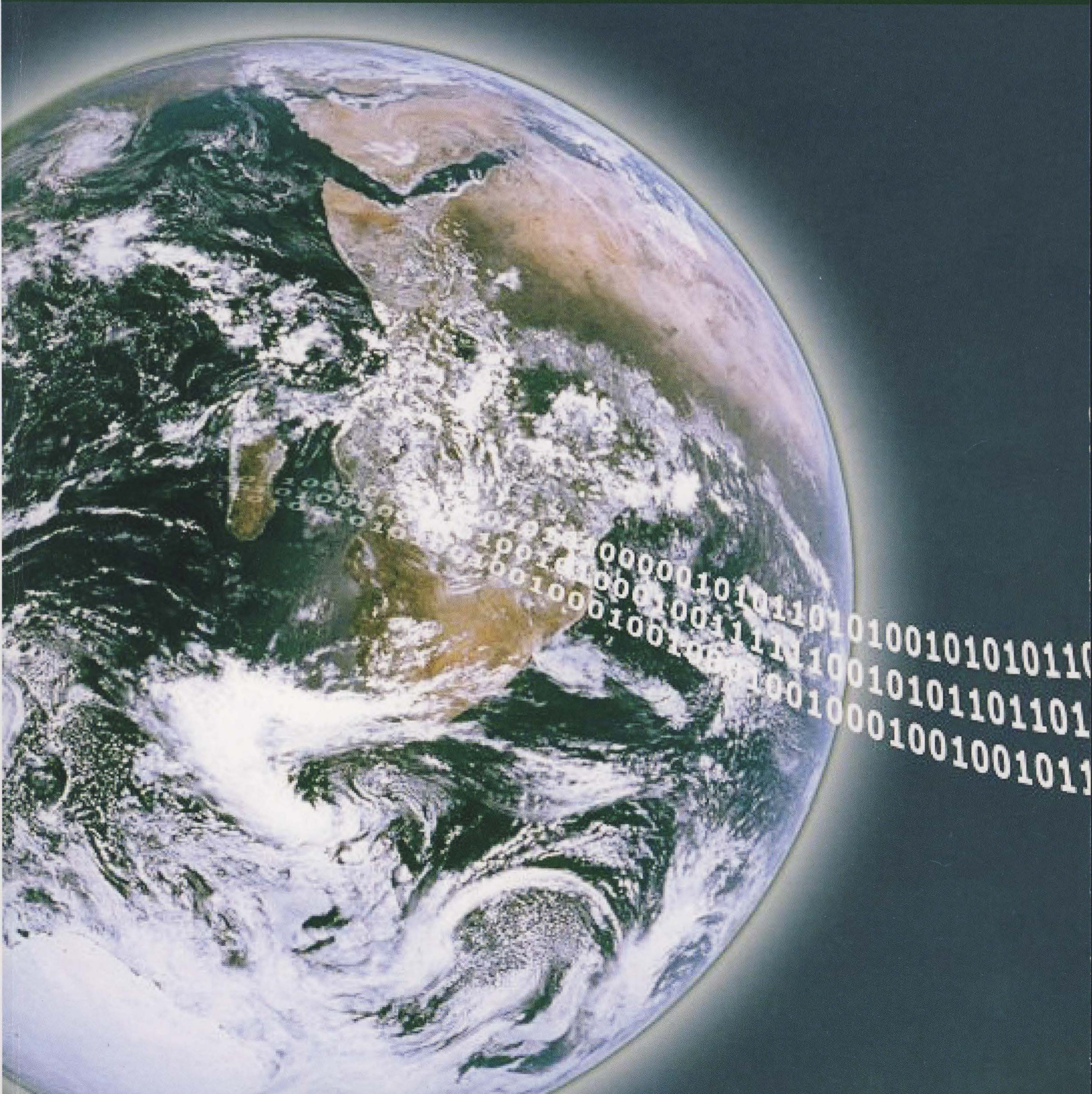
ISSN 1882-5796

The 4th South East Asian Technical University Consortium (SEATUC) Symposium



BIO-MATHEMATICAL CONCEPTS IN DNA SPLICING SYSTEM

Fong Wan Heng[1], Nor Haniza Sarmin[2] and Yuhani Yusof[3]

