in splicing systems.

In this paper we continue our investigation on the generative power of probabilistic splicing systems.

This paper is organized as follows. *Section 2* contains some necessary definitions and results from the theories of formal languages and splicing systems that are used in sequel. *Section 3* explains the specific features of probabilistic splicing systems in two examples and establishes some basic and also important results concerning the generative power of probabilistic splicing systems. *Section 4* discusses our main results, cites some open problems and indicates possible topics for future research in this direction.

2 Preliminaries

In this section we recall some prerequisites, by giving basic notions and notations of the theories of formal languages, and splicing systems, which are used in sequel. The reader is referred to [6,19,20] for more detailed information.

Throughout the paper we use the following general notations. The symbol $\in$ denotes the membership of an element to a set while the negation of set membership is denoted by $\notin$. The inclusion is denoted by $\subseteq$ and the strict (proper) inclusion is denoted by $\subset$. The empty set is denoted by $\emptyset$. The sets of integers, positive rational numbers and real numbers are denoted by $\mathbb{Z}$, $\mathbb{Q}^+$ and $\mathbb{R}$ respectively. The cardinality of a set X is denoted by $|X|$. The power set of X is denoted by 2^X.

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* (see [19]), hold

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

Further, we recall some basic notations of (iterative) splicing systems. Let V be an alphabet, and $\#, \$$ two special symbols. A splicing rule over V is a string of the form

$$r = u_1\#u_2\$u_3\#u_4 \text{ where } u_i \in V^*, 1 \le i \le 4.$$

For such a rule $r \in R$ and strings $x, y, z \in V^*$, we write

$$(x, y) \vdash_r z$$

if and only if

$$x = x_1u_1u_2x_2,\ y = y_1u_3u_4y_2, \text{ and } z = x_1u_1u_4y_2,$$

for some $x_1, x_2, y_1, y_2 \in V^*$.

The string z is said to be obtained by splicing x, y, as indicated by the rule r; the strings u_1u_2 and u_3u_4 are called

the *sites* of the splicing. We call x the *first term* and y the *second term* of the splicing operation.

An H *scheme* (a *splicing scheme*) is a pair $\sigma = (V,R)$, where V is an alphabet and $R \subseteq V^*\#V^*\$V^*\#V^*$ is a set of splicing rules. For a given H scheme $\sigma = (V,R)$ and a language $L \subseteq V^*$, we write

$$\sigma(L) = \{z \in V^* : (x,y) \vdash_r z, \text{ for some } x,y \in L, r \in R\},$$

and we define

$$\sigma^*(L) = \bigcup_{i\geq 0} \sigma^i(L)$$

by

$$\sigma^0(L) = L,$$
$$\sigma^{i+1}(L) = \sigma^i(L) \cup \sigma(\sigma^i(L)), i \geq 0.$$

An *extended H system* is a construct $\gamma = (V,T,A,R)$, where V is an alphabet, $T \subseteq V$ is the *terminal* alphabet, $A \subseteq V^*$ is the set of *axioms*, and $R \subseteq V^*\#V^*\$V^*\#V^*$ is the set of *splicing rules*. The system is said to be *non-extended* when $T = V$. An alphabet $x \in V$ is said to be *non-terminal* when $x \notin T$. The language generated by γ is defined by

$$L(\gamma) = \sigma^*(A) \cap T^*.$$

The symbols $\mathbf{EH}(F_1,F_2)$ denotes the family of languages generated by extended H system $\gamma = (V,T,A,R)$ with $A \in F_1$ and $R \in F_2$ where

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

The following theorem (for details, see [6]) shows the relations of the families of languages generated by splicing systems to the families of Chomsky languages.

Theorem 2.1. The relations in the following table hold, where at the intersection of the row marked with F_1 with the column marked with F_2 there appear either the family $\mathbf{EH}(F_1,F_2)$ or two families F_3, F_4 such that $F_3 \subset \mathbf{EH}(F_1,F_2) \subseteq F_4$.

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

3 Definitions, examples and results

First, we slightly modify our notions and notations related to probabilistic splicing systems initially defined in [18]. Second, we recall some results obtained from [18] and present new ones related to the generative capacity of probabilistic splicing systems.

