MJFAS MALAYSIAN JOURNAL OF FUNDAMENTAL AND APPLIED SCIENCES
PRINT ISSN: 2289-5981 | ONLINE ISSN: 2289-599X

RESEARCH ARTICLE

Static Watson-Crick regular grammar

Aqilahfarhana Abdul Rahman [a, *], Wan Heng Fong [a], Nor Haniza Sarmin [a], Sherzod Turaev [b], Nurul Liyana Mohamad Zulkufli [b]

[a] *Department of Mathematical Sciences, Faculty of Science, Universiti Teknologi Malaysia, 81310 UTM Skudai, Johor, Malaysia*
[b] *Department of Computer Science, Faculty of Information and Communication Technology, International Islamic University Malaysia, 53100 Kuala Lumpur, Malaysia*

* *Corresponding author: aqilahfarhana_13@yahoo.com*

Article history
Received 3 May 2018
Revised 25 August 2018
Accepted 11 September 2018
Published Online 25 October 2018

Graphical abstract

$$\begin{tikzcd}
SREG & & WKREG \arrow[ll, dashed] \\
& & REG = RSL(n) = OSL(n) \arrow[ull] \arrow[u]
\end{tikzcd}$$

Abstract

DNA computing, or more generally, molecular computing, is a recent development at the interface of computer science and molecular biology. In DNA computing, many computational models have been proposed in the framework of formal language theory and automata such as Watson-Crick grammars and sticker systems. A Watson-Crick grammar is a grammar model that generates double stranded strings, whereas a sticker system is a DNA computing model of the ligation and annealing operations over DNA strands using the Watson-Crick complementarity to form a complete double stranded DNA sequence. Most of the proposed DNA computing models make use of this concept, including the Watson-Crick grammars and sticker systems. Watson-Crick grammars and their variants can be explored using formal language theory which allows the development of new concepts of Watson-Crick grammars. In this research, a new variant of Watson-Crick grammar called a *static Watson-Crick regular grammar* is introduced as an analytical counterpart of sticker systems. The computation of a sticker system starts from a given set of incomplete double stranded sequence to form a complete double stranded sequence. Here, a static Watson-Crick regular grammar differs from a dynamic Watson-Crick regular grammar in generating double stranded strings: the latter grammar produces each strand string "independently" and only check for the Watson-Crick complementarity of a generated complete double stranded string at the end, while the former grammar generates both strand strings "dependently", i.e., checking for the Watson-Crick complementarity for each complete substring. In this paper, computational properties of static Watson-Crick regular grammars are investigated to correlate with the Chomsky hierarchy and hierarchy of the families of dynamic Watson-Crick regular languages. The relationship between families of languages generated by static Watson-Crick regular grammars with several variants of sticker systems, Watson-Crick regular grammars and Chomsky grammars are presented by showing the hierarchy.

Keywords: Sticker system, Watson-Crick grammar, regular grammar, computational power, Chomsky hierarchy

INTRODUCTION

Deoxyribonucleic acid (DNA) computing appears to be a challenge in designing new types of computers which differ from their classical counterparts in a fundamental way, which is to solve a wide spectrum of computationally intractable problems. DNA molecules are double stranded structures composed of four nucleotides; A (adenine), C (cytosine), G (guanine), and T (thymine), paired as $A-T$ and $C-G$ according to the so-called Watson-Crick complementarity. Another feature of DNA molecules is the massive parallelism of DNA strands which allows the construction of many copies of DNA strands and carries out operations on encoded information simultaneously. The use of these two fundamental features of DNA molecules has illustrated that DNA-based computers can solve many computationally intractable problems such as Hamiltonian path problem [1], the satisfiability problems ([2],[3],[4]), the maximal clique problem [5], NP-complete graph based problems [6] and others.

DNA molecular operations motivate the introduction of different formal language tools such as recognition devices (automata) and generative devices (splicing systems, sticker systems, grammars and others), and the investigation of structures and properties of biological sequences. In 1987, Head [7] introduced the first formal tool (theoretical model) for DNA-based computation that uses the splicing operation, known as *splicing systems*. Following that, in 1998 Kari *et al.* [8] introduced another model of DNA computation which is the *sticker system*, a language generating device based on the sticker operation. The sticker operation was first used and performed in Adleman's experiment in 1994 to show how biological experiments are used to solve the Hamiltonian path problem for a simple graph [1]. The operation starts from a set of incomplete double stranded sequences (axioms) and two sets of single stranded complementary sequences. When compared to the splicing operation in a splicing system, the sticker operation is more advanced because it uses no enzymes and requires no strands extensions [9].