[1]Ibnu Sina Institute for Fundamental Science Studies

[2, 3]Department of Mathematics, Faculty of Science

Universiti Teknologi Malaysia, 81310 UTM Skudai, Johor, Malaysia

ABSTRACT

Splicing system is a formal characterization of the generative capacity of specified enzymatic activities operating on specified set of double-stranded DNA molecules. The formalism of this splicing system is illustrated under the framework of Formal Language Theory which is a branch of applied discrete mathematics and theoretical computer science. The mathematical formalism that will be used in this research is the splicing system concept that was first initiated by Head in 1987. In this research, some new bio-mathematical concepts involved in the DNA splicing system will be presented. These concepts include factor of a language and constant of a language. The regularity of these concepts will also be presented with their proofs.

1. INTRODUCTION

Splicing system was first introduced by Head in 1987. A splicing system comprises of the set A of alphabets in Deoxyribonucleic Acid (DNA), set I of initial strings, and sets B and C of triples, also known as the rules. The set of alphabets include Adenine (A), Guanine (G), Cytosine (C) and Thymine (T), which are the bases of DNA in their nucleotide chain. When restriction enzymes are added to DNA molecules, these molecules will be cleaved at certain restriction site of the molecule. After the cleavage, the resulting fragments can be adjoined together to form new strings of DNA molecules. This process is also called the ligating process. The language which results from a splicing system is called a splicing language.

The relation between DNA splicing system and Formal Language Theory can be shown through a mathematical modeling of splicing system. DNA molecules are modeled as strings in Formal Language Theory, while enzymatic operations are modeled as a set of splicing rules. The recombinant behavior of DNA is modeled as a language (normally denoted as L) in a splicing system.

Splicing languages are regular, but not all regular languages are splicing languages (Gatterdam, 1989). In the following two sections, the factor of a language, constant of a language and their regularity will be discussed using some concepts of automaton.

2. FACTOR OF LANGUAGE

Let A be a finite set to serve as the alphabet. L is a regular language and $Fac(L)$ is the set of all factors of L. The definition of $Fac(L)$ is given in the following.

Definition 2.1: ***Fac*(*L*)**
$Fac(L) = \{x \text{ in } A^*: uxv \text{ in } L \text{ for some } u, v \text{ in } A^*\}$.

Suppose that $L(M')$ is the language recognized by the automaton M'. We can show that $L(M') = Fac(L)$ as given in the following theorem.

Theorem 2.1
$L(M') = Fac(L)$.

Proof.

First, let $w \in L(M')$. There are $p \in P$ and $q \in Q$ for which pwq is a recognition path in M'. Since p is accessible in M and q is coaccessible in M, there are states $i \in I$ and $t \in T$, u, $v \in A^*$ for which $upwqv$ is a labeled path from state i to state t. Thus $uwv \in L(M) = L$. Hence $w \in Fac(L)$.

Next, let $w \in Fac(L)$. Then there exist u, v in A^* such that $uwv \in L = L(M)$. So, there are $i \in I$, $t \in T$, for which uwv is a labeled path from state i to state t. Let p, $q \in Q$ for which $upwqv$ is a labeled path from state i to state t. Then pwq is a recognition path in M'. So $w \in L(M')$. □

Since the next few theorems will involve the concept of regularity, the definition of a regular language is given in the following.

Definition 2.2 (Linz, 2001): Regular
A language L is called **regular** if and only if there exist a deterministic finite accepter M such that $L = L(M)$.

Regular expressions are constructed from primitive constituents by repeatedly applying certain recursive rules. The definitions of primitive regular expressions and regular expressions are given next.

Definition 2.3 (Linz, 2001): Primitive Regular Expressions, Regular Expressions
Let Σ be a given alphabet. Then
(1) ϕ, λ and $a \in \Sigma$ are all regular expressions. These are called primitive regular expressions.
(2) If r_1 and r_2 are regular expressions, so are r_1+r_2, $r_1 \bullet r_2$, r_1^* and $(r_1)(r_2)$.
(3) A string is a regular expression if and only if it can be derived from the primitive regular expressions by a finite number of applications of the rules in (2).