Definition 3.1. A *probabilistic H* (*splicing*) *system* is a 5-tuple $\gamma = (V,T,A,R,p)$ where V,T,R are defined as for an extended H system, $p : V^* \to [0,1]$ is a probability function, and A is a finite subset of V^+ such that

$$\sum_{x\in A} p(x) = 1.$$

Further, we define a probabilistic splicing operation, an iterative splicing over probabilistic languages and the language generated by a probabilistic splicing system.

Definition 3.2. For strings with probabilities $(x,p(x))$, $(y,p(y))$, $(z,p(z)) \in V^* \times [0,1]$ and $r \in R$, we say that

$$[(x,p(x)),(y,p(y))] \vdash_r (z,p(z))$$

if and only if $(x,y) \vdash_r z$ and $p(z) = p(x)p(y)$.

Definition 3.3. Let $L \subseteq V^*$ be a language and $p : V^* \to [0,1]$ be a probability function. We define a *probabilistic language on* L by

$$L_p = \{(x,p(x)) : x \in L \text{ and } p(x) \text{ is the probability of } x\}.$$

Definition 3.4. For a given H scheme $\sigma = (V,R)$ and a language $L_p \subseteq V^+ \times [0,1]$, we write

$$\sigma(L_p) = \{(z,p(z)) \in V^* \times [0,1] : [(x,p(x)),(y,p(y))]$$
$$\vdash_r (z,p(z)) \text{ for some } (x,p(x)),(y,p(y)) \in L_p \text{ and } r \in R\},$$

and define

$$\sigma^*(L_p) = \bigcup_{i\geq 0} \sigma^i(L_p)$$

where

$$\sigma^0(L_p) = L_p,$$
$$\sigma^{i+1}(L_p) = \sigma^i(L_p) \cup \sigma(\sigma^i(L_p)), i \geq 0.$$

Definition 3.5. The *probabilistic language* generated by a probabilistic splicing system γ is defined as

$$L_p(\gamma) = \{(x,p(x)) \in \sigma^*(A_p) : x \in T^*\}.$$

Remark 3.1. *We should mention that different splicings may result in the same string with different probabilities. There are some techniques to remove this "ambiguity" from the strings. For instance, we can consider a second operation, such as addition or maximum, together with the multiplication. Then, the cumulative probability or maximum probability of a string generated by the splicing system can be used as the probability of the string.*

Another idea is based on the selection of the strings generated by a splicing system: we choose cut-points (thresholds) such as numbers, segments or discrete sets with different modes such as "greater than", "smaller than", "equal to", "belongs to", etc., and consider some splicings are successful and select the produced strings

into the language if the probabilities satisfy the threshold requirements. In our current research we use the latter approach because we are merely interested in the generative capacity of probabilistic splicing systems, i.e., the powers of the families of crispy languages generated by the probabilistic splicing systems.

Let $L_p(\gamma)$ be the language generated by a probabilistic splicing system $\gamma = (V,T,A,R,p)$. We consider as thresholds (cut-points) sub-segments and discrete subsets of [0, 1] as well as real numbers in [0, 1]. We define the following two types of *threshold languages* with respect to thresholds $\omega \in [0,1]$ and $\Omega \subseteq [0,1]$:

$$L_p(\gamma,\diamond) = \{z \in T^* : (z,p(z)) \in \sigma_p^*(A) \wedge p(z) \diamond \omega\},$$
$$L_p(\gamma,\star) = \{z \in T^* : (z,p(z)) \in \sigma_p^*(A) \wedge p(z) \star \Omega\}$$

where $\diamond \in \{=,\neq,\geq,>,<,\leq\}$ and $\star \in \{\in,\notin\}$ are called the *threshold modes*.

We denote the family of languages generated by probabilistic splicing systems of type (F_1,F_2) by $p\mathbf{EH}(F_1,F_2)$ where

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

Instead of $p\mathbf{EH}(F_1,F_2)$, we also use the simplified notation $p\mathbf{EH}(F)$ of the family languages generated by probabilistic splicing systems with finite sets of axioms where $F \in \{\mathbf{FIN},\mathbf{REG},\mathbf{CF},\mathbf{LIN},\mathbf{CS},\mathbf{RE}\}$ shows the language family of splicing rules.

