Probabilistic Splicing Systems

Sherzod Turaev[1], Mathuri Selvarajoo[2], Mohd Hasan Selamat[3], Nor Haniza Sarmin[2,4], and Wan Heng Fong[4]

[1] Department of Computer Science
Kulliyah of Information and Communication Technology
International Islamic University Malaysia
50728 Kuala Lumpur, Malaysia
turaev@sherzod.info
[2] Department of Mathematical Sciences, Faculty of Science
Universiti Teknologi Malaysia
81310 UTM Johor Bahru, Johor, Malaysia
nhs@utm.my
[3] Faculty of Computer Science and Information Technology
Universiti Putra Malaysia
43400 UPM Serdang, Selangor, Malaysia
hasan@fsktm.upm.edu.my
[4] Ibnu Sina Institute for Fundamental Science Studies
Universiti Teknologi Malaysia
81310 UTM Johor Bahru, Johor, Malaysia
fwh@ibnusina.utm.my

Abstract. In this paper we introduce splicing systems with probabilities, i.e., *probabilistic splicing systems*, and establish basic properties of language families generated by this type of splicing systems. We show that a simple extension of splicing systems with probabilities may increase the computational power of splicing systems with finite components.

1 Introduction

DNA molecules are double stranded helicoidal structures composed of four nucleotides A (*adenine*), C (*cytosine*), G (*guanine*), and T (*thymine*), paired A-T, C-G according to the so-called *Watson-Crick complementary*. Watson-Crick complementary and *massive parallelism*, the other fundamental and distinctive feature of DNA molecules, are taken as the main characteristics of *DNA computing*.

Adleman's [1] famous biological experiment, which could solve Hamiltonian Path Problem using these two features, indeed gave a high hope for the future of DNA computing. Since there have been obtained a number exciting results showing the power of DNA computing, for instance, Lipton [2] showed that how to use DNA to solve the problem to find 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.

N.T. Nguyen et al. (Eds.): *Adv. Methods for Comput. Collective Intelligence*, SCI 457, pp. 259–268.
DOI: 10.1007/978-3-642-34300-1_25

One of the early theoretical proposals for DNA based computation was made by Head [4] who used the *splicing operation* – a formal model of the cutting and recombination of DNA molecules under the influence of restriction enzymes. 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 verse. This process can be formalized as an operation on strings, described by a so-called *splicing rule* of the form $(u_1, u_2; v_1, v_2)$ where u_1u_2 and v_1v_2 are the subsequences in question and the cuts are located between u_1 and u_2 as well as v_1 and v_2. These rules are the basis of a computational model (language generating device) called a *splicing system* or also *H system*. A system starts from a given set of strings (*axioms*) and produce a *language* by iterated splicing according to a given set of splicing rules.

Since splicing systems with finite sets of axioms and rules generate only regular languages (see [5]), several restrictions in the use of rules have been considered (see [6]), which increase the computational power up to the recursively enumerable languages. This is important from the point of view of DNA computing: splicing systems with restrictions can be considered as theoretical models of *universal programmable DNA based computers*.

Different problems appearing computer science and related areas motivates to consider suitable models for the solution of these problems. For instance, in order to develop accurate tools for natural and programming language processing, the probabilistic models have been widely used. In fact, adding probabilities to grammars allows eliminating ambiguity and leads 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) started in the 1960s (for instance, see [7–11]). The recent results on probabilistic grammars and automata can be found, for instance, in [12–14].

In general, the probability of a generated (accepted) string is computed by the multiplication of the probabilities of those rules (transitions) which participated in the derivation (acceptance) of the string (though, in [11], the computation of probabilities is defined slightly different). Different threshold probabilistic languages can also be defined by using different thresholds (numbers, sets, etc.) and their modes (order relations, a membership to a threshold set, etc.)

The interesting and natural fact that the probabilistic concepts in formal language and automata theories can also be adapted in DNA computing theory, i.e., we can define probabilistic splicing and sticker systems as well as probabilistic Watson-Crick automata.

