Closure Properties of Static Watson-Crick Linear and Context-Free Grammars

Aqilahfarhana Abdul Rahman,[1, a)] Wan Heng Fong,[1, b)] Nor Haniza Sarmin,[1, c)] and Sherzod Turaev[2, d)]

[1)] *Department of Mathematical Sciences, Faculty of Science, Universiti Teknologi Malaysia, 81310 UTM Johor Bahru, Malaysia.*
[2)] *Department of Computer Science and Software Engineering, College of Information Technology, United Arab Emirates University, P.O.Box 15551, Al-Ain, United Arab Emirates.*

[a)] *Corresponding author: aqilahfarhana2@graduate.utm.my*
[b)] *Electronic mail: fwh@utm.my*
[c)] *Electronic mail: nhs@utm.my*
[d)] *Electronic mail: sherzod@uaeu.ac.ae*

Abstract. In DNA computing, a sticker system is a computing mechanism involving the Watson-Crick complementarity of DNA molecules. The sticker system is known as a language generating device based on the sticker operation which is analyzed through the concept of formal language theory. The grammar of a formal language can be described by determining finite sets of variables, terminal symbols and production rules. Research on the grammar which uses the Watson-Crick complementarity has been done previously, known as Watson-Crick grammars. As an improvement to the Watson-Crick grammars, the static Watson-Crick grammars have been proposed as an analytical counterpart of sticker system which consist of regular grammar, linear grammar and context-free grammar. In this research, the closure properties of static Watson-Crick linear and context-free grammars are investigated. The result shows that the families of languages generated by static Watson-Crick linear and context-free grammars are closed under different operations.

INTRODUCTION

Computers have been used in every possible way over the years. However, since computer technology has its physical limit, some researchers and engineers have looked for solutions to these traditional models. The limitation of traditional silicon-based computer involves the speed and density, design complexity, non-recurring and high cost, power consumption and heat dissipation. As an alternative to traditional computer, new computation technology is proposed which includes biological (DNA), optical, molecular and quantum computing techniques.

DNA computing is based on the double-stranded structure of DNA molecules. There are two fundamental features which are necessary in performing the computation using DNA, namely Watson-Crick (WK) complementarity and massive parallelism. The former characteristic provides far-reaching findings regarding the data on the DNA strands and the other strand can be decoded according to the complementarity by inspecting the data encrypted on a single-strand. Meanwhile, the latter characteristic allows the building of many DNA strands with multiple operations conducted concurrently on the encoded data.

There are two different theoretical models of DNA computing which have been introduced by Kari et al. [2] and Paun et al. [3] namely sticker systems and Watson-Crick automata, respectively. Sticker system is a language generating device which is based on the sticking operation and it is a model of techniques used by Adleman [4]. The sticking operation begins with a well-started sequence and then prolongs to the right or to the left of the generated sequences by using a set of single or double-stranded complementary sequence until a complete sequence is obtained. In Watson-Crick finite automata, the two reading heads and the two input tapes are developed based on the concept of finite automata. In order to increase the generative power, some additional restrictions have been proposed for sticker systems [5, 6, 7] and Watson-Crick automata [8, 9, 10].

Different formal languages (formal grammars) have been widely used in the computational research of molecular processes. In 2012, Watson-Crick regular grammar has been introduced by Subramanian et al. [11]. Later, this idea has been used and modified in 2015 and extended to the linear grammar and context-free grammar [12]. Then, the static Watson-Crick grammars are proposed as an analytical counterpart of the sticker system based on Watson-Crick grammar [13]. Static Watson-Crick grammars generate both stranded strings by checking for the complementarity of each complete substring, while Watson-Crick grammars generate each-stranded string by checking for the Watson-Crick complementarity of a complete generated double-stranded string at the end of the derivation.

In this research, we mainly focus on static Watson-Crick linear grammars and static Watson-Crick context-free grammars. We investigate the closure properties of the family of languages generated by both grammars. The study of closure properties is necessary for checking whether the language generated by the grammar belongs to the family of the languages, which is significant in grammatical theory and practices.

This paper is organized as follows: Section 1 gives the background and introduction of the paper. Section 2 presents definitions and notations from the theories of formal languages, Watson-Crick grammars and static Watson-Crick grammars. The closure properties of both grammars are presented in Section 3. Lastly, a summary and future study for this research are given in Section 4.

