MATEMATIKA: MJIAM, 2019, Volume 35, Number 3, 283–296
© Penerbit UTM Press. All rights reserved

Static Watson-Crick Linear Grammars and Its Computational Power

[1]**Aqilahfarhana Abdul Rahman**, [2]**Wan Heng Fong**[*], [3]**Nor Haniza Sarmin**, [4]**Sherzod Turaev** and [5]**Nurul Liyana Mohamad Zulkufli**

[1,2,3]Department of Mathematical Sciences, Universiti Teknologi Malaysia
81310 UTM Johor Bahru, Malaysia

[4]Department of Computer Science and Software Engineering, College of Information Technology, United Arab Emirates University, P.O. Box 15551, Al-Ain, United Arab Emirates

[5]Department of Computer Science, Faculty of Information and Communication Technology, International Islamic University Malaysia, 53100 Kuala Lumpur, Malaysia

[*]Corresponding author: fwh@utm.my

Article history
Received: 3 October 2018
Received in revised form: 10 June 2019
Accepted: 8 July 2019
Published online: 1 December 2019

Abstract DNA computing, or more generally, molecular computing, is a recent development on computations using biological molecules, instead of the traditional silicon-chips. Some computational models which are based on different operations of DNA molecules have been developed by using the concept of formal language theory. The operations of DNA molecules inspire various types of formal language tools which include sticker systems, grammars and automata. Recently, the grammar counterparts of Watson-Crick automata known as Watson-Crick grammars which consist of regular, linear and context-free grammars, are defined as grammar models that generate double-stranded strings using the important feature of Watson-Crick complementarity rule. In this research, a new variant of static Watson-Crick linear grammar is introduced as an extension of static Watson-Crick regular grammar. A static Watson-Crick linear grammar is a grammar counterpart of sticker system that generates the double-stranded strings and uses rule as in linear grammar. There is a difference in generating double-stranded strings between a dynamic Watson-Crick linear grammar and a static Watson-Crick linear grammar. A dynamic Watson-Crick linear grammar produces each stranded string independently and only check for the Watson-Crick complementarity of a generated complete double-stranded string at the end; while the static Watson-Crick linear grammar generates both stranded strings dependently, i.e., checking for the Watson-Crick complementarity for each complete substring. The main result of the paper is to determine some computational properties of static Watson-Crick linear grammars. Next, the hierarchy between static Watson-Crick languages, Watson-Crick languages, Chomsky languages and families of languages generated by sticker systems are presented.

Keywords Sticker system; computational power; Watson-Crick grammar; linear grammar.

Mathematics Subject Classification 68Q45, 92B05.

1 Introduction

In DNA computing techniques, there are two fundamental features which are necessary to overcome the limitation of traditional silicon-based computing technologies known as Watson-Crick complementarity and massive parallelism of DNA strands. DNA computing is based on the double stranded structure of DNA molecule composed of four nucleotides as its bases known as adenine (A), thymine (T), cytosine (C), and guanine (G), paired as A-T and C-G according to Watson-Crick complementarity. Massive parallelism of DNA strands, which is another feature of DNA molecules, allows construction of many copies of DNA strands where numerous operations are carried out on the encoded information simultaneously.

Some computational models which are based on different operations of DNA molecules have been developed such as Watson-Crick automata and sticker systems [1]. Watson-Crick automata [2] is an extension from finite automata which works on a double-stranded string. The input of both strands are scanned separately in a corresponding way reading from left to right which is controlled by a state. On the other hand, Adleman [3] has used sticker operation in his experiment of computing a Hamiltonian path problem in a graph by using DNA molecules. In 1998, Kari *et al.* [4] proposed the concept of sticker system as a language generating device using sticker operation.

Historically, some grammar models that were introduced did not use the fundamental feature of Watson-Crick (WK) complementarity of DNA molecules [5–7]. Then, a broad variety of dynamic WK grammars that use this fundamental feature have been proposed [8–10]. Although these grammars use different restrictions of production rules, they still have some weaknesses when generating double-stranded strings. These model produce each stranded string independently and did not fully illustrate the synthesis of DNA molecules. Motivated by the Watson-Crick grammars, this research introduces a new variant of static WK grammars known as a static WK linear grammar, as an extension of static WK regular grammar [11]. A static WK linear grammar is a grammar counterpart of sticker system which uses WK complementarity feature of DNA molecules; starting from the incomplete double-stranded sequence and iteratively using sticking operation until complete double-stranded sequence is obtained.

This paper is organised as follows: Section 1 gives the background and introduction of the paper. Section 2 presents some necessary definitions and notations from the theories of formal languages, automata and sticker systems. The definitions of WK grammars and static WK regular grammar are also stated. In addition, the definition of static WK linear grammar is given in Section 3. In Section 4, some examples are provided to illustrate the computational power of static WK linear grammar. Besides, some new facts on the computational power of static WK linear grammar and the hierarchy between static WK languages, WK languages, Chomsky languages and families of languages generated by sticker systems are presented. Lastly, the conclusion is given in Section 5.

2 Preliminaries

This section includes some preliminary concepts which involves the basic terms, theorem and definitions that are used in this paper. The reader may refer to [2,12,13] for detailed information regarding on the basic concepts of formal language theory, automata and sticker systems.

In this paper, the symbol $\subseteq$ denotes the inclusion while $\subset$ denotes the strict (proper) inclusion. The membership of an element to a set is denoted by $\in$ and the empty set is denoted by the symbol $\emptyset$. Let T be a finite alphabet, then T^* is the set of all finite strings (words) over T. A string with no symbols, or we called it as empty string, is denoted by λ. The set T^* always contain λ and to exclude the empty string, the symbol T^+ is defined as the set of all nonempty finite strings over T where $T^+ = T^* - \lambda$.

Next, a *Chomsky grammar* is defined as a quadruple $G = (N, T, S, P)$, where the alphabet N is defined as the *nonterminal* alphabet, T is the *terminal* alphabet, $S \in N$ is the axiom or start alphabet, and $P \subseteq (N \cup T)^* N (N \cup T)^* \times (N \cup T)^*$ is the set of production rules of G. The rule $(x, y) \in P$ is written in the form of $x \to y$ where $x \in (N \cup T)^+$ and $y \in (N \cup T)^*$. We say that u *directly derives* v or v is derived from u with respect to G which is written as $u \Rightarrow v$ if and only if $u = u_1 x u_2, v = u_1 y u_2$ for some $u_1 u_2 \in (N \cup T)^*$ and $x \to y \in P$. The set of all terminal strings is the language generated by the grammar which defined by $L(G) = \{w \in T^* : S \Rightarrow^* w\}$.