On the other hand, Freund *et al.* [10] proposed the *Watson-Crick automata* (WKA) which is one of the mathematical models used in DNA computation. WKA is an extension of finite automata with the addition of two reading heads on double stranded sequences. The symbols at the corresponding positions from the two strands of input are related complementarily, similar with the Watson-Crick complementarity of DNA nucleotides. The development of the Watson-Crick automata in the grammatical area starts in 2012 when Subramanian *et al.* [11] introduced the Watson-Crick (WK) regular grammar and it has been modified in [12]. The research is motivated by the synthesis processes in DNA replication which can be simulated by derivations in the WK grammars. Although these WK grammars use different restriction of production rules, they all generate double-stranded strings dynamically: the WK complementarity can only be checked after generating both the strands of a complete double-stranded string. In other words, the WK grammars produce each stranded string "independently" and only check for the WK complementarity of a complete generated double stranded string at the end. On the other hand, they cannot fully describe the replication and synthesis behavior of DNA molecules. Motivated by the WK regular grammar, a static WK (sWK) regular grammar is proposed as an analytical counterpart of the sticker system and uses rules as in a regular grammar. This new theoretical model generates both stranded strings "dependently", i.e., checking for WK complementarity of each complete substring and fully illustrate the replication of DNA in DNA molecules.

This paper is organized as follows: Section 1 introduces the background of the research. In Section 2, some preliminary concepts including the basic terms, theorem and definitions of formal language theory, sticker systems and automata are presented. Next, the definition of sWK grammars and the language generated by these grammars together with some examples are discussed and shown in Section 3. In Section 4, the computational power of sWK regular grammars is investigated and presented.

In the next section, some preliminaries which are used in this paper are discussed.

PRELIMINARIES

This section includes some preliminary concepts which involve the basic terms, theorem and definitions that are used in this paper. The reader may refer to ([13],[14],[15]) for detailed information regarding the basic concepts of formal language theory, sticker systems and automata. 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 $\emptyset$. The power set of X is denoted by 2^X.

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 contains λ and to exclude the empty string, the symbol T^+ is defined as the set of all nonempty finite strings over T where $T^+ = T^* - \{\lambda\}$.

A Chomsky grammar (sometimes simply called a grammar) is a set of rule formation for rewriting strings. Therefore, a grammar acts as a mechanism to describe languages mathematically; in other words, acting as a language generator. 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 $x \to y \in P$.

A grammar normally generates many strings by applying the rules in different orders. Here, the set of all terminal strings is the language generated by the grammar which is 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 [14]:

(i) *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 $x \in (N \cup T)^+$.
(ii) *context-free*, if each rule $u \to v \in P$ has $u \in N$.
(iii) *linear*, if each rule $u \to v \in P$ has $u \in N$ and $v \in T^* \cup T^* N T^*$.
(iv) *right-linear*, if each rule $u \to v \in P$ has $u \in N$ and $v \in T^* \cup T^* N$.
(v) *left-linear*, if each rule $u \to v \in P$ has $u \in N$ and $v \in T^* \cup N T^*$.
(vi) *regular*, if each rule $u \to v \in P$ has $u \in N$ and $v \in T \cup TN \cup \{\lambda\}$.

The family of languages generated by regular grammars is equal to the family of languages generated by right- or left-linear grammars. All those families of languages generated by context-sensitive grammars, context-free grammars, linear grammars and regular grammars 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. Hence, the following strict inclusion holds for *Chomsky hierarchy*.

Theorem 1. [14]

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

As mentioned before, Watson-Crick automata (WKA) or Watson-Crick finite automata (WKFA) is an extension of finite automata. A finite automata (FA) can be defined as a quintuple

$$M = (Q, V, q_0, F, \delta),$$

where Q is a finite set of states, V is a finite set of symbols called the input alphabet, $q_0 \in Q$ is the initial state, $F \subseteq Q$ is the set of final states and $\delta : Q \times V \to 2^Q$ is called a transition function. The family of languages accepted by finite automata is denoted as **FA**.