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

PRELIMINARIES

In this section, some basic notations, terms and definitions related to formal language theory, sticker systems, Watson-Crick grammars and static Watson-Crick grammars are presented. For further details, the reader can refer to [12-14].

A string over a set T is a finite sequence of symbols from the alphabet T. The set of all finite strings is denoted as T^*. The set of symbols from which the strings are constructed is called the alphabet, T. An empty string is denoted by λ. The symbol T^+ is the set of all nonempty finite strings over T which excludes the empty string. Therefore, a language is defined as a subset of the set of strings where $L \subseteq T^*$. The membership of an element to a set is denoted by $\in$ and the empty set is denoted by $\emptyset$.

A Chomsky grammar is defined as a quadruple $G = (N,T,S,P)$ where the alphabet N is the finite set of variables, T is the finite set of terminal alphabets, $S \in N$ is the 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 rules of $(x,y) \in P$ are written in the form of $x \to y$ where $x \in (N\cup T)^+$ and $y \in (N\cup T)^*$. Here, *u directly derives v* or v is derived from u with respect to G, written as $u \Rightarrow v$, if and only if $u = u_1xu_2, v = u_1yu_2$, for some $u_1u_2 \in (N\cup T)^*$ and $x \to y \in P$. The set of all terminal strings is the language generated by the grammar which is defined by $L(G) = \{w \in T^* : S \Rightarrow^* w\}$. The Chomsky grammars are classified into recursively enumerable, context-sensitive, context-free, linear and regular grammars. The families of languages generated by *recursive enumerable*, *context-sensitive*, *context-free*, *linear* and *regular* grammars are denoted as **RE**, **CS**, **CF**, **LIN** and **REG** respectively.

The definition of a Watson-Crick grammar is presented in the following [12]:

Definition 1. A Watson-Crick 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 the symbol V is replaced with T for $\langle V^*/V^*\rangle$. The sticker system is developed by using sticker operation on DNA molecules. Let V be an alphabet and let ρ be a symmetric relation where $\rho \in V \times V$ over V. The set

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

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 double-stranded sequences. The pair of $\begin{pmatrix} x \\ y \end{pmatrix}$ indicates there is no relation between the elements x and y; while $\begin{bmatrix} x \\ y \end{bmatrix}$ indicates that the elements in the upper and lower strand are complement and have the same length.

The set of incomplete molecules is denoted as $W_\rho(V) = L_\rho(V) \cup R_\rho(V) \cup LR_\rho(V)$ where

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

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

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

Another notion of $LR_\rho(V)$ is introduced in this research, where

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

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

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 a 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 the one to the right such that the sticker operation is associative. Moreover, a sequence $x_1 \Rightarrow x_2 \Rightarrow \ldots \Rightarrow x_k$ 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. A sticker language is a language generated by the sticker system, γ which is defined by $L(\gamma) = \{w \in \binom{V}{V}_\rho^* | x \Rightarrow^* w, x \in A\}$. Next, the following shows the definition of static Watson-Crick grammars [13]. For static Watson-Crick regular grammars, we state only for right-linear grammars since the definition is similar with the left-linear grammars.

Definition 2. A static Watson-Crick grammar $G = (N, T, \rho, S, P)$ is called:

i. right-linear grammar if each production is in the form of $S \longrightarrow \begin{bmatrix} u \\ v \end{bmatrix} \binom{x}{y} A$ where $A \in N - \{S\}$, $\begin{bmatrix} u \\ v \end{bmatrix} \binom{x}{y} \in R_\rho(T)$; $A \longrightarrow \binom{x}{y} B$ where $A, B \in N - \{S\}$ and $\binom{x}{y} \in LR_\rho^*(T)$; $A \longrightarrow \binom{x}{y} \begin{bmatrix} u \\ v \end{bmatrix}$ where $A \in N - \{S\}$, $\binom{x}{y} \begin{bmatrix} u \\ v \end{bmatrix} \in L_\rho(T)$; or $S \longrightarrow \binom{\lambda}{\lambda}$.