From the definition, the next lemma follows immediately.

Lemma 3.1.

$$\mathbf{EH}(\mathbf{FIN},F) \subseteq p\mathbf{EH}(F)$$

for all families $F \in \{\mathbf{FIN},\mathbf{REG},\mathbf{CF},\mathbf{LIN},\mathbf{CS},\mathbf{RE}\}$.

Proof. Let $\gamma = (V,T,A,R)$ be a splicing system generating the language $L(\gamma) \in \mathbf{EH}(\mathbf{FIN},F)$ with $F \in \{\mathbf{FIN},\mathbf{REG},\mathbf{CF},\mathbf{LIN},\mathbf{CS},\mathbf{RE}\}$.

Let $A = \{x_1,x_2,\ldots,x_n\}$, $n \geq 1$. We define a probabilistic splicing system $\gamma' = (V,T,A',R,p)$ where the set of axioms is defined by

$$A' = \{(x_i,p(x_i)) : x_i \in A, 1 \leq i \leq n)\}$$

where $p(x_i) = 1/n$ for all $1 \leq i \leq n$, then

$$\sum_{i=1}^{n} p(x_i) = 1.$$

We define the threshold language generated by γ' as $L_p(\gamma',>0)$, then it is not difficult to see that $L(\gamma) = L_p(\gamma',>0)$. □

Example 3.1. Let us consider a probabilistic splicing system

$$\gamma_1 = (\{a,b,c,d\},\{a,b,c\},\{(cad,2/7),(dbc,5/7)\},R_1,p_1)$$

where

$$R_1 = \{r_1 = a\#d\$c\#ad, r_2 = db\#c\$d\#b, r_3 = a\#d\$d\#b\}. \quad (1)$$

It is easy to see that the first rule in (1) can only be applied to the string cad, and the second rule in (1) to the string dbc. For instance,

$$[(cad,2/7),(cad,2/7)] \vdash_{r_1} (caad,(2/7)^2),$$

and

$$[(dbc,5/7),(dbc,5/7)] \vdash_{r_2} (dbbc,(5/7)^2).$$

In general, for any $k \geq 1$ and $m \geq 1$,

$$[(ca^kd,(2/7)^k),(cad,2/7)] \vdash_{r_1} (ca^{k+1}d,(2/7)^{k+1}),$$

and

$$[(db^mc,(5/7)^m),(dbc,5/7)] \vdash_{r_2} (db^{m+1}c,(5/7)^{m+1}).$$

From the strings ca^kd, $k \geq 1$ and db^mc, $m \geq 1$ by applying the rule r_3, we obtain

$$[(ca^kd,(2/7)^k),(db^mc,(5/7)^m)] \vdash_{r_3} (ca^kb^mc,(2/7)^k)(5/7)^m.$$

Thus,

$$L_p(\gamma_1) = \{(ca^kb^m,(2/7)^k(5/7)^m) : k,m \geq 1\}.$$

We obtain the following threshold languages generated by this probabilistic splicing system with different thresholds and modes:

Since all the initial probabilities are nonzero, all multiplications of probabilities are also nonzero, consequently, $L_p(\gamma_1,=0) = \emptyset$, and

$$L_p(\gamma_1,>0) = L(\gamma_1')$$

where γ' is the "crispy" variant of γ, i.e., γ without probabilities. Then, $L_p(\gamma_1,>0)$ is *regular*.

If we choose $>$ as a mode and $(10/49)^i$, $i \geq 1$, as a cut-point, then $L_p(\gamma_1,>(10/49)^i)$ is a *finite* language, i.e.,

$$L_p(\gamma_1,>(10/49)^i) = \{ca^kb^mc : 1 \leq k,m \leq i\}.$$

We can obtain *context-free* languages if we consider some discrete subsets of $[0,1]$, for instance,

$$L_p(\gamma_1,\in\{(10/49)^n : n \geq 1\}) = \{ca^nb^nc : n \geq 1\}$$

and

$$L_p(\gamma_1,\notin\{(10/49)^n : n \geq 1\}) = \{ca^kb^mc : k > m \geq 1\} \cup \{ca^kb^mc : m > k \geq 1\}.$$