For the definition of regular language, deterministic finite accepter is used instead of deterministic finite automaton since deterministic finite automaton (Kelly, 1995) is also known as deterministic finite accepter in (Linz, 2001).

The following theorem is on the regularity of $Fac(L)$ given that L is a regular language.

Theorem 2.2
If L is regular language, then $Fac(L)$ is regular.

Proof.
Since L is regular, let L be recognized by the trimmed automaton $M = (Q, I, T)$, where Q is the set of states, I is the set of initial states, and T is the set of terminal states. So $L(M) = L$. Let M' be the automaton (Q, Q, Q) where all states are both initial and final. By Theorem 2.1, $L(M') = Fac(L)$. So, $Fac(L)$ is recognized by the automaton M'. Thus $Fac(L)$ is regular. □

However, the converse of Theorem 2.2 is not true as shown using a counterexample in Theorem 2.3.

Theorem 2.3
If Fac (L) is regular, then L is not necessarily regular.

Proof.
Let $A = \{a,b\}$. $L = \{w$ in A^*: $|w|_a = |w|_b\}$ is not regular, but $Fac(L) = A^*$ is regular. For any $w \in A^* = Fac(L)$, let $|w|_a \geq |w|_b$, then $wb...b \in L$, where the number of bs is equal to $|w|_a - |w|_b$. Thus $|wb...b|_a = |wb...b|_b$, $w \in L$ but L is not regular. Therefore, $Fac(L)$ is regular does not imply that L is regular. □

In the next section, some characteristics regarding constant of a language are given.

3. CONSTANT OF A LANGUAGE

Let $Con(L)$ be the set of all constant factors of L. The definition of $Con(L)$ is given as follows.

Definition 3.1 $Con(L)$
$Con(L) = \{x$ in $Fac(L)$: pxq and uxv in L imply that pxv and uxq are also in $L\}$.

We can show that any word containing an element in $Con(L)$ is also in $Con(L)$ as given in the following theorem.

Theorem 3.1
If $c \in Con(L)$, then $w = xcy \in Con(L)$.

Proof.
If $pwq \in L$ and $uwv \in L$, then $pwq = pcxyq \in L$ and $uwv = uxcyv \in L$. Since $c \in Con(L)$, $pxcyv = pwv \in L$ and $uxcyq = uwq \in L$. Thus, $w \in Con(L)$. □

The language recognized by the automaton P' are the words which are not constant relative to the language L. This is shown in the following theorem.

Theorem 3.2
$L = (P') = \{w \in A^*$: w is not constant relative to $L\}$.

Proof.
First, we show $L(P') \subseteq = \{w \in A^*$: w is not constant relative to $L\}$. Let $w \in L$ (P'), $(p_1, p_2) \xrightarrow{w} (q_1, q_2)$ in P', $q_1 \neq q_2, p_1 \xrightarrow{w} q_1, p_2 \xrightarrow{w} q_2 \in M$. Since $q_1 \neq q_2$ $q_1 \xrightarrow{s} r_1, q_2 \xrightarrow{s} r_2$ where exactly one of $\{r_1, r_2\}$ is terminal and one is not. Note that there are u, v in A^* for which $i \xrightarrow{u} p_1, i \xrightarrow{v} p_2$. Without loss of generality, let r_1 be terminal and let r_2 be not terminal. Since q is coaccessible then there is a word $t \in A^*$ for which $q_2 \xrightarrow{t} r_3$ with r_3 terminal. Thus $i \xrightarrow{uws} r_1, i \xrightarrow{vwt} r_3$. So uws, vwt $\in L$. If w were a constant, then vws would lie in L, which would contradict r_2 being not terminal. Thus, w is not constant relative to L.

Next, we show $\{w \in A^*$: w is not constant relative to $L\}$ $\subseteq$ $L(P')$. Let w be not constant. Then there are $u_1 w u_2 \in L$ and $v_1 w v_2 \in L$ which are both accepted by $L(M)$ for which either $u_1 w v_2 \notin L$ or $v_1 w u_2 \notin L$. Thus there exist $iu_1 p_1 w q_1 u_2 t_1, iv_1 p_2 w q_2 v_2 t_2$.