ii. linear grammar if each production is in the form of $S \rightarrow \begin{bmatrix} u_1 \\ v_1 \end{bmatrix} \binom{x_1}{y_1} A \binom{x_2}{y_2} \begin{bmatrix} u_2 \\ v_2 \end{bmatrix}$ where $A \in N - \{S\}$, $\begin{bmatrix} u_1 \\ v_1 \end{bmatrix} \binom{x_1}{y_1} \in R_\rho(T)$ and $\binom{x_2}{y_2} \begin{bmatrix} u_2 \\ v_2 \end{bmatrix} \in L_\rho(T)$; $A \rightarrow \binom{x_1}{y_1} B \binom{x_2}{y_2}$ where $A, B \in N - \{S\}$ and $\binom{x_1}{y_1}, \binom{x_2}{y_2} \in LR_\rho^*(T)$; $A \rightarrow \binom{x_1}{y_1}$ where $A \in N - \{S\}$ and $\binom{x_1}{y_1} \in LR_\rho^*(T)$; or $S \longrightarrow \binom{\lambda}{\lambda}$.

iii. context-free grammar if each production is in the form of $S \rightarrow x_1 A_1 x_2 A_2 \cdots x_k A_k x_{k+1}$ where $A_i \in N - \{S\}$ for $1 \leq i \leq k, x_1 \in R_\rho(T), x_i \in LR_\rho^+(T)$ for $2 \leq i \leq k$ and $x_{k+1} \in L_\rho(T)$; $A \rightarrow y_1 B_1 y_2 B_2 \cdots y_t B_t y_{t+1}$ where $A, B_i \in N - \{S\}$ for $1 \leq i \leq t, y_i \in LR_\rho^+(T)$ for $2 \leq i \leq t+1$ and $y_1, y_{t+1} \in LR_\rho^*(T)$; $A \longrightarrow x$ where $A \in N - \{S\}$ and $x \in LR_\rho^*(T)$; or $S \longrightarrow \binom{\lambda}{\lambda}$.

The language generated by a static WK (regular, linear, context-free) grammar G is defined as $L(G) = \{u : \begin{bmatrix} u \\ v \end{bmatrix} \in WK_\rho(T)$ and $S \underset{G}{\Rightarrow^*} \begin{bmatrix} u \\ v \end{bmatrix}\}$. The families of static Watson-Crick regular, linear and context-free languages are denoted by **SREG**, **SLIN** and **SCF** respectively.

Next, we recall some closure properties which are used in this study. The union of two languages L_1 and L_2, denoted as $L_1 \cup L_2$, is the set of all strings which includes the elements in both L_1 and L_2. Next, the concatenation of two languages L_1 and L_2 is the set of all strings that is obtained when any element of L_1 is concatenate with any element of L_2 where $L_1 L_2 = \{xy : x \in L_1, y \in L_2\}$. The star-closure (or Kleene-star closure) of a language is defined as $L^* = L^0 \cup L^1 \cup L^2 \cup \cdots$ and the mirror image (reverse) of a language is the set of all string reversals where $L^R = \{w^R : w \in L\}$.

In the next section, the closure properties of static Watson-Crick linear and context-free grammars are presented.

RESULTS AND DISCUSSIONS

In this section, the closure properties of static Watson-Crick linear and context-free grammars are discussed. First, we discussed on the closure properties for static Watson-Crick linear grammars. Both static Watson-Crick linear and context-free languages have the same definition on the properties of union and mirror image. The definition on the union of the static Watson-Crick linear and context-free languages is defined next.

Definition 3. Union
Let $L(G_1)$ and $L(G_2)$ be two languages generated by static Watson-Crick linear (or context-free) grammars. The union of the two languages $L(G_1)$ and $L(G_2)$ is $L(G_1) \cup L(G_2) = \{x | x_1 \in L(G_1) \,\text{or}\, x_2 \in L(G_2)\}$.

In the next lemma, we proved that the union between two static Watson-Crick linear languages is also a static Watson-Crick linear language.

Lemma 1. If L_1 and L_2 are **SLIN**, then $L_1 \cup L_2$ is also a **SLIN**.
Proof. Let $G_1 = (N_1, T, \rho, S_1, P_1)$ and $G_2 = (N_2, T, \rho, S_2, P_2)$ be static Watson-Crick linear grammars generating languages L_1 and L_2 respectively, such that $L_1 = L(G_1)$ and $L_2 = L(G_2)$. Assume that $N_1 \cap N_2 \neq \emptyset$. Let $G = (N, T, \rho, S, P)$ be a static Watson-Crick linear grammar which generates L such that $L = L(G)$. We set $N = N_1 \cup N_2 \cup \{S\}$ which contains all nonterminals of G_1 and G_2 including the start symbol for which $S \notin N_1 \cup N_2$. Next, a set of production rules is set where $P = P_1 \cup P_2 \cup \{S \longrightarrow S_1\} \cup \{S \longrightarrow S_2\}$. A string w is in $L(G)$ when there is a derivation $S \Rightarrow S_i \underset{G_i}{\Rightarrow}^* w$ for $i = 1$ or 2. Thus, $L(G)$ is the union of languages L_1 and L_2 such that $L(G) = L_1 \cup L_2$.