On the other hand, WKFA is defined as a 6-tuple

$$M = (Q, V, q_0, F, \delta, \rho),$$

where Q, V, q_0, F are defined as for a finite automata, $\rho \subseteq V \times V$ is the complementarity relation and $\delta : Q \times \langle V^* \times V^* \rangle \to 2^Q$ is called a transition function such that $\delta\left(q, \binom{u}{v}\right) \neq \emptyset$ only for finitely many triples $(q, u, v) \in Q \times V^* \times V^*$. The language accepted by WKFA M is

$$L(M) = \left\{u : \begin{bmatrix} u \\ v \end{bmatrix} \in WK_\rho(V)\right\} \text{ and } q_0 \begin{bmatrix} u \\ v \end{bmatrix} \to^* \begin{bmatrix} u \\ v \end{bmatrix} q,$$

where $q \in F$. The family of languages accepted by finite automata is denoted as **WKFA.**

The definitions of Watson-Crick grammars are presented as follows.

Definition 1. [12] A Watson-Crick (WK) grammar $G = (N, T, \rho, S, P)$ is called

- *regular* if each production has the form $A \to \langle u/v \rangle$ where $A, B \in N$ and $\langle u/v \rangle \in \langle T^*/T^* \rangle$.
- *linear* if each production has the form $A \to \langle u_1/v_1 \rangle B \langle u_2/v_2 \rangle$ or $A \to \langle u/v \rangle$ where $A, B \in N$ and $\langle u/v \rangle, \langle u_1/v_1 \rangle, \langle u_2/v_2 \rangle \in \langle T^*/T^* \rangle$.
- *context-free* if each production has the form $A \to \alpha$ 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 $V^* \times V^*$.

In order to generate or form a complete double sequence of DNA, the sticker system uses a sticker operation on DNA molecules. Let V be an alphabet (a finite set of abstract symbols) for a symmetric relation $\rho \in V \times V$ over V (of complementarity). The symbol V^* represents a set of all strings which includes the empty string denoted as λ, composed of elements of V and V^+ is the set $V^* - \{\lambda\}$. The set

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

denotes the *Watson-Crick domain* associated to alphabet V and complementarity relation ρ. The elements $\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \in WK_\rho(V)$ are 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 complement and have the same length.

Apart from that, the set of incomplete molecules is denoted as:

$$W_\rho(V) = L_\rho(V) \cup R(V) \cup LR_\rho(V),$$

Where

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

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

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

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

Definition 2. [14] 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 *dominoes*).

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 on the right such that the sticker operation is associative. Moreover, a sequence $x_1 \Rightarrow x_2 \Rightarrow \cdots \Rightarrow x_k$ 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 it is defined by

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

There are several restricted variants of the sticker system which are arbitrary, one-sided, regular, simple, simple and one-sided, or simple and regular denoted as $\mathbf{ASL}(\boldsymbol{\alpha})$, $\mathbf{OSL}(\boldsymbol{\alpha})$, $\mathbf{RSL}(\boldsymbol{\alpha})$, $\mathbf{SSL}(\boldsymbol{\alpha})$, $\mathbf{SOSL}(\boldsymbol{\alpha})$, $\mathbf{SRSL}(\boldsymbol{\alpha})$ respectively where $\alpha \in \{n, p, b\}$ and the letters n, p, b represent no restrictions, primitive and delay computation, respectively [14]. In 1998, Păun and Rozenberg [16] investigated the generative power of several variants of sticker systems and the results of characterizations of regular, linear, and recursively enumerable languages were obtained. The definitions of $\mathbf{RSL}(\boldsymbol{\alpha})$ and $\mathbf{OSL}(\boldsymbol{\alpha})$ are listed in the following.

Defintion 3. [14] A language generated from a sticker system is defined as a *regular sticker language* (**RSL**) if for each pair $(u, v) \in D$, we have either $u = \lambda$.

Defintion 4. [14] A language generated from a sticker system is defined as a *one-sided sticker language* (**OSL**) if for each pair $(u, v) \in D$, we have either $u = \lambda$ or $v = \lambda$.

The relationship between the family of regular sticker language in sticker system with the family of regular language in Chomsky hierachy is given in the next corollary.

Corollary 1. [14] $\mathbf{RSL}(\boldsymbol{\alpha}) = \mathbf{OSL}(\boldsymbol{\alpha}) = \mathbf{REG}, \alpha \in \{n, p, b\}$.

In the next section, the definition of sWK regular grammars with some examples are presented.