Case 1: If $u_1 w v_2 \notin L$, there exist $iu_1 p_1 w q_1 v_2 r, r \neq t_2$ that is, r is not terminal. From $iv_1 p_2 w q_2 v_2 t_2$ ar $iu_1 p_1 w q_1 v_2 r, q_1 \neq q_2$. Therefore in P', there exist a pa $(p_1, p_2) w (q_1, q_2)$, where (p_1, p_2) is an initial state of state t. Then pwq is a recognition path in M'. So $w \in L(M')$. □

Since the next few theorems will involve the concept of regularity, the definition of a regular language is given in the following.

Definition 2.2 (Linz, 2001): Regular
A language L is called **regular** if and only if there exist a deterministic finite accepter M such that $L = L(M)$.

Regular expressions are constructed from primitive constituents by repeatedly applying certain recursive rules. The definitions of primitive regular expressions and regular expressions are given next.

Definition 2.3 (Linz, 2001): Primitive Regular Expressions, Regular Expressions
Let Σ be a given alphabet. Then
(1) ϕ, λ and $a \in \Sigma$ are all regular expressions. These are called primitive regular expressions.
(2) If r_1 and r_2 are regular expressions, so are r_1+r_2, $r_1 \bullet r_2$, r_1^* and $(r_1)(r_2)$.
(3) A string is a regular expression if and only if it can be derived from the primitive regular expressions by a finite number of applications of the rules in (2).

For the definition of regular language, deterministic finite accepter is used instead of deterministic finite automaton since deterministic finite automaton (Kelly, 1995) is also known as deterministic finite accepter in (Linz, 2001).

The following theorem is on the regularity of $Fac(L)$ given that L is a regular language.

Theorem 2.2
If L is regular language, then $Fac(L)$ is regular.

Proof.
Since L is regular, let L be recognized by the trimmed automaton $M = (Q, I, T)$, where Q is the set of states, I is the set of initial states, and T is the set of terminal states. So $L(M) = L$. Let M' be the automaton (Q, Q, Q) where all states are both initial and final. By Theorem 2.1, $L(M') = Fac(L)$. So, $Fac(L)$ is recognized by the automaton M'. Thus $Fac(L)$ is regular. □

However, the converse of Theorem 2.2 is not true as shown using a counterexample in Theorem 2.3.

Theorem 2.3
If $Fac(L)$ is regular, then L is not necessarily regular.

Proof.
Let $A = \{a,b\}$. $L = \{w$ in A^*: $|w|_a = |w|_b\}$ is not regular, but $Fac(L) = A^*$ is regular. For any $w \in A^* = Fac(L)$, let $|w|_a \geq |w|_b$, then $wb...b \in L$, where the number of bs is equal to $|w|_a - |w|_b$. Thus $|wb...b|_a = |wb...b|_b$, $w \in L$ but L is not regular. Therefore, $Fac(L)$ is regular does not imply that L is regular. □

In the next section, some characteristics regardin[g] constant of a language are given.

3. CONSTANT OF A LANGUAGE

Let $Con(L)$ be the set of all constant factors of L. The definition of $Con(L)$ is given as follows.

Definition 3.1 $Con(L)$
$Con(L) = \{x$ in $Fac(L)$: pxq and uxv in L imply that $p[xv]$ and uxq are also in $L\}$.

We can show that any word containing an element i[n] $Con(L)$ is also in $Con(L)$ as given in the following theorem.

Theorem 3.1
If $c \in Con(L)$, then $w = xcy \in Con(L)$.

Proof.
If $pwq \in L$ and $uwv \in L$, then $pwq = pcxyq \in L$ and $uwv = uxcyv \in L$. Since $c \in Con(L)$, $pxcyv = pwv \in L$ and $uxcyq = uwq \in L$. Thus, $w \in Con(L)$. □

The language recognized by the automaton P' are the words which are not constant relative to the language L. This is shown in the following theorem.

Theorem 3.2
$L = (P') = \{w \in A^*$: w is not constant relative to $L\}$.