The Chomsky grammar is classified depending on their respective form of production rules. A grammar $G = (N, T, S, P)$ is called *context-sensitive* if each rule $u \to v \in P$ has $u = u_1 A u_2$, $v = u_1 x u_2$, for $u_1, u_2 \in (N \cup T)^*, A \in N$ and $x \in (N \cup T)^+$; *context-free* if each rule $u \to v \in P$ has $u \in N$; *linear* if each rule $u \to v \in P$ has $u \in N$ and $v \in T^* \cup T^* N T^*$; *regular* if each rule $u \to v \in P$ has $u \in N$ and $v \in T \cup TN \cup \{\lambda\}$.

All those families of languages generated by *context-sensitive*, *context-free*, *linear* and *regular* are denoted as **CS**, **CF**, **LIN** and **REG** respectively. Other than that, **RE** and **FIN** represent the family of recursive enumerable language and finite language. The languages are classified into four language types, type 0 to type 3 which make up the Chomsky hierarchy. Unrestricted grammars which generate recursively enumerable languages are referred to as Type 0. Besides, Type 1 grammars refer to context-sensitive grammars, Type 2 grammars refer to context-free grammars and Type 3 grammars refer to regular grammars which generate context-sensitive languages, context-free languages and regular languages respectively. Hence, the following strict inclusion holds for *Chomsky hierarchy*:

Theorem 1 $\mathbf{FIN} \subset \mathbf{REG} \subset \mathbf{LIN} \subset \mathbf{CF} \subset \mathbf{CS} \subset \mathbf{RE}$.

The definitions of WK grammars are presented as follows.

Definition 1 [8] *A Watson Crick (WK) grammar* $G = (N, T, S, P, \rho)$ *is called*

(i) *regular if each production has the form* $A \longrightarrow \langle u/v \rangle B$ *or* $A \longrightarrow \langle u/v \rangle$ *where* $A, B \in N$ *and* $\langle u/v \rangle \in \langle T^*/T^* \rangle$.

(ii) *linear if each production has the form* $A \longrightarrow \langle u_1/v_1 \rangle B \langle u_2/v_2 \rangle$ *or* $A \longrightarrow \langle u/v \rangle$ *where* $A, B \in N$ *and* $\langle u_1/v_1 \rangle, \langle u_2/v_2 \rangle, \langle u/v \rangle \in \langle T^*/T^* \rangle$.

(iii) *context-free if each production has the form* $A \longrightarrow \alpha$ *where* $A \in N$ *and* $\alpha \in (N \cup \langle T^*/T^* \rangle)^*$.

The notation $\langle u/v \rangle$ represents the element $(u, v) \subseteq V \times V$ in the set of pairs of strings and $\langle T^*/T^* \rangle$ is written instead of $\langle V^*/V^* \rangle$.

In order to generate or form a complete double sequence of DNA, the sticker system uses sticker operation on DNA molecules. Let V be an alphabet (a finite set of abstract symbol) and a symmetric relation $\rho \in V \times V$ over V (of complementarity). The set

$$WK_\rho(V) = \begin{bmatrix} V \\ V \end{bmatrix}_\rho^* \text{where} \begin{bmatrix} V \\ V \end{bmatrix}_\rho = \{\begin{bmatrix} a \\ b \end{bmatrix} | a, b \in V, \begin{pmatrix} a \\ b \end{pmatrix} \in \rho\}.$$

denotes the *Watson-Crick domain* associated to alphabet V and complementarity relation ρ. The elements $\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \in WK_\rho(V)$ is called well-formed double stranded sequences, or also known as double stranded sequences. The string w_1 is the upper strand and w_2 is the lower strand of the molecule. Note that there is a difference between $\begin{pmatrix} a \\ b \end{pmatrix}$ and $\begin{bmatrix} a \\ b \end{bmatrix}$; for the pair of $\begin{pmatrix} a \\ b \end{pmatrix}$, there is no relation between the elements a and b, while $\begin{bmatrix} a \\ b \end{bmatrix}$ indicates that the elements in the upper strand and lower strand are complement and have the same length.

Apart from that, the set of incomplete molecules is denoted as: $W_\rho(V) = L_\rho(V) \cup R_\rho(V) \cup LR_\rho(V)$ where

$$L_\rho(V) = (\begin{pmatrix} \lambda \\ V^* \end{pmatrix} \cup \begin{pmatrix} V^* \\ \lambda \end{pmatrix}) \begin{bmatrix} V \\ V \end{bmatrix}_\rho^*,$$

$$R_\rho(V) = \begin{bmatrix} V \\ V \end{bmatrix}_\rho^* (\begin{pmatrix} \lambda \\ V^* \end{pmatrix} \cup \begin{pmatrix} V^* \\ \lambda \end{pmatrix}),$$

$$LR_\rho(V) = (\begin{pmatrix} \lambda \\ V^* \end{pmatrix} \cup \begin{pmatrix} V^* \\ \lambda \end{pmatrix}) \begin{bmatrix} V \\ V \end{bmatrix}_\rho^+ (\begin{pmatrix} \lambda \\ V^* \end{pmatrix} \cup \begin{pmatrix} V^* \\ \lambda \end{pmatrix}).$$

In this research, we modified the definition of $LR_\rho(V)$ according to our grammar, where

$$LR_\rho^*(T) = (\begin{pmatrix} \lambda \\ T^* \end{pmatrix} \cup \begin{pmatrix} T^* \\ \lambda \end{pmatrix}) \begin{bmatrix} T \\ T \end{bmatrix}_\rho^* (\begin{pmatrix} \lambda \\ T^* \end{pmatrix} \cup \begin{pmatrix} T^* \\ \lambda \end{pmatrix}),$$

$$LR_\rho^+(T) = (\begin{pmatrix} \lambda \\ T^* \end{pmatrix} \cup \begin{pmatrix} T^* \\ \lambda \end{pmatrix}) \begin{bmatrix} T \\ T \end{bmatrix}_\rho^+ (\begin{pmatrix} \lambda \\ T^* \end{pmatrix} \cup \begin{pmatrix} T^* \\ \lambda \end{pmatrix}),$$

and the alphabet V which is defined in $W_\rho(V)$ is changed to the alphabet T according to the definition in the Chomsky grammar. Next, the sticker system is defined as follows.