Next, the definition on the concatenation of the static Watson-Crick linear languages is presented.

Definition 4. Concatenation
Let $L(G_1)$ be a language generated by static Watson-Crick regular grammars and $L(G_2)$ be a language generated by static Watson-Crick linear grammars. The concatenation of the two languages $L(G_1)$ and $L(G_2)$ is $L(G_1) \cdot L(G_2) = \{xy | x \in L(G_1) \,\text{and}\, y \in L(G_2)\}$.

Next, we proved that the concatenation between static Watson-Crick regular and linear languages is also a static Watson-Crick linear language.

Lemma 2. If L_1 is a **SREG** and L_2 is a **SLIN**, then $L_1 \cdot L_2$ is a **SLIN**.
Proof. Let $G_1 = (N_1, T, \rho, S_1, P_1)$ be a static Watson-Crick regular grammar which generates the language L_1 and let $G_2 = (N_2, T, \rho, S_2, P_2)$ be a static Watson-Crick linear grammar which generates the language L_2. Define $G = (N, T, \rho, S, P)$ as a static Watson-Crick linear grammar which generates L such that $L = L(G)$. The start symbol initiates the derivation in both G_1 and G_2 by setting $S = S_1$ for G_1. The derivation of terminal string generates w in the form of uv, where $u \in L_1$ and $v \in L_2$. The derivation of u uses only rules from P_1 and v rules from P_2. In the last production rule of P_1, we set the start symbol S_2 in P_2 as the nonterminal symbol to start the derivation of $v \in L_2$. The production rule P is defined as $P = (P_1 - \{A \longrightarrow \binom{x}{y} \in P_1\}) \cup \{A \longrightarrow \binom{x}{y} S_2 : A \longrightarrow \binom{x}{y} \in P_1\} \cup P_2$. Next, N contains all nonterminals of N_1 and N_2 such that $N = N_1 \cup N_2$. Hence, $L(G)$ is the concatenation of the languages L_1 and L_2 such that $L(G) = L_1 \cdot L_2$.

In formal language theory, the pumping lemma is used to prove that a language does not belong to the familiy of the languages. The following shows the theorem of pumping lemma for linear languages which is used to prove Lemma 3, where $|uvyz|$ and $|vy|$ denote the length of the strings $uvyz$ and vy respectively.

Theorem 1. [14] Let L be an infinite linear language. There exists some positive integer m, such that for any w in L with $|w| \geq m$ are decomposed as $w = uvxyz$ with $|uvyz| \leq m$, $|vy| \geq 1$ such that uv^ixy^iz in L for all $i = 0, 1, 2, ...$

The positive integer m can be referred as the pumping length in which every word $w \in L$ of length at least m can be written as $w = uvxyz$ with $|uvyz| \leq m$, $|vy| \geq 1$ such that uv^ixy^iz in L for all $i = 0, 1, 2, ...$ In the following lemma, we proved that the concatenation of two static Watson-Crick linear languages is not a static Watson-Crick linear language.

Lemma 3. If L_1 and L_2 are **SLIN**, then $L_1 \cdot L_2$ is not a **SLIN**.
Proof. Let $G_1 = (N_1, T, \rho, S_1, P_1)$ be a static Watson-Crick linear grammar which generates the language L_1 and let

$G_2 = (N_2, T, \rho, S_2, P_2)$ be a static Watson-Crick linear grammar which generates the language L_2. Then, we define $G = (N, T, \rho, S, P)$ which generates the language $L(G) = L_1 \cdot L_2$. We prove by contradiction. First, assume L is linear. Let $w = uvxyz$ be a string in L of length at least m, where m is the pumping length of positive integer such that for any $w \in L$, $|w| \geq m$. By pumping lemma, there exist u, v, x, y, z with $|uvyz| \leq m$, $|vy| \geq 1$ such that $w^i = uv^ixy^iz$, $i = 0, 1, 2, \ldots n$ hold. The strings v and y must be located within m symbols of the left and right ends of w, respectively, while the middle string x can be of arbitrary length. By considering all possible ways that satisfy the conditions above, it is shown that the string w is not in L. Hence, **SLIN** is not closed under concatenation.