Example 3.2. Consider the probabilistic splicing system

$$\gamma_2 = (\{a,b,c,w,x,y,z\}, \{a,b,c,w,zA_2,R_2,p_2\})$$

where

$$A_2 = \{(wax,3/19),(xby,5/19),(ycz,11/19)\}$$

and

$$\begin{aligned} R_2 = \{&r_1 = wa\#x\$w\#a,\ r_2 = xb\#y\$x\#b,\\ &r_3 = yc\#z\$y\#c,\ \ r_4 = a\#x\$x\#b,\\ &r_5 = b\#y\$y\#c\}. \end{aligned}$$

We obtain the following strings from the first axiom and the rule r_1:

$$(wa^kx,(3/19)^k), k \geq 1,$$

from the second axiom and the rule r_2:

$$(xb^my,(5/19)^m), m \geq 1,$$

from the third axiom and the rule r_3:

$$(yc^nz,(11/19)^n), n \geq 1.$$

The nonterminals x and y in these strings are eliminated by rules r_4 and r_5, i.e.,

$$\begin{aligned} [(wa^kx,(3/19)^k),(xb^my,(5/19)^m)] &\vdash_{r_4}\\ (wa^kb^my,(3/19)^k&(5/19)^m) \end{aligned}$$

and

$$\begin{aligned} [(wa^kb^my,(3/19)^k(5/19)^m),(yc^nz,(11/19)^n)] &\vdash_{r_5}\\ (wa^kb^mc^nz,(3/19)^k&(5/19)^m(11/19)^n). \end{aligned}$$

Then the language generated by the probabilistic splicing system γ_2 is

$$\begin{aligned} L_p(\gamma_2) = \{&(wa^kb^mc^nz,(3/19)^k(5/19)^m(11/19)^n) :\\ &k,m,n \geq 1\}. \end{aligned}$$

Using different cut-points and modes we can obtain the following threshold languages.

(1) $L_p(\gamma_2,=0) = \emptyset$.

(2) $L_p(\gamma_2,>0) = L(\gamma_2') \in \mathbf{REG}$ where γ_2' is the crispy variant of γ_2.

(3) $L_p(\gamma_2,>\tau^i) = \{wa^kb^mc^nz : 1 \leq k,m,n \leq i\} \in \mathbf{FIN}$ where $\tau = 165/6859$.

(4) For $\Omega = \{(165/6859)^n : n \geq 1\}$,

$$L_p(\gamma_2,\in\Omega) = \{wa^nb^nc^nz : n \geq 1\} \in \mathbf{CS}-\mathbf{CF}$$

and

$$L_p(\gamma_2,\notin\Omega) = \{wa^kb^mc^nz : k \neq m, k \neq n, m \neq n\}$$

that is also in $\mathbf{CS}-\mathbf{CF}$.

The examples above illustrate that the use of thresholds with probabilistic splicing systems increase the generative power of splicing systems with finite components. We should also mention two simple but interesting facts of probabilistic splicing systems. First, as Proposition 3.1 and second, as Proposition 3.2, stated in the following.

Proposition 3.1. If the probability of each axiom $x \in A$ in a probabilistic splicing system $\gamma = (V,T,A,R,p)$ is nonzero, then the threshold language $L_p(\gamma,=0)$ is the empty set, i.e., $L_p(\gamma,=0) = \emptyset$.

Proposition 3.2. If the probability of each axiom $x \in A$ in a probabilistic splicing system $\gamma = (V,T,A,R,p)$ is not greater than 1, then every threshold language $L_p(\gamma,>\nu)$ with $\nu > 0$ is finite.

From Theorem 2.1, Lemma 3.1, Example 3.1 and 3.2, we obtain the following theorems.

Theorem 3.2.

$$\mathbf{REG} \subset p\mathbf{EH}(\mathbf{FIN}) \subseteq p\mathbf{EH}(F) = \mathbf{RE}$$

where $F \in \{\mathbf{FIN},\mathbf{REG},\mathbf{CF},\mathbf{LIN},\mathbf{CS},\mathbf{RE}\}$.

