Probabilistic simple splicing systems
Mathuri Selvarajoo, Fong Wan Heng, Nor Haniza Sarmin, and Sherzod Turaev

Citation: AIP Conference Proceedings 1602, 760 (2014); doi: 10.1063/1.4882571
View online: http://dx.doi.org/10.1063/1.4882571
View Table of Contents: http://scitation.aip.org/content/aip/proceeding/aipcp/1602?ver=pdfcov
Published by the AIP Publishing

Articles you may be interested in
Some characteristics of probabilistic one-sided splicing systems

An analysis of four variants of splicing system

Regular languages, regular grammars and automata in splicing systems
AIP Conf. Proc. 1522, 856 (2013); 10.1063/1.4801217

SOME CHARACTERIZATIONS IN SPICING SYSTEMS
AIP Conf. Proc. 1309, 411 (2010); 10.1063/1.3525142

Probabilistic and thermodynamic aspects of dynamical systems
Chaos 8, 311 (1998); 10.1063/1.166313
Probabilistic Simple Splicing Systems

Mathuri Selvarajoo\textsuperscript{a}, Fong Wan Heng\textsuperscript{b}, Nor Haniza Sarmin\textsuperscript{a} and Sherzod Turaev\textsuperscript{c}

\textsuperscript{a}Department of Mathematical Sciences, Faculty of Science, Universiti Teknologi Malaysia, 81310 UTM Johor Bahru, Johor, Malaysia
\textsuperscript{b}Ibnu Sina Institute for Fundamental Science Studies, Universiti Teknologi Malaysia, 81310 UTM Johor Bahru, Johor, Malaysia
\textsuperscript{c}Department of Computer Science, Kulliyah of Information and Communication Technology, International Islamic University Malaysia, 53100 Kuala Lumpur, Malaysia

Abstract. A \textit{splicing system}, one of the early theoretical models for DNA computing was introduced by Head in 1987. Splicing systems are based on the splicing operation which, informally, cuts two strings of DNA molecules at the specific recognition sites and attaches the prefix of the first string to the suffix of the second string, and the prefix of the second string to the suffix of the first string, thus yielding the new strings. For a specific type of splicing systems, namely the \textit{simple splicing systems}, the recognition sites are the same for both strings of DNA molecules. It is known that splicing systems with finite sets of axioms and splicing rules only generate regular languages. Hence, different types of restrictions have been considered for splicing systems in order to increase their computational power. Recently, probabilistic splicing systems have been introduced where the probabilities are initially associated with the axioms, and the probabilities of the generated strings are computed from the probabilities of the initial strings. In this paper, some properties of \textit{probabilistic simple splicing systems} are investigated. We prove that probabilistic simple splicing systems can also increase the computational power of the splicing languages generated.

Keywords: DNA Computing; Probability; Simple Splicing Systems; Regular Languages; Computational Power
PACS: 02.20-a, 02.50.Cw, 87.14.gk

INTRODUCTION

\textit{Deoxyribonucleic acid} (DNA) is the genetic material of organisms in a chain of nucleotides. The nucleotides differ by their chemical bases that are \textit{adenine} (A), \textit{guanine} (G), \textit{cytosine} (C), and \textit{thymine} (T). DNA bases pair up with each other, A with T and C with G, to form units called base pairs. So, nucleotides can be arranged in two long strands that form a spiral called a \textit{double helix}. The structure of the double helix is somewhat like a ladder. DNA can be represented as strings over four alphabets, i.e. $D=\{[A/T], [C/G], [G/C], [T/A]\}$. \textit{Restriction enzymes}, found naturally in bacteria, can cut DNA fragments at specific sequences, known as \textit{restriction sites}; while another enzyme, ligase, can re-join DNA fragments that have complementary ends. This recombination behavior of restriction enzymes and ligases was modeled in the form of splicing systems and splicing languages by Head in 1987 [1]. This model has been defined to investigate the recombinant behavior of DNA molecules in the presence of restriction enzymes and ligases.