A sticker system is a construct $\gamma = (V, \rho, A, D)$, where V is an alphabet, $\rho \in V \times V$ is a symmetric relation, A is finite subset of $LR_\rho(V)$ (called *axioms*) and D is a finite subset of $W_\rho(V) \times W_\rho(V)$ (called *domimoes*). For the two sequences $x, y \in LR_\rho(V)$, $x \Rightarrow y$ if and only if $y = \mu(u,\mu(x, v))$ for some $(u, v) \in D$, where μ is defined as the sticking operation. Hence, $\mu(u,\mu(x, v)) = \mu(\mu(u, x), v)$ since the prolongation to the left is independent as to the one at the right such that the sticker operation is associative. Moreover, a sequence $x_1 \Rightarrow x_2 \Rightarrow ... \Rightarrow x_k$ is obtained and is called a *computation* in γ as $x_1 \in A$ and $x_k \in WK_\rho(V)$. Thus, a complete computation, σ is represented as $x_1 \Rightarrow^* x_k$ when there is no sticky end in the last sequence. The language generated by the sticker system, γ is called a sticker language and is defined by

$$L(\gamma) = \{w \in \begin{pmatrix} V \\ V \end{pmatrix}_\rho^* | x \Rightarrow^* w, x \in A\}.$$

There are several restricted variants of sticker system which are arbitrary, one-sided, regular, simple, simple and one-sided, or simple and regular denoted as ASL(α), OSL(α), RSL(α), SSL(α), SOSL(α), SRSL(α) respectively where $\alpha \in \{n, p, b\}$ and the letters *n,p,b* represent no restrictions, primitive and delay computation, respectively.

In the next section, the definition of static WK linear grammar with an example are presented.

3 Static Watson-Crick Linear Grammar

In this section, the definition of static WK regular grammar introduced in [12] is used to extend the basic concept to linear grammars. Here, we define a grammar called static WK linear grammar, a grammar counterpart of the sticker system that has rules as in a linear grammar in the following.

Definition 2 *A static Watson-Crick linear grammar is a 5 -tuple $G = (N, T, \rho, S, P)$ where N, T are disjoint nonterminal and terminal alphabets, respectively, $\rho \in T \times T$ is a symmetric relation (WK complementarity), $S \in N$ is the start symbol (axiom) and P is a finite set of production in the form of*

(i) $S \rightarrow \begin{bmatrix} u_1 \\ v_1 \end{bmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} A \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} \begin{bmatrix} u_2 \\ v_2 \end{bmatrix}$ *where* $A \in N - \{S\}$, $\begin{bmatrix} u_1 \\ v_1 \end{bmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} \in R_\rho(T)$ *and* $\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} \begin{bmatrix} u_2 \\ v_2 \end{bmatrix} \in L_\rho(T)$;

(ii) $A \rightarrow \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} B \begin{pmatrix} x_2 \\ y_2 \end{pmatrix}$ *where* $A, B \in N - \{S\}$ *and* $\begin{pmatrix} x_1 \\ y_1 \end{pmatrix}, \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} \in LR^*_\rho(T)$; *or*

(iii) $A \rightarrow \begin{pmatrix} x_1 \\ y_1 \end{pmatrix}$ *where* $A \in N - \{S\}$ *and* $\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} \in LR^*_\rho(T)$.

Remark 1. The elements $\begin{bmatrix} u \\ v \end{bmatrix}$ in the set of all pairs of strings $T \times T$ can be classified into two cases, whether in the form of $\begin{bmatrix} u \\ v \end{bmatrix} \neq \begin{bmatrix} \lambda \\ \lambda \end{bmatrix}$ or $\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \lambda \\ \lambda \end{bmatrix}$.

The derivation for a static WK linear grammar is presented as follows.

Definition 3 *Let $G = (N, T, \rho, S, P)$ be a static WK linear grammar. We say that α derives β in G, denoten or written as $\alpha \Rightarrow \beta$ if and only if*

(i) $\alpha = S$ *and* $\beta = \begin{bmatrix} u_1 \\ v_1 \end{bmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} A \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} \begin{bmatrix} u_2 \\ v_2 \end{bmatrix}$ *where* $\alpha \Rightarrow \beta \in P$;

(ii) $\alpha = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} A \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} \begin{bmatrix} w_3 \\ w_4 \end{bmatrix}$ *and* $\beta = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} \begin{pmatrix} x_3 \\ y_3 \end{pmatrix} B \begin{pmatrix} x_4 \\ y_4 \end{pmatrix} \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} \begin{bmatrix} w_3 \\ w_4 \end{bmatrix}$ *where* $A, B \in N - \{S\}$, $\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} \in R_\rho(T)$, $\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} \begin{bmatrix} w_3 \\ w_4 \end{bmatrix} \in L_\rho(T)$ *and* $A \rightarrow \begin{pmatrix} x_3 \\ y_3 \end{pmatrix} B \begin{pmatrix} x_4 \\ y_4 \end{pmatrix} \in P$; *or*

(iii) $\alpha = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} A \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} \begin{bmatrix} w_3 \\ w_4 \end{bmatrix}$ *and* $\beta = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} \begin{pmatrix} x_3 \\ y_3 \end{pmatrix} \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} \begin{bmatrix} w_3 \\ w_4 \end{bmatrix}$ *where* $A, B \in N - \{S\}$, $\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} \in R_\rho(T)$, $\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} \begin{bmatrix} w_3 \\ w_4 \end{bmatrix} \in L_\rho(T)$ *and* $A \to \begin{pmatrix} x_3 \\ y_3 \end{pmatrix} \in P$.