Theorem 3.3.

$$p\mathbf{EH}(\mathbf{FIN}) - \mathbf{CF} \neq \emptyset.$$

Further, we investigate the relations between "usual" splicing systems and probabilistic splicing systems.

Theorem 3.4.

$$\mathbf{EH}(\mathbf{FIN},\mathbf{FIN}) \subset p\mathbf{EH}(\mathbf{FIN},\mathbf{FIN}).$$

Proof. From Lemma 3.1,

$$\mathbf{EH}(\mathbf{FIN},\mathbf{FIN}) \subseteq p\mathbf{EH}(\mathbf{FIN},\mathbf{FIN}).$$

Since the language $L_p(\gamma_1,\in\{(10/49)^n : n \geq 1\})$ in Example 3.1 belongs to $p\mathbf{EH}(\mathbf{FIN},\mathbf{FIN})$ but not to $\mathbf{EH}(\mathbf{FIN},\mathbf{FIN})$, we get the proper inclusion. □

Since

$$L_p(\gamma_1,\in\{(10/49)^n : n \geq 1\}) \in p\mathbf{EH}(\mathbf{FIN},\mathbf{FIN})$$

in Example 3.1 is context-free, and

$$L_p(\gamma_2,\in\{(165/6859)^n : n \geq 1\}) \in p\mathbf{EH}(\mathbf{FIN},\mathbf{FIN})$$

in Example 3.2 is not context-free, we obtain the following strict inclusions.

Corollary 3.1.

$$\mathbf{EH}(\mathbf{REG},\mathbf{FIN}) \subset p\mathbf{EH}(\mathbf{REG},\mathbf{FIN})$$

and

$$\mathbf{EH}(\mathbf{CF},\mathbf{FIN}) \subset p\mathbf{EH}(\mathbf{CF},\mathbf{FIN}).$$

Since by Theorem 2.1, $\mathbf{EH}(F_1,F_2)=\mathbf{RE}$ for all

$$F_1 \in \{\mathbf{FIN},\mathbf{REG},\mathbf{LIN},\mathbf{CF},\mathbf{CS},\mathbf{RE}\}$$

and

$$F_2 \in \{\mathbf{REG},\mathbf{LIN},\mathbf{CF},\mathbf{CS},\mathbf{RE}\},$$

the following theorem also holds.

Theorem 3.5.

$$\mathbf{EH}(F_1,F_2)=p\mathbf{EH}(F_1,F_2)$$

where $F_1 \in \{\mathbf{FIN},\mathbf{REG},\mathbf{LIN},\mathbf{CF},\mathbf{CS},\mathbf{RE}\}$ and $F_2 \in \{\mathbf{REG},\mathbf{LIN},\mathbf{CF},\mathbf{CS},\mathbf{RE}\}$.

Since $\mathbf{EH}(F,\mathbf{FIN})=\mathbf{RE}$ for $F \in \{\mathbf{CS},\mathbf{RE}\}$, again by Theorem 3.1, we have the next theorem.

Theorem 3.6. For $F \in \{\mathbf{CS},\mathbf{RE}\}$

$$\mathbf{EH}(F,\mathbf{FIN})=p\mathbf{EH}(F,\mathbf{FIN}).$$

Next, we study the *regularity* of probabilistic splicing systems with finite sets of axioms and splicing rules with respect to different cut-points and modes.

Theorem 3.7. Let $\gamma=(V,T,A,R,p)$ be a probabilistic splicing system where $|A|=1$ and $R \in \mathbf{FIN}$. Then the threshold language generated by γ is a regular for all cut-points $\Omega \subseteq [0,1]$ and $\omega \in [0,1]$, and for all modes $\star \in \{\in,\notin\}$ and $\diamond \in \{=,\neq,>,\geq,\leq,<\}$.

Proof. Let $\gamma=(V,T,A,R,p)$ be a probabilistic splicing system where $A=\{(w,p(w))\}$. We denote the crispy splicing systems by γ' i.e., $\gamma'=(V,T,\{w\},R)$. By definition, $p(w)=1$. It is not difficult to see that for any $x \in \sigma_p^i(A)$, $i \geq 1$, we also have $p(x)=1$.

Further, we consider all possible cut-points and modes:

(1) $L_p(\gamma,>1)=\emptyset$.