MAIN DEFINITIONS

In this section, the definition of static Watson-Crick regular grammars which consist of right-linear and left-linear grammars are introduced.

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

(i) $S \to \begin{bmatrix} u \\ v \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix} A$ where $A \in N - \{S\}$, $\begin{bmatrix} u \\ v \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \in R_\rho(T)$;

(ii) $A \to \begin{pmatrix} x \\ y \end{pmatrix} B$ where $A, B \in N - \{S\}$, $\begin{pmatrix} x \\ y \end{pmatrix} \in LR_\rho{}^*(T)$;

or

(iii) $A \to \begin{pmatrix} x \\ y \end{pmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$ where $A \in N - \{S\}$, $\begin{pmatrix} x \\ y \end{pmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \in L_\rho(T)$.

Definition 6. A static Watson-Crick (sWK) left-linear grammar is a 5-tuple $G = (N, T, \rho, S, P)$ where N, T are disjoint alphabets of nonterminal and terminal respectively, $\rho \in T \times T$ is a symmetric relation (Watson-Crick complementarity), $S \in N$ is a start symbol (axiom) and P is a finite set of production rules in the form of

(i) $S \to A \begin{pmatrix} x \\ y \end{pmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$ where $A \in N - \{S\}$, $\begin{pmatrix} x \\ y \end{pmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \in L_\rho(T)$;

(ii) $A \to B \begin{pmatrix} x \\ y \end{pmatrix}$ where $A, B \in N - \{S\}$, $\begin{pmatrix} x \\ y \end{pmatrix} \in LR_\rho{}^*(T)$;

or

(iii) $A \to \begin{bmatrix} u \\ v \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$ where $A \in N - \{S\}$, $\begin{bmatrix} u \\ v \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \in R_\rho(T)$.

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 $\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \lambda \\ \lambda \end{bmatrix}$.

Next, the derivation step for sWK regular grammar and the language generated by this grammar are given.

Definition 7. Let $G = (N, T, \rho, S, P)$ be a sWK regular grammar. We say that α derives β in G, denoted/written as $\alpha \underset{G}{\Rightarrow} \beta$ (it is clear from the context that we omit G and write $\alpha \Rightarrow \beta$) if and only if

(i) $\alpha = S$ and $\beta = \begin{bmatrix} u \\ v \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix} A$ where $\alpha \Rightarrow \beta \in P$;

(ii) $\alpha = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} A$ and $\beta = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} B$ where $A, B \in N - \{S\}$, $\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} \in R_\rho(T)$ and $A \to \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} B \in P$;

or

(iii) $\alpha = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} A$ and $\beta = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$ where $A \in N - \{S\}$, $\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} \in R_\rho(T)$ and $A \to \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \in P$.

The reflexive and transitive closure of $\underset{G}{\Rightarrow}$ or $(\Rightarrow)$ is denoted by $\underset{G}{\Rightarrow}^*$ or $(\Rightarrow^*)$.

Definition 8. The language generated by a sWK regular grammar G denoted by $L(G)$, is defined as

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

The family of languages generated by a sWK regular grammar is denoted by **SREG**.

The following example is illustrated to show the family of languages generated by **SREG**.

Example 1. Let

$G = (\{S, A, B, C, D\}, \{a, b, c, d\}, \{(a, a), (b, b), (c, c), (d, d)\}, S, P)$

be a sWK regular grammar and P consists of the following rules:

$$S \to \begin{pmatrix} a \\ \lambda \end{pmatrix} A,$$

$$A \to \begin{pmatrix} \lambda \\ a \end{pmatrix} \begin{bmatrix} a \\ a \end{bmatrix} \begin{pmatrix} a \\ \lambda \end{pmatrix} A, \quad A \to \begin{pmatrix} \lambda \\ a \end{pmatrix} \begin{bmatrix} b \\ b \end{bmatrix} \begin{pmatrix} b \\ \lambda \end{pmatrix} B,$$

$$B \to \begin{pmatrix} \lambda \\ b \end{pmatrix} \begin{bmatrix} b \\ b \end{bmatrix} \begin{pmatrix} b \\ \lambda \end{pmatrix} B, \quad B \to \begin{pmatrix} \lambda \\ b \end{pmatrix} \begin{bmatrix} c \\ c \end{bmatrix} \begin{pmatrix} c \\ \lambda \end{pmatrix} C,$$

$$C \to \begin{pmatrix} \lambda \\ c \end{pmatrix} \begin{bmatrix} c \\ c \end{bmatrix} \begin{pmatrix} c \\ \lambda \end{pmatrix} C, \quad C \to \begin{pmatrix} \lambda \\ c \end{pmatrix} \begin{bmatrix} d \\ d \end{bmatrix} \begin{pmatrix} d \\ \lambda \end{pmatrix} D,$$

$$D \to \begin{pmatrix} \lambda \\ d \end{pmatrix} \begin{bmatrix} d \\ d \end{bmatrix}.$$

From this, we obtain the derivation:

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

Hence, G generates the language

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

Next, the computational power related to sWK regular grammars is discussed in the following section.