The reflexive and transitive closure of $\underset{G}{\Rightarrow}$ or $(\Rightarrow)$ is denoted by $\underset{G}{\Rightarrow}^*$ or $(\Rightarrow^*)$. The language generated by a static WK linear grammar G, denoted by $L(G)$, is defined as

$$L(G) = \{u : \begin{bmatrix} u \\ v \end{bmatrix} \in WK_\rho(T) \text{ and } S \Rightarrow [\ G\ ]^* \begin{bmatrix} u \\ v \end{bmatrix}\}.$$

The family of languages generated by static WK linear grammar is denoted by **SLIN**. Next, an example is illustrated to show the family of languages generated by static WK linear grammar.

Example 1 Let $G = (\{S, A, B\}, \{a, b, c\}, \{(a, a), (b, b), (c, c)\}, S, P)$ be a static WK linear grammar, where P consists of the following rules:

$$\begin{aligned}
S &\to \begin{bmatrix} a \\ a \end{bmatrix} \begin{pmatrix} a \\ \lambda \end{pmatrix} A \begin{pmatrix} \lambda \\ b \end{pmatrix} \begin{bmatrix} b \\ b \end{bmatrix}, \\
A &\to \begin{pmatrix} \lambda \\ a \end{pmatrix} \begin{bmatrix} a \\ a \end{bmatrix} \begin{pmatrix} a \\ \lambda \end{pmatrix} A \begin{pmatrix} \lambda \\ b \end{pmatrix} \begin{bmatrix} b \\ b \end{bmatrix} \begin{pmatrix} b \\ \lambda \end{pmatrix} \\
A &\to \begin{pmatrix} \lambda \\ a \end{pmatrix} \begin{bmatrix} c \\ c \end{bmatrix} \begin{pmatrix} c \\ \lambda \end{pmatrix} B \begin{pmatrix} b \\ \lambda \end{pmatrix} \\
B &\to \begin{pmatrix} \lambda \\ c \end{pmatrix} \begin{bmatrix} c \\ c \end{bmatrix} \begin{pmatrix} c \\ \lambda \end{pmatrix} B \\
B &\to \begin{pmatrix} \lambda \\ c \end{pmatrix}.
\end{aligned}$$

From this, we obtain the derivation:

$$\begin{aligned}
S \Rightarrow \begin{bmatrix} a \\ a \end{bmatrix} \begin{pmatrix} a \\ \lambda \end{pmatrix} A \begin{pmatrix} \lambda \\ b \end{pmatrix} \begin{bmatrix} b \\ b \end{bmatrix} &\Rightarrow^* \begin{bmatrix} a^{n-1} \\ a^{n-1} \end{bmatrix} \begin{pmatrix} a \\ \lambda \end{pmatrix} A \begin{pmatrix} \lambda \\ b \end{pmatrix} \begin{bmatrix} b^{n-1} \\ b^{n-1} \end{bmatrix} \\
&\Rightarrow \begin{bmatrix} a^n c \\ a^n c \end{bmatrix} \begin{pmatrix} c \\ \lambda \end{pmatrix} B \begin{bmatrix} b^n \\ b^n \end{bmatrix} \\
&\Rightarrow^* \begin{bmatrix} a^n c^{m-1} \\ a^n c^{m-1} \end{bmatrix} \begin{pmatrix} c \\ \lambda \end{pmatrix} B \begin{bmatrix} b^n \\ b^n \end{bmatrix} \\
&\Rightarrow \begin{bmatrix} a^n c^m b^n \\ a^n c^m b^n \end{bmatrix}.
\end{aligned}$$

Therefore, $L(G) = \{a^n c^m b^n \mid n, m \geq 2\}$.

In the next section, the generative power of static WK linear grammar is discussed.

4 Generative Power of Static Watson-Crick Linear Grammar

In this section, the relationship between the family of static WK linear grammar with family in the Chomsky hierarchy, sticker system and also WK grammar are determined. The following lemma follows immediately from Definition 2, where **LIN** indicates the set of languages generated by linear grammars.

Lemma 1 *The following inclusion holds:*

$$\boldsymbol{LIN} \subseteq \boldsymbol{SLIN}.$$

Proof For a linear grammar $G = (N, T, S, P)$, its static WK variant $G' = (N, T, \rho, S, P')$ is defined where $\rho = \{(a, a)|a \in T\}$ and for each production $A \to \alpha \in P$, every terminal string x in α is changed to $\begin{bmatrix} x \\ x. \end{bmatrix}$. Then, it is easy to see that $L(G') = L(G)$. □

In the next lemma, we show the relationship between the static WK regular grammar and static WK linear grammar. According to the Chomsky hierarchy, the linear languages are more powerful than the regular languages. Thus, this idea is true for the inclusion between static WK regular and linear grammars such that all the languages generated by static WK regular grammars can be generated directly by static WK linear grammars.

Lemma 2 *The following inclusion holds:*

$$\boldsymbol{SREG} \subseteq \boldsymbol{SLIN}.$$

Next, Example 2 shows the language generated by a static WK linear grammar. Through this example, we can relate the generative power in Lemma 2 by using the same language as in Lemma 4 [11] where $L(G) = \{a^n b^n c^n | n \geq 2\}$.

Example 2 Consider the static WK linear grammar $G = (\{S, A, B, C, D, E\}, \{a, b, c\}, S, \rho, P)$ where P contains the following productions:

$$\text{(i) } S \to \begin{bmatrix} a \\ a \end{bmatrix} \binom{a}{\lambda} A \binom{c}{\lambda} \begin{bmatrix} c \\ c \end{bmatrix}, \qquad \text{(v) } C \to \binom{b}{\lambda} B,$$

$$\text{(ii) } A \to \binom{a}{\lambda} A \binom{c}{\lambda}, \qquad \text{(vi) } C \to \binom{\lambda}{b} D \binom{\lambda}{c},$$

$$\text{(iii) } A \to \binom{b}{\lambda} B, \qquad \text{(vii) } D \to \binom{\lambda}{b} D \binom{\lambda}{c},$$

$$\text{(iv) } B \to \binom{\lambda}{a} C, \qquad \text{(viii) } D \to \begin{bmatrix} b \\ b \end{bmatrix}.$$

Here, we need to show that the language $L(G) = \{a^n b^n c^n | n \geq 2\} \in$ **SLIN**. We define the production rules of static WK linear grammar as follows:

Step 1. From rule (i):

$$S \Rightarrow \begin{bmatrix} a \\ a \end{bmatrix} \begin{pmatrix} a \\ \lambda \end{pmatrix} A \begin{pmatrix} c \\ \lambda \end{pmatrix} \begin{bmatrix} c \\ c \end{bmatrix}. \tag{1}$$

Step 2. Derivation (1) can be continued with rule (ii) or rule (iii). Without loss of generality, we apply rule (ii) $k \geq 0$ times and apply rule (iii):

$$S \Rightarrow^* \begin{bmatrix} a \\ a \end{bmatrix} \begin{pmatrix} a \\ \lambda \end{pmatrix} \begin{pmatrix} a^k \\ \lambda \end{pmatrix} \begin{pmatrix} b \\ \lambda \end{pmatrix} B \begin{pmatrix} c^k \\ \lambda \end{pmatrix} \begin{pmatrix} c \\ \lambda \end{pmatrix} \begin{bmatrix} c \\ c \end{bmatrix} = \begin{bmatrix} a \\ a \end{bmatrix} \begin{pmatrix} a^{k+1} \\ \lambda \end{pmatrix} \begin{pmatrix} b \\ \lambda \end{pmatrix} B \begin{pmatrix} c^{k+1} \\ \lambda \end{pmatrix} \begin{bmatrix} c \\ c \end{bmatrix}. \tag{2}$$

Step 3. Derivation (2) can only be continued with rule (iv):

$$S \Rightarrow^* \begin{bmatrix} a \\ a \end{bmatrix} \begin{pmatrix} a^{k+1} \\ \lambda \end{pmatrix} \begin{pmatrix} b \\ \lambda \end{pmatrix} \begin{pmatrix} \lambda \\ a \end{pmatrix} C \begin{pmatrix} c^{k+1} \\ \lambda \end{pmatrix} \begin{bmatrix} c \\ c \end{bmatrix} = \begin{bmatrix} aa \\ aa \end{bmatrix} \begin{pmatrix} a^{k} \\ \lambda \end{pmatrix} \begin{pmatrix} b \\ \lambda \end{pmatrix} C \begin{pmatrix} c^{k+1} \\ \lambda \end{pmatrix} \begin{bmatrix} c \\ c \end{bmatrix}. \tag{3}$$

Step 4. Derivation (3) can be continued with rule (v) or (vi). Rule (v) must be applied k times to complete the lower strand of $\begin{pmatrix} a^k \\ \lambda \end{pmatrix}$, which results in applying rule (iv) to applied k times, and then we apply rule (vi):

$$S \Rightarrow^* \begin{bmatrix} aa \\ aa \end{bmatrix} \begin{pmatrix} a^{k} \\ a^k \end{pmatrix} \begin{pmatrix} b \\ \lambda \end{pmatrix} \begin{pmatrix} b^k \\ \lambda \end{pmatrix} C \begin{pmatrix} c^{k+1} \\ \lambda \end{pmatrix} \begin{bmatrix} c \\ c \end{bmatrix} = \begin{bmatrix} a^{k+2} \\ a^{k+2} \end{bmatrix} \begin{bmatrix} b \\ b \end{bmatrix} \begin{pmatrix} b^k \\ \lambda \end{pmatrix} D \begin{pmatrix} c^{k} \\ \lambda \end{pmatrix} \begin{bmatrix} cc \\ cc \end{bmatrix}. \tag{4}$$

Step 5. To complete derivation (4), we apply rule (vii) k times to complete the lower strand of $\begin{pmatrix} b^k \\ \lambda \end{pmatrix}$ and $\begin{pmatrix} c^k \\ \lambda \end{pmatrix}$. The derivation is completed with rule (viii):

$$S \Rightarrow^* \begin{bmatrix} a^{k+2}b^{k+2}c^{k+2} \\ a^{k+2}b^{k+2}c^{k+2} \end{bmatrix}.$$

Thus, $L(G) = \{a^n b^n c^n | n \geq 2\} \in$ **SLIN**. □

The theorem below also follows immediately from Example 2. Theorem 2 shows that the family of languages generated by linear grammar is strictly included in the family of languages generated by static WK linear grammar, and that static WK linear grammar can generate some non context-free languages.

Theorem 2 *The following inclusions hold:*

$$\boldsymbol{LIN} \subset \boldsymbol{SLIN} \text{ and } \boldsymbol{SLIN} - \boldsymbol{CF} \neq \emptyset.$$

Proof From Example 2, the static WK linear grammar can generate some non context-free languages. Since context-free languages can generate the linear grammars and context-free grammars, then the language $L(G) = \{a^n b^n c^n | n \geq 2\} \in$ **SLIN** cannot be generated by linear grammars and context-free grammars. Thus, the theorem follows from Lemma 1 and Example 2. □

On the other hand, we can relate the language in Example 1 with static WK regular grammar whereby $L(G) = \{a^n c^m b^n | n, m \geq 2\} \in$ **SREG** as shown in Example 3.

Example 3 Let $G = (\{S, A, B, C, D, E, F\}, \{a, b, c\}, \{(a, a), (b, b), (c, c)\}, S, P)$ be a static WK regular grammar where P contains the following productions:

$$\begin{array}{ll}
\text{(i) } S \to \binom{aa}{\lambda} A, & \text{(vii) } D \to \binom{b}{\lambda} C, \\
\text{(ii) } A \to \binom{a}{\lambda} A, & \text{(viii) } D \to \binom{\lambda}{c} E, \\
\text{(iii) } A \to \binom{cc}{\lambda} B, & \text{(ix) } E \to \binom{\lambda}{c} E, \\
\text{(iv) } B \to \binom{c}{\lambda} B, & \text{(x) } E \to \binom{\lambda}{b} F, \\
\text{(v) } B \to \binom{b}{\lambda} C, & \text{(xi) } F \to \binom{\lambda}{b} F, \\
\text{(vi) } C \to \binom{\lambda}{a} D, & \text{(xii) } F \to \binom{\lambda}{\lambda}.
\end{array}$$

From this, we obtain the derivation:

$$S \Rightarrow^* \binom{a^{n-1}}{\lambda} A \Rightarrow \binom{a^n}{\lambda} A \Rightarrow^* \binom{a^n c^{m-1}}{\lambda} B \Rightarrow \binom{a^n c^m}{\lambda} B \Rightarrow \binom{a^n c^m}{\lambda} \binom{b}{\lambda} C$$
$$\Rightarrow^* \binom{a^n c^m b^{n-1}}{a^{n-1}} \binom{b}{\lambda} C \Rightarrow \begin{bmatrix} a^n \\ a^n \end{bmatrix} \binom{c^m b^n}{\lambda} \binom{\lambda}{c} E \Rightarrow^* \begin{bmatrix} a^n c^m \\ a^n c^m \end{bmatrix} \binom{b^n}{b^{n-1}} F \Rightarrow \begin{bmatrix} a^n c^m b^n \\ a^n c^m b^n \end{bmatrix}.$$

Therefore, $L(G) = \{a^n c^m b^n \mid n, m \geq 2\}$.