(2) For all $\omega \in [0,1)$, $L_p(\gamma,>\omega)=L(\gamma') \in \mathbf{REG}$.

(3) For all $\omega \in [0,1]$, $L_p(\gamma,<\omega)=\emptyset$.

(4) For all $\omega \in [0,1)$, $L_p(\gamma,=\omega)=\emptyset$.

(5) $L_p(\gamma,=1)=L(\gamma') \in \mathbf{REG}$.

(6) For all $[\omega_1,\omega_2] \subseteq [0,1)$, $L_p(\gamma,\in[\omega_1,\omega_2])=\emptyset$.

(7) For all $[\omega_1,\omega_2] \subseteq [0,1)$, $L_p(\gamma,\notin[\omega_1,\omega_2])=\emptyset$.

(8) $L_p(\gamma,\in[0,1])=L(\gamma') \in \mathbf{REG}$. □

One can notice that all threshold languages generated by the probabilistic splicing system γ is the empty set or a regular language.

Theorem 3.8. Let $\gamma=(V,T,A,R,p)$ be a probabilistic splicing system with finite sets of axioms and splicing rules. If $0<p(w)<1$ for each $(w,p(w)) \in A$, then, for any $\omega \in [0,1]$, $L_p(\gamma,>\omega)$ is a finite language and $L_p(\gamma,\leq\omega)$ is a regular language.

Proof. Let $\gamma=(V,T,A,R,p)$ be a probabilistic splicing system where

$$A=\{(w_1,p_1),(w_2,p_2),\ldots,(w_n,p_n)\}.$$

where $0<p_i<1$ for all $1 \leq i \leq n$. It immediately follows that $n \geq 2$. Let $\omega \in [0,1]$. One can easily see that for any positive integer k,

$$\prod_{j=1}^{k} p_{i_j} > \prod_{j=1}^{k+1} p_{i_j}$$

where $p_{i_j} \in \{p_1,p_2,\ldots,p_n\}$. Then there exists a positive integer m such that

$$\prod_{j=1}^{m} p_{i_j} < \omega$$

for all $p_{i_j} \in \{p_1,p_2,\ldots,p_n\}$ where $1 \leq j \leq m$.

For any string $x \in \sigma_p^i(A)$, $i \geq m$, that is obtained from some strings of $\sigma_p^{i-1}(A)$ using more than or equal to m splicing operations, we have $p(x)<\omega$. Thus, $L_p(\gamma,>\omega)$ contains a finite number of strings, i.e., $L_p(\gamma,>\omega)$ is finite.

Since

$$L_p(\gamma,\leq\omega)=L_p(\gamma)-L_p(\gamma,>\omega),$$

the language $L_p(\gamma,\leq\omega)$ is regular. □

From Theorem 3.7 and 3.8, the following theorem follows.

Theorem 3.9. A non-regular language L can only be generated by a probabilistic splicing system γ with finite sets of axioms and splicing rules when the cut-point is a discrete subset of $[0,1]$ and the mode is $\in$ or $\notin$.

4 Conclusions

In this paper we studied the generative capacity of probabilistic splicing systems with different continuous and discrete probabilistic distributions. We showed that any continuous distribution does not increase the generative capacity of the probabilistic splicing systems with finite sets of axioms and splicing rules, whereas some discrete distributions increase the generative power up to context-sensitive languages.

We should mention that the incomparability of the context-free language family with the family of languages generated by probabilistic splicing systems finite components and the best upper bound for this language family remain open.

Acknowledgment

The first author would like to thank Ministry of Education, Malaysia for the financial funding through MyBrain15 scholarship. The second author acknowledges the financial support by Fundamental Research Grant Scheme **FRGS13-066-0307**, International Islamic University Malaysia, Ministry of Education, Malaysia. The third and fourth authors would also like to thank the Ministry of Education, Malaysia and Research Management Center, Universiti Teknologi Malaysia for the financial funding through Research University Fund Vote No. **08H07** and **08H45**.

The authors are grateful to the anonymous referees for a careful checking of the details and for helpful comments that improved this paper.