Proof.
First, we show $L(P') \subseteq = \{w \in A^*$: w is not constan[t] relative to $L\}$. Let $w \in L(P')$, $(p_1, p_2) \xrightarrow{w} (q_1, q_2)$ in P', $q_1 \neq q_2, p_1 \xrightarrow{w} q_1, p_2 \xrightarrow{w} q_2 \in M$. Since $q_1 \neq q_2$, $q_1 \xrightarrow{s} r_1, q_2 \xrightarrow{s} r_2$ where exactly one of $\{r_1, r_2\}$ is terminal and one is not. Note that there are u, v in A^* for which $i \xrightarrow{u} p_1, i \xrightarrow{v} p_2$. Without loss of generality, let r_1 be terminal and let r_2 be not terminal. Since q is coaccessible then there is a word $t \in A^*$ for which $q_2 \xrightarrow{t} r_3$ with r_3 terminal. Thus $i \xrightarrow{uws} r_1, i \xrightarrow{vwt} r_3$. So uws, $vwt \in L$. If w were a constant, then vws would lie in L, which would contradict r_2 being not terminal. Thus, w is not constant relative to L.

Next, we show $\{w \in A^*$: w is not constant relative to $L\} \subseteq L(P')$. Let w be not constant. Then there are $u_1 w u_2 \in L$ and $v_1 w v_2 \in L$ which are both accepted by $L(M)$ for which either $u_1 w v_2 \notin L$ or $v_1 w u_2 \notin L$. Thus, there exist $iu_1 p_1 w q_1 u_2 t_1, iv_1 p_2 w q_2 v_2 t_2$.

Case 1: If $u_1 w v_2 \notin L$, there exist $iu_1 p_1 w q_1 v_2 r, r \neq t_2$, that is, r is not terminal. From $iv_1 p_2 w q_2 v_2 t_2$ and $iu_1 p_1 w q_1 v_2 r, q_1 \neq q_2$. Therefore in P', there exist a path $(p_1, p_2) w (q_1, q_2)$, where (p_1, p_2) is an initial state of P'

and (q_1, q_2) is a terminal state of P'. Thus, $w \in L(P')$.

Case 2: If $v_1 w u_2 \notin L$, there exist $i v_1 p_2 w q_2 u_2 r', r' \neq t_1$, that is, r' is not terminal. From $i u_1 p_1 w q_1 u_2 t_1$ and $i v_1 p_2 w q_2 u_2 r', q_1 \neq q_2$. Therefore in P', there exist a path $(p_1, p_2) w (q_1, q_2)$, where (p_1, p_2) is an initial state of P' and (q_1, q_2) is a terminal state of P'. Thus, $w \in L(P')$. Therefore, $w \in L(P')$ in both cases. □

Theorem 3.3 states that *Con*(*L*) is a necessary condition for *L* being regular.

Theorem 3.3
If *L* is regular language, then *Con*(*L*) is regular.

Proof.
Let *M* be a minimal intrinsic deterministic automaton recognizing *L* where $M = (Q, I, T, E)$. Therefore $L(M) = L$. Construct $P = M \times M = (Q \times Q, I \times I, T \times T, E')$ where $E' \subseteq (Q \times Q) \times A \times (Q \times Q)$, $(p_1, p_2) \xrightarrow{a} (q_1, q_2)$ is in E' where $(p_1, a, q_1) \in E$ and $(p_2, a, q_2) \in E$. Define $P' = (Q \times Q, Q \times Q, \{(p,q) \in Q \times Q : p \neq q\}, E')$. From Theorem 3.2, $L(P') = \{w \in A^* : w$ is not constant relative to $L\}$. Thus the set of non-constants, $A^* \setminus Con(L)$ is regular. Consequently *Con*(*L*) is regular. □

However, *Con*(*L*) is not a sufficient condition for *L* being regular as shown using a counterexample in Theorem 3.4.

Theorem 3.4
If *Con*(*L*) is regular, then *L* is not necessarily regular.