In this paper we introduce *probabilistic splicing systems*. In such 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. In order to overcome the ambiguity (the same string may have different probabilities), we can use another operation such as addition. We also define different threshold probabilistic languages using as a threshold

segments, sets, real numbers and as a mode the order and equality relations, a membership to a threshold set.

This paper is organized as follows. Section 2 contains some necessary definitions and results from the theories of formal languages and splicing systems. Section 3 introduces the concept of a probabilistic splicing systems and threshold probabilistic languages, explains the specific features of probabilistic splicing systems in two examples and establishes some basic results concerning to the generative power of probabilistic splicing systems. It shows that probabilistic splicing systems with finite components generate not only regular languages but also context-free and context-sensitive languages. Section 4 discusses the obtained 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 formal languages, and splicing systems, which are used in sequel. The reader is referred to [6, 15, 16] for 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 $\subset$. $\emptyset$ denotes the empty set. 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**, **FIN**, respectively. For these language families, the next strict inclusions, named *Chomsky hierarchy*, hold

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

Further, we recall some basic notations and notations of (iterative) splicing systems.

Let V be an alphabet, and $\#, \$ \notin V$ two special symbols. A *splicing rule* over V is a string of the form

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

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

$$(x, y) \vdash_r z \text{ iff } x = x_1 u_1 u_2 x_2,\ y = y_1 u_3 u_4 y_2,$$
$$\text{and } z = x_1 u_1 u_4 y_2,$$

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

We say that z is obtained by splicing x, y, as indicated by the rule r; $u_1 u_2$ and $u_3 u_4$ are called the *sites* of the splicing. We call x the *first term* and y the

second term of the splicing operation. When understood from the context, we omit the specification of r and we write $\vdash$ instead of $\vdash_r$.

An *H 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 define

$$\begin{aligned}
\sigma(L) =& \{z \in V^* \mid (x, y) \vdash_r z, \\
& \qquad \text{for some } x, y \in L, r \in R\}, \\
\sigma^0(L) =& L, \\
\sigma^{i+1}(L) =& \sigma^i(L) \cup \sigma(\sigma^i(L)), i \geq 0, \\
\sigma^*(L) =& \bigcup_{i \geq 0} \sigma^i(L).
\end{aligned}$$

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*. When $T = V$, the system is said to be *non-extended*. The language generated by γ is defined by

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

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

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

Theorem 1 ([6]). *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

In this section we define probabilistic splicing systems which is specified with probabilities assigned to each string generated by the splicing system and the multiplication operation over the probabilities. Moreover, we define threshold probabilistic splicing languages and show that probabilistic splicing systems with finite components can also generate context-free and context-sensitive languages.

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

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

We define a probabilistic splicing operation as follows:

Definition 3. *For strings* $(x, p(x)), (y, p(y)), (z, p(z)) \in V^* \times [0, 1]$ *and* $r \in R$,

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

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

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. *Then the language generated by the splicing system* γ *is defined as*

$$L_p(\gamma) = \{z \in T^* \mid (z, p(z)) \in \sigma^*(A)\}.$$

Remark 5. We should mention that splicing operations may result in the same string with different probabilities. Since, in this paper, we focus on strings whose probabilities satisfy some *threshold* requirements, i.e., the probabilities are merely used for the selection of some strings, this "ambiguity" does not effect on the selection. When we investigate the properties connected with the probabilities of the strings, we can define another operation together with the multiplication, for instance, the addition over the probabilities of the same strings, which removes the ambiguity problem.

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) subsegments 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 $\Omega \subseteq [0, 1]$ and $\omega \in [0, 1]$:

$$\begin{aligned} L_p(\gamma, *\omega) &= \{z \in T^* \mid (z, p(z)) \in \sigma^*(A) \wedge p(z) * \omega\}, \\ L_p(\gamma, \star\Omega) &= \{z \in T^* \mid (z, p(z)) \in \sigma^*(A) \wedge p(z) \star \Omega\}, \end{aligned}$$

where $* \in \{=, \geq, >, \leq, <\}$ and $\star \in \{\in, \notin\}$ are called *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

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

Remark 6. In this paper we focus on probabilistic splicing systems with the finite set of axioms, since we consider a finite initial distribution of probabilities over the set of axioms. Moreover, it is natural in practical point of view: only splicing

systems with finite components can be chosen as a theoretical models for DNA based computation devices. Thus, we use the simplified notation $p\mathbf{EH}(F)$ of the language family generated by probabilistic splicing systems with finite set of axioms instead of $p\mathbf{EH}(F_1, F_2)$, where $F \in \{\mathbf{FIN}, \mathbf{REG}, \mathbf{CF}, \mathbf{LIN}, \mathbf{CS}, \mathbf{RE}\}$ shows the family of languages for splicing rules.