RESULTS AND DISCUSSION

Here, the computational power of a sWK regular grammar is investigated by finding the relationship between **SREG** with the families in the Chomsky hierarchy, sticker system and also in Watson-Crick grammar.

The following lemma immediately follows from the definition of a sWK regular grammar.

Lemma 1. REG ⊆ SREG.

Proof. For a regular grammar $G = (N, T, S, P)$, its sWK variant $G' = (N, T, \rho, S, P')$ is defined as follows:

(i) $\rho = \{(a, a) \mid a \in T\}$,

(ii) 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 $L(G') = L(G)$. ■

Since the sWK grammar is a grammar counterpart of sticker system, the relation between the regular sticker language, **RSL(*n*)** with **SREG** is shown as follows in order to investigate the generative power between these two languages.

Lemma 2. The following inclusion holds:

RSL(*n*) ⊆ SREG.

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

1. $S \to xB$ for each $x \in A$.
2. $B \to vB$ for each $(\lambda, v) \in D$.
3. $B \to \begin{pmatrix} \lambda \\ \lambda \end{pmatrix}$.

First, we show that $L(\gamma) \subseteq L(G)$. Suppose $w \in L(\gamma)$. Then, there are some $x \in A$ and $(\lambda, v_1), (\lambda, v_2), \cdots, (\lambda, v_k)$ such that

$$\begin{bmatrix} w \\ w \end{bmatrix} = \mu(\mu(\cdots \mu(\mu(x, v_1), v_2) \cdots), v_{k-1}) v_{k.} \qquad (1)$$

The computation in (1) can be simulated by the following derivation in G:

$$\begin{aligned} &S \Rightarrow xB \Rightarrow xv_1B \Rightarrow xv_1v_2B \Rightarrow \\ &\Rightarrow xv_1v_2 \cdots v_{k-1}B \Rightarrow xv_1v_2 \cdots v_{k-1}v_kB \\ &\Rightarrow xv_1v_2 \cdots v_{k-1}v_k. \end{aligned}$$

Second, we show that $L(G) \subseteq L(\gamma)$. If $w \in L(G)$ and $w = x_1x_2x_3 \cdots x_k$ for some $x_1 \in R_\rho(T), x_2x_3 \cdots x_k \in LR_\rho{}^*(T)$, then

$$S \Rightarrow x_1B \Rightarrow^* x_1x_2x_3 \cdots x_kB \Rightarrow x_1x_2x_3 \cdots x_k. \qquad (2)$$

By construction of $G, x_1 \in A$ and for each x_i, $2 \leq i \leq k$ and $(\lambda, x_i) \in D$. Thus, (2) can easily be changed to the computation in γ:

$$\begin{bmatrix} w \\ w \end{bmatrix} = \mu(\mu(\cdots \mu(\mu(x, v_1), v_2) \cdots), v_{k-1}) v_{k.} \qquad ■$$

Lemma 3. The following proper inclusion holds:

RSL(*n*) ⊂ SREG.

Proof. From Lemma 2 and Corollary 1,

RSL(*n*) = REG ⊆ SREG. (3)

Next, we show that the language $L = \{a^n b^n \mid n \geq 2\} \in$ **SREG**, which implies the strictness of the inclusion in (3). Consider the sWK regular grammar $G = (\{S, A, B, C, D\}, \{a, b\}, \rho, S, P)$ where P contains the following productions:

(i)	$S \to \begin{bmatrix} a \\ a \end{bmatrix} \begin{pmatrix} a \\ \lambda \end{pmatrix} A,$	(v)	$C \to \begin{pmatrix} b \\ \lambda \end{pmatrix} B,$
(ii)	$A \to \begin{pmatrix} a \\ \lambda \end{pmatrix} A,$	(vi)	$C \to \begin{pmatrix} \lambda \\ b \end{pmatrix} D,$
(iii)	$A \to \begin{pmatrix} b \\ \lambda \end{pmatrix} B,$	(vii)	$D \to \begin{pmatrix} \lambda \\ b \end{pmatrix} D,$
(iv)	$B \to \begin{pmatrix} \lambda \\ a \end{pmatrix} C,$	(viii)	$D \to \begin{bmatrix} b \\ b \end{bmatrix}.$

Thus, the derivation for each of the production rules is defined as follows:

Step 1. From rule (i):

$$S \Rightarrow \begin{bmatrix} a \\ a \end{bmatrix} \begin{pmatrix} a \\ \lambda \end{pmatrix} A. \qquad (4)$$

Step 2. Derivation (4) can be continued with rule (ii) or rule (iii). Without the loss of generality, we apply rule (ii) $k \geq 0$ times and apply rule (iii):

$$S \Rightarrow^* \begin{bmatrix} a \\ a \end{bmatrix} \begin{pmatrix} a \\ \lambda \end{pmatrix} \begin{pmatrix} a^k \\ \lambda \end{pmatrix} \begin{pmatrix} b \\ \lambda \end{pmatrix} B = \begin{bmatrix} a \\ a \end{bmatrix} \begin{pmatrix} a^{k+1} \\ \lambda \end{pmatrix} \begin{pmatrix} b \\ \lambda \end{pmatrix} B. \qquad (5)$$

Step 3. Derivation (5) can only be continued with rule (iv):

$$S \Rightarrow^* \begin{bmatrix} a \\ a \end{bmatrix} \begin{pmatrix} a^{k+1} \\ \lambda \end{pmatrix} \begin{pmatrix} b \\ \lambda \end{pmatrix} \begin{pmatrix} \lambda \\ a \end{pmatrix} C = \begin{bmatrix} aa \\ aa \end{bmatrix} \begin{pmatrix} a^k \\ \lambda \end{pmatrix} \begin{pmatrix} b \\ \lambda \end{pmatrix} C. \qquad (6)$$

Step 4. Derivation (6) can be continued with rule (v) and rule (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):

$$S \Rightarrow^* \begin{bmatrix} aa \\ aa \end{bmatrix} \begin{pmatrix} a^k \\ a^k \end{pmatrix} \begin{pmatrix} b \\ \lambda \end{pmatrix} \begin{pmatrix} b^k \\ \lambda \end{pmatrix} C = \begin{bmatrix} a^{k+2} \\ a^{k+2} \end{bmatrix} \begin{bmatrix} b \\ b \end{bmatrix} \begin{pmatrix} b^k \\ \lambda \end{pmatrix} D. \qquad (7)$$

Step 5. To complete derivation (7), we apply rule (vii) k times and the derivation is completed with rule (viii):

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

Thus, $L(G) = \{a^n b^n | n \geq 2\}$. ■

Next, we show that **SREG** can generate some non-context free language, as shown in the Lemma 4.

Lemma 4.

SREG – **CF** $\neq \emptyset$.

Proof. We will show that the language $L = \{a^n b^n c^n | n \geq 2\} \in$ **SREG**. Consider the sWK regular grammar $G = (\{S, A, B, C, D\}, \{a, b, c\}, \rho, S, P)$ where P contains the following productions:

$$\begin{array}{llll}
\text{(i)} & S \to \begin{bmatrix}a\\a\end{bmatrix}\binom{a}{\lambda}A, & \text{(vi)} & C \to \binom{c}{\lambda}D, \\
\text{(ii)} & A \to \binom{a}{\lambda}A, & \text{(vii)} & D \to \binom{\lambda}{b}E, \\
\text{(iii)} & A \to \binom{bb}{\lambda}B, & \text{(viii)} & E \to \binom{c}{\lambda}D, \\
\text{(iv)} & B \to \binom{\lambda}{a}C, & \text{(ix)} & E \to \binom{\lambda}{c}E, \\
\text{(v)} & C \to \binom{b}{\lambda}B, & \text{(x)} & E \to \binom{\lambda}{\lambda}.
\end{array}$$

The derivation using the above production rules is as follows:

Step 1. From rule (i):

$$S \Rightarrow \begin{bmatrix}a\\a\end{bmatrix}\binom{a}{\lambda}A. \tag{8}$$

Step 2. Derivation (8) can be continued with rule (ii) or rule (iii). Without the loss of generality, we apply rule (ii) $k \geq 0$ times and apply rule (iii):

$$S \Rightarrow^* \begin{bmatrix}a\\a\end{bmatrix}\binom{a}{\lambda}\binom{a^k}{\lambda}\binom{bb}{\lambda}B = \begin{bmatrix}a\\a\end{bmatrix}\binom{a^{k+1}}{\lambda}\binom{bb}{\lambda}B. \tag{9}$$

Step 3. Derivation (9) can only be continued with rule (iv):

$$S \Rightarrow^* \begin{bmatrix}a\\a\end{bmatrix}\binom{a^{k+1}}{\lambda}\binom{bb}{\lambda}\binom{\lambda}{a}C = \begin{bmatrix}aa\\aa\end{bmatrix}\binom{a^k}{\lambda}\binom{bb}{\lambda}C. \tag{10}$$

Step 4. Derivation (10) can be continued with rule (v) and rule (vi). Rule (v) must be applied k times to complete the lower strand of $\binom{a^k}{\lambda}$, which results in applying rule (iv) k times, and then we apply rule (vi):

$$S \Rightarrow^* \begin{bmatrix}aa\\aa\end{bmatrix}\binom{a^k}{a^k}\binom{bb}{\lambda}C = \begin{bmatrix}a^{k+2}\\a^{k+2}\end{bmatrix}\binom{b^{k+2}}{\lambda}\binom{c}{\lambda}D. \tag{11}$$

Step 5. Derivation (11) can be continued with rule (vii) and rule (viii). Rule (vi) must be applied k times to complete the lower strand of $\binom{b^k}{\lambda}$, which results in applying rule (vi) k times, and then we apply rule (viii):

$$S \Rightarrow^* \begin{bmatrix}a^{k+2}\\a^{k+2}\end{bmatrix}\binom{b^{k+2}}{\lambda}\binom{c}{\lambda}D = \begin{bmatrix}a^{k+2}\\a^{k+2}\end{bmatrix}\begin{bmatrix}b^{k+2}\\b^{k+2}\end{bmatrix}\begin{bmatrix}c\\c\end{bmatrix}\binom{c^k}{\lambda}D. \tag{12}$$

Step 6. To complete derivation (12), we apply rule (ix) k times to complete the lower strand of $\binom{c^k}{\lambda}$ and the derivation is completed with rule (x):

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

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

Since **RSL(*n*)** ⊂ **SREG** as in Lemma 3, then the following lemma shows the relation between **WKREG** and **SREG**.

Lemma 5. The following inclusion holds:

WKREG ⊆ **SREG**.

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

Let $P_1 = \left\{A \to \binom{x}{y}B \in P | x \neq \lambda \text{ and } y \neq \lambda\right\}$ and $P_2 = \left\{A \to \binom{x}{y} \in P | x \neq \lambda \text{ and } y \neq \lambda\right\}$.

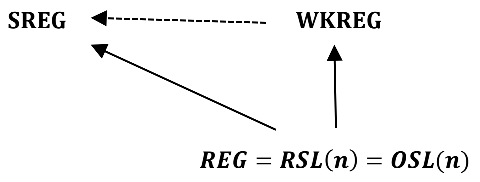
(i) For each production $r: A \to \binom{x}{y}B \in P_1$, we introduce the new productions $A \to \binom{x}{\lambda}A_r$, $A_r \to \binom{\lambda}{y}B$ where A_r is a new nonterminal.

(ii) For each production $r: A \to \binom{x}{y} \in P_2$, we introduce the new productions $A \to \binom{x}{\lambda}A_r$ and $A_r \to \binom{\lambda}{y}$.

Sets $P_1{}'$ and $P_2{}'$ contain the productions defined in (i) and (ii) above respectively. We set $P' = \big(P - (P_1 \cup P_2)\big) \cup P_1{}' \cup P_2{}' \cup \{S' \to S\}$ where S' is a new nonterminal. Next, N' contains all nonterminals of N and new nonterminals introduced in the construction of new productions. In a derivation $S \Rightarrow^* w$ in G, each application of production $r: A \to \binom{x}{y}B \in P_1$ or $r: A \to \binom{x}{y} \in P_2$ is replaced with the application of the sequence of productions $A \to \binom{x}{\lambda}A_r$, $A_r \to \binom{\lambda}{y}B \in P_1{}'$ or $A \to \binom{x}{\lambda}A_r$, $A_r \to \binom{\lambda}{y} \in P_2{}'$ resulting in a derivation $S' \Rightarrow S \Rightarrow^* w$ in G'. The similar construction also holds for the inverse case. ■

All the results above are sumarized in the following theorem.