In [4], the relationship between WK linear language (**WKLIN**) and the family of arbitrary sticker languages with no restriction **ASL (n)** has been established, where **ASL (n)** $\subseteq$ **WKLIN**. Here, we show that **ASL (n)** can be simulated by **SLIN** in the following proposition.

Proposition 1 *The following hold:*

$$\boldsymbol{ASL(n)} \subseteq \boldsymbol{SLIN}.$$

Proof Let $\gamma = (T, \rho, A, D)$ be an arbitrary sticker system. We construct a static WK linear grammar $G = (N, T, \rho, S, P)$ with $L(\gamma) = L(G)$ where $N = \{S, B\}$ and P contains the productions in the form of:

(i) $S \to uBv$ for all $\binom{u}{v} \in D$ where $u \in R_\rho(T)$ and $v \in L_\rho(T)$,

(ii)$B \to uBv$ for all $\binom{u}{v} \in D$ where $u, v \in LR_\rho(T)$,

(iii) $B \to x$ for all $x \in A$.

First, we need to show that $L(\gamma) \subseteq L(G)$. Suppose $w \in L(\gamma)$. Then, there are some $x \in A$ and $(u_1, v_1), (u_2, v_2), \ldots, (u_k, v_k)$ such that

$$\begin{aligned} x &\Rightarrow \mu(\mu(u_1, x), v_1) \\ &\Rightarrow \mu(\mu(u_2, \mu(\mu(u_1, x), v_1)), v_2) \\ &\Rightarrow^* \mu(\mu(\mu_k \ldots (\mu(\mu(u_2, \mu(\mu(u_1, x), v_1)), v_2)) \ldots), v_k) \\ &\Rightarrow \begin{bmatrix} w \\ w \end{bmatrix}. \end{aligned} \tag{5}$$

The computation in (5) can be simulated by the following derivation in G in a reversed order, starting from the last used pair in D and progressing toward the center of the sequence, where an axiom x in A will be used such that

$$\begin{aligned} S &\Rightarrow u_k B v_k \Rightarrow u_k u_{k-1} B v_{k-1} v_k \Rightarrow \ \ldots \\ &\Rightarrow {}^* u_k u_{k-1} \ldots u_1 B v_1 \ldots v_{k-1} v_k \\ &\Rightarrow u_k u_{k-1} \ldots u_1 x v_1 \ldots v_{k-1} v_k \\ &\Rightarrow \begin{bmatrix} w \\ w \end{bmatrix}. \end{aligned}$$

The similar construction also holds for the inverse case where $L(G) \subseteq L(\gamma)$. Therefore, we can conclude that every derivation in G can also be simulated by a computation in γ and vice versa. □

The following example illustrates Proposition 1.

Example 4 Let $\gamma = (T, \rho, A, D)$ be an arbitrary sticker system, $ASL(n)$ with

$$L(\gamma) = \{ \begin{bmatrix} a \\ a \end{bmatrix}^n \begin{bmatrix} b \\ b \end{bmatrix}^m \begin{bmatrix} c \\ c \end{bmatrix}^n | n \geq 2, m \geq 3\}$$

where

$$\begin{aligned} &\rho = \{(a, a), (b, b), (c, c) | a, b, c \in T\}, \\ &A = \{ \begin{pmatrix} \lambda \\ b \end{pmatrix} \begin{bmatrix} b \\ b \end{bmatrix} | \begin{pmatrix} \lambda \\ b \end{pmatrix} \begin{bmatrix} b \\ b \end{bmatrix} \in LR_\rho(T)\}, \\ &D = \{(\begin{pmatrix} \lambda \\ b \end{pmatrix} \begin{bmatrix} b \\ b \end{bmatrix} \begin{pmatrix} b \\ \lambda \end{pmatrix}, \begin{pmatrix} \lambda \\ \lambda \end{pmatrix}) \text{where} \begin{pmatrix} \lambda \\ b \end{pmatrix} \begin{bmatrix} b \\ b \end{bmatrix} \begin{pmatrix} b \\ \lambda \end{pmatrix} \in u_1, \begin{pmatrix} \lambda \\ \lambda \end{pmatrix} \in v_1, \\ &\quad (\begin{pmatrix} \lambda \\ a \end{pmatrix} \begin{bmatrix} b \\ b \end{bmatrix} \begin{pmatrix} b \\ \lambda \end{pmatrix}, \begin{pmatrix} c \\ \lambda \end{pmatrix}) \text{where} \begin{pmatrix} \lambda \\ a \end{pmatrix} \begin{bmatrix} b \\ b \end{bmatrix} \begin{pmatrix} b \\ \lambda \end{pmatrix} \in u_2, \begin{pmatrix} c \\ \lambda \end{pmatrix} \in v_2, \\ &\quad (\begin{pmatrix} \lambda \\ a \end{pmatrix} \begin{bmatrix} a \\ a \end{bmatrix} \begin{pmatrix} a \\ \lambda \end{pmatrix}, \begin{pmatrix} \lambda \\ c \end{pmatrix} \begin{bmatrix} c \\ c \end{bmatrix} \begin{pmatrix} c \\ \lambda \end{pmatrix}) \text{where} \begin{pmatrix} \lambda \\ a \end{pmatrix} \begin{bmatrix} a \\ a \end{bmatrix} \begin{pmatrix} a \\ \lambda \end{pmatrix} \in u_3, \begin{pmatrix} \lambda \\ c \end{pmatrix} \begin{bmatrix} c \\ c \end{bmatrix} \begin{pmatrix} c \\ \lambda \end{pmatrix} \in v_3, \\ &\quad (\begin{bmatrix} a \\ a \end{bmatrix} \begin{pmatrix} a \\ \lambda \end{pmatrix}, \begin{pmatrix} \lambda \\ c \end{pmatrix} \begin{bmatrix} c \\ c \end{bmatrix}) \text{where} \begin{bmatrix} a \\ a \end{bmatrix} \begin{pmatrix} a \\ \lambda \end{pmatrix} \in u_4, \begin{pmatrix} \lambda \\ c \end{pmatrix} \begin{bmatrix} c \\ c \end{bmatrix} \in v_4\}. \end{aligned}$$