From the definition the next lemma follows immediately.

Lemma 7

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

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 an extended splicing system generating the language $L(\gamma) \in \mathbf{EH}(\mathbf{FIN}, F)$ where $F \in \{\mathbf{FIN}, \mathbf{REG}, \mathbf{CF}, \mathbf{LIN}, \mathbf{CS}, \mathbf{RE}\}$.

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

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

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 8. Let us consider the system

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

where

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

It is not difficult 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 $m \geq 1$,

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

and

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

From the strings $ca^k d$, $k \geq 1$, and $db^m c$, $m \geq 1$, by the rule r_3,

$$[(ca^k d, (2/7)^k), (db^m c, (5/7)^m)] \vdash_{r_3} (ca^k b^m c, (2/7)^k (5/7)^m).$$

Thus,

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

We consider the threshold languages generated by this probabilistic splicing systems with different thresholds and modes:

(1) $L_p(\gamma_1, = 0) = \emptyset$;

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

(3) $L_p(\gamma_1, > \nu^i) = \{ca^kb^mc \mid 1 \leq k, m \leq i\}$;

(4) $L_p(\gamma_1, \in \{\nu^n \mid n \geq 1\}) = \{ca^nb^nc \mid n \geq 1\}$;

(5) $L_p(\gamma_1, \notin \{\nu^n \mid n \geq 1\}) = \{ca^kb^mc \mid k > m \geq 1\} \cup \{ca^kb^mc \mid m > k \geq 1\}$

where $\nu = 10/49$, $i \geq 1$ is a fixed integer, and γ_1' is the "crisp" variant of the splicing system γ_1, i.e., γ_1 without probabilities. We can see that the second language is **regular**, the third language is **finite**, the fourth and fifth languages are not regular but **context-free**. □

Example 9. Consider the probabilistic splicing system

$$\gamma_2 = (\{a, b, c, w, x, y, z\}, \{a, b, c, w, z\}, A_2, R_2, p_2)$$

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

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

Using the first axiom and rule r_1, we obtain strings

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

the second axiom and rule r_2,

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

the third axiom and rule r_3,

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

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

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

and

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

Then the language generated by the probabilistic splicing system γ_2

$$L_p(\gamma_2) = \{(wa^k b^m c^n z, \tau_1^k \tau_2^m \tau_3^n) \mid k, m, n \geq 1\}$$

where $\tau_1 = 3/19$, $\tau_2 = 5/19$ and $\tau_3 = 11/19$.

Further, we consider the following threshold languages:

$$L_p(\gamma_2, > 0) = L(\gamma_2') \in \mathbf{REG}$$

where γ_2' is the "crisp" variant of the splicing system γ_2.

$$L_p(\gamma_2, > \tau^i) = \{wa^k b^m c^n z \mid 1 \leq k, m, n \leq i\} \in \mathbf{FIN}$$

where $\tau = 165/6859$, and $i \geq 1$ is a fixed positive integer.

Now, let $\Omega = \{(165/6859)^n \mid n \geq 1\}$, then

$$L_p(\gamma_2, \in \Omega) = \{wa^n b^n c^n z \mid n \geq 1\} \in \mathbf{CS} - \mathbf{CF}$$

and

$$L_p(\gamma_2, \notin \Omega) = \{wa^k b^m c^n z \mid k, m, n \geq 1 \wedge k \neq m, m \neq n, k \neq n\} \in \mathbf{CS} - \mathbf{CF}. \quad \square$$

The examples above illustrate that the use of thresholds with probabilistic splicing systems increase 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 and second as Proposition 4, stated in the following.