Example 1 shows that **SLIN** is not closed under concatenation.

Example 1. Let $L(G_1) = \{a_1^n b_1^n c_1^n : n \geq 2\}$ and $L(G_2) = \{a_2^n b_2^m c_2^m d_2^n : n \geq 2, m \geq 1\}$ be **SLIN**. The concatenation of these two languages yield $L(G) = L(G_1) \cdot L(G_2) = \{a_1^n b_1^n c_1^n a_2^n b_2^m c_2^m d_2^n : n \geq 2, m \geq 1\}$ which is not a **SLIN**. To prove by using pumping lemma, assume the language is linear and apply the condition to the string $w = a_1^{m+1} b_1^{m+1} c_1^{m+1} a_2^{m+1} b_2^m c_2^m d_2^{m+1}$. The inequality $|uvyz| \leq m$ shows that in this case, the strings u, v, y, z must consist $a_1's$ and $c_1's$. If we pump the string once, we get $a_1^{(m+1)+k} b_1^{m+1} c_1^{m+1} a_2^{m+1} b_2^m c_2^m d_2^{(m+1)+j}$, with either $k \geq 1$ or $j \geq 1$, a result that is not in L. Hence, by contradiction, the language is not linear.

The concatenation between two strings which is generated by static Watson-Crick linear grammars shows that it is not included in **SLIN**. This concludes that **SLIN** is not closed under concatenation. On the other hand, the star-closure can be represented as the concatenation of L with itself for i times. Since **SLIN** is not closed under concatenation, then we conclude that **SLIN** is not closed under star-closure.

Next, the definition on the mirror image (reversal) of the static Watson-Crick linear and context-free languages is presented.

Definition 5. Mirror image
Let $L(G)$ be a language generated by static Watson-Crick linear (or context-free) grammars. The mirror image of a language is the set of all string reversals where $L(G)^R = \{w^R | w \in L(G)\}$.

Following that, we proved that the mirror image of the static Watson-Crick linear language is also a static Watson-Crick linear language.

Lemma 4. If $L_1(G)$ is a **SLIN**, then $L_1^R(G)$ is also a **SLIN**.
Proof. Let L_1 be a language which is generated by a static Watson-Crick linear grammar, $L(G_1) \in$ **SLIN**. We need to

show that $L_1^R(G) \in$ **SLIN** . Define $G = (N_1, T, \rho, S_1, P_1)$ as a static Watson-Crick linear grammar which generates the language $L_1^R(G)$. We set the production rule of P_1 in a reversal form such that $(uv)^R = v^R u^R$ where R is the reversal. The production rule P_1 consists of the following forms:

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

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

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

Hence, the reversal of L_1 is the language generated by static Watson-Crick linear grammars, i.e. $L_1^R(G) \in \mathbf{SLIN}$.

All the results above are summarized in the following theorem.

Theorem 2. The family of static Watson-Crick linear languages is closed under union, concatenation with static Watson-Crick regular language and mirror image.

Next, the closure properties of static Watson-Crick context-free languages are presented as follows. The definition on the union of the static Watson-Crick context-free languages is defined as in Definition 3.

In the following lemma, we proved that the union between two static Watson-Crick context-free languages is also a static Watson-Crick context-free language.

Lemma 5. If L_1 and L_2 are **SCF**, then $L_1 \cup L_2$ is also a **SCF**.
Proof. Let $G_1 = (N_1, T, \rho, S_1, P_1)$ and $G_2 = (N_2, T, \rho, S_2, P_2)$ be static Watson-Crick context-free grammars generating languages L_1 and L_2 respectively, such that $L_1 = L(G_1)$ and $L_2 = L(G_2)$. Without loss of generality, assume that $N_1 \cap N_2 \neq \emptyset$. Let $G = (N, T, \rho, S, P)$ be a static Watson-Crick context-free grammar which generates L such that $L = L(G)$. We set $N = N_1 \cup N_2 \cup \{S\}$ which contains all nonterminals of G_1 and G_2 including the start symbol for which $S \notin N_1 \cup N_2$. Next, a set of production rules is set where $P = P_1 \cup P_2 \cup \{S \longrightarrow S_1\} \cup \{S \longrightarrow S_2\}$. A string w is in $L(G)$ when there is a derivation $S \Rightarrow S_i \underset{G_i}{\Rightarrow^*} w$ for $i = 1$ or 2. Thus, $L(G)$ is the union of languages L_1 and L_2 such that $L(G) = L_1 \cup L_2$.