The computation starts with an axiom and the sequence is prolonged to the left and to the right directions using the pair of dominoes, $(u_1, v_1), (u_2, v_2), (u_3, v_3)$ and (u_4, v_4). Let $n = 4$ and $m = 5$. The computations are as follows:

1. $\mu\{\binom{\lambda}{b}\begin{bmatrix}b\\b\end{bmatrix}, [\binom{\lambda}{b}\begin{bmatrix}b\\b\end{bmatrix}\binom{b}{\lambda}, \binom{\lambda}{\lambda}]\} = \binom{\lambda}{b}\begin{bmatrix}b\\b\end{bmatrix}\binom{b}{\lambda}\binom{\lambda}{b}\begin{bmatrix}b\\b\end{bmatrix}$.
2. $\mu\{\binom{\lambda}{b}\begin{bmatrix}b\\b\end{bmatrix}^3, [\binom{\lambda}{a}\begin{bmatrix}b\\b\end{bmatrix}\binom{b}{\lambda}, \binom{c}{\lambda}]\} = \binom{\lambda}{a}\begin{bmatrix}b\\b\end{bmatrix}\binom{b}{\lambda}\binom{\lambda}{b}\begin{bmatrix}b\\b\end{bmatrix}^3\binom{c}{\lambda}$.
3. $\mu\{\binom{\lambda}{a}\begin{bmatrix}b\\b\end{bmatrix}^5\binom{c}{\lambda}, [\binom{\lambda}{a}\begin{bmatrix}a\\a\end{bmatrix}\binom{a}{\lambda}, \binom{\lambda}{c}\begin{bmatrix}c\\c\end{bmatrix}\binom{c}{\lambda}]\} =$
   $\binom{\lambda}{a}\begin{bmatrix}a\\a\end{bmatrix}\binom{a}{\lambda}\binom{\lambda}{a}\begin{bmatrix}b\\b\end{bmatrix}^5\binom{c}{\lambda}\binom{\lambda}{c}\begin{bmatrix}c\\c\end{bmatrix}\binom{c}{\lambda}$.
4. $\mu\{\binom{\lambda}{a}\begin{bmatrix}a\\a\end{bmatrix}^2\begin{bmatrix}b\\b\end{bmatrix}^5\begin{bmatrix}c\\c\end{bmatrix}^2\binom{c}{\lambda}, [\begin{bmatrix}a\\a\end{bmatrix}\binom{a}{\lambda}, \binom{\lambda}{c}\begin{bmatrix}c\\c\end{bmatrix}]\} =$
   $\begin{bmatrix}a\\a\end{bmatrix}\binom{a}{\lambda}\binom{\lambda}{a}\begin{bmatrix}a\\a\end{bmatrix}^2\begin{bmatrix}b\\b\end{bmatrix}^5\begin{bmatrix}c\\c\end{bmatrix}^2\binom{c}{\lambda}\binom{\lambda}{c}\begin{bmatrix}c\\c\end{bmatrix}$.

Therefore, $\binom{\lambda}{a}\begin{bmatrix}b\\b\end{bmatrix} \Rightarrow \binom{\lambda}{b}\begin{bmatrix}b\\b\end{bmatrix}^3 \Rightarrow \binom{\lambda}{b}\begin{bmatrix}b\\b\end{bmatrix}^5\binom{c}{\lambda} \Rightarrow \binom{\lambda}{a}\begin{bmatrix}a\\a\end{bmatrix}^2\begin{bmatrix}b\\b\end{bmatrix}^5\begin{bmatrix}c\\c\end{bmatrix}^2\binom{c}{\lambda} \Rightarrow$ $\begin{bmatrix}a\\a\end{bmatrix}^4\begin{bmatrix}b\\b\end{bmatrix}^5\begin{bmatrix}c\\c\end{bmatrix}^4$. Hence, the language generated from the computation is $L(\gamma) = \{\begin{bmatrix}a\\a\end{bmatrix}^n\begin{bmatrix}b\\b\end{bmatrix}^m\begin{bmatrix}c\\c\end{bmatrix}^n$ $|n \geq 2, m \geq 3\}$.

Next, we build a static WK linear grammar $G = (N, T, \rho, S, P)$ from γ , where P contains the following rules:

$$S \to \begin{bmatrix}a\\a\end{bmatrix}\binom{a}{\lambda}A\binom{\lambda}{c}\begin{bmatrix}c\\c\end{bmatrix}, \text{ which is obtained from } (u_4, v_4) \in D,$$

$$A \to \binom{\lambda}{a}\begin{bmatrix}a\\a\end{bmatrix}\binom{a}{\lambda}A\binom{\lambda}{c}\begin{bmatrix}c\\c\end{bmatrix}\binom{c}{\lambda}, \text{ which is obtained from } (u_3, v_3) \in D,$$

$$A \to \binom{\lambda}{a}\begin{bmatrix}b\\b\end{bmatrix}\binom{b}{\lambda}B\binom{c}{\lambda}, \text{ which is obtained from } (u_2, v_2) \in D,$$

$$B \to \binom{\lambda}{b}\begin{bmatrix}b\\b\end{bmatrix}\binom{b}{\lambda}B, \text{ which is obtained from } (u_1, v_1) \in D,$$

$$B \to \binom{\lambda}{b}\begin{bmatrix}b\\b\end{bmatrix}, \text{ which is obtained from } A.$$

From this, we obtain the same derivation of G as in Example 1 where $L(G) = \{a^n b^m c^n | n \geq 2, m \geq 3\}$. The static WK linear grammar G generates the same sequences of double-stranded string as the sticker system do. It follows that every derivation in G can also be simulated by γ.

Next, the results show that the family of WK linear languages is included in the family of static WK linear languages.