Theorem 2. The relation in Fig. 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.



$$\begin{tikzcd}
\mathbf{SREG} & \mathbf{WKREG} \arrow[l, dashed] \\
& \boldsymbol{REG = RSL(n) = OSL(n)} \arrow[ul] \arrow[u]
\end{tikzcd}$$

Fig. 1 The hierarchy of sWK, WK, Chomsky and sticker language families.

CONCLUSION

In this paper, we defined a new theoretical model known as the static Watson-Crick regular grammar and investigated its computational power. Based on the results obtained, we can conclude that:

(i) the family of regular languages is strictly included in the family of static Watson-Crick regular languages,
(ii) static Watson-Crick regular grammars can generate non context-free languages,
(iii) the family of Watson-Crick regular languages is included in the family of static Watson-Crick regular languages.

ACKNOWLEDGEMENT

The first author would like to thank UTM Zamalah for funding her studies at UTM under the Zamalah Scholarship. The second and third authors would also like to thank the Ministry of Education (MOE) and Research Management Centre (RMC), Universiti Teknologi Malaysia (UTM) for the financial funding through Research University Grant Vote No. 13H18. The fourth author is grateful to IIUM for financial support through Research Initiative Grant Scheme RIGS16-368-0532.

REFERENCES

[1] L.M. Adleman, Molecular Computation of Solutions to Combinatorial Problems, Science 266 (1994) 1021-1024.
[2] D. Boneh, C. Dunworth, R.J. Lipton, J. Sgall, On the Computational Power of DNA, Discrete Applied Mathematics 71 (1996) 79-94.
[3] R.J. Lipton, DNA Solution of Hard Computational Probelms, Science 268 (1995) 542–545.
[4] N. Bouhmala, A Variable Neighborhood Walksat-based Algorithm for MAX-SAT Problems, The Scientific World Journal (2014) 1-11.

[5] M. Darehmiraki, A New Solution for Maximal Clique Problem, Biosystems 95(2) (2009) 145-149.

[6] M. Razzazi, M. Roayaei, Using Sticker Model of DNA Computing to Solve Domatic Partition, Kernel and Induced Path Problems, Journal Information Sciences 181(17) (2011) 3581-3600.

[7] T. Head, Formal Language Theory and DNA: An Analysis of the Generative Capacity of Specific Recombinant Behaviors, Bulletin of Mathematical Biology 49(6) (1987) 737-759.

[8] L. Kari, G. Paun, G. Rozenberg, A. Salomaa, S. Yu, DNA Computing, Sticker Systems and Universality, Acta Informatica 35(5) (1998) 401-420.

[9] Y.S. Gan, W.H. Fong, N.H. Sarmin, The Generative Power of Weighted One-Sided and Regular Sticker Systems, In AIP Conference Proceedings, 2014, 1602 (1) p. 855-862.

[10] R. Freund, G. Paun, G. Rozenberg, A. Salomaa, Watson-Crick Finite Automata, in: Proc. 3rd DIMACS Workshop on DNA based Computers, Philadelphia, 1997, p. 297-328.

[11] K.G. Subramanian, S. Hemalatha, I. Venkat, On Watson-Crick Automata, In: Proc. the Second International Conference on Computational Science, Engineering and Information Technology, Coimbatore, India, 2012, p. 151-156.

[12] N.L.M. Zulkufli, S. Turaev, M.I.M. Tamrin, M. Azeddine, Closure Properties of Watson-Crick Grammars, In AIP Conference Proceedings, 2015, 1691 (1) p. 040032.

[13] E. Czeizler, E. Czeizler, A Short Survey on Watson-Crick Automata, Bulletin of the EATCS 88 (2006). 104-119.

[14] G. Păun, G. Rozenberg, A. Salomaa, DNA Computing: New Computing Paradigms, Springer, 1998, p. 82-153.

[15] P. Linz, An Introduction to Formal Languages and Automata, Jones and Barlett Publishers, 2006, p. 16.

[16] G. Păun, G. Rozenberg, Sticker Systems, Theoretical Computer Science 204 (1998) 183-203.