Proof.
Let $A = \{a, b\}$. The set $E = \{w \text{ in } A^* : |w|_a = |w|_b\}$ is not regular. Let $w \in Con(L)$, $uwv \in E$ and $pwq \in E$. Let $w = ab$, $u = aa$, $v = bb$, $p = abab$, $q = abab$. From $uwv \in E$ and $pwq \in E$, that is *aaabbb* and *ababababab*, $uwq = aaababab \notin E$. Thus $ab \notin Con(L)$ and in fact, $Con(L) = \phi$.

In general, we can show that for $\overbrace{a \ldots a}^{|w|_b} w \overbrace{b \ldots b}^{|w|_a} \in E$, the number of *a*'s $= |w|_b + |w|_a + 0$ and the number of *b*'s $= 0 + |w|_b + |w|_a$.

For $\overbrace{b \ldots b}^{|w|_a} w \overbrace{a \ldots a}^{|w|_b} \in E$, the number of *a*'s $= 0 + |w|_a + |w|_b$ and the number of *b*'s $= |w|_a + |w|_b + 0$. If $w \in A^*, w \in Con(L), a \ldots a w a \ldots a \notin E$ since the number of *a*'s $= |w|_b + |w|_a + |w|_b$ and the number of *b*'s $= 0 + |w|_b + 0$, which has too many *a*'s. Similarly, $b \ldots b w b \ldots b \notin E$ since the number of *a*'s $= 0 + |w|_a + 0$ and the number of *b*'s $= |w|_a + |w|_b + |w|_a$ which has too many *b*'s. Therefore $Con(L)) = \phi$. □

CONCLUSION

In this research, some new bio-mathematical concepts in DNA splicing system are introduced. These include factor of a language and constant of a language. Since all splicing languages are regular, but not all regular languages are splicing languages, the regularity of the new concepts are discussed through theorems and their proofs.

ACKNOWLEDGEMENT

The authors would like to thank Ibnu Sina Institute for the financial funding to this symposium. The authors would also like to thank the Research Management Centre, Universiti Teknologi Malaysia, Skudai, Johor, Malaysia for the financial funding through Vote No. 77924 and 78482. The third author would also like to acknowledge Universiti Malaysia PAHANG for the study leave and Ministry of Higher Education for SLAI scholarship.

REFERENCES

Gatterdam, R. W., Splicing Systems and Regularity, *International Journal of Computer Math*., vol. 31, pp. 63-67, 1989.

Head, T., Formal Language Theory and DNA: An Analysis of the Generative Capacity of Specific Recombinant Behaviors, *Bulletin of Mathematical Biology*, vol. 49, pp. 737-759, 1987.

Kelly, D., *Automata and Formal Languages*, Prentice-Hall Inc., USA, 1995.

Linz, P., *An Introduction to Formal Languages and Automata*, Jones and Bartlett Publishers Inc., USA, 2001.

NOMENCLATURE

A^*	strings obtained by concatenating zero or more symbols from *A*
I	set of initial strings
B, *C*	rules in splicing system
Con(*L*)	the set of all constant factors of *L*
Fac(*L*)	the set of all factors of *L*
L	language
L(*M*)	language recognized by an automaton *M'*
$\lvert w\rvert_a$	number of *a*s in *w*
$p \to q$	state *p* to state *q*

Fong Wan Heng received the B.Sc. (Industrial Mathematics) in 2003, M.Sc. (2004) and Ph.D (2008) in Mathematics from Universiti Teknologi Malaysia.

She is a lecturer at Ibnu Sina Institute for Fundamental Science Studies, Universiti Teknologi Malaysia, Skudai, Johor. Her current interests include Group Theory and DNA Splicing System.

Nor Haniza Sarmin received the B.Sc. (1989), M.A. (1990) and Ph.D (1998) in Mathematics from State University of New York, Binghamton, USA.

She is an Associate Professor at Department of Mathematics, Faculty of Science, Universiti Teknologi Malaysia, Skudai, Johor. Her current interests include Group Theory and DNA Splicing System.

Yuhani Yusof received the B.Sc. (Industrial Mathematics) in 2003 and M. Sc. (Mathematics) in 2005 from Universiti Teknologi Malaysia.

She is a Ph.D (Mathematics) student in Universiti Teknologi Malaysia, Skudai, Johor. Her current interest is DNA Splicing System.