Lemma 3 *The following hold:*

$$\boldsymbol{WKLIN} \subseteq \boldsymbol{SLIN}.$$

Proof Let $G = (N, T, \rho, S, P)$ be a WK linear grammar. From G, we build a (sticker) static WK linear grammar where $G' = (N', T, \rho, S', P')$ such that $L(G) = L(G')$.

Let $P_1 = \{A \to \binom{u_1}{v_1} B \binom{u_2}{v_2} \in P | u_1 \neq \lambda, v_1 \neq \lambda \text{ or } u_2 \neq \lambda, v_2 \neq \lambda\}$ and $P_2 = \{A \to \binom{u}{v} \in P | u \neq \lambda,\, v \neq \lambda\}$.

(i) For each production $r : A \to \binom{u_1}{v_1} B \binom{u_2}{v_2} \in P_1$, we define the new productions, $A \to \binom{u_1}{\lambda} B_r \binom{u_2}{\lambda}$ and $B_r \to \binom{\lambda}{v_1} B \binom{\lambda}{v_2}$ where B_r is a new nonterminal.

(ii) For each production $r : A \to \binom{u}{v} \in P_2$, we introduce the new productions $A \to \binom{u}{\lambda} B_r$, $B_r \to \binom{\lambda}{v}$.

Using the similar arguments in the proof of Lemma 5 in [11] , we can show that $L(G) = L(G')$. Hence, $\mathbf{WKLIN} \subseteq \mathbf{SLIN}$. □

The inclusion for **LIN** and **WKLIN** has already been proved in Theorem 13 [9] where $\mathbf{LIN} \subset \mathbf{WKLIN}$. By combining the above results with the results from [8, 11], we obtain the following relation. The relation in Figure 1 holds where the solid arrows represent the proper inclusions of the lower families into the upper families, while the dotted arrow represents the inclusions.

$$\begin{tikzcd}
\mathbf{WKLIN} \arrow[r, dashed] & \mathbf{SLIN} & \mathbf{ASL(n)} \arrow[l, dashed] \\
\mathbf{WKREG} \arrow[u] \arrow[r, dashed] & \mathbf{SREG} & \mathbf{LIN = ASL(b)} \arrow[u] \arrow[ul] \arrow[ull] \\
 & & \boldsymbol{REG = RSL(n) = OSL(n)} \arrow[u] \arrow[ul] \arrow[ull]
\end{tikzcd}$$

Figure 1: The Hierarchy of Static WK, WK, Chomsky and Sticker Language Families

5 Conclusion

In this paper, we define a static Watson-Crick linear grammar, which is one of the variants of static Watson-Crick grammars and determine its computational power in order to correlate with other family of languages. Based on the results obtained, we can conclude that the family of linear languages is strictly included in the family of static Watson-Crick linear languages and static Watson-Crick linear grammars can generate non context-free languages. Also, the arbitrary sticker languages is included in static Watson-Crick linear languages and the family of

Watson-Crick linear languages is included in the family of static Watson-Crick linear languages. However, the relation between the family of static Watson-Crick linear languages with the family of static Watson-Crick regular languages and the family of linear languages with the family of static Watson-Crick regular languages remain open. This research can be further studied by defining other variants of grammars such as static WK context-free grammars, regulated variants of static WK grammars and the computational power of the variants of grammars which is useful for DNA based computing devices and algorithmic techniques.

Acknowledgments

The first author would like to thank Universiti Teknologi Malaysia (UTM) for the Zamalah Scholarship. The second and third authors would also like to thank the Ministry of Education (MOE) and Research Management Centre (RMC), UTM for the financial funding through Fundamental Research Grant Scheme Vote No. 5F022.

References

[1] Kari, L., Seki, S. and Sosik, P. DNA computing–foundations and implications. In *Handbook of Natural Computing*. New York: Springer-Verlag, Berlin. 2012. 1073–1127.

[2] Paun, G., Rozenberg, G. and Salomaa, A. *DNA Computing: New Computing Paradigms*. New York: Springer-Verlag Berlin, 1998.

[3] Adleman, L. M. Molecular computation of solutions to combinatorial problems. *Science*. 1994. 266(5187): 1021–1024.

[4] Kari, L., Paun, G., Rozenberg, G., Salomaa, A. and Yu, S. DNA computing, sticker systems and universality. *ActaInformatica*. 1998. 35(5): 401–420.

[5] Varun, R.K. and Gupta, S. Analyzing the ambiguity in RNA structure using probabilistic approach. *International Journal of Information Technology*. 2012. 5(1): 107–110.

[6] Sutapa, D. and Mukhopadhyay, S. A composite method based on formal grammar and DNA structural features in detecting human polymerase II promoter region. *PLoS One*. 2013. 8(2): e54843.

[7] Algwaiz, A., Ammar, R. and Rajasekaran, S. Framework for data mining of big data using probabilistic grammars. In *e-Learning (econf) Fifth International Conference on*. 2015. 241–246.

[8] Zulkufli, N.L.M., Turaev, S., Tamrin, M.I.M and Azeddine, M. Closure properties of Watson-Crick grammars. *In AIP Conference Proceedings* 2015. 1691(1): 040032.

[9] Zulkufli, N.L.M., Turaev, S., Tamrin, M.I.M. and Messikh, A. Generative power and closure properties of Watson-Crick grammars. *Applied Computational Intelligence and Soft Computing*. 2016. 1–12.

[10] Zulkufli, N.L.M., Turaev, S., Tamrin, M.I.M., Messikh, A. and Alshaikhli, I.F.T. Computational properties of Watson-Crick context-free grammars. In *Advanced Computer Science Applications and Technologies (ACSAT), 2015 4th International Conference*. 2015. 186–191.

[11] Rahman, A.F.A, Fong, W.H., Sarmin, N. H., Turaev, S. and Zulkufli, N.L.M. Static Watson-Crick regular grammar. *Malaysian Journal of Applied and Fundamental Sciences.* 2018. 14: 457–462.

[12] Czeizler, E. and Czeizler, E. A short survey on Watson-Crick automata. *Bulletin of the EATCS* . 2006. 88(3): 104-119.

[13] Linz, P. *An Introduction to Formal Languages and Automata.* United States: Jones and Barlett Publishers. 2006.