Next, the definition on the concatenation of the static Watson-Crick context-free languages is presented.

Definition 6. Concatenation
Let $L(G_1)$ and $L(G_2)$ be a language generated by static Watson-Crick context-free grammars. The concatenation of the two languages $L(G_1)$ and $L(G_2)$ is $L(G_1) \cdot L(G_2) = \{xy | x \in L(G_1) \,\text{and}\, y \in L(G_2)\}$.

Next, Lemma 6 shows that the concatenation between two static Watson-Crick context-free languages is also a static Watson-Crick context-free language.

Lemma 6. If L_1 and L_2 are **SCF**, then $L_1 \cdot L_2$ is also a **SCF**.
Proof. Let $G_1 = (N_1, T, \rho, S_1, P_1)$ and $G_2 = (N_2, T, \rho, S_2, P_2)$ be static Watson-Crick context-free grammars generating languages L_1 and L_2 respectively, such that $L_1 = L(G_1)$ and $L_2 = L(G_2)$. Let $G = (N, T, \rho, S, P)$ be a static Watson-Crick context-free grammar which generates L such that $L = L(G)$. The start symbol initiates the derivation in both G_1 and G_2 by setting $S = S_1$ for G_1. The derivation of terminal string generates w in the form of uv, where $u \in L_1$ and $v \in L_2$. The derivation of u uses only rules from P_1 and v rules from P_2. In the last production rule of P_1, we set the start symbol S_2 in P_2 as the nonterminal symbol to start the derivation of $v \in L_2$. The production rule P is defined as $P = (P_1 - \{A \longrightarrow x \in P_1\}) \cup \{A \longrightarrow xS_2 : A \longrightarrow x \in P_1\} \cup P_2$. Next, N contains all nonterminals of N_1 and N_2 such that $N = N_1 \cup N_2$. Thus, the concatenation of language $L(G) = L_1 \cdot L_2 \in \mathbf{SCF}$.

Next, the definition on the star-closure of the static Watson-Crick context-free languages is presented.

Definition 7. Star-closure
Let $L(G)$ be a language generated by static Watson-Crick context-free grammars. The star-closure of language $L(G)$ is $L(G)^* = \{L^0 \cup L^1 \cup L^2 \cup \cdots\}$.

Next, Lemma 7 shows that the star-closure of a static Watson-Crick context-free language is also a static Watson-Crick context-free language.

Lemma 7. If L_1 is a **SCF**, then L_1^* is also a **SCF**.

Proof. Let $G = (N_1, T, \rho, S, P_1)$ be a static Watson-Crick context-free grammar generating language L_1^*. We set $S = S_1$ and $N = N_1$. The production rule P is defined in the form of $P_1 = P_1 \cup (P_1 - \{A \longrightarrow x \in P_1\}) \cup \{A \longrightarrow xS_1 : A \longrightarrow x \in P_1\} \cup \{S_1 \longrightarrow \lambda\}$. The S_1 symbol generates the number of copies of P_1. Each of these initiates the derivation of a string in L_1. The concatenation of any number of strings from L_1 yields L_1^*. Therefore, the star-closure of language L is $L(G) = L_1^*$.

The definition on the mirror image (reversal) of the static Watson-Crick linear languages is presented as in Definition 5. In the following lemma, we proved that the mirror image of the static Watson-Crick context-free language is also a static Watson-Crick context-free language.