References

[1] L. Adleman, Science, **266**, 1021–1024 (1994).
[2] R. Lipton, Science, **268**, 542–545 (1995).
[3] D. Boneh, C. Dunworth, R. Lipton and J. Sgall, Discrete Applied Mathematics. Special Issue on Computational Molecular Biology, **71**, 79–94 (1996).
[4] T. Head, Bull. Math. Biology, **49**, 737–759 (1987).
[5] D. Pixton, Discrete Applied Mathematics, **69**, 101–124 (1996).
[6] Gh. Păun, G. Rozenberg and A. Salomaa, DNA computing. New computing paradigms, Berlin, Springer-Verlag, 1998.
[7] C. Nick, D. Christopher, Probabilistic models of language processing and acquisition. TRENDS in Cognitive Sciences, **10**, 335–344 (2006).
[8] T. Booth and R. Thompson, IEEE Transactions on Computers, **22**, 442–450 (1973).
[9] C.A. Ellis, Probabilistic Languages and Automata. PhD thesis, Department of Computer Science, University of Illinois, 1969.
[10] K.S. Fu and T. Li, International Journal of Informafion Science, **1**, 403–419 (1969).
[11] M. Rabin, Information and Control, **6**, 230–245 (1963).
[12] A. Salomaa, Information and Control, **15**, 529–544 (1969).
[13] N. Smith and M. Johnson, IEEE Transactions on Computers, **33**, 477–491 (2007).
[14] A. Kornai, Journal of Logic, Language and Information, **20**, 317–328 (2011).
[15] T. Fowler, The generative power of probabilistic and weighted context-free grammars. In: Kanazawa M, Kornai A, Kracht M, Seki H, (Eds.), The Mathematics of Language, Springer-Verlag, Berlin, 2011.
[16] M.Selvarajoo, W.H. Fong, N.H. Sarmin and S. Turaev, Probabilistic Sticker Systems, Malaysian Journal of Fundamental and Applied Sciences, **9**, 150–155 (2013).
[17] M.Selvarajoo, W.H. Fong, N.H. Sarmin and S. Turaev, Probabilistic Semi-Simple Splicing System and Its Characteristics, Jurnal Teknologi (Sciences and Engineering) Special Edition, **62**, 21–26 (2013).
[18] S. Turaev, M. Selvarajoo, W.H. Fong and N.H. Sarmin, Probabilistic splicing systems, Advanced Methods for Computational Collective Intelligence, Volume 457, Springer, Berlin, Heidelberg, 2013.
[19] G. Rozenberg and A. Salomaa (Eds.), Handbook of Formal Languages, Volumes 1-3, Springer, Berlin, 1997.
[20] J. Dassow and G. Paun, Regulated Rewriting in Formal Language Theory, Springer-Verlag, Berlin, 1989.
[21] M. Marin and T. Kutsia, Front. Comput. Sci. China, **4**, 173–184 (2010).

Mathuri Selvarajoo is a PhD candidate in Department of Mathematical Sciences, Universiti Teknologi Malaysia. Her current research interest includes DNA computing, Formal Language Theory, and application of probability in DNA computing models.

Sherzod Turaev obtained his PhD in 2010 from Universitat Rovira i Virgili, Tarragona, Spain. He is currently an Assistant Professor at Department of Computer Science, Faculty of Information and Communication Technology, International Islamic University Malaysia. His research interests include Petri net controlled grammars, descriptional complexity of formal languages and automata, and weighted DNA computing.

Fong Wan Heng obtained her PhD in 2008 from Universiti Teknologi Malaysia and is a senior lecturer at Ibnu Sina Institute for Fundamental Science Studies, Universiti Teknologi Malaysia. She has been the project leader for several research projects. Her research is focused on splicing system and DNA computing. Her research interest also includes Formal Language Theory and its application in DNA computing.

Nor Haniza Sarmin obtained her PhD in 1998 from State University of New York at Binghamton, USA and is a Professor at Department of Mathematical Sciences, Universiti Teknologi Malaysia. She has been the project leader for many research projects. Her research is focused on Group Theory and its applications, and splicing systems on DNA molecules. Her research interest also includes nonabelian tensor product and nonabelian tensor squares of groups, homological functors, capability of groups, commutativity degree of groups, Formal Language Theory and its application in DNA computing.