Proposition 10. *For any probabilistic splicing system γ, the threshold language $L_p(\gamma, = 0)$ is the empty set, i.e., $L_p(\gamma, = 0) = \emptyset$.*

Proposition 11. *If for each splicing rule r in a probabilistic splicing system γ, $p(r) < 1$, then every threshold language $L_p(\gamma, > \eta)$ with $\eta > 0$ is finite.*

From Theorem 1, Lemma 7 and Examples 8,9, we obtain the following two theorems.

Theorem 12

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

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

Theorem 13

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

4 Conclusions

In this paper we introduced probabilistic splicing systems by associating probabilities with strings and also established some basic but important facts. We showed that an extension of splicing systems with probabilities increases the generative power of splicing systems with finite components, in particular cases, probabilistic splicing systems can generate non-context-free languages.

The problem of strictness of the second inclusion in Theorem 12 and the incomparability of the family of context-free languages with the family of languages generated by probabilistic splicing systems with finite components (the inverse inequality of that in Theorem 13) remain open.

We should mention the possible and interesting topics for the study in probabilistic DNA computing in general and in probabilistic splicing systems in particular:

- Since extended splicing systems with finite set of splicing rules can generate languages in different language families in Chomsky hierarchy (see Theorem 1), it would be interesting to consider splicing systems with an infinite probability distribution over the set of axioms, and investigate the properties of the generated languages.
- It is also interesting to define probabilistic sticker systems and probabilistic Watson-Crick automata with different thresholds and modes.
- Another interesting topic in this direction is to consider splicing systems extended with fuzzy characteristic functions and weights, which can also be defined in the same way as probabilistic splicing systems.

Acknowledgements. This work has been supported by Ministry of Higher Education Fundamental Research Grant Scheme FRGS /1/11/SG/UPM/01/1 and University Putra Malaysia via RUGS 05-01-10-0896RU/F1.

References

1. Adleman, L.: Molecular computation of solutions to combinatorial problems. Science, 1021–1024 (November 11, 1994)
2. Lipton, R.J.: Using DNA to solve NP–complete problems. Science, 542–545 (April 28, 1995)
3. Boneh, D., Dunworth, C., Lipton, R., Sgall, J.: On the computational power of DNA. Discrete Applied Mathematics, Special Issue on Computational Molecular Biology, 79–94 (1996)
4. Head, T.: Formal language theory and DNA: An analysis of the generative capacity of specific recombination behaviors. Bull. Math. Biology, 737–759 (1987)
5. Pixton, D.: Regularity of splicing languages. Discrete Applied Mathematics, 101–124 (1996)
6. Păun, G., Rozenberg, G., Salomaa, A.: DNA computing. New computing paradigms. Springer (1998)
7. Booth, T., Thompson, R.: Applying probability measures to abstract languages. IEEE Transactions on Computers 22, 442–450 (1973)

8. Ellis, C.A.: Probabilistic Languages and Automata. PhD thesis, Department of Computer Science, University of Illinois (1969)
9. Fu, K.S., Li, T.: On stochastic automata and languages. Inter. J. Information Sci. (1969)
10. Rabin, M.: Probabilistic automata. Information and Control 6, 230–245 (1963)
11. Salomaa, A.: Probabilistic and weighted grammars. Information and Control 15, 529–544 (1969)
12. Smith, N., Johnson, M.: Weighted and probabilistic context-free grammars are equally expressive. IEEE Transactions on Computers 33, 477–491 (2007)
13. Kornai, A.: Probabilistic grammars and languages. Journal of Logic, Language and Information 20, 317–328 (2011)
14. Fowler, T.: The Generative Power of Probabilistic and Weighted Context-Free Grammars. In: Kanazawa, M., Kornai, A., Kracht, M., Seki, H. (eds.) MOL 12. LNCS, vol. 6878, pp. 57–71. Springer, Heidelberg (2011)
15. Rozenberg, G., Salomaa, A.: Handbook of formal languages, vol. 1-3. Springer (1997)
16. Dassow, J., Păun, G.: Regulated rewriting in formal language theory. Springer, Berlin (1989)