Since unrestricted splicing systems have the generating power limitations, several restrictions of splicing operation have been considered to increase the generating power of splicing systems. This is important from the point of view in DNA computing: splicing systems with restrictions can be considered as theoretical models of \textit{universal programmable DNA based computers} as stated by Adleman in [2]. There are few operations acting on strings that can form an expression to denote the language. The pattern of string will be classified in the family of languages according to the Chomsky hierarchy using grammars. Some of the classes for family of languages are defined by Pixton [3], whereas relations on several variants of splicing systems are mentioned by Paun [4].

Probabilistic concepts in formal language and automata theories can also be adapted in DNA computing theory. It has been proven that the languages generated by probabilistic splicing system have a higher computational power as compared to the languages generated by splicing systems without any restriction on the rules as stated by Turaev et al. [5]. The same result is obtained when probability is used as a restriction in sticker operation by Selvarajoo et al.[6]. Hence, by applying the probability concept on splicing systems and sticker systems, we can obtain new results as well as we can use them as molecular models for stochastic processes.

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 as shown by Turaev et al. and Selvarajoo et al. [5,6]. In this paper, we consider a probabilistic variant of simple splicing systems.
This paper explains some necessary definitions and results from the theories of formal languages, splicing systems, and introduction on probabilistic splicing system in sequel. Next, some definitions of probabilistic simple splicing systems are given. Moreover, some examples and important results concerning the computational power of the languages generated by probabilistic simple splicing system are established. The conclusion of this research is then discussed at the end of the paper.

PRELIMINARIES

In this section we recall some prerequisites by giving basic notions of the theories of formal languages and splicing systems which are used in sequel. The reader is referred to Turaev et al., Linz and Mateescu et al.\[5, 7, 8\] 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 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

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

**Definition 1** (Head [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.

**Theorem 1** (Head [2]) 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 $F_1, F_2$ such that $F_3 \subseteq EH(F_1, F_2) \subseteq F_4$

<table>
<thead>
<tr>
<th>$F_2$ \ $F_1$</th>
<th>FIN</th>
<th>REG</th>
<th>CF</th>
<th>LIN</th>
<th>CS</th>
<th>RE</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIN</td>
<td>REG</td>
<td>RE</td>
<td>RE</td>
<td>RE</td>
<td>RE</td>
<td>RE</td>
</tr>
<tr>
<td>REG</td>
<td>REG</td>
<td>RE</td>
<td>RE</td>
<td>RE</td>
<td>RE</td>
<td>RE</td>
</tr>
<tr>
<td>CF</td>
<td>LIN,CF</td>
<td>RE</td>
<td>RE</td>
<td>RE</td>
<td>RE</td>
<td>RE</td>
</tr>
<tr>
<td>LIN</td>
<td>CF</td>
<td>RE</td>
<td>RE</td>
<td>RE</td>
<td>RE</td>
<td>RE</td>
</tr>
<tr>
<td>CS</td>
<td>RE</td>
<td>RE</td>
<td>RE</td>
<td>RE</td>
<td>RE</td>
<td>RE</td>
</tr>
<tr>
<td>RE</td>
<td>RE</td>
<td>RE</td>
<td>RE</td>
<td>RE</td>
<td>RE</td>
<td>RE</td>
</tr>
</tbody>
</table>

We further define the concept of probabilistic splicing systems.

**Definition 2** (Turaev et al. [5]) A probabilistic splicing system $(pEH)$ is a 5-tuple $\gamma = (V, T, A, R, p)$ where $V, T, R$ are defined as for a usual extended H system, $p : V^+ \rightarrow [0, 1]$ is a probability function, and $A$ is a finite subset of $V^+ \times [0, 1]$ such that

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

**Definition 3** (Turaev et al. [5]) A Probabilistic Splicing Operation is defined as below:

For strings $(x, p(x)), (y, p(y)), (z, p(z)) \in V^+ \times [0, 1]$, and $r \in R$, we say that
Thus, the probability of the string \( z \in V^* \) obtained by splicing operation on two strings \( x, y \in V^* \) is computed by multiplying their probabilities.

**Definition 4** (Turaev et al. [5]) The language generated by probabilistic splicing system \( \gamma \) is defined as

\[
L_p(\gamma) = \left\{ z \in T^* \mid (z, p(z)) \in \sigma^*(A) \right\}.
\]

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 \([0,1]\) and \( \omega \in [0,1] \):

\[
L_p(\gamma, \ast \omega) = \left\{ z \in T^* \mid (z, p(z)) \in \sigma^*(A) \text{ and } p(z) \ast \omega \right\},
\]

\[
L_p(\gamma, \Diamond \omega) = \left\{ z \in T^* \mid (z, p(z)) \in \sigma^*(A) \text{ and } p(z) \Diamond \omega \right\},
\]

where \( \ast \in \{ =, \neq, \leq, >, <, \leq \} \) and \( \Diamond \in \{ \in, \notin \} \) are called threshold modes.

We denote the family of languages generated by multiplicative probabilistic splicing system of type \((F_1, F_2)\) by \( pEH(F_1, F_2) \) where 

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

We also use the simplified notation \( pEH(F) \) of the language family generated by probabilistic splicing systems with finite set of axioms instead of \( pEH(F_1, F_2) \), where \( F \in \{ \text{FIN, REG, CF, LIN, CS, RE} \} \) shows the family of languages for splicing rules.

**Definition 5** (Mateescu et al. [8]) 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 \((a, 1)\) or those of \( A \) are called axioms.

**RESULTS**

In this section we will discuss our results. First, we define probabilistic simple splicing system which is specified with probabilities assigned to each string generated by the splicing systems and the multiplication operation over the probabilities.

**Definition 6** A probabilistic simple splicing system is a 4-tuple

\[
\gamma = (V, M, A, p),
\]

where \( V \) is defined as for a usual extended \( H \) system, \( M \) is the rule in the form \((a, 1)\) for \( a \in A \) and \( p \) is a probabilistic function defined by \( p : V^* \rightarrow [0,1] \) and \( A \) is a finite subset of \( V^+ \times [0,1] \) such that

\[
\sum_{(s, p(x)) \in A} p(x) = 1.
\]

**Definition 7** A Probabilistic Simple Splicing Operation is defined as below:

For strings \((x, p(x)), (y, p(y)), (z, p(z)) \in V^* \times [0,1], \) and \( r \in M, \) we say that

\[
[(x, p(x)), (y, p(y))] \rightarrow_r (z, p(z))
\]
if and only if $(x, y) \rightarrow_r z$, $p(z) = p(x) \cdot p(y)$ and $r = (a, l; a, l) \in M$.

**Definition 8** The language generated by probabilistic simple splicing system $\gamma$ is defined as

$$L_\gamma(\gamma) = \left\{ z \in V^* | (z, p(z)) \in \sigma^*(A) \right\}.$$

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

**Lemma 1** $SEH(\text{FIN}, F) \subseteq pSEH(F)$ for all families $F \in \{\text{FIN, REG, CF, LIN, CS, RE}\}$.

**Proof.** Let $\gamma = (V, M, A)$, be a simple splicing system generating the language $L(\gamma) \in SEH(\text{FIN}, F)$ where $F \in \{\text{FIN, REG, CF, LIN, CS, RE}\}$.

Let $A = \{x_1, x_2, \ldots, x_n\}$, $n \geq 1$. We define a probabilistic simple splicing system $\gamma' = (V, M, A, p)$ where the set of axioms is defined by

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

where $p(x_i) = \frac{1}{m}$ for all $1 \leq i \leq n$, then $\sum_{i=1}^{n} p(x_i) = 1$.

We define the threshold language generated by $\gamma'$ as $L_\gamma(\gamma', > 0)$, then it is not difficult to see that

$$L(\gamma) = L_\gamma(\gamma', > 0) = pSEH(F).$$

Next, an example is given to illustrate the application of probability to the simple splicing system. Here, the symbol $\vdash$ denote the splicing operation on the strings.

**Example 1** Consider the semi-simple splicing system

$$\gamma_1 = \left\{(a, b, c), ((a, l; a, l), (b, l; b, l), (a, l; a, l), (ab, bc, abb, bce) \left\{ \begin{array}{c}
\frac{2}{28} \\
\frac{3}{28} \\
\frac{5}{28} \\
\frac{7}{28}
\end{array} \right\} \right\}.$$  

We obtain

$$L_\gamma(\gamma_1, \eta) = \left\{ a^n b^n c^n \left| \begin{array}{c}
\left( \begin{array}{c}
2 \cdot 3 \\
28^3
\end{array} \right) \\
\left( \begin{array}{c}
5 \cdot 7 \cdot 11 \\
28^3
\end{array} \right)^{n-1}
\end{array} \right\} n \geq 1, \right\}, \text{ where } \eta = \left\{ \begin{array}{c}
2 \cdot 3 \\
28^3
\end{array} \right\} \left( \begin{array}{c}
5 \cdot 7 \cdot 11 \\
28^3
\end{array} \right)^{n-1}.$$  

The way to obtain the string is by performing the splicing operation using the rules to the axioms.

Case 1 : Using strings $ab$ & $aab$ with rule $(a, l; a, l)$.

i : for the string $ab$, $p(ab) = \left\{ \begin{array}{c}
\frac{2}{28} \\
\left( \begin{array}{c}
2 \cdot 3 \\
28^3
\end{array} \right) \\
\left( \begin{array}{c}
5 \cdot 7 \cdot 11 \\
28^3
\end{array} \right)^{n-1}
\end{array} \right\}$

ii : for the string $aab$, $p(aab) = \left\{ \begin{array}{c}
\frac{5}{28} \\
\left( \begin{array}{c}
5 \cdot 7 \cdot 11 \\
28^3
\end{array} \right)^{n-1}
\end{array} \right\}$

iii : for the both strings $ab \left\{ p(ab) = \left( \begin{array}{c}
\frac{2}{28} \\
\frac{5}{28}
\end{array} \right) \right\}$ and $aab \left\{ p(aab) = \left( \begin{array}{c}
\frac{5}{28} \\
\frac{5}{28}
\end{array} \right) \right\}$.

\[ \left[ a\left( \frac{2}{28} \right), a\left( \frac{5}{28} \right) \right] \vdash \left[ (ab), \left( \begin{array}{c}
\frac{2}{28} \\
\frac{5}{28}
\end{array} \right) \right]. \]
iv : from the string produced in (iii) and string \(aab\), i.e.
\[
p(aab) = \left( \frac{5}{28} \right)
\]
using the same rule,
\[
\left[ \left( \frac{22}{28} \right), \left( \frac{5}{28} \right) \right]\left( \frac{22}{28} \right) = \frac{3}{28}
\]
and
\[
\left( \frac{3}{28} \right)
\]
and using the same rule,
\[
\left[ \left( \frac{22}{28} \right), \left( \frac{5}{28} \right) \right]\left( \frac{22}{28} \right) = \frac{3}{28}
\]

v : from the string produced in (iv) and string \(aab\), i.e.
\[
p(aaab) = \left( \frac{5}{28} \right)
\]
using the same rule,
\[
\left[ \left( \frac{22}{28} \right), \left( \frac{5}{28} \right) \right]\left( \frac{22}{28} \right) = \frac{3}{28}
\]
and
\[
\left( \frac{3}{28} \right)
\]
and using the same rule,
\[
\left[ \left( \frac{22}{28} \right), \left( \frac{5}{28} \right) \right]\left( \frac{22}{28} \right) = \frac{3}{28}
\]

vi : for each new string produced in (v), i.e.
\[
p(a^{k-1}b) = \left( \frac{5}{28}, \frac{3}{28} \right)
\]
and string (ii)
\[
p(a) = \left( \frac{5}{28} \right)
\]
and using the same rule,
\[
\left[ \left( \frac{22}{28} \right), \left( \frac{5}{28} \right) \right]\left( \frac{22}{28} \right) = \frac{3}{28}
\]
and
\[
\left( \frac{3}{28} \right)
\]
and using the same rule,
\[
\left[ \left( \frac{22}{28} \right), \left( \frac{5}{28} \right) \right]\left( \frac{22}{28} \right) = \frac{3}{28}
\]

Case 2 : Using strings \(ab & abb\) with rule \((b, 1; b, 1)\).

i : for the string \(ab\), \(p(ab) = \left( \frac{2}{28}, \frac{3}{28} \right)\)
\[
\left[ \left( \frac{22}{28} \right), \left( \frac{5}{28} \right) \right]\left( \frac{22}{28} \right) = \frac{3}{28}
\]

ii : for the string \(abb\), \(p(abb) = \left( \frac{7}{28}, \frac{3}{28} \right)\)
\[
\left[ \left( \frac{22}{28} \right), \left( \frac{5}{28} \right) \right]\left( \frac{22}{28} \right) = \frac{3}{28}
\]

iii : for the both strings \(ab\) \(p(ab) = \left( \frac{2}{28}, \frac{3}{28} \right)\) and \(abb\) \(p(abb) = \left( \frac{7}{28}, \frac{3}{28} \right)\)
\[
\left[ \left( \frac{22}{28} \right), \left( \frac{5}{28} \right) \right]\left( \frac{22}{28} \right) = \frac{3}{28}
\]

iv : from the string produced in (iii) and string \(abb\), i.e.
\[
p(abb) = \left( \frac{2}{28}, \frac{3}{28} \right)
\]
and
\[
\left[ \left( \frac{22}{28} \right), \left( \frac{5}{28} \right) \right]\left( \frac{22}{28} \right) = \frac{3}{28}
\]

v : from the string produced in (iv) and string \(abb\), i.e.
\[
p(abbb) = \left( \frac{2}{28}, \frac{3}{28} \right)
\]
and
\[
\left[ \left( \frac{22}{28} \right), \left( \frac{5}{28} \right) \right]\left( \frac{22}{28} \right) = \frac{3}{28}
\]

vi : for each new string produced in (v), i.e.
\[
p(a^{k-1}b) = \left( \frac{2}{28}, \frac{3}{28} \right)
\]
and using the same rule,
\[
\left[ \left( \frac{22}{28} \right), \left( \frac{5}{28} \right) \right]\left( \frac{22}{28} \right) = \frac{3}{28}
\]
and
\[
\left( \frac{3}{28} \right)
\]
and using the same rule,
\[
\left[ \left( \frac{22}{28} \right), \left( \frac{5}{28} \right) \right]\left( \frac{22}{28} \right) = \frac{3}{28}
\]

Case 3 : Using strings \(bc & bcc\) with rule \((c, 1; c, 1)\).

i : for the string \(bc\), \(p(bc) = \left( \frac{3}{28}, \frac{3}{28} \right)\)
\[
\left[ \left( \frac{22}{28} \right), \left( \frac{5}{28} \right) \right]\left( \frac{22}{28} \right) = \frac{3}{28}
\]
ii : for the string $bcc$, $p(bcc) = \left( \frac{11}{28} \right)$

iii : for the both strings $bc \left( p(bc) = \left( \frac{3}{28} \right) \right)$ and $bcc \left( p(bcc) = \left( \frac{11}{28} \right) \right)$,

\[ \left[ bc \left( \frac{3}{28} \right), bc \left( \frac{11}{28} \right) \right] \mapsto \left[ \text{bcc} \left( \frac{3}{28} \right), \text{bcc} \left( \frac{11}{28} \right) \right], \]

iv : from the string produced in (iii) and string $bcc$, i.e. $bcc \left( p(bcc) = \left( \frac{11}{28} \right) \right)$ and $bcc \left( p(bcc) = \left( \frac{3}{28} \right) \right)$, using the same rule,

\[ \left[ \text{bcc} \left( \frac{3}{28} \right), \text{bcc} \left( \frac{11}{28} \right) \right] \mapsto \left[ \text{bcc} \left( \frac{3}{28} \right), \text{bcc} \left( \frac{11}{28} \right) \right]. \]

v : from the string produced in (iv) and string $bcc$, i.e. $bcc \left( p(bcc) = \left( \frac{11}{28} \right) \right)$ and $bcc \left( p(bcc) = \left( \frac{3}{28} \right) \right)$, using the same rule,

\[ \left[ bc \left( \frac{3}{28} \right), bc \left( \frac{11}{28} \right) \right] \mapsto \left[ bc \left( \frac{3}{28} \right), bc \left( \frac{11}{28} \right) \right]. \]

vi : for each new string produced in (v), i.e. $bcc \left( p(bcc) = \left( \frac{11}{28} \right) \right)$ and string (ii)

\[ \left. \left[ \text{bcc} \left( \frac{3}{28} \right), \text{bcc} \left( \frac{11}{28} \right) \right] \right\} \mapsto \left[ \text{bcc} \left( \frac{3}{28} \right), \text{bcc} \left( \frac{11}{28} \right) \right]. \]

From the resulting string above $a^k b^n \left( \begin{array}{c} \left( \frac{2}{28} \right) \left( \frac{5}{28} \right)^{k-1} \\ \left( \frac{2}{28} \right) \left( \frac{5}{28} \right)^n \end{array} \right)$ and string from Case 3 $bc^n \left( p(bc^n) = \left( \frac{3}{28} \right) \left( \frac{11}{28} \right)^{n-1} \right)$ using rule $(b,1;b,1)$,

\[ \left[ a^k b^m \left( \frac{2}{28} \left( \frac{5}{28} \right)^{k-1} \right), a^m b^n \left( \frac{2}{28} \left( \frac{5}{28} \right)^n \right) \right] \mapsto \left[ a^k b^m \left( \frac{2}{28} \left( \frac{5}{28} \right)^{k-1} \right), a^m b^n \left( \frac{2}{28} \left( \frac{5}{28} \right)^n \right) \right]. \]

Using the threshold properties, we can conclude the following:

i : $\eta = 0 \Rightarrow L(\gamma_1) = \emptyset \in \text{REG}$,

ii : $\eta > 0 \Rightarrow L(\gamma_1) \in \text{REG}$,
The examples above illustrate that the use of thresholds with probabilistic simple splicing systems increases the generative power of splicing systems with finite components. We should also mention two simple but interesting facts of probabilistic simple splicing systems, as stated in the following:

**Proposition 1** For any probabilistic simple splicing system \( \gamma \), the threshold language \( L(\gamma, 0) \) is the empty set, i.e. \( L(\gamma, 0) = \emptyset \).

**Proposition 2** If for each splicing rule \( r \) in a probabilistic simple splicing system \( \gamma \), \( p(r) < 1 \), then every threshold language \( L(\gamma, \eta) \) with \( \eta > 0 \) is finite.

From Theorem 1, Lemma 1 and Example 1, we obtain the following two conjectures.

**Theorem 1** \( \text{REG} \subset pSEH(\text{FIN}) \subset pSEH(F) = \text{RE} \) where \( F \in \{\text{REG, CF, LIN, CS, RE} \} \).

**Theorem 2** \( pSEH(\text{FIN}) - \text{CF} \neq \emptyset \).

**CONCLUSIONS**

In this paper we studied probabilistic simple splicing systems and established some basic but important facts related to probabilistic simple splicing systems. We showed that a probability extension increases the generative power of simple splicing systems with finite components. In particular cases, probabilistic simple splicing systems can even generate non-context-free languages. The problem of strictness of the second inclusion in Theorem 1 and the incomparability of the family of context-free languages with the family of languages generated by probabilistic simple splicing systems with finite components in Theorem 2 remain open.

**ACKNOWLEDGMENTS**

The first author would like to thank the Malaysian Ministry of Education (MOE) for the financial funding support through MyBrain15 scholarship. The second and third authors would also like to thank the Malaysian Ministry of Education (MOE) and Research Management Center (RMC), UTM for their financial funding through Research University Fund Vote No. 07J41.

**REFERENCES**