Lemma 8. If $L_1(G)$ is a **SCF**, then $L_1^R(G)$ is also a **SCF**.
Proof. Let $L_1(G) \in$ **SCF**. We need to show that $L_1^R(G) \in$ **SCF** . Define $G = (N_1, T, \rho, S_1, P_1)$ which generates the language $L_1^R(G)$ We set the production rule of P_1 in a reversal form such that $(uv)^R = v^R u^R$ where R is the reversal. The production rule P_1 consists of the following form:

i. $S \rightarrow x_{k+1}A_k x_k \cdots A_2 x_2 A_1 x_1$ where $A_i \in N - \{S\}$ for $1 \leq i \leq k, x_{k+1} \in R_\rho(T), x_i \in LR_\rho^+(T)$ for $2 \leq i \leq k$ and $x_1 \in L_\rho(T)$, and $S \rightarrow x_1 A_1 x_2 A_2 \cdots x_k A_k x_{k+1} \in P_1$;

ii. $A \rightarrow y_{t+1} B_t y_t \cdots B_2 y_2 B_1 y_1$ where $A, B_i \in N - \{S\}$ for $1 \leq i \leq t, y_i \in LR_\rho^+(T)$ for $2 \leq i \leq t+1$ and $y_1, y_{t+1} \in LR_\rho^*(T)$, and $A \rightarrow y_1 B_1 y_2 B_2 \cdots y_t B_t y_{t+1} \in P_1$; or

iii. $A \longrightarrow x$ where $A \in N - \{S\}$ and $x \in LR_\rho^*(T)$.

Hence, the reversal of L_1 is the language generated by static Watson-Crick context-free grammars, i.e. $L_1^R(G) \in$ **SCF**.

All the results above are summarized in the following theorem.

Theorem 3. The family of static Watson-Crick context-free language is closed under union, concatenation, star-closure and mirror image.

CONCLUSION

In this paper, we have shown that the family of static Watson-Crick linear languages are closed under union, concatenation with static Watson-Crick regular languages and mirror image; and the family of static Watson-Crick context-free languages are closed under union, concatenation, star-closure and mirror image. Besides, static Watson-Crick linear languages are not closed under concatenation. Thus, we found that static Watson-Crick linear languages are not closed under star-closure. Hence, the closure properties of grammars depend on the computational power of static Watson-Crick grammars. This research can be further studied by finding the closure of homomorphism, intersection, complement and so on for both static Watson-Crick linear grammars and static Watson-Crick context-free grammars.

ACKNOWLEDGMENTS

The first author would like to thank UTM Zamalah for funding her studies at Universiti Teknologi Malaysia (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), UTM for the financial funding through Fundamental Research Grant Scheme (FRGS) Vote No. 5F022.

REFERENCES

1. H. S. Hassan and M. Asghar, "Limitation of silicon based computation and future prospects," IEEE, 559–561 (2010).

2. L. Kari, G. Pǎun, G. Rozenberg, A. Salomaa, and S. Yu, Acta Informatica **35**, 401 – 420 (1998).
3. G. Pãun, G. Rozenberg, and A. Salomaa, *DNA Computing: New Computing Paradigms*, (Springer-Verlag, New York,1998).
4. L. M. Adleman, Science 1021 – 1024 (1994).
5. N. M. Sebry, N. Z. A. Hamzah, N. H. Sarmin, W. H. Fong, and S. Turaev, Malaysia Journal of Fundamental and Applied Sciences **8** (2012).
6. M. Selvarajoo, W. H. Fong, N. H. Sarmin, and S. Turaev, Malaysia Journal of Fundamental and Applied Sciences **9** (2013).
7. Y. S. Gan, W. H. Fong, N. H. Sarmin, and S. Turaev, *AIP Conference Proceedings*, Vol. 1602 (2014).
8. E. Petre, Journal of Automata, Languages and Combinatories **8**, 59 – 70 (2003).
9. M. I. M. Tamrin, S. Turaev, and T. M. T. Sembok, *AIP Conference Proceedings*, Vol. 1605 (2014).
10. L. Hegedüs, B. Nagy, and Ö. Eğecioğlu, Natural Computing **11**, 361 – 368 (2012).
11. K. Subramanian, S. Hemalatha, and I. Venkat, *Proceedings of the Second International Conference on Computational Science, Engineering and Information Technology*, (2012).
12. N. L. M. Zulkufli, S. Turaev, M. I. M. Tamrin, and M. Azeddine, *AIP Conference Proceedings*, Vol. 1691 (2015).
13. W. H. Fong, A. A. Rahman, N. H . Sarmin, and S. Turaev, International Jounral of Online Engineering **15** (2019).
14. P. Linz, *An Introduction to Formal Languages and Automata* (Jones and Barlett Publishers, United